Oracle Private Cloud Appliance

Concepts Guide for Release 3.0.2

F59007-02
January 2023

ORACLE

Oracle Private Cloud Appliance Concepts Guide for Release 3.0.2,

F59007-02

Contents

2 Hardware Overview

3 Appliance Administration Overview

4 Identity and Access Management Overview

7 Compute Instance Concepts

8 Block Volume Storage Overview

9 File Storage Overview

10 Object Storage Overview

Preface

This publication is part of the customer documentation set for Oracle Private Cloud Appliance Release 3.0.2. Note that the documentation follows the release numbering scheme of the appliance software, not the hardware on which it is installed. All Oracle Private Cloud Appliance product documentation is available at https://docs.oracle.com/en/engineered-systems/private-cloud-appliance/index.html.

Oracle Private Cloud Appliance Release 3.x is a flexible general purpose Infrastructure as a Service solution, engineered for optimal performance and compatibility with Oracle Cloud Infrastructure. It allows customers to consume the core cloud services from the safety of their own network, behind their own firewall.

Audience

This documentation is intended for owners, administrators and operators of Oracle Private Cloud Appliance. It provides architectural and technical background information about the engineered system components and services, as well as instructions for installation, administration, monitoring and usage.

Oracle Private Cloud Appliance has two strictly separated operating areas, known as enclaves. The Compute Enclave offers a practically identical experience to Oracle Cloud Infrastructure: It allows users to build, configure and manage cloud workloads using compute instances and their associated cloud resources. The Service Enclave is where privileged administrators configure and manage the appliance infrastructure that provides the foundation for the cloud environment. The target audiences of these enclaves are distinct groups of users and administrators. Each enclave also provides its own separate interfaces.

It is assumed that readers have experience with system administration, network and storage configuration, and are familiar with virtualization technologies. Depending on the types of workloads deployed on the system, it is advisable to have a general understanding of container orchestration, and UNIX and Microsoft Windows operating systems.

Feedback

Provide feedback about this documentation at https://www.oracle.com/goto/docfeedback.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, code in examples, text that appears on the screen, or text that you enter.
`$` prompt	The dollar sign (`$`) prompt indicates a command run as a non-root user.
`#` prompt	The pound sign (`#`) prompt indicates a command run as the `root` user.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Architecture and Design

Welcome to Oracle Private Cloud Appliance. This chapter provides an overview of Oracle's platform for building cloud services and applications inside your own data center.

Introduction to Oracle Private Cloud Appliance

Oracle Private Cloud Appliance is engineered to deliver a comprehensive suite of cloud infrastructure services within the secure environment of your on-premises network. The system integrates all required hardware and software components, and has been tested, configured and tuned for optimal performance by Oracle engineers. In essence, it is a flexible general purpose IaaS (Infrastructure as a Service) solution in the sense that it supports the widest variety of workloads. Its pluggable platform provides an excellent foundation to layer PaaS (Platform as a Service) and SaaS (Software as a Service) solutions on top of the infrastructure.

This release of Private Cloud Appliance provides API compatibility with Oracle's public cloud solution, Oracle Cloud Infrastructure. You access the core IaaS services using the same methods, tools and interfaces as with Oracle Cloud Infrastructure. An installation of Private Cloud Appliance represents a customer region. Workloads are portable between your private and public cloud environments, but the private cloud is disconnected from Oracle Cloud Infrastructure and thus runs its own control plane components in order to host its set of compatible services.

As an engineered system, Private Cloud Appliance complies with the highest business continuity and serviceability requirements. It has the capabilities to monitor all components, detect potential problems, send out alerts, and automatically log a service request. Subsequent troubleshooting and repair can be performed without affecting the uptime of the environment.

System upgrades are also designed for minimum disruption and maximum availability. Health checks are performed before an upgrade to ensure that all components are in an acceptable state. The upgrade process is modular and allows components – such as firmware, operating systems, containerized services or the system's main database – to be upgraded individually or as an integrated multi-component workflow.

Compatibility with Oracle Cloud Infrastructure

A principal objective of Private Cloud Appliance is to allow you to consume the core Oracle Cloud Infrastructure services from the safety of your own on-premises network, behind your own firewall. The infrastructure services provide a foundation for building PaaS and SaaS solutions; the deployed workloads can be migrated between the public and the private cloud infrastructure with minimal or no modification required. For this purpose, Private Cloud Appliance offers API compatibility with Oracle Cloud Infrastructure.

As a rack-scale system, Private Cloud Appliance can be considered the smallest deployable unit of Oracle Cloud Infrastructure, aligned with the physical hierarchy of the public cloud design:

Hierarchy Concept	Oracle Cloud Infrastructure Design	Oracle Private Cloud Appliance Mapping
Realm	A Realm is a superset of Regions, and the highest physical subdivision of the Oracle cloud. There are no cross-realm features. Oracle Cloud Infrastructure currently consists of a Realm for Commercial Regions and a Realm for Government Cloud Regions.	The concept of a Realm exists in Private Cloud Appliance, but it has no practical function. It allows the appliance to participate in any Realm.
Region	A Region is a geographic area. An Oracle Cloud Infrastructure Region is composed of at least three Availability Domains. It is possible to migrate or replicate data and resources between Regions.	Private Cloud Appliance is designed as a single Region. Because this private region is disconnected from any other systems, it has no practical function. Domain and system identifiers are used in system configuration instead, and mapped to the region and realm values.
Availability Domain	An Availability Domain consists of one or more data centers. Availability Domains are isolated from each other; they have independent power and cooling infrastructure and separate internal networking. A failure in one Availability Domain is highly unlikely to impact others. Availability Domains within the same region are interconnected through an encrypted network with high bandwith and low latency. This is a critical factor in providing high availability and disaster recovery.	Each Private Cloud Appliance is configured as an Availability Domain. Multiple installations are distinct from each other: they do not function as Availability Domains within the same region.
Fault Domain	A Fault Domain is a grouping of infrastructure components within an Availability Domain. The goal is to isolate downtime events due to failures or maintenance, and make sure that resources in other Fault Domains are not affected. Each Availability Domain contains three Fault Domains. Fault Domains provide anti-affinity: the ability to distribute instances so that they do not run on the same physical hardware.	Private Cloud Appliance adheres to the public cloud design: each Availability Domain contains three Fault Domains. A Fault Domain corresponds with one or more physical compute nodes.

Private Cloud Appliance also aligns with the logical partitioning of Oracle Cloud Infrastructure. It supports multiple tenancies, which are securely isolated from each other by means of tunneling and encapsulation in the appliance underlay network. Tenancies are hosted on the same physical hardware, but users and resources that belong to a given tenancy cannot interact with other tenancies. In addition, the Compute Enclave – which refers to all tenancies collectively, and to the cloud resources created and managed within them – is logically isolated from the Service

Enclave, from where the appliance infrastructure is controlled. Refer to Enclaves and Interfaces for more information.

The Compute Enclave interfaces provide access in the same way as Oracle Cloud Infrastructure. Its CLI is identical while the browser UI offers practically the same user experience. API support is also identical, but limited to the subset of cloud services that Private Cloud Appliance offers.

The consistency of the supported APIs is a crucial factor in the compatibility between the public and private cloud platforms. It ensures that the core cloud services support resources and configurations in the same way. More specifically, Private Cloud Appliance supports the same logical constructs for networking and storage, manages user identity and access in the same way, and offers the same compute shapes and images for instance deployment as Oracle Cloud Infrastructure. As a result, workloads set up in a private cloud environment are easily portable to Private Cloud Appliance and vice versa. However, due to the disconnected operating mode of the private cloud environment, workloads must be migrated offline.

Enclaves and Interfaces

From a cloud user perspective, the Private Cloud Appliance Compute Enclave offers a practically identical experience to Oracle Cloud Infrastructure. However, the appliance also runs its own specific and separate administration area known as the Service Enclave. This section describes the boundaries between the enclaves and their interfaces, which are intended for different groups of users and administrators with clearly distinct access profiles.

Enclave Boundaries

The **Compute Enclave** was deliberately designed for maximum compatibility with Oracle Cloud Infrastructure. Users of the Compute Enclave have certain permissions to create and manage cloud resources. These privileges are typically based on group membership. The Compute Enclave is where workloads are created, configured and hosted. The principal building blocks at the users' disposal are compute instances and associated network and storage resources.

Compute instances are created from a *compute image*, which contains a preconfigured operating system and optional additional software. Compute instances have a particular *shape*, which is a template of virtual hardware resources such as CPUs and memory. For minimal operation, a compute instance needs a boot volume and a connection to a virtual cloud network (VCN). As you continue to build the virtual infrastructure for your workload, you will likely add more compute instances, assign private and public network interfaces, set up NFS shares or object storage buckets, and so on. All those resources are fully compatible with Oracle Cloud Infrastructure and can be ported between your private and public cloud environments.

The **Service Enclave** is the part of the system where the appliance infrastructure is controlled. Access is closely monitored and restricted to privileged administrators. It runs on a cluster of three management nodes. Because Private Cloud Appliance is operationally disconnected from Oracle Cloud Infrastructure, it needs a control plane of its own, specific to the design and scale of the appliance. The API is specific to Private Cloud Appliance, and access is very strictly controlled. Functionality provided by the Service Enclave includes hardware and capacity management, service delivery, monitoring and tools for service and support.

Both enclaves are strictly isolated from each other. Each enclave provides its own set of interfaces: a web UI, a CLI and an API per enclave. An administrator account with full access

to the Service Enclave has no permissions whatsoever in the Compute Enclave. The administrator creates the tenancy with a primary user account for initial access, but has no information about the tenancy contents and activity. Users of the Compute Enclave are granted permission to use, manage and create cloud resources, but they have no control over the tenancy they work in, or the hardware on which their virtual resources reside.

Access Profiles

Each enclave has its own interfaces. To access the Compute Enclave, you use either the Compute Web UI or OCI CLI. To access the Service Enclave, you use either the Service Web UI or Service CLI.

> **Note:**
>
> You access the graphical interfaces of both enclaves using a web browser. For support information, please refer to the Oracle software web browser support policy.

The properties of your account determine which operations you are authorized to perform and which resources you can view, manage or create. Whether you use the web UI or the CLI makes no difference in terms of permissions. All operations result in requests to a third, central interface of the enclave: the API. Incoming requests from the Service API or Compute API are evaluated and subsequently authorized or rejected by the API service.

Different categories of users interact with the appliance for different purposes. At the enclave level we distinguish between administrators of the appliance infrastructure on the one hand, and users managing cloud resources within tenancies on the other hand. Within each enclave, different access profiles exist that offer different permissions.

In the Service Enclave, only a select team of administrators should be granted full access. There are other administrator roles with restricted access, for example for those responsible specifically for system monitoring, capacity planning, availability, upgrade, and so on. For more information about administrator roles, see Administrator Access. Whenever Oracle accesses the Service Enclave to perform service and support operations, an account with full access must be used.

When a tenancy is created, it has only one Compute Enclave user account: the tenancy administrator, who has full access to all resources in the tenancy. Practically speaking, every additional account with access to the tenancy is a regular Compute Enclave user account, with more or less restrictive permissions depending on group membership and policy definitions. It is the task of the tenancy administrator to set up additional user accounts and user groups, define a resource organization and management strategy, and create policies to apply that strategy.

Once a resource management strategy has been defined and a basic configuration of users, groups and compartments exists, the tenancy administrator can delegate responsibilities to other users with elevated privileges. You could decide to use a simple policy allowing a group of administrators to manage resources for the entire organization, or you might prefer a more granular approach. For example, you can organize resources in a compartment per team or project and let a compartment

administrator manage them. In addition, you might want to keep network resources and storage resources contained within their own separate compartments, controlled respectively by a network administrator and storage administrator. The policy framework of the Identity and Access Management service offers many different options to organize resources and control access to them. For more information, refer to the chapter Identity and Access Management Overview.

When creating scripts or automation tools to interact directly with the API, make sure that the developers understand the authentication and authorization principles and the strict separation of the enclaves. Basic API reference documentation is available for both enclaves.

To view the API reference, append `/api-reference` to the base URL of the Compute Web UI or Service Web UI. For example:

- Service Web UI base URL: https://adminconsole.myprivatecloud.example.com.

  Service Enclave API reference: https://adminconsole.myprivatecloud.example.com*/api-reference*.
- Compute Web UI base URL: https://console.myprivatecloud.example.com.

  Compute Enclave API reference: https://console.myprivatecloud.example.com*/api-reference*.

Layered Architecture

For the architecture of Oracle Private Cloud Appliance a layered approach is taken. At the foundation are the hardware components, on which the core platform is built. This, in turn, provides a framework for administrative and operational services exposed to different user groups. The layers are integrated but not monolithic: they can be further developed at different rates as long as they maintain compatibility. For instance, supporting a new type of server hardware or extending storage functionality are enhancements that can be applied separately, and without redeploying the entire controller software stack.

Hardware Layer

The hardware layer contains all physical system components and their firmware and operating systems.

- The three management nodes form a cluster that runs the base environment for the controller software.
- The compute nodes provide the processing capacity to host compute instances.
- The storage appliance provides disk space for storage resources used by compute instances. It also provides the storage space required by the appliance internally for its operation.
- The network switches provide the physical connections between all components and the uplinks to the public (data center) network.

Platform Layer

Private Cloud Appliance uses a service-based deployment model. The product is divided into functional areas that run as services; each one within its own container. The platform provides the base for this model. Leveraging the capabilities of Oracle Cloud Native

Environment, the management node cluster orchestrates the deployment of the containerized services, for which it also hosts the image registry.

In addition, the platform offers a number of fundamental services of its own, that are required by all other services: message transport, secrets management, database access, logging, monitoring, and so on. These fundamental services are standardized so that all services deployed on top of the platform can plug into them in the same way, which makes new service integrations easier and faster.

The platform also plays a central role in hardware administration, managing the data exchange between the hardware layer and the services. Information about the hardware layer, and any changes made to it, must be communicated to the services layer to keep the inventory up-to-date. When operations are performed at the service layer, an interface is required to pass down commands to the hardware. For this purpose, the platform has a tightly secured API that is only exposed internally and requires the highest privileges. This API interacts with management interfaces such as the server ILOMs and the storage controllers, as well as the inventory database and container orchestration tools.

For additional information about this layer in the appliance architecture, refer to Platform Layer Overview.

Infrastructure Services Layer

This layer contains all the services deployed on top of the platform. They form two functionally distinct groups: user-level cloud services and administrative services.

Cloud services offer functionality to users of the cloud environment, and are very similar in operation to the corresponding Oracle Cloud Infrastructure services. They constitute the Compute Enclave, and enable the deployment of customer workloads through compute instances and associated resources. Cloud services include the compute and storage services, identity and access management, and networking.

The administrative services are either internal or restricted to administrators of the appliance. These enable the operation of the cloud services and provide support for them. They constitute the Service Enclave. Administrator operations include system initialization, compute node provisioning, capacity expansion, tenancy management, upgrade, and so on. These operations have no externalized equivalent in Oracle Cloud Infrastructure, where Oracle fulfills the role of the infrastructure administrator.

Platform Layer Overview

In the Oracle Private Cloud Appliance architecture, the platform layer is the part that provides a standardized infrastructure for on-premises cloud services. It controls the hardware layer, enables the services layer, and establishes a secure interface that allows those layers to interact in a uniform and centralized manner. Common components and features of the infrastructure services layer are built into the platform, thus simplifying and accelerating the deployment of those microservices.

Fundamental Services

To fulfill its core role of providing the base infrastructure to deploy on-premises cloud services, the platform relies on a set of fundamental internal services of its own. This section describes their function.

Hardware Management

When a system is initialized, low level platform components orchestrate the provisioning of the management node cluster and the compute nodes. During this process, all nodes, including the controllers of the ZFS Storage Appliance, are connected to the required administration and data networks. When additional compute nodes are installed at a later stage, the same provisioning mechanism integrates the new node into the global system configuration. Additional disk trays are also automatically integrated by the storage controllers.

The first step in managing the hardware is to create an inventory of the rack's components. The inventory is a separate database that contains specifications and configuration details for the components installed in the rack. It maintains a history of all components that were ever presented to the system, and is updated continuously with the latest information captured from the active system components.

The services layer and several system components need the hardware inventory details so they can interact with the hardware. For example, a component upgrade or service deployment process needs to send instructions to the hardware layer and receive responses. Similarly, when you create a compute instance, a series of operations needs to be performed at the level of the compute nodes, network components and storage appliance to bring up the instance and its associated network and storage resources.

All the instructions intended for the hardware layer are centralized in a hardware management service, which acts as a gateway to the hardware layer. The hardware management service uses the dedicated and highly secured platform API to execute the required commands on the hardware components: server ILOMs, ZFS storage controllers, and so on. This API runs directly on the management node operating system. It is separated from the container orchestration environment where microservices are deployed.

Service Deployment

Oracle Private Cloud Appliance follows a granular, service-based development model. Functionality is logically divided into separate microservices, which exist across the architectural layers and represent a vertical view of the system. Services have internal as well as externalized functions, and they interact with each other in different layers.

These microservices are deployed in Kubernetes containers. The container runtime environment as well as the registry are hosted on the three-node management cluster. Oracle Cloud Native Environment provides the basis for container orchestration, which includes the automated deployment, configuration and startup of the microservices containers. By design, all microservices consist of multiple instances spread across different Kubernetes nodes and pods. Besides high availability, the Kubernetes design also offers load balancing between the instances of each microservice.

Containerization simplifies service upgrades and functional enhancements. The services are tightly integrated but not monolithic, allowing individual upgrades on condition that compatibility requirements are respected. A new version of a microservice is published to the container registry and automatically propagated to the Kubernetes nodes and pods.

Common Service Components

Some components and operational mechanisms are required by many or all services, so it is more efficient to build those into the platform and allow services to consume them when they are deployed on top of the platform. These common components, add a set of essential

features to each service built on top of the platform, thus simplifying service development and deployment.

- **Message Transport**

  All components and services are connected to a common transport layer. It is a message broker that allows components to send and receive messages written in a standardized format. This message transport service is deployed as a cluster of three instances for high availability and throughput, and uses TLS for authentication and traffic encryption.

- **Secret Service**

  Secrets used programmatically throughout the system, such as login credentials and certificates, are centrally managed by the secret service. All components and services are clients of the secret service: after successful authentication the client receives a token for use with every operation it attempts to execute. Policies defined within the secret service determine which operations a client is authorized to perform. Secrets are not stored in a static way; they have a limited lifespan and are dynamically created and managed.

  During system initialization, the secret service is unsealed and prepared for use. It is deployed as an active/standby cluster on the management nodes, within a container at the platform layer, but outside of the Kubernetes microservices environment. This allows the secret service to offer its functionality to the platform layer at startup, before the microservices are available. All platform components and microservices must establish their trust relationship with the secret service before they are authorized to execute any operations.

- **Logging**

  The platform provides unified logging across the entire system. For this purpose, all services and components integrate with the Fluentd data collector. Fluentd collects data from a pre-configured set of log files and stores it in a central location. Logs are captured from system components, the platform layer and the microservices environment, and made available through the Loki log aggregation system for traceability and analysis.

- **Monitoring**

  For monitoring purposes, the platform relies on Prometheus to collect metric data. Since Prometheus is deployed inside the Kubernetes environment, it has direct access to the microservices metrics. Components outside Kubernetes, such as hardware components and compute instances, provide their metric data to Prometheus through the internal network and the load balancer. The management nodes and Kubernetes itself can communicate directly with Prometheus.

- **Analytics**

  Logging and monitoring data are intended for infrastructure administrators. They can consult the data through the Service Web UI, where a number of built-in queries for health and performance parameters are visualized on a dashboard. Alerts are sent when a key threshold is exceeded, so that appropriate countermeasures can be taken.

- **Database**

  All services and components store data in a common, central database. It is a MySQL cluster database with instances deployed across the three management nodes and running on bare metal. Availability, load balancing, data synchronization and clustering are all controlled by internal components of the MySQL cluster. For

optimum performance, data storage is provided by LUNs on the ZFS Storage Appliance, directly attached to each of the management nodes. Access to the database is strictly controlled by the secret service.

- **Load Balancing**

  The management nodes form a cluster of three active nodes, meaning they are all capable of simultaneously receiving inbound connections. The ingress traffic is controlled by a statically configured load balancer that listens on a floating IP address and distributes traffic across the three management nodes. An instance of the load balancer runs on each of the management nodes.

  In a similar way, all containerized microservices run as multiple pods within the container orchestration environment on the management node cluster. Kubernetes provides the load balancing for the ingress traffic to the microservices.

Physical Resource Allocation

When users deploy compute instances – or virtual machines – these consume physical resources provided by the hardware layer. The hypervisor manages the allocation of those physical resources based on algorithms that enable the best performance possible for a given configuration.

The compute nodes in Private Cloud Appliance have a Non-Uniform Memory Access (NUMA) architecture, meaning each CPU has access not only to its own local memory but also the memory of the other CPUs. Each CPU socket and its associated local memory banks are called a *NUMA node*. Local memory access, within the same NUMA node, always provides higher bandwidth and lower latency.

In general, the memory sharing design of NUMA helps with the scaling of multiprocessor workloads, but it may also adversely affect virtual machine performance if its resources are distributed across multiple NUMA nodes. Hypervisor policies ensure that a virtual machine's CPU and memory reside on the same NUMA node whenever possible. Using specific CPU pinning techniques, each virtual machine is pinned to one or multiple CPU cores so that memory is accessed locally on the NUMA node where the virtual machine is running. Cross-node memory transports are avoided or kept at a minimum, so the users of compute instances benefit from optimal performance across the entire system.

If a virtual machine can fit into a single NUMA node on a hypervisor host then this NUMA node is used for pinning, referred to as *strict pinning*. If a virtual machine cannot fit into single NUMA node on a hypervisor host, but can fit into multiple NUMA nodes, then multiple NUMA nodes are used for pinning, referred to as *loose pinning*.

> **Caution:**
>
> The CPU pinning applied through the hypervisor to optimize CPU and memory allocation to compute instances is not configurable by an appliance administrator, tenancy administrator or instance owner.

Private Cloud Appliance detects the NUMA topology of compute nodes during their provisioning (or any upgrade path from one release to another) and stores this information for use during the deployment of compute instances. The NUMA details of each compute instance are stored in the instance configuration. The NUMA settings are preserved when compute instances are migrated to another host compute node, but they might be overridden

and dynamically adjusted if the target compute node is unable to accommodate that particular configuration.

Compute instances running on a Private Cloud Appliance that is not yet NUMA-aware, can take advantage of the optimization policies as soon as the system has been patched or upgraded. There is no action required from the administrator to align instances with NUMA topology; the existing instance configurations are made compatible as part of the process.

High Availability

Oracle Engineered Systems are built to eliminate single points of failure, allowing the system and hosted workloads to remain operational in case of hardware or software faults, as well as during upgrades and maintenance operations. Private Cloud Appliance has redundancy built into its architecture at every level: hardware, controller software, master database, services, and so on. Features such as backup, automated service requests and optional disaster recovery further enhance the system's serviceability and continuity of service.

Hardware Redundancy

The minimum base rack configuration contains redundant networking, storage and server components to ensure that failure of any single element does not affect overall system availability.

Data connectivity throughout the system is built on redundant pairs of leaf and spine switches. Link aggregation is configured on all interfaces: switch ports, host NICs and uplinks. The leaf switches interconnect the rack components using cross-cabling to redundant network interfaces in each component. Each leaf switch also has a connection to each of the spine switches, which are also interconnected. The spine switches form the backbone of the network and enable traffic external to the rack. Their uplinks to the data center network consist of two cable pairs, which are cross-connected to two redundant ToR (top-of-rack) switches.

The management cluster, which runs the controller software and system-level services, consists of three fully active management nodes. Inbound requests pass through the virtual IP of the management node cluster, and are distributed across the three nodes by a load balancer. If one of the nodes stops responding and fences from the cluster, the load balancer continues to send traffic to the two remaining nodes until the failing node is healthy again and rejoins the cluster.

Storage for the system as well as for the cloud resources in the environment is provided by the internal ZFS Storage Appliance. Its two controllers form an active-active cluster, providing high availability and excellent throughput at the same time. The ZFS pools are built on disks in a mirrored configuration for optimum data protection. This applies to the standard high-capacity disk tray as well as an optional SSD-based high-performance tray.

System Availability

The appliance controller software and services layer are deployed on the three-node management cluster, and take advantage of the high availability that is inherent to the cluster design. The Kubernetes container orchestration environment also uses clustering for both its own controller nodes and the service pods it hosts. Multiple replicas of the microservices are running at any given time. Nodes and pods are

distributed across the management nodes, and Kubernetes ensures that failing pods are replaced with new instances to keep all services running in an active/active setup.

All services and components store data in a common, central database. It is a MySQL cluster database with instances deployed across the three management nodes. Availability, load balancing, data synchronization and clustering are all controlled by internal components of the MySQL cluster.

A significant part of the system-level infrastructure networking is software-defined, just like all the virtual networking at the VCN and instance level. The configuration of virtual switches, routers and gateways is not stored and managed by the switches, but is distributed across several components of the network architecture. The network controller is deployed as a highly available containerized service.

The upgrade framework leverages the hardware redundancy and the clustered designs to provide rolling upgrades for all components. In essence, during the upgrade of one component instance, the remaining instances ensure that there is no downtime. The upgrade is complete when all component instances have been upgraded and returned to normal operation.

Compute Instance Availability

When compute instances go down because of a compute node failure, the system attempts to recover them automatically when the host compute node returns to normal operation. If automatic recovery is not successful, the instances need to be restarted manually. Instances can be restarted in a different fault domain, on a different compute node, on condition that the instances are no longer in an active state on their initial host compute node.

At the level of a compute instance, high availability refers to the automated recovery of an instance in case the underlying infrastructure fails. The state of the compute nodes, hypervisors and compute instances is monitored continually; each compute node is polled with a 5 minute interval. When compute instances go down, the system takes measures to recover them automatically.

If an individual compute instance crashes due to internal issues, the hypervisor attempts to restart the instance on the same compute node.

In the scenario where a compute node goes down because of an unplanned reboot, when the compute node successfully returns to normal operation, instances are restarted on the same host compute node. At the next polling interval, if instances are found that should be running but are in a different state, the start command is issued again. If any instances have crashed and remain in that state, the hypervisor attempts to restart them up to 5 times. Instances that were not running before the compute node became unavailable, remain shut down when the compute node is up and running again.

> **Note:**
>
> When a compute node becomes unavailable, the affected *running* compute instances remain in that configuration state in the Compute Web UI. When the compute node has rebooted and the instance recovery operations are executed, their configuration state changes to "shut down" and eventually back to "running" once the instance is available again.

If a compute node is lost due to a failure, its compute instances are evacuated to other compute nodes. A compute node is considered failing when it has been disconnected from the data network or has been in powered-off state for more than 10 minutes. This 10-minute timeout corresponds with two unsuccessful polling attempts, and is the threshold for placing the compute node in `FAIL` state and its agent in `EVACUATING` state. This condition is required before the *reboot migration* can start.

Reboot migration implies that all compute instances from the failing compute node are stopped and restarted on another compute node. Once this process begins, the scenario of a compute node reboot – where instances are restored on the same host – is no longer possible. When migration is complete, the failing compute node's agent indicates that instances have been evacuated. If the compute node eventually reboots successfully, it must go through a cleanup process that removes all stale instance configurations and associated virtual disks. After cleanup, the compute node can host new compute instances again.

During the entire reboot migration, the instances remain in "moving" configuration state. Once migration is completed, the instance configuration state is changed to "running". Instances that were stopped before the failure are not migrated, since they are not associated with any compute node.

Fault domain preference is strictly enforced with instance migration. If a compute instance cannot be migrated to another compute node in the same fault domain due to insufficient capacity, the instance is stopped. Other fault domains are not considered. The instance must be restarted manually in another fault domain, or in its current fault domain when sufficient resources have been made available again.

In case of planned maintenance, the administrator must first disable provisioning for the compute node in question, and apply a maintenance lock. When the compute node is under a provisioning lock, the administrator can live-migrate all running compute instances to another compute node. If no target compute node is available in the same fault domain, instances may be migrated to another fault domain. If live migration fails, the administrator must manually shut down the instance and restart it on a different compute node. Maintenance mode can only be activated when there are no more running instances on the compute node. All compute instance operations on this compute node are disabled. A compute node in maintenance mode cannot be provisioned or deprovisioned.

Continuity of Service

Private Cloud Appliance offers several features that support and further enhance high availability. Health monitoring at all levels of the system is a key factor. Diagnostic and performance data is collected from all components, then centrally stored and processed, and made available to administrators in the form of visualizations on standard dashboards. In addition, alerts are generated when metrics exceed their defined thresholds.

Monitoring allows an administrator to track system health over time, take preventative measures when required, and respond to issues when they occur. In addition, systems registered with My Oracle Support provide phone-home capabilities, using the collected diagnostic data for fault monitoring and targeted proactive support. Registered systems can also submit automated service requests with Oracle for specific problems reported by the appliance.

To mitigate data loss and support the recovery of system and services configuration in case of failure, consistent and complete backups are made regularly. A backup can

also be executed manually, for example to create a restore point just before a critical modification. The backups are stored in a dedicated NFS share on the ZFS Storage Appliance, and allow the entire Service Enclave to be restored when necessary.

Optionally, workloads deployed on the appliance can be protected against downtime and data loss through the implementation of disaster recovery. To achieve this, two Private Cloud Appliance systems need to be set up at different sites, and configured to be each other's replica. Resources under disaster recovery control are stored separately on the ZFS Storage Appliances in each system, and replicated between the two. When an incident occurs at one site, the environment is brought up on the replica system with practically no downtime. Oracle recommends that disaster recovery is implemented for all critical production systems.

2
Hardware Overview

This chapter provides an overview of the hardware components that the Oracle Private Cloud Appliance comprises: a base rack with servers, switches and storage. Different sections describe the role of each component. Special attention is given to the appliance network infrastructure, and the way it integrates with the data center environment and, optionally, an Oracle Exadata system.

Base Rack Components

The current Private Cloud Appliance hardware platform, with factory-installed software release 3.0.1, consists of an Oracle Rack Cabinet 1242 base, populated with the hardware components identified in the figure below. This figure shows a full configuration, however you can customize your system to include different storage or compute capacity as needed.

Starting with release 3.0.1, the flex bay concept is available on Private Cloud Appliance. Flex bays are dedicated 4 rack unit sections within the rack that can be used for flexible expansion of your system; adding either storage or compute resources. For each flex bay, you can choose to add 1-4 compute nodes, 1 Oracle Storage Drive Enclosure DE3-24C, or 1-2 Oracle Storage Drive Enclosure DE3-24P. A flex bay can house either storage or compute resources, but not both in the same bay.

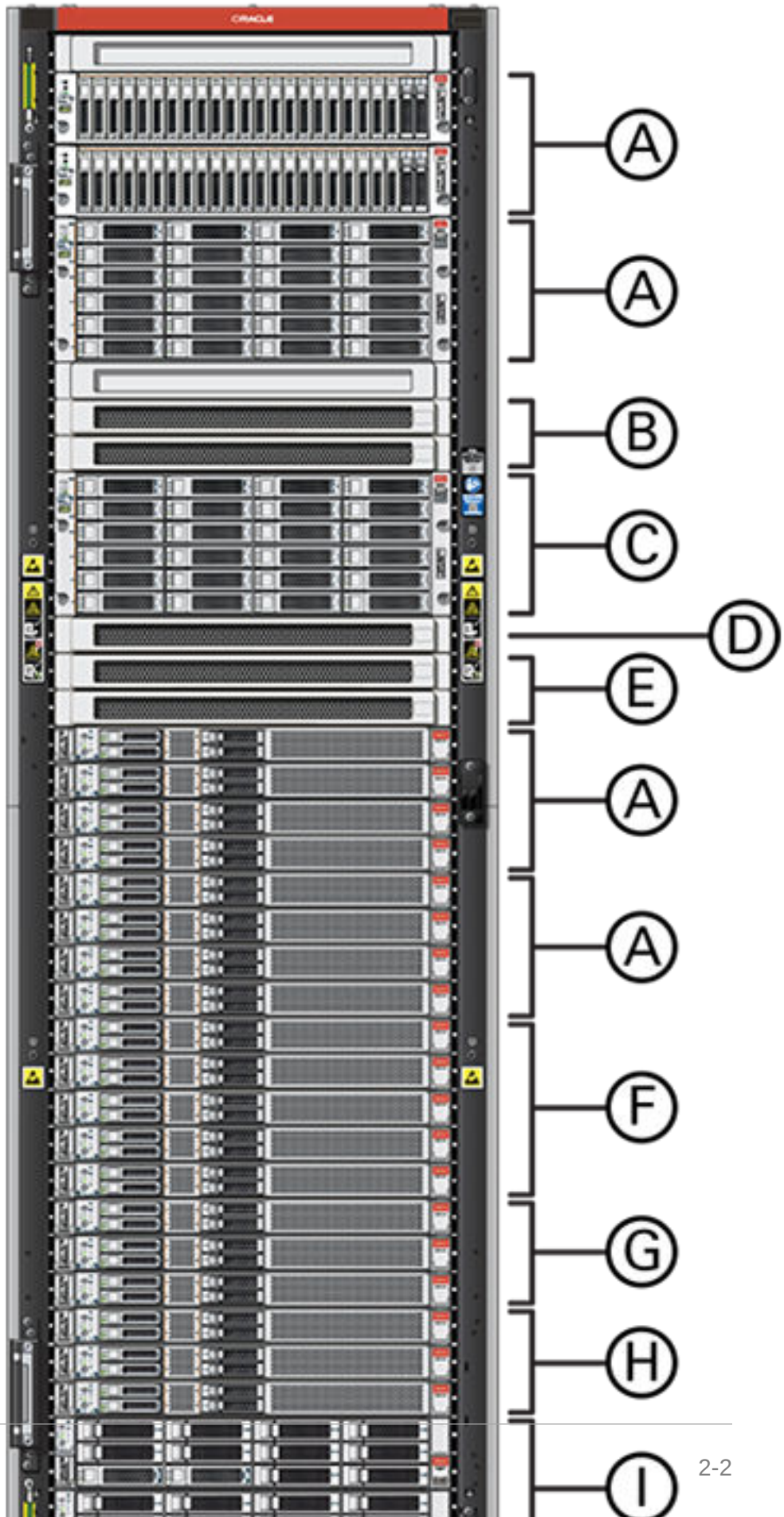

A
A
B
C
D
E
A
A
F
G
H
I

Figure Legend

Callout	Quantity	Description
A	1 - 4	flex bay can accommodate 1-4 compute nodes, or 1-2 storage enclosures
B	2	leaf switch
C	1	Oracle Storage Drive Enclosure DE3-24C disk shelf
D	1	management switch
E	2	spine switch
F	1 - 5	can accommodate 1 - 5 compute nodes
G	3	compute nodes 3 required for minimum configuration
H	3	management nodes
I	2	storage controllers

Servers

These sections describe the management and compute nodes employed by Private Cloud Appliance.

Management Nodes

At the heart of each Private Cloud Appliance installation are three management nodes. They are installed in rack units 5, 6 and 7 and form a cluster for high availability: all servers are capable of running the same controller software and system-level services, have equal access to the system configuration, and all three servers manage the system as a fully active cluster. For details about the management node components, refer to Server Components.

The management nodes, running the controller software, provide a foundation for the collection of services responsible for operating and administering Private Cloud Appliance. Responsibilities of the management cluster include monitoring and maintaining the system hardware, ensuring system availability, upgrading software and firmware, backing up and restoring the appliance, and managing disaster recovery. For an overview of mangement node services, refer to Appliance Administration Overview. See the Oracle Private Cloud Appliance Administrator Guide for instructions about performing management functions.

The part of the system where the appliance infrastructure is controlled is called the Service Enclave, which runs on the management node cluster and can be accessed through the Service CLI or the Service Web UI. Access is closely monitored and restricted to privileged administrators. For more information, see Administrator Access. Also refer to the chapter Working in the Service Enclave in the Oracle Private Cloud Appliance Administrator Guide.

Compute Nodes

The compute nodes in the Private Cloud Appliance are part of the hardware layer and provide the processing power and memory capacity to host compute instances. Management of the hardware layer is provided by the platform and services layer of the appliance. For more details about the Layered Architecture approach, see Architecture and Design.

When a system is initialized, compute nodes are automatically discovered by the admin service and put in the Ready-to-Provision state. Administrators can then provision the compute nodes through the Service Enclave, and they are ready for use. When additional compute nodes are installed at a later stage, the new nodes are discovered, powered on, and discovered automatically by the same mechanism.

As an administrator, you can monitor the health and status of compute nodes from the Private Cloud Appliance Service Web UI or Service CLI, as well as performing other operations such as assigning compute nodes to tenancies and upgrading compute node components. For general administration information see Appliance Administration Overview . For more information about managing compute resources, see Compute Instance Concepts.

The minimum configuration of the base rack contains three compute nodes, but it can be expanded by three nodes at a time up to 18 compute nodes, if you choose to use all of your flex bay space for compute nodes. The system can support up to 20 compute nodes total, with two slots reserved for spares that you can request through an exception process. Contact your Oracle representative about expanding the compute node capacity in your system. For hardware configuration details of the compute nodes, refer to Server Components.

Server Components

The table below lists the components of the server models that may be installed in a Private Cloud Appliance rack and which are supported by the current software release.

Quantity	Oracle Server X9-2 Management Node	Oracle Server X9-2 Compute Node
1	Oracle Server X9-2 base chassis	Oracle Server X9-2 base chassis
2	Intel Xeon Gold 5318Y CPU, 24 core, 2.10GHz, 165W	Intel Xeon Platinum 8358 CPU, 32 core, 2.60GHz, 250W
16	16x 64 GB DDR4-3200 DIMMs (1TB total)	16x 64 GB DDR4-3200 DIMMs (1TB total)
2	240GB M.2 SATA boot devices configured as RAID1 mirror	240GB M.2 SATA boot devices configured as RAID1 mirror
2	3.84 TB NVMe SSD storage devices configured as RAID1 mirror	
1	Ethernet port for remote management	Ethernet port for remote management
1	Dual-port 100Gbit Ethernet NIC module in OCPv3 form	Dual-port 100Gbit Ethernet NIC module in OCPv3 form
2	Redundant power supplies and fans	Redundant power supplies and fans

Network Infrastructure

For network connectivity, Private Cloud Appliance relies on a physical layer that provides the necessary high-availability, bandwidth and speed. On top of this, a distributed network fabric composed of software-defined switches, routers, gateways and tunnels enables secure and segregated data traffic – both internally between cloud resources, and externally to and from resources outside the appliance.

Device Management Network

The device management network provides internal access to the management interfaces of all appliance components. These have Ethernet connections to the 1Gbit management switch, and all receive an IP address from each of these address ranges:

- 100.96.0.0/23 – IP range for the Oracle Integrated Lights Out Manager (ILOM) service processors of all hardware components
- 100.96.2.0/23 – IP range for the management interfaces of all hardware components

To access the device management network, you connect a workstation to port 2 of the 1Gbit management switch and statically assign the IP address 100.96.3.254 to its connected interface. Alternatively, you can set up a permanent connection to the Private Cloud Appliance device management network from a data center administration machine, which is also referred to as a *bastion host*. From the bastion host, or from the (temporarily) connected workstation, you can reach the ILOMs and management interfaces of all connected rack components. For information about configuring the bastion host, see the "Optional Bastion Host Uplink" section of the Oracle Private Cloud Appliance Installation Guide.

Note that port 1 of the 1Gbit management switch is reserved for use by support personnel only.

Data Network

The appliance data connectivity is built on redundant 100Gbit switches in two-layer design similar to a leaf-spine topology. The leaf switches interconnect the rack hardware components, while the spine switches form the backbone of the network and provide a path for external traffic. Each leaf switch is connected to all the spine switches, which are also interconnected. The main benefits of this topology are extensibility and path optimization. A Private Cloud Appliance rack contains two leaf and two spine switches.

The data switches offer a maximum throughput of 100Gbit per port. The spine switches use 5 interlinks (500Gbit); the leaf switches use 2 interlinks (200Gbit) and 2x2 crosslinks to each spine. Each server node is connected to both leaf switches in the rack, through the `bond0` interface that consists of two 100Gbit Ethernet ports in link aggregation mode. The two storage controllers are connected to the spine switches using 4x100Gbit connections.

For external connectivity, 5 ports are reserved on each spine switch. Four ports are available to establish the uplinks between the appliance and the data center network; one port is reserved to optionally segregate the administration network from the data traffic.

Uplinks

The connections between the Private Cloud Appliance and the customer data center are called *uplinks*. They are physical cable connections between the two spine switches in the appliance rack and one or – preferably – two next-level network devices in the data center infrastructure. Besides the physical aspect, there is also a logical aspect to the uplinks: how traffic is routed between the appliance and the external network it is connected to.

Physical Connection

On each spine switch, ports 1-4 can be used for uplinks to the data center network. For speeds of 10Gbps or 25Gbps, the spine switch port must be split using an MPO-to-4xLC breakout cable. For speeds of 40Gbps or 100Gbps each switch port uses a single MPO-to-

MPO cable connection. The correct connection speed must be specified during initial setup so that the switch ports are configured with the appropriate breakout mode and transfer speed.

The uplinks are configured during system initialization, based on information you provide as part of the installation checklist. Unused spine switch uplink ports, including unused breakout ports, are disabled for security reasons. The table shows the supported uplink configurations by port count and speed, and the resulting total bandwidth.

Uplink Speed	Number of Uplinks per Spine Switch	Total Bandwidth
10 Gbps	1, 2, 4, 8, or 16	20, 40, 80, 160, or 320 Gbps
25 Gbps	1, 2, 4, 8, or 16	50, 100, 200, 400, or 800 Gbps
40 Gbps	1, 2, or 4	80, 160, or 320 Gbps
100 Gbps	1, 2, or 4	200, 400, or 800 Gbps

Regardless of the number of ports and port speeds configured, you also select a topology for the uplinks between the spine switches and the data center network. This information is critical for the network administrator to configure link aggregation (port channels) on the data center switches. The table shows the available options.

Topology	Description
Triangle	In a triangle topology, all cables from both spine switches are connected to a single data center switch.
Square	In a square topology, two data center switches are used. All outbound cables from a given spine switch are connected to the same data center switch.
Mesh	In a mesh topology, two data center switches are used as well. The difference with the square topology is that uplinks are created in a cross pattern. Outbound cables from each spine switch are connected in pairs: one cable to each data center switch.

Reference Topologies

The physical topology for the uplinks from the appliance to the data center network is dependent on bandwidth requirements and available data center switches and ports. Connecting to a single data center switch implies that you select a triangle topology. To increase redundancy you distribute the uplinks across a pair of data center switches, selecting either a square or mesh topology. Each topology allows you to start with a minimum bandwidth, which you can scale up with increasing need. The maximum bandwidth is 800 Gbps, assuming the data center switches, transceivers and cables allow it.

The diagrams below represent a subset of the supported topologies, and can be used as a reference to integrate the appliance into the data center network. Use the diagrams and the notes to determine the appropriate cabling and switch configuration for your installation.

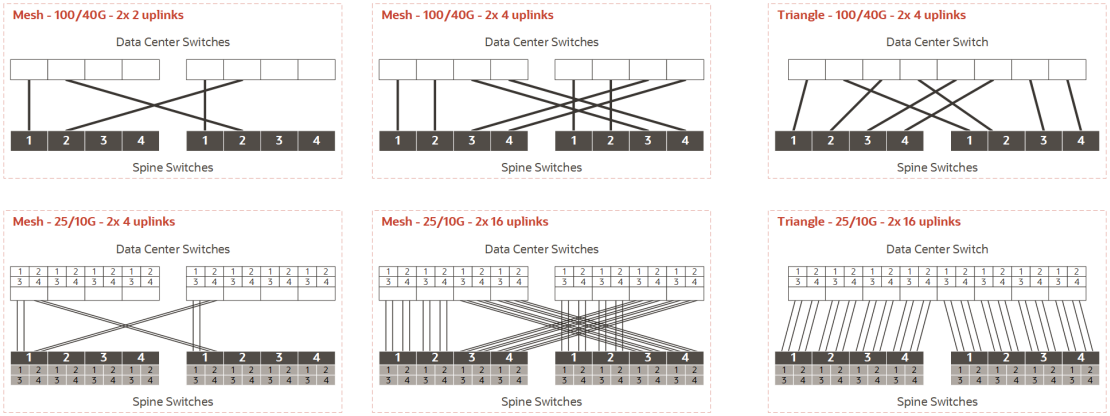


Diagram Notes

On the appliance side there are two spine switches that must be connected to the data center network. Both spine switches must have identical port and cable configurations. In each example, the spine switches are shown at the bottom, with all uplink ports identified by their port number. The lines represent outgoing cable connection to the data center switches, which are shown at the top of each example without port numbers.

Cabling Pattern and Port Speed

There are six examples in total, organized in two rows by three columns.

- The top row shows cabling options based on full-port 100Gbps or 40Gbps connections. The bottom row shows cabling options using breakout ports at 25Gbps or 10Gbps speeds; the smaller boxes numbered 1-4 represent the breakout connections for each of the four main uplink ports per spine switch.
- The third column shows a triangle topology with full-port connections and breakout connections. The difference with column two is that all uplinks are connected to a single data center switch. The total bandwidth is the same, but the triangle topology lacks data center switch redundancy.
- There are no diagrams for the square topology. The square cabling configuration is like the mesh examples, but without the crossing patterns. Visually, all connectors in the diagrams would be parallel. In a square topology, all outgoing cables from one spine switch are connected port for port to the same data center switch. Unlike mesh, square implies that each spine switch is peered with only one data center switch.

Link Count

When connecting the uplinks, you are required to follow the spine switch port numbering. Remember that both spine switches are cabled identically, so each uplink or connection corresponds with a pair of cables.

- With one cable per spine port, using 100 or 40 Gbps transceivers, the first uplink pair uses spine switch ports numbered "1", the second uses port 2, and so on. In this configuration, the maximum number of uplinks is four per spine switch.
- When breakout cables are used, with 25 or 10Gbps port speeds, the first uplink pair uses port 1/1. With two or four uplinks per spine switch there is still only one full port in use. When you increase the uplink count to 8 per spine switch, ports 1/1-2/4 are in use. At 16 uplinks per spine switch, all breakout connections of all four reserved ports will be in use.

- In a mesh topology, a particular cabling pattern must be followed: connect the first half of all uplinks to one data center switch, and the second half to the other data center switch. For example: if you have four uplinks then the first two go to the same switch; if you have eight uplinks (not shown in the diagrams) then the first four go to the same switch; if you have 16 uplinks then the first eight go to the same switch.

Mesh Topology Implications

- In a mesh topology, the spine switch configuration expects that the first half of all uplinks is connected to one data center switch, and the second half to the other data center switch. When you initially connect the appliance to your data center network, it is straightforward to follow this pattern.
- However, if you increase the number of uplinks at a later time, the mesh cabling pattern has a significant implication for the existing uplinks. Compare the diagrams in the first two columns: when you double the uplink count, half of the existing connections must be moved to the other data center switch. For 100/40Gbit uplinks, re-cabling is only required when you increase the link count from 2 to 4. Due to the larger number of cables, 25/10Gbit uplinks require more re-cabling: when increasing uplink count from 2 to 4, from 4 to 8, and from 8 to 16.

Logical Connection

The logical connection between the appliance and the data center is implemented entirely in layer 3. In the OSI model (Open Systems Interconnection model), layer 3 is known as the network layer, which uses the source and destination IP address fields in its header to route traffic between connected devices.

Private Cloud Appliance supports two logical connection options: you must choose between static routing and dynamic routing. Both routing options are supported by all three physical topologies.

Connection Type	Description
Static Routing	When static routing is selected, all egress traffic goes through a single default gateway IP address configured on data center network devices. This gateway IP address must be in the same subnet as the appliance uplink IP addresses, so it is reachable from the spine switches. The data center network devices can use SVIs (Switch Virtual Interfaces) with VLAN IDs in the range of 2-3899. All gateways configured within a virtual cloud network (VCN) will automatically have a route rule to direct all traffic intended for external destination to the IP address of the default gateway.

Connection Type	Description
Dynamic Routing	When dynamic routing is selected, BGP (Border Gateway Protocol) is used to establish a TCP connection between two Autonomous Systems: the appliance network and the data center network. This configuration requires a registered or private ASN (Autonomous System Number) on each side of the connection. Private Cloud Appliance BGP configuration uses ASN 136025 by default, this can be changed during initial configuration. For BGP routing, two routing devices in the data center must be connected to the two spine switches in the appliance rack. Corresponding interfaces (port channels) between the spine switches and the data center network devices must be in the same subnet. It is considered good practice to use a dedicated /30 subnet for each point-to-point circuit, which is also known as a route hand-off network. This setup provides redundancy and multipathing. Dynamic routing is also supported in a triangle topology, where both spine switches are physically connected to the same data center network device. In this configuration, two BGP sessions are still established: one from each spine switch. However, this approach reduces the level of redundancy.

Supported Routing Designs

The table below shows which routing designs are supported depending on the physical topology in your data center and the logical connection you choose to implement.

Note that link aggregation across multiple devices (vPC or MLAG) is only supported with static routing. When dynamic routing is selected, link aggregation is restricted to ports of the same switch.

When the uplinks are cabled in a mesh topology, a minimum of 2 physical connections per spine switch applies. To establish BGP peering, 2 subnets are required. If the uplink count changes, the port channels are reconfigured but the dedicated subnets remain the same.

Logical Connection	Physical Topology	Routing Design		
		Single Subnet	Dual Subnet	vPC/MLAG
Static Routing	Square	Yes	Yes	Yes
	Mesh	Yes	Yes	Yes
	Triangle	Yes	Yes	Yes
Dynamic Routing	Square	Yes	–	–
	Mesh	–	Yes	–
	Triangle	Yes	–	–

Uplink Protocols

The uplinks to the data center run a variety of protocols to provide redundancy and reduce link failure detection and recovery times on these links. These protocols work with the triangle, square, or mesh topologies.

The suite of uplink protocols include:

- Bidirectional Forwarding Detection (BFD)
- Virtual Router Redundancy Protocol (VRRP)
- Hot Spare Router Protocol (HSRP)
- Equal Cost Multi-Path (ECMP)

Each is briefly described in the following sections of this topic.

BFD

In most router networks, connection failures are detected by loss of the “hello” packets sent by routing protocols. However, detection by this method often takes more than one second, routing a lot of packets on high-speed links to a destination that they cannot reach, which burdens link buffers. Increasing the “hello” packet rate burdens the router CPU.

Bidirectional Forwarding Detection (BFD) is a built-in mechanism that alerts routers at the end of a failed link that there is a problem more quickly than any other mechanism, reducing the load on buffers and CPUs. BFD works even in situations where there are switches or hubs between the routers.

BFD requires no configuration and has no user-settable parameters.

VRRPv3

The Virtual Router Redundancy Protocol version 3 (VRRPv3) is a networking protocol that uses the concept of a virtual router to group physical routers together and make them appear as one to participating hosts. This increases the availability and reliability of routing paths through automatic default gateway selections on an IP subnetwork.

With VRRPv3, the primary/active and secondary/standby routers act as one virtual router. This virtual router becomes the default gateway for any host on the subnet participating in VRRPv3. One physical router in the group becomes the primary/active touter for packet forwarding. However, if this router fails, another physical router in the group takes over the forwarding role, adding redundancy to the router configuration. The VRRPv3 “network” is limited to the local subnet and does not advertise routes beyond the local subnet.

HSRP

Cisco routers often use a redundancy protocol called the Hot Spare Router Protocol (HSRP) to improve router availability. Similar to the methods of VRRP, HSRP groups physical routers into a single virtual router. The failure of a physical default router results in another router using HSRP to take over the default forwarding of packets without stressing the host device.

ECMP

Equal Cost Multi-Path (ECMP) is a way to make better use of network bandwidth, especially in more complex router networks with many redundant links.

Normally, router networks with multiple router paths to another destination network choose one active route to a gateway router as the “best” path and use the other paths as a standby in case of failure. The decision about which path to a network gateway router to use is usually determined by its “cost” from the routing protocol perspective. In cases where the cost over several links to reach network gateways are equal, the router simply chooses one based on some criteria. This makes routing decisions easy but wastes network bandwidth as network links on paths not chosen sit idle.

ECMP is a way to send traffic on multiple path links with equal cost, making more efficient use of network bandwidth.

Administration Network

In an environment with elevated security requirements, you can optionally segregate administrative appliance access from the data traffic. The administration network physically separates configuration and management traffic from the operational activity on the data network by providing dedicated secured network paths for appliance administration operations. In this configuration, the entire Service Enclave can be accessed only over the administration network. This also includes the monitoring, metrics collection and alerting services, the API service, and all component management interfaces.

Setting up the administration network requires additional Ethernet connections from the next-level data center network devices to port 5 on each of the spine switches in the appliance. Inside the administration network, the spine switches must each have one IP address and a virtual IP shared between the two. A default gateway is required to route traffic, and NTP and DNS services must be enabled. The management nodes must be assigned host names and IP addresses in the administration network – one each individually and one shared between all three.

A separate administration network can be used with both static and dynamic routing. The use of a VLAN is supported, but when combined with static routing the VLAN ID must be different from the one configured for the data network.

Reserved Network Resources

The network infrastructure and system components of Private Cloud Appliance need a large number of IP addresses and several VLANs for internal operation. It is critical to avoid conflicts with the addresses in use in the customer data center as well as the CIDR ranges configured in the virtual cloud networks (VCNs).

These IP address ranges are reserved for internal use by Private Cloud Appliance:

Reserved IP Addresses	Description
CIDR blocks in Shared Address Space	The Shared Address Space, with IP range 100.64.0.0/10, was implemented to connect customer-premises equipment to the core routers of Internet service providers. To allocate IP addresses to the management interfaces and ILOMs (Oracle Integrated Lights Out Manager) of hardware components, two CIDR blocks are reserved for internal use: 100.96.0.0/23 and 100.96.2.0/23.

Reserved IP Addresses	Description
CIDR blocks in Class E address range	Under the classful network addressing architecture, Class E is the part of the 32-bit IPv4 address space ranging from 240.0.0.0 to 255.255.255.255. At the time, it was reserved for future use, so it cannot be used on the public Internet. To accommodate the addressing requirements of all infrastructure networking over the physical 100Gbit connections, the entire 253.255.0.0/16 subnet is reserved. It is further subdivided into multiple CIDR blocks in order to group IP addresses by network function or type. The various CIDR blocks within the 253.255.0.0/16 range are used to allocate IP addresses for the Kubernetes containers running the microservices, the virtual switches, routers and gateways enabling the VCN data network, the hypervisors, the appliance chassis components, and so on.
Link Local CIDR block	A link-local address belongs to the 169.254.0.0/16 IP range, and is valid only for connectivity within a host's network segment, because the address is not guaranteed to be unique outside that network segment. Packets with link-local source or destination addresses are not forwarded by routers. The link-local CIDR block 169.254.239.0/24, as well as the IP address 169.254.169.254, are reserved for functions such as DNS requests, compute instance metadata transfer, and cloud service endpoints.

All VCN traffic – from one VCN to another, as well as between a VCN and external resources – flows across the 100Gbit connections and is carried by VLAN 3900. Traffic related to server management is carried by VLAN 3901. All VLANs with higher IDs are also reserved for internal use, and VLAN 1 is the default for untagged traffic. The remaining VLAN range of 2-3899 is available for customer use.

Oracle Exadata Integration

Optionally, Private Cloud Appliance can be integrated with Oracle Exadata for a high-performance combination of compute capacity and database optimization. In this configuration, database nodes are directly connected to reserved ports on the spine switches of the Private Cloud Appliance. Four 100Gbit ports per spine switch are reserved and split into 4x25Gbit breakout ports, providing a maximum of 32 total cable connections. Each database node is cabled directly to both spine switches, meaning up to 16 database nodes can be connected to the appliance. It is allowed to connect database nodes from different Oracle Exadata racks.

Once the cable connections are in place, the administrator configures an *Exadata network*, which enables traffic between the connected database nodes and a set of compute instances. These prerequisites apply:

- The Exadata network must not overlap with any subnets in the on-premises network.
- The VCNs containing compute instances that connect to the database nodes, must have a dynamic routing gateway (DRG) configured.
- The relevant subnet route tables must contain rules to allow traffic to and from the Exadata network.

The Exadata network configuration determines which Exadata clusters are exposed and which subnets have access to those clusters. Access can be enabled or disabled per Exadata cluster and per compute subnet. In addition, the Exadata network can be exposed through the appliance's external network, allowing other resources within the on-premises network to connect to the database nodes through the spine switches of the appliance. The Exadata network configuration is created and managed through the Service CLI.

Storage

Private Cloud Appliance includes an Oracle ZFS Storage Appliance ZS9-2 as the base storage option, with the capability to expand storage within the rack as needed.

Oracle ZFS Storage Appliance ZS9-2

The Oracle ZFS Storage Appliance ZS9-2, which consists of two controller servers installed at the bottom of the appliance rack and disk shelf about halfway up, fulfills the role of 'system disk' for the entire appliance. It is crucial in providing storage space for the Private Cloud Appliance software.

The default disk shelf in the appliance has 100+ TB of customer usable storage for public object storage, customer compute images, and customer block storage.

The hardware configuration of the Oracle ZFS Storage Appliance ZS9-2 is as follows:

- Two clustered storage controller heads
- One fully populated disk chassis with twenty 18TB hard disks
- Four cache disks installed in the disk shelf: 2x 200GB SSD and 2x 7.68TB SSD
- Mirrored configuration, for optimum data protection

The storage appliance is connected to the management subnet and the storage subnet. Both heads form a cluster in active-active configuration to guarantee continuation of service in the event that one storage head should fail. The storage heads provide two IPs in the storage subnet: one for the default capacity storage pool, the other to access the optional performance (SSD) storage pool. Four management IP addresses are provided: one local to each controller head and one for each storage pool that follows the pool resources between controllers over takeover or failback events for convenient maintenance access. The primary mirrored capacity storage pool contains two projects, named `PCA` and `private_ostore_project`.

Additional Storage

You can optionally increase the storage in your system by adding disk shelves to a system flex bay. The available storage options for expansion are the Oracle Storage Drive Enclosure DE3-24C and the Oracle Storage Drive Enclosure DE3-24P.

The supported hardware configuration of the Oracle Storage Drive Enclosure DE3-24C is as follows:

- Fully populated disk chassis with twenty 18TB hard disks
- Four cache disks installed in the disk shelf: 2x 200GB SSD and 2x 7.68TB SSD

The supported hardware configuration of the Oracle Storage Drive Enclosure DE3-24P is as follows:

- Twenty 7.68TB SSD
- Two cache disks installed in the disk shelf: 2x 200GB SSD
- Two drive bay fillers

Storage devices included in additional Oracle Storage Drive Enclosure DE3-24C enclosures are automatically added to the primary capacity storage pool, while devices included in Oracle Storage Drive Enclosure DE3-24P enclosures are automatically added to the optional high performance storage pool, once the enclosure is installed and cabled to the storage controller.

3
Appliance Administration Overview

The appliance infrastructure is controlled from a system area that is securely isolated from the workspace where cloud resources are created and managed. This administration area is called the *Service Enclave*, and is available only to privileged administrators. This chapter describes how administrators access the Service Enclave and which features and functions are available to configure the appliance and keep it in optimum operating condition.

Administrator Access

An appliance administrator is a highly privileged user with access to the physical components of Oracle Private Cloud Appliance. There is no functional relationship between an appliance administrator account and a tenancy administrator account; these are entirely separate entities. While appliance administrators may be authorized to create and delete tenancies, their account does not grant any permission to access a tenancy or use its resources. An appliance administrator has no access whatsoever to user data or instances.

Access to the administrative functionality is provided through separate interfaces: a Service Web UI, a Service CLI and a Service API, which are all highly restricted. The resources and functions an administrator can access is controlled by authorization groups, which are configured for access control using policies. For more information, see Administrator Account Management in the Oracle Private Cloud Appliance Administrator Guide.

Appliance administrator accounts can be created locally, but Private Cloud Appliance also supports federating with an existing identity provider, so people can log in with their existing id and password. User groups from the identity provider must be mapped to the appliance administrator groups, to ensure that administrator roles are assigned correctly to each authorized account.

A single federated identity provider is supported for appliance administrator accounts. The process of establishing a federation trust with the identity provider is the same as for identity federation at the tenancy level. This is described in the chapter Identity and Access Management Overview. Refer to the section Federating with Identity Providers.

Authorization Groups and Their Policies

As an administrator, the specific functions you can perform is dependent on the *authorization group* to which you belong. Authorization groups have access control polices attached to them which define the resources and functions to which you have access. When you write your access policies (policy statements), you can define resources and functions individually, or you can use authorization families.

There are three default authorization groups for Oracle Private Cloud Appliance:

- SuperAdmin

  Users have unrestricted access to the Service Enclave. They are authorized to perform all available operations, including the setup of other administrator accounts and management of authorization groups and families.

- Initial

Users have limited access to the Service Enclave. They are authorized to create the initial administrator account and view information about the appliance, but do not have read access to any other resources.

- Day0

  Users have specific access to operations related to the initial setup of the appliance – a process also referred to as the "day zero configuration."

When you are configuring additional administrative access, you can use either a default authorization group or create a new authorization group. Every authorization group must have at least one policy statement that allows users who belong to this group access to resources. An authorization group without a policy statement is valid, but its users would not have access to any resources.

If you are upgrading your appliance from a previous release, you might have users in legacy authorization groups. These legacy groups still exist after an upgrade. For continuity, the upgrade process creates the necessary authorization families and policies to ensure the users in a legacy group retain the same level of access.

The following table lists the legacy authorization groups and the access privileges associated with each.

Legacy Authorization Group	Description
Admin	The *Admin* role grants permission to list, create, modify and delete practically all supported object types. Permissions excluded from this role are: administrator account and authorization group management, and disaster recovery operations.
Monitor	Administrators with a *Monitor* role are authorized to execute read-only commands. For example, using the `get` API calls, they can list and filter for objects of a certain type. Some objects related to specific features, such as the disaster recovery items, are excluded because they require additional privileges.
DR Admin	The *DrAdmin* role grants the same permissions as the Admin role, with the addition of all operations related to disaster recovery.
Day Zero Config	The *Day0Config* role only provides specific access to operations related to the initial setup of the appliance – a process also referred to as the "day zero configuration". The state of the system determines which operations an administrator with this role is allowed to perform. For example, when the system is ready for the primary administrator account to be created, only that specific command is available. Then, when the system is ready to register system initialization data, only the commands to set those parameters are available.
Internal	This role is reserved for internal system use.

Authorization groups must have associated access control polices. You can create individual policies for an authorization group or you can use authorization families. Using an authorization family allows you to create policies that you can reuse across authorization groups. The default authorization groups use predefined policies, which are created using authorization families.

You can create policy statements from the Service Web UI or Service CLI. Each policy statement must contain the following:

- Name - 1 to 255 characters
- Action - Inspect, Read, Use, or Manage
- Resource / Authorization Family - One or more resources or one authorization family
- (**Service CLI only**) Authorization Group - the ID of the group

The following table contains information about the actions you can take on a resource.

Action	Type of Access
`inspect`	Ability to list resources, without access to any confidential information or user-specified metadata that may be part of that resource.
`read`	Includes `inspect` plus the ability to get user-specified metadata and the actual resource itself.
`use`	Includes `read` plus the ability to work with existing resources. The actions vary by resource type.
`manage`	Includes all permissions for the resource.

To learn how to create authorization groups and policies, see the "Managing Administrator Privileges" topic in the Administrator Account Management section of the Oracle Private Cloud Appliance Administrator Guide.

Authorization Families

Authorization groups must have at least one associated access control policy, known as a policy statement. Each policy statement provides a type of action for one or more resources. You can list individual resources in your statements or use *authorization families*.

Authorization families allow you to group resources and functions that make logical sense in the management of your appliance. There are two types of authorization families you can use in policy statements:

- *Resource families* are used to define appliance resources, such as servers, storage, and network infrastructure.
- *Function families* are used to define appliance functions, such as compartment, user, and compute management.

The default authorization groups use predefined resource and function families in their policy statements. The following table lists these predefined authorization families and how they are used in the default authorization groups' policies.

Authorization Family Name	Authorization Family Type	Used In Policies For...	Users In Group Can...
Day0	Function Family	SuperAdmin authorization group	• set Day0 system, static routing, dynamic routing, and network parameters • get management node, compute node, and ZFS health from ILOM • unlock and lock the appliance
Initial	Function Family	Initial authorization group	create the initial admin account
SuperAdmin	Function Family	SuperAdmin authorization group	manage all appliance functions
Day0	Resource Family	SuperAdmin authorization group	read system information and networking configuration
Initial	Resource Family	Initial authorization group	read system information
SuperAdmin	Resource Family	SuperAdmin authorization group	manage all resources on appliance

To learn how to create authorization families, see the "Managing Administrator Privileges" topic in the Administrator Account Management section of the Oracle Private Cloud Appliance Administrator Guide.

Status and Health Monitoring

The overall health status of the system is continually monitored, using real-time data from the hardware and platform layers. System health checks and monitoring data are the foundation of problem detection. When an unhealthy condition is found, administrators use this information to begin troubleshooting. If necessary, they register a service request with Oracle for assistance in resolving the problem. If the Private Cloud Appliance is registered for Oracle Auto Service Request (ASR), certain hardware failures cause a service request and diagnostic data to be automatically sent to Oracle support.

Monitoring

Independently of the built-in health checks, an administrator can consult the monitoring data at any time to verify the overall status of the system or the condition of a particular component or service. This is done through the Grafana interface, by querying the system-wide metric data stored in Prometheus.

Grafana provides a visual approach to monitoring: it allows you to create dashboards composed of a number of visualization panels. Each panel corresponds with a single

metric query or a combination of metric queries, displayed in the selected format. Options include graphs, tables, charts, diagrams, gauges, and so on. For each metric panel, thresholds can be defined. When the query result exceeds or drops below a given threshold, the display color changes, providing a quick indication of which elements are healthy, require investigation, or are malfunctioning.

Oracle provides a set of pre-defined dashboards that allow administrators to start monitoring the system as soon as it is up and running. The default monitoring dashboards are grouped into the following categories:

Monitoring Dashboard Set	Description
Service Advisor	Appliance-specific collection of dashboards for monitoring the Kubernetes container orchestration environment, the containerized services it hosts, and the system health check services.
Service Level Monitoring	A read-only collection of dashboards that provide statistic data for all the microservices.
Kubernetes Monitoring	An additional collection of dashboards provided by Oracle's cloud native monitoring and visualization experts. These provide vast and detailed information about the Kubernetes cluster and its services.

The default dashboards contain metric data for the system's physical components – servers, switches, storage providers and their operating systems and firmware – as well as its logical components – controller software, platform, Kubernetes cluster and microservices, compute instances and their virtualized resources. This allows the administrator or support engineer to verify the health status of both component categories independently, and find correlations between them. For example, a particular microservice might exhibit poor performance due to lack of available memory. The monitoring data indicates whether this is a symptom of a system configuration issue, a lack of physical resources, or a hardware failure. The monitoring system has an alerting service capable of detecting and reporting hardware faults. The administrator may optionally configure a notification channel to receive alerts based on rules defined in the monitoring system.

As part of service and support operations, Oracle may ask you to report specific metric data displayed in the default dashboards. For this reason, the default dashboard configurations should always be preserved. However, if some of the monitoring functionality is inadvertently modified or broken, the defaults can be restored. In a similar way, it is possible for Oracle to create new dashboards or modify existing ones for improved monitoring, and push them to your operational environment without the need for a formal upgrade procedure.

All the open source monitoring and logging tools described here have public APIs that allow customers to integrate with their existing health monitoring and alerting systems. However, Oracle does not provide support for such custom configurations.

Fault Domain Observability

When it comes to keeping the appliance infrastructure, the compute instances and their related resources running in a healthy state, the *Fault Domain* is an extremely important concept. It groups a set of infrastructure components with the goal of isolating downtime events due to failures or maintenance, making sure that resources in other Fault Domains are not affected.

In line with Oracle Cloud Infrastructure, there are always three Fault Domains in a Private Cloud Appliance. Each of its Fault Domains corresponds with one or more physical compute

nodes. Apart from using Grafana to consult monitoring data across the entire system, an administrator can also access key capacity metrics for Fault Domains directly from the Service Enclave:

- Number of compute nodes per Fault Domain
- Total and available amount of RAM per Fault Domain
- Total and available number of vCPUs per Fault Domain
- Unassigned system CPU and RAM capacity

The Fault Domain metrics reflect the actual physical resources that can be consumed by compute instances hosted on the compute nodes. The totals do not include resources reserved for the operation of the hypervisor: 40GB RAM and 4 CPU cores (8 vCPUs).

In addition to the three Fault Domains, the Service CLI displays an "Unassigned" category. It refers to installed compute nodes that have not been provisioned, and thus are not part of a Fault Domain yet. For unassigned compute nodes the memory capacity cannot be calculated, but the CPU metrics are displayed.

System Health Checks

Health checks are the most basic form of detection. They run at regular intervals as Kubernetes CronJob services, which are very similar to regular UNIX cron jobs. A status entry is created for every health check result, which is always one of two possibilities: *healthy* or *not healthy*. All status information is stored for further processing in Prometheus; the unhealthy results also generate log entries in Loki with details to help advance the troubleshooting process.

Health checks are meant to verify the status of specific system components, and to detect status changes. Each health check process follows the same basic principle: to record the current condition and compare it to the expected result. If they match, the health check passes; if they differ, the health check fails. A status change from healthy to *not healthy* indicates that troubleshooting is required.

For the purpose of troubleshooting, there are two principal data sources at your disposal: logs and metrics. Both categories of data are collected from all over the system and stored in a central location: logs are aggregated in Loki and metrics in Prometheus. Both tools have a query interface that allows you to filter and visualize the data: they both integrate with Grafana. Its browser-based interface can be accessed from the Service Web UI.

To investigate what causes a health check to fail, it helps to filter logs and metric data based on the type of failure. Loki categorizes data with a labeling system, displaying log messages that match the selected log label. Select a label from the list to view the logs for the service or application you are interested in. This list allows you to select not only the health checks but also the internal and external appliance services.

In addition, the latest status from each health check is displayed in the Platform Health Check dashboard, which is part of the Service Advisor dashboard set provided by default in Grafana.

Private Cloud Appliance runs the health checks listed below.

Health Check Service	Description
cert-checker	Verifies on each management node that no certificates have expired.
flannel-checker	Verifies that the Flannel container network service is fully operational on each Kubernetes node.
kubernetes-checker	Verifies the health status of Kubernetes nodes and pods, as well as the containerized services and their connection endpoints.
mysql-cluster-checker	Verifies the health status of the MySQL cluster database.
l0-cluster-services-checker	Verifies that low-level clustering services and key internal components (such as platform API, DHCP) in the hardware and platform layer are fully operational.
network-checker	Verifies that the system network configuration is correct.
registry-checker	Verifies that the container registry is fully operational on each management node.
vault-checker	Verifies that the secret service is fully operational on each management node.
etcd-checker	Verifies that the etcd service is fully operational on each management node.
zfssa-analytics-exporter	Reports ZFS Storage Appliance cluster status, active problems, and management path connection information. It also reports analytics information for a configurable list of datasets.

Centralized Logging

The platform provides unified logging across the entire system. The Fluentd data collector retrieves logs from components across the entire system and stores them in a central location, along with the appliance telemetry data. As a result, all the necessary troubleshooting and debugging information is maintained in a single data store, and does not need to be collected from different sources when an issue needs to be investigated. The overall health of the system is captured in one view, a Grafana dashboard, meaning there is no need for an administrator to check individual components.

Whenever an issue is found that requires assistance from Oracle, the administrator logs a service request. A support bundle is usually requested as part of that process. Thanks to the centralized logging, the support bundle is straightforward to generate, and remains possible even if system operation is severely compromised. Generating the support bundle is a scripted operation that produces a single compressed archive file. The administrator does need to manually upload the archive file containing the consolidated logs and other diagnostic data.

If the Private Cloud Appliance is registered for ASR, certain hardware failures cause a service request and diagnostic data to be automatically sent to Oracle support.

Upgrade

Upgrading components of Private Cloud Appliance is the responsibility of the appliance administrator. The system provides a framework to verify the state of the appliance prior to an upgrade, and to execute an upgrade procedure as a workflow that initiates each individual task and tracks its progress until it completes. Thanks to built-in redundancy at all system

levels, the appliance components can be upgraded without service interruptions to the operational environment.

The source content for upgrades – packages, archives, deployment charts and so on – is delivered through an ISO image. During the preparation of the upgrade environment, the ISO image is downloaded to shared storage, and the upgrader itself is upgraded to the latest version included in the ISO. At this stage the appliance is ready for upgrades to be applied.

The administrator can perform an upgrade either through the Service Web UI or the Service CLI, and must select one of two available options: individual component upgrade or full management node cluster upgrade.

Pre-Checks

All upgrade operations are preceded by a verification process to ensure that system components are in the correct state to be upgraded. For these pre-checks, the upgrade mechanism relies on the platform-level health checks. Even though health checks are executed continually for monitoring purposes, they must be run specifically before an upgrade operation. The administrator is not required to run the pre-checks manually; they are executed by the upgrade code when an upgrade command is entered. All checks must pass for an upgrade to be allowed to start, even if a single-component upgrade is selected.

Certain upgrade procedures require that the administrator first sets a provisioning lock and maintenance lock. While the locks are active, no provisioning operations or other conflicting activity can occur, meaning the upgrade process is protected against potential disruptions. Once the upgrade has completed, the maintenance and provisioning locks must be released so the system returns to full operational mode.

Single Component Upgrade

Private Cloud Appliance upgrades are designed to be modular, allowing individual components to be upgraded rather than the entire system at once. With single component upgrade, the following component options are available:

- **ILOM firmware**

  Use this option to upgrade the Oracle Integrated Lights Out Manager (ILOM) firmware of a specific server within the appliance. After the firmware is upgraded successfully, the ILOM is automatically rebooted. However, the administrator must manually restart the server for all changes to take effect.

- **Switch firmware**

  Use this option to upgrade the operating software of the switches. You must specify which switch category to upgrade: the leaf switches, the spine switches, or the management switch.

- **ZFS Storage Appliance firmware**

  Use this option to upgrade the operating software on the ZFS Storage Appliance. Both controllers, which operate in an active-active cluster configuration, are upgraded as part of the same process.

- **Host operating system**

Use this option to upgrade the Oracle Linux operating system on a management node. It triggers a yum upgrade on the selected management node, and is configured to use a yum repository populated through the ISO image.

- **Clustered MySQL database**

  Use this option to upgrade the MySQL database on all management nodes. The database installation is rpm-based and thus relies on the yum repository that is populated through the ISO image. The packages for the database are deliberately kept out of the host operating system upgrade, because the timing of the database upgrade is critical. The database upgrade workflow manages the backup operations and the cluster state, and stops and restarts the relevant services. It ensures all the steps are performed in the correct order on each management node.

- **Kubernetes cluster**

  Use this option to upgrade the Kubernetes cluster, which is the container orchestration environment where services are deployed. The Kubernetes cluster runs on all the management nodes and compute nodes; its upgrade involves three major operations:

  – Upgrading the Kubernetes packages and all dependencies: kubeadm, kubelet, kubectl and so on.

  – Upgrading the Kubernetes container images: kube-apiserver, kube-controller-manager, kube-proxy and so on.

  – Updating any deprecated Kubernetes APIs and services YAML manifest files.

- **Secret service**

  The process to upgrade the secret service on all management nodes consists of two parts. It involves a rolling upgrade of the two main secret service components: the etcd key value store and the Vault secrets manager. Both are upgraded in no particular order and independently of each other, using the new image files made available in the podman registry.

- **Platform services**

  Use this option to upgrade the containerized services running within the Kubernetes cluster on the management nodes. The service upgrade mechanism is based on Helm, the Kubernetes equivalent of a package manager. For services that need to be upgraded, new container images and Helm deployment charts are delivered through an ISO image and uploaded to the internal registry. None of the operations up to this point have an effect on the operational environment.

  At the platform level, an upgrade is triggered by restarting the pods that run the services. The new deployment charts are detected, causing the pods to retrieve the new container image when they restart. If a problem is found, a service can be rolled back to the previous working version of the image.

  > **Note:**
  >
  > In specific circumstances it is possible to upgrade certain platform services individually, by adding an optional JSON string to the command. This option should not be used unless Oracle provides explicit instructions to do so.

- **Compute node**

  Use this option to perform a yum upgrade of the Oracle Linux operating system on a compute node. Upgrades include the `ovm-agent` package, which contains appliance-

specific code to optimize virtual machine operations and hypervisor functionality. You must upgrade the compute nodes one by one; there can be no concurrent upgrade operations.

Full Management Node Cluster Upgrade

Upgrades of individual components are largely self-contained. The full management node cluster upgrade integrates a number of those component upgrades into a global workflow that executes the component upgrades in a predefined order. With a single command, all three management nodes in the cluster are upgraded sequentially and component by component. This means an upgrade of a given component is executed on each of the three management nodes before the global workflow moves to the next component upgrade.

The order in which components are upgraded is predefined because of dependencies, and must not be changed. During the full management node cluster upgrade, the following components are upgraded:

1. Management node host operating system
2. Clustered MySQL database
3. Secret service
4. Kubernetes cluster
5. Platform services

Patching

Patching refers to the ability to apply security enhancements and functional updates to Private Cloud Appliance independently of regular product releases. Patches are delivered as RPM packages through a series of dedicated channels on the Unbreakable Linux Network (ULN). To gain access to these channels, you need a Customer Support Identifier (CSI) and a ULN subscription.

The appliance is not allowed to connect directly to Oracle's ULN servers. The supported process is to set up a ULN mirror on a system inside the data center. The patch channels are then synchronized on the ULN mirror, where the management nodes can access the RPMs. Compute nodes need access to a subset of the RPMs, which are copied to a designated location on the internal shared storage and kept up-to-date.

Patches are installed using a mechanism similar to upgrade. A key difference is that patching commands include the path to the ULN mirror. The Service CLI provides a separate patch command for each supported component type. In the Service Web UI the administrator applies a patch by creating an upgrade request for a given component type, but selecting the *Patch* option instead of Upgrade.

ULN patches may be delivered for any of the component types that are part of a traditional product release: operating system updates for management and compute nodes, platform updates, microservices updates, firmware updates, compute image updates, and so on.

For more detailed information and step-by-step instructions, refer to the Oracle Private Cloud Appliance Patching Guide

Backup and Restore

The integrated Private Cloud Appliance backup service is intended to protect the system configuration against data loss and corruption. It does not create backups of the customer environment, but is instead geared toward storing the data required for system and service operation, so that any crucial service or component can be restored to its last-known healthy state. In line with the microservice-based deployment model, the backup service orchestrates the various backup operations across the entire system and ensures data consistency and integrity, but it does not define the individual component backup requirements. That logic is part of the component backup plugins.

The backup plugin is the key element that determines which files must be backed up for a given system component or service, and how the data must be collected. For example, a simple file copy may work for certain files while a snapshot is required for other data, or in some cases a service may need to be stopped to allow the backup to be created. The plugin also determines the backup frequency and retention time. Each plugin registers with the backup service so that it is aware of the active plugins and can schedule the required backup operations in a consistent manner as Kubernetes CronJobs. Plugins are aggregated into a backup profile; the backup profile is the task list that the backup service executes when a backup job is launched.

The backup data collected through the plugins is then stored by the backup service in a dedicated NFS share on the internal ZFS Storage Appliance, using ZFS encryption to ensure that the data at rest is secure. If required, the backup files can optionally be replicated to an external storage location.

When restoring a service or component from a backup, the service again relies on the logic provided by the plugin. A component restore process has two major phases: verification and data management. During the verification phase, the backup is evaluated for completeness and appropriateness in relation to the current condition of the component. Next, during the data management phase, the required actions are taken to stop or suspend a component, replace the data, and restart or resume normal operation. As with backup, the operations to restore the data are specific to the component in question.

The default backup and restore implementation is to execute a global backup profile that covers the MySQL cluster database, the ZFS Storage Appliance configuration, a snapshot of the ZFS projects on the storage appliance, and all registered component backup plugins. The default profile is executed daily at midnight UTC and has a 14-day retention policy. Backups are stored in `/nfs/shared_storage/backups/backup_*`. All restore operations must be performed manually on a per-component basis.

> **Note:**
>
> In release 3.0.1 of Private Cloud Appliance, the Backup and Restore service is not available through the Service Web UI or Service CLI, which also implies that administrators cannot configure the backup schedule.
>
> Automated restore operations based on the backup plugins are currently not possible. If a manual restore from a backup is required, please contact Oracle for assistance.

Disaster Recovery

The goal of disaster recovery is to provide high availability at the level of an installation site, and to protect critical workloads hosted on a Private Cloud Appliance against outages and data loss. The implementation requires two Private Cloud Appliance systems installed at different sites, and a third system running an Oracle Enterprise Manager installation with Oracle Site Guard.

The two Private Cloud Appliance systems are both fully operational environments on their own, but at the same time configured to be each other's replica. A dedicated network connection between the two peer ZFS Storage Appliances – one in each rack – ensures reliable data replication at 5-minute intervals. When an incident is detected in either environment, the role of Oracle Site Guard is to execute the failover workflows, known as operation plans.

Setting up disaster recovery is the responsibility of an appliance administrator or Oracle engineer. It involves interconnecting all participating systems, and configuring the Oracle Site Guard operation plans and the replication settings on both Private Cloud Appliance systems. The administrator determines which workloads and resources are under disaster recovery control by creating and managing *DR configurations* through the Service CLI on the two appliances.

The DR configurations are the core elements. The administrator adds critical compute instances to a DR configuration, so that they can be protected against site-level incidents. Storage and network connection information is collected and stored for each instance included in the DR configuration. With the creation of a DR configuration, a dedicated ZFS project is set up for replication to the peer ZFS Storage Appliance, and the compute instance resources involved are moved from the default storage location to this new ZFS project. A DR configuration can be refreshed at all times to pick up changes that might have occurred to the instances it includes.

Next, site mapping details are added to the DR configuration. All relevant compartments and subnets must be mapped to their counterparts on the replica system. A DR configuration cannot work unless the compartment hierarchy and network configuration exist on both Private Cloud Appliance systems.

When an incident occurs, failover operations are launched to bring up the instances under disaster recovery control on the replica system. This failover is not granular but a site-wide process involving these steps:

1. The site-level incident causes running instances to fail abruptly. They cannot be shut down gracefully.
2. Reversing the roles of the primary and replica ZFS Storage Appliance.
3. Recovering the affected compute instances of the primary system by starting them on the replica system.
4. Cleaning up the primary system: removing stopped instances, frozen DR configurations, and so on.
5. Setting up reverse DR configurations based on the ZFS project and instance metadata.

A failover is the result of a disruptive incident being detected at one of the installation sites. However, Oracle Site Guard also supports switchover, which is effectively the same process but manually triggered by an administrator. In a controlled switchover scenario the first step in the process is to safely stop the running instances on the

primary system to avoid data loss or corruption. Switchover is typically used for planned maintenance or testing. After a failover or switchover, when both sites are fully operational, a failback is performed to return the two Private Cloud Appliance systems to their original configuration.

Serviceability

Serviceability refers to the ability to detect, diagnose and correct issues that occur in an operational system. Its primary requirement is the collection of system data: general hardware telemetry details, log files from all system components, and results from system and configuration health checks. For a more detailed description of monitoring, system health and logging, refer to Status and Health Monitoring.

As an engineered system, Private Cloud Appliance is designed to process the collected data in a structured manner. To provide real-time status information, the system collects and stores metric data from all components and services in a unified way using Prometheus. Centrally aggregated data from Prometheus is visualized through metric panels in a Grafana dashboard, which permits an administrator to check overall system status at a glance. Logs are captured across the appliance using Fluentd, and collected in Loki for diagnostic purposes.

When a component status changes from healthy to not healthy, the alerting mechanism can be configured to send notifications to initiate a service workflow. If support from Oracle is required, the first step is for an administrator to open a service request and provide a problem description.

If the Private Cloud Appliance is registered for ASR, certain hardware failures cause a service request and diagnostic data to be automatically sent to Oracle support. The collection of diagnostic data is also called a support bundle. A Service Enclave administrator can also create and send a service request and supporting diagnostic data separate from ASR. For more information about ASR and support bundles, see Status and Health Monitoring in the *Oracle Private Cloud Appliance Administrator Guide*.

To resolve the reported issue, Oracle may need access to the appliance infrastructure. For this purpose, a dedicated service account is configured during system initialization. For security reasons, this non-root account has no password. You must generate and provide a service key to allow the engineer to work on the system on your behalf. Activity related to the service account leaves an audit trail and is clearly separated from other user activity.

Most service scenarios for Oracle Engineered Systems are covered by detailed action plans, which are executed by the assigned field engineer. When the issue is resolved, or if a component has been repaired or replaced, the engineer validates that the system is healthy again before the service request is closed.

This structured approach to problem detection, diagnosis and resolution ensures that high-quality service is delivered, with minimum operational impact, delay and cost, and with maximum efficiency.

4
Identity and Access Management Overview

The Identity and Access Management service (IAM) lets you control who has access to the cloud resources within your tenancies. It is the task of a tenancy administrator to control what type of access a user group has, and to which specific resources that access applies. The responsibility to manage and maintain access control can be delegated to other privileged users, for instance by granting them full access to a subcompartment of the tenancy.

Appliance administrator accounts are managed separately and provide access to appliance administration functions. This functionality is not related to the tenancy-level IAM service. For more information, refer to the section Administrator Access.

Users and Groups

When a tenancy is created, a default user account is added to allow you to log in and perform initial setup tasks. This default user is included in a group named *Administrators*, which provides full access to all resources and operations within the tenancy. The group cannot be deleted and must always contain at least one user.

Once logged in, the tenancy administrator can start adding more users and organize them into groups. A group is a set of users who have the same type of access to a particular set of resources. The general principle is that users have no access rights at all, unless they have been explicitly granted permission.

User accounts can be created locally in the tenancy, but Private Cloud Appliance also supports federating with an existing identity provider. In this configuration, a tenancy administrator sets up a *federation trust* relationship between the tenancy and the identity provider, allowing users log in with their existing id and password. Each existing user group from the identity provider can be mapped to a group in the tenancy, so that existing group definitions can be re-used to authorize access to cloud resources. For more information, see Federating with Identity Providers.

The permission to access a resource and perform an operation is defined in a policy. The policy for the Administrators group states that the users in this group are allowed to manage all resources in the tenancy. For all other user groups as well, a policy must be defined to manage their specific permissions. For more information, see How Policies Work.

To group and isolate resources, you organize them into compartments. Compartments are primary building blocks in a tenancy. You can compare them to the directories in a file system structure, where the tenancy is equivalent to the root directory. Compartments also help control and secure the access to resources. Unlike administrators, regular users only see the compartments to which they have access. Policy statements further refine the type of access. For more information, see Organizing Resources in Compartments.

As an example of how users, groups, compartments, resources and policies interact with each other, consider the following scenario. You decide to create groups for different teams in your organization and assign a separate compartment for each team's resources. You allow each team to create and use instances within their compartment but prevent them from accessing the resources of another team. In addition, you might prefer to let a network administrator manage all network resources in the tenancy. To achieve this, you create all network-related resources in a dedicated network compartment, which only a network

administrator is allowed to manage. Other users need to be allowed to use the network resources in their configurations, but they should not have permission to modify the network setup.

User Credentials

Different types of credentials are managed through the Identity and Access Management (IAM) service:

- Account password: for signing in to the Compute Web UI to work with cloud resources in the tenancy. Note that passwords for federated users are not managed through IAM because the identity provider controls their login activity.
- API signing key: for sending API requests, which require authentication. These keys must be created in PEM format.

Account Passwords

When creating a new user account, the tenancy administrator generates a one-time password and delivers it to the user in a secure way. When users sign in for the first time, they are prompted to change this password. After 7 days the one-time password expires and an administrator will need to generate a new one.

After signing in successfully with the new password, users can start working with cloud resources in the tenancy, in accordance with the permissions they have been granted.

All users are allowed to change their own password, which can be done through the Compute Web UI. Users who forgot their password must request a tenancy administrator to reset the password for them.

After 10 unsuccessful login attempts in a row, a user is automatically locked out of the system. A tenancy administrator needs to unblock the account.

API Signing Keys

Users who need to make API requests must have an RSA public key added to their user profile. Both the private and public key must be in PEM format, with a minimum length of 2048 bits. Users either generate a private/public key pair through the Compute Web UI and download the private key, or generate the key pair on their local machine and upload the public key to their profile.

Alternatively, a tenancy administrator can generate the API keys and complete the profile setup for all users. This is a requirement for non-human user accounts: systems that make API requests without human operation. For such systems, the administrator needs to create a user account with signing keys, but without password.

On the system from where API requests are sent, a directory named `.oci` must be created inside the user home directory. The `.oci` directory must contain a configuration file with required parameters for interaction with the API server. Make sure it lists the correct path to where the private key file is stored, if it is not in the same directory. API requests are signed using the private key.

A user account can contain a maximum of 3 API signing keys at a time. API signing keys are different from the SSH keys you use to access a compute instance.

Federating with Identity Providers

Many companies use an identity provider to manage user logins and passwords and to authenticate users for access to secure websites, services, and resources. To access the Private Cloud Appliance Compute Web UI, people must also sign in with a user name and password. An administrator can *federate* with a supported identity provider so that each user can use an existing login and password, rather than having to create new credentials to access and use cloud resources.

Federation involves setting up a trust relationship between the identity provider and Private Cloud Appliance. When the administrator has established that relationship, any user who goes to the Compute Web UI is prompted with a *single sign-on* experience offered by the identity provider.

Supported Identity Providers

For identity federation, Private Cloud Appliance supports the Security Assertion Markup Language (SAML) 2.0 protocol. However, the only identity provider supported in version 3.0.1 is Microsoft Active Directory, via Active Directory Federation Services.

Your organization can have multiple Active Directory accounts; for example one for each division of the organization. You can federate multiple Active Directory accounts with Private Cloud Appliance, but each federation trust that you set up must be for a single Active Directory account. Identity federation is available in the Compute Web UI as well as the Service Web UI.

Private Cloud Appliance version 3.0.1 does not currently support the System for Cross-domain Identity Management (SCIM), a standard protocol that enables user provisioning across identity systems. Without SCIM, user accounts from an identity provider cannot be automatically provisioned in a tenancy and synchronized. Consequently, the credentials of federated users cannot be managed within the tenancy.

Experience for Federated Users

When browsing to the Compute Web UI, the user is prompted to enter the name of the tenancy. Federated users then choose which identity provider to use, and are redirected to the identity provider's sign-in interface for authentication. After entering the user name and password that they already set up, they are authenticated by the identity provider, and redirected back to the Private Cloud Appliance Compute Web UI. From here, they can access the resources in their tenancy, in accordance with the permissions granted to them.

Federated user accounts are created and managed within the identity provider; they are not replicated or synchronized within Private Cloud Appliance. Federated users are granted access based on their membership of identity provider groups that are mapped to Private Cloud Appliance groups in the tenancy. Unlike local users, federated users cannot manage their credentials through the Compute Web UI. They have no User Settings page, cannot change or reset their password, and cannot upload API signing keys to use the API and CLI.

Federation Process Overview

To set up the federation trust between Private Cloud Appliance and an identity provider, you perform certain steps of the procedure on both systems respectively. In general, identity federation consists of these steps:

1. Configuring groups in the identity provider, so they can be mapped to groups in the Private Cloud Appliance tenancy.
2. Downloading the SAML metadata document from the identity provider and collecting the names of the groups you intend to map.
3. Setting up the new identity provider in your Private Cloud Appliance tenancy. You need to upload the identity provider metadata file. Create the group mappings at this stage, or return to add them later.
4. Downloading the Private Cloud Appliance federation metadata document from the Federation page in your tenancy.
5. Setting up Private Cloud Appliance as a trusted application or trusted relying party in your identity provider. You need to upload the Private Cloud Appliance federation metadata document, or provide the URL to it.

   The identity provider SAML authentication response must be configured to include *name ID* and *groups*, which are parameters required by Private Cloud Appliance.
6. Setting up IAM policies for the mapped groups.
7. Providing federated users the name of the tenancy and the URL to the Private Cloud Appliance Compute Web UI.

Organizing Resources in Compartments

After initial setup, your tenancy only contains a root compartment. A tenancy administrator needs to perform setup tasks and establish an organization plan. The plan should include the compartment hierarchy for organizing your resources and the definitions of the user groups that need access to the resources. These two things impact how you write policies to manage access, and therefore should be considered together.

Understanding Compartments

Think carefully about how you want to use compartments to organize and isolate your cloud resources. Compartments are fundamental to that process. Most resources can be moved between compartments. However, it's important to think through your compartment design for your organization up front, before implementing anything.

When planning your compartment structure, keep the following things in mind:

- You can create a hierarchy up to six compartments deep under the tenancy or root compartment.
- Compartments are logical, not physical. Related resource components can be placed in different compartments. For example, your cloud network subnets with access to an internet gateway can be secured in a separate compartment from other subnets in the same cloud network.
- Compartments behave like a filter for viewing resources. When you select a compartment, the Compute Web UI only shows you resources that are in that compartment. To view resources in another compartment, you must first select that compartment.

  This experience is different when viewing users, groups, and federation providers. Those reside in the tenancy itself, not in an individual compartment. A policy can

be attached to either the tenancy or a compartment. Where it is attached controls who has access to modify or delete it.

- The Compute Web UI only displays the compartments and resources that you have permission to access. Only an administrator can view all compartments and work with all resources.
- At the time you create a resource (for example, instance, block storage volume, VCN, subnet), you must decide in which compartment to put it.

Considerations for Granting Resource Access

Another primary consideration when planning the setup of your tenancy is who should have access to which resources. Defining how different groups of users need to access the resources helps you plan how to organize your resources most efficiently, making it easier to write and maintain your access policies.

For example, you might have users who need to:

- View the Compute Web UI, but not be allowed to edit or create resources
- Create and update specific resources across several compartments; for example: network administrators who need to manage your cloud networks and subnets
- Launch and manage instances and block volumes, but not have access to your cloud network
- Have full permissions on all resources, but only in a specific compartment
- Manage other users' permissions and credentials

For information on accessing resources across tenancies, see Cross-Tenancy Policies.

Examples of Compartment Organization

This section describes examples of how compartments can be structured to align with your resource management goals.

All Resources in the Root Compartment

If your organization is small, or if you are still in the proof-of-concept stage, you might consider placing all of your resources in the root compartment or the tenancy. This approach makes it easy for you to quickly view and manage all your resources. You can still write policies and create groups to restrict permissions on specific resources to only the users who need access.

High-level tasks to set up the single compartment approach:

1. Create a sandbox compartment. Even though your plan is to maintain your resources in the root compartment, Oracle recommends setting up a sandbox compartment so that you can give users a dedicated space to try out features. In the sandbox compartment you can grant users permissions to create and manage resources, while maintaining stricter permissions on the resources in your tenancy (root) compartment.
2. Create groups and policies.
3. Add users.

Separately Managed Project Compartments

Consider this approach if your organization has multiple departments that you want to manage separately, or if several distinct projects exist that would be easier to manage separately.

In this approach, you can add a dedicated administrators group for each project compartment who can set the access policies for just that project. Users and groups still must be added at the tenancy level. You can give one group control over all their resources, while not allowing them administrator rights to the root compartment or any other projects. This way, you enable different groups to set up their own "sub-clouds" for their own resources and manage them independently.

High-level tasks to set up the multi-project approach:

1. Create a sandbox compartment. Oracle recommends setting up a sandbox compartment so that you can give users a dedicated space to try out features. In the sandbox compartment you can grant users permissions to create and manage resources, while maintaining stricter permissions on the resources in your tenancy and other compartments.
2. Create a compartment for each project. For example: ProjectA, ProjectB.
3. Create an administrators group for each project. For example: ProjectA_Admins.
4. Create a policy for each administrators group. For example:

   ```
   Allow group ProjectA_Admins to manage all-resources in compartment ProjectA
   ```

5. Add users.
6. Let the administrators for ProjectA and ProjectB create subcompartments within their designated compartment to manage resources.
7. Let the administrators for ProjectA and ProjectB create the policies to manage the access to their compartments.

Working with Compartments

This section describes the basics of compartment management.

Creating Compartments

When creating a compartment, you must provide a name that is unique within the tenancy. It can be up to 100 characters long, and include letters, numbers, periods, hyphens, and underscores. You must also provide a description for the compartment, which must be 1-400 characters long and is non-unique and changeable. The new compartment is automatically assigned a unique identifier called an Oracle Cloud ID (OCID).

You can create subcompartments within compartments, to create hierarchies that are up to six levels deep.

Compartment Access Control

After creating a compartment, you need to write at least one policy for it, otherwise it is only accessible to administrators and users with permissions set at the tenancy level.

When creating a compartment inside another compartment, the subcompartment inherits access permissions from compartments higher up its hierarchy.

When you create an access policy, you need to specify which compartment to attach it to. This controls who can later modify or delete the policy. Depending on your compartment hierarchy design, you might attach it to the tenancy, a parent, or to the specific compartment itself.

For information on accessing resources across tenancies, see Cross-Tenancy Policies.

Adding Resources to Compartments

When you start working with cloud resources in the Compute Web UI, you must choose from a list which compartment to work in. That list is filtered to show only the compartments in the tenancy that you have permission to access.

To place a new resource in a compartment, you simply specify that compartment when creating the resource. The compartment is one of the required parameters to create a resource. When working in the Compute Web UI, make sure you are first viewing the compartment where you want to create the resource.

Keep in mind that most IAM resources reside in the tenancy. This is always the case for users and groups, but compartments and policies are also often attached to the tenancy. Tenancy-level resources cannot be created or managed from a specific compartment.

Moving Resources Between Compartments

Most resources can be moved after they are created.

Some resources have attached resource dependencies. When the parent resource is moved to another compartment, the attached dependencies do not all behave in the same way. Some are moved immediately with their parent resources, and some are moved asynchronously, meaning they are not visible in the new compartment until the move is complete. For other resources, the attached resource dependencies are not automatically moved to the new compartment. You can move these attached resources independently.

After you move a resource to a new compartment, the policies that govern the new compartment apply immediately and affect access to the resource. See the service documentation for individual resources to familiarize yourself with the behavior of each resource and its attachments.

Deleting Compartments

To delete a compartment, it must be empty of all resources. Before you initiate deleting a compartment, be sure that all its resources have been moved, deleted, or terminated, including any policies attached to the compartment.

The delete action is asynchronous and initiates a work request, which typically takes several minutes to complete. While a compartment is in the *deleting* state it is not displayed in the compartment picker. If the work request fails, the compartment is not deleted and it returns to the *active* state. You can track the progress of a work request during the operation, or check afterwards whether it completed successfully or if errors occurred.

After a compartment is deleted, its state is updated to *deleted* and a random string of characters is appended to its name. For example, CompartmentA might become CompartmentA.qR5hP2BD. Renaming the compartment allows you to reuse the original name for a different compartment. Deleted compartments are listed in the Compartments

page for 365 days, but are removed from the compartment picker. If any policy statements reference the deleted compartment, these statements are updated to reflect the new name.

If the work request indicates that the compartment failed to delete, verify that you have removed all the resources. Verify that there are no policies in the compartment. If you cannot locate any resources in the compartment, check with your administrator; you might not have permission to view all resources.

Moving a Compartment to a Different Parent Compartment

To move a compartment, you must belong to a group that has `manage all-resources` permissions on the lowest shared parent compartment of the current compartment and the destination compartment.

You can move a compartment to a different parent compartment within the same tenancy. When you move a compartment, all its contents, meaning its subcompartments and resources, are moved with it. This section also describes the implications of moving a compartment. Ensure that you are aware of these before you move a compartment.

When moving a compartment, these restrictions apply:

- You cannot move a compartment to a destination compartment with the same name as the compartment being moved.
- Two compartments within the same parent cannot have the same name. Therefore you cannot move a compartment to a destination compartment where a compartment with the same name already exists.

Policy Implications

After you move a compartment to a new parent compartment, the access policies of the new parent take effect and the policies of the previous parent no longer apply. Before you move a compartment, ensure that:

- You are aware of the policies that govern access to the compartment in its current position.
- You are aware of the polices in the new parent compartment that will take effect when you move the compartment.

Groups with access to resources in the current compartment lose their permissions when the compartment is moved; groups with permissions in the destination compartment gain access. Ensure that you are aware not only of what groups lose permissions when you move a compartment, but also what groups will gain permissions. If necessary, adjust the policy statements to avoid certain users inadvertently losing access.

Tagging Implications

Tags are not automatically updated after a compartment has been moved. If you have implemented a tagging strategy based on compartment, you must update the tags on the resources after moving the compartment.

For example, assume that CompartmentA has a child compartment named CompartmentB. CompartmentA is set up with tag defaults so that every resource in CompartmentA is tagged with TagA. Therefore CompartmentB and all its resources

are tagged with this default tag, TagA. When you move CompartmentB to CompartmentC, it will still have the default tags from CompartmentA. If you have set up default tags for CompartmentC, you need to add them to the resources in the moved compartment.

How Policies Work

A policy is a document that specifies who can access which cloud resources in your tenancy, and how. A policy simply allows a group to work in certain ways with specific types of resources in a particular compartment. If you are not familiar with users, groups, or compartments, refer to the respective sections in the chapter Identity and Access Management Overview.

Policy Basics

To govern control of your resources, you need at least one policy. Each policy consists of one or more policy statements that follow this basic syntax:

```
Allow group <group_name> to <verb> <resource-type> in compartment <compartment_name>
```

Policy statements begin with the word `Allow`. Policies only allow access; they cannot deny it. Instead, all access is implicitly denied, meaning users can only do what they have been granted permission for. A tenancy administrator defines the groups and compartments; the available resource types are determined by Oracle. The use and meaning of verbs in policy statements is described in Policy Syntax.

If you want a policy to apply to the tenancy and not a compartment inside the tenancy, change the end of the policy statement as follows:

```
Allow group <group_name> to <verb> <resource-type> in tenancy
```

A basic feature of policies is the concept of *inheritance*: compartments inherit any policies from their parent compartment. If a group has a particular level of access to certain resource types in a compartment, then those same permissions apply in all subcompartments of the compartment where this policy is applied. The simplest example is the Administrators group in the tenancy: the built-in policy allows the administrators to manage all the resources in the tenancy root compartment. Because of policy inheritance, the administrators have full access to all operations and all resources in every compartment.

Resource Types

The resource types that you can use in policies are either *individual* or *family* types. The family resource types make policy writing easier, as they include multiple individual resource types that are often managed together. For example, the `virtual-network-family` type brings together a variety of types related to the management of VCNs: `vcns`, `subnets`, `route-tables`, `security-lists`, etc. If you need to write a more granular policy, use an individual resource type to give access to only those specific resources. Note that there are other ways to make policies more granular, such as the ability to specify conditions under which the access is granted.

With future service updates, it is possible that resource type definitions are changed or added. These are typically reflected automatically in the resource family type for that service, so your policies remain current.

Some operations require access to multiple resource types. For example, launching an instance requires the permission to create instances and to work with a cloud network. Or

creating a volume backup requires access to both the volume and the volume backup. That means you have separate statements to give access to each resource type.

These individual statements do not have to be in the same policy. A user can gain the required access from being in different groups. For example, a user could be in one group that gives the required level of access to the `volumes` resource type, and in another group that gives the required access to the `volume-backups` resource type. The sum of the individual statements, regardless of their location in the overall set of policies, allows the user to create a volume backup.

Policy Attachment

Another basic feature of policies is the concept of *attachment*. When you create a policy you must attach it to a compartment; or to the tenancy, which is the root compartment. Where you attach it controls who can then modify it or delete it. If you attach a policy to the tenancy, it can only be modified by the Administrators group, and not by users with only access to a subcompartment.

If you instead attach the policy to a child compartment, then anyone with access to manage the policies *in that compartment* can change or delete it. In practical terms, you can give compartment administrators – a group with access to manage all resources in the compartment – access to manage their own compartment's policies, without giving them broader access to manage policies that reside in the tenancy.

To attach a policy to a compartment, you must be in that compartment when you create the policy. As part of a policy statement you specify the compartment it applies to, so if you try to attach the policy to a different compartment you get an error. Policy attachment occurs at the time of creation, which means a policy can be attached to one compartment only.

Policy Syntax

The overall syntax of a policy statement is as follows:

```
Allow <subject> to <verb> <resource-type> in <location> where <conditions>
```

Additional spaces or line breaks in the statement have no effect.

Subject

Specify a group by name or OCID. You can specify multiple groups separated by commas. To cover all users in the tenancy, specify `any-user`.

These examples show how you can specify the subject in a policy statement.

- To specify a single group by name:

  ```
  Allow group A-admins to manage all-resources in compartment Project-A
  ```

- To specify multiple groups by name (a space after the comma is optional):

  ```
  Allow group A-admins, B-admins to manage all-resources in compartment
  Projects
  ```

- To specify a single group by OCID (the OCID is shortened for brevity):

  ```
  Allow group id ocid1.group.....<group1_unique_id>
  to manage all-resources in compartment Project-A
  ```

- To specify multiple groups by OCID (the OCIDs are shortened for brevity):

```
Allow
group id ocid1.group.....<group1_unique_id>,
group id ocid1.group.....<group2_unique_id>
to manage all-resources in compartment Projects
```

- To specify any user in the tenancy:

```
Allow any-user to inspect users in tenancy
```

Verb

Specify a single verb.

```
Allow group A-admins to manage all resources in compartment Project-A
```

The policy syntax supports the following verbs, ordered by increasing permissions:

Verb	Type of Access	Target User
`inspect`	Ability to list resources, without access to any confidential information or user-specified metadata that may be part of that resource. **Notes:** • The operation to list policies includes the contents of the policies themselves. • The list operations for the Networking resource types return all the information, including the contents of security lists and route tables.	Third-party auditors
`read`	Includes `inspect` plus the ability to get user-specified metadata and the actual resource itself.	Internal auditors
`use`	Includes `read` plus the ability to work with existing resources. The actions vary by resource type. Includes the ability to update the resource, except for resource types where the "update" operation has the same effective impact as the "create" operation; for example `UpdatePolicy`, `UpdateSecurityList`, etc. In those cases the "update" ability is available only with the `manage` verb. In general, the verb `use` does not include the ability to create or delete that type of resource.	Day-to-day end users of resources
`manage`	Includes all permissions for the resource.	Administrators

The verb gives a certain general type of access. For example, `inspect` lets you list and get resources. You then join that type of access with a particular resource type in a policy. For example, allow group XYZ to `inspect compartments` in the tenancy. As a result, that group gains access to a specific set of permissions and API operations; for example `ListCompartments`, `GetCompartment`.

Resource Type

Specify a single resource-type, which can be:

- An individual resource type; for example: `vcns`, `subnets`, `instances`, `volumes`, etc.

- A family resource type; for example: `virtual-network-family`, `instance-family`, `volume-family`, etc.

  A family resource type covers a variety of individual resource types that are typically used together.
- `all-resources`: Covers all resources in the compartment or tenancy.

These examples show how you can specify the resource type in a policy statement.

- To specify a single resource type:

  ```
  Allow group HelpDesk to manage users in tenancy
  ```
- To specify multiple resource types, use separate statements:

  ```
  Allow group A-users to manage instance-family in compartment Project-A
  Allow group A-users to manage volume-family in compartment Project-A
  ```
- To specify all resources in the compartment or tenancy:

  ```
  Allow group A-admins to manage all-resources in compartment Project-A
  ```

Here is an overview of the family resource types can be used in policy statements:

Family Resource Type	Description
`compute-management-family`	This aggregate resource covers the following individual resource types: `instance-configurations`, `instance-pools`, `cluster-networks`.
`instance-family`	This aggregate resource covers the following individual resource types: `app-catalog-listing`, `console-histories`, `instances`, `instance-console-connection`, `instance-images`, `volume-attachments`.
`volume-family`	This aggregate resource covers all individual resource types related to block volumes: `volumes`, `volume-attachments`, `volume-backups`, `boot-volume-backups`, `backup-policies`, `backup-policy-assignments`, `volume-groups`, `volume-group-backups`.
`virtual-network-family`	This aggregate resource covers all individual resource types related to the networking service. For example: VCNs, subnets, route tables, gateways, VNICs, network security groups, and so on.
`file-family`	This aggregate resource covers all individual resource types related to the file storage service: `file-systems`, `mount-targets`, `export-sets`.
`object-family`	This aggregate resource covers all individual resource types related to the object storage service: `objectstorage-namespaces`, `buckets`, `objects`.

Location

Specify a single compartment by name or OCID. Or simply specify `tenancy` to cover the entire tenancy. Remember that users, groups, and compartments reside in the tenancy. Policies can be attached to either the tenancy or a child compartment.

The location is required in the statement. If you want to attach a policy to a compartment, you must be in that compartment when you create the policy.

These examples show how you can specify the location in a policy statement.

- To specify a compartment by name:

```
Allow group A-admins to manage all-resources in compartment Project-A
```

- To specify a compartment by OCID:

```
Allow group A-admins to manage all-resources
in compartment id ocid1.compartment.oc1..aaaaaaaaexampleocid
```

- To specify multiple compartments, use separate statements:

```
Allow group InstanceAdmins to manage instance-family in compartment Project-A
Allow group InstanceAdmins to manage instance-family in compartment Project-B

Allow group InstanceAdmins to manage instance-family
in compartment id ocid1.compartment.oc1..aaaaaaaayzexampleocid
Allow group InstanceAdmins to manage instance-family
in compartment id ocid1.compartment.oc1..abcabcabcexampledocid
```

Conditions

Specify one or more conditions. With multiple conditions, use `any` or `all` for a logical OR or AND, respectively.

These are the types of values you can use in conditions:

Value Type	Examples

String	Single quotation marks are required around the value.
	`'johnsmith@example.com'`
	`'ocid1.compartment.oc1..aaaaaaaaph...ctehnqg756a'`

Pattern	`/HR*/` - matches strings that start with "HR" `/*HR/` - matches strings that end with "HR" `/*HR*/` - matches strings that contain "HR"

These examples show how you can specify conditions in a policy statement.

> **Note:**
>
> In the example statements, the condition to match group names makes it impossible for GroupAdmins to list all users and groups. The list operation does not involve specifying a group, which means there is no value to match the condition variable `target.group.name`. To resolve this, a statement including the `inspect` verb is added.

- The following policy enables the GroupAdmins group to create, update, or delete any groups with names that start with "A-Users-":

```
Allow group GroupAdmins to manage groups in tenancy where target.group.name = /A-Users-*/
Allow group GroupAdmins to inspect groups in tenancy
```

- The following policy enables the NetworkAdmins group to manage cloud networks in any compartment except the one specified:

```
Allow group NetworkAdmins to manage virtual-network-family in tenancy
where target.compartment.id != 'ocid1.compartment.oc1..aaaaaaaaexampleocid'
```

- The following policy uses multiple conditions and lets GroupAdmins create, update, or delete any groups whose names start with "A-", except for the A-Admins group itself:

```
Allow group GroupAdmins to manage groups in tenancy
where all {target.group.name=/A-*/,target.group.name!='A-Admins'}
Allow group GroupAdmins to inspect groups in tenancy
```

Common Policies

This section includes some common policies you might want to use in your organization. These policies use example group and compartment names. Make sure to replace them with your own names.

Let the help desk manage users

Type of access: Ability to create, update, and delete users and their credentials. It does not include the ability to put users in groups.

Where to create the policy: In the tenancy, because users reside in the tenancy.

```
Allow group HelpDesk to manage users in tenancy
```

Let auditors inspect your resources

Type of access: Ability to list the resources in all compartments. Be aware that:

- The operation to list IAM policies includes the contents of the policies themselves
- The list operations for Networking resource types return all the information (for example, the contents of security lists and route tables)
- The operation to list instances requires the `read` verb instead of `inspect`, and the contents include the user-provided metadata

Where to create the policy: In the tenancy. Because of the concept of policy inheritance, auditors can then inspect both the tenancy and all compartments beneath it. Or you could choose to give auditors access to only specific compartments if they don't need access to the entire tenancy.

```
Allow group Auditors to inspect all-resources in tenancy
Allow group Auditors to read instances in tenancy
```

Let network admins manage a cloud network

Type of access: Ability to manage all components in Networking. This includes cloud networks, subnets, gateways, security lists, route tables, and so on.

Where to create the policy: In the tenancy. Because of the concept of policy inheritance, NetworkAdmins can then manage a cloud network in any compartment. To reduce the scope of access to a particular compartment, specify that compartment instead of the tenancy.

```
Allow group NetworkAdmins to manage virtual-network-family in tenancy
```

Let users launch compute instances

Type of access: Ability to do everything with instances launched into the cloud network and subnets in compartment XYZ, and attach/detach any existing volumes that already exist in compartment ABC. The first statement also lets the group create and manage instance images in compartment ABC.

Where to create the policy: The easiest approach is to put this policy in the tenancy. If you want the admins of the individual compartments (ABC and XYZ) to have control over the individual policy statements for their compartments, these policy statements need to be split across two policies and attached to the compartment they apply to.

```
Allow group InstanceLaunchers to manage instance-family in compartment ABC
Allow group InstanceLaunchers to use volume-family in compartment ABC
Allow group InstanceLaunchers to use virtual-network-family in compartment XYZ
```

Let users manage compute instance configurations, instance pools, and cluster networks

Type of access: Ability to do all things with instance configurations, instance pools, and cluster networks in all compartments.

Where to create the policy: In the tenancy, so that the access is easily granted to all compartments by way of policy inheritance. To reduce the scope of access to just the instance configurations, instance pools, and cluster networks in a particular compartment, specify that compartment instead of the tenancy.

```
Allow group InstancePoolAdmins to manage compute-management-family in tenancy
```

If a group needs to create instance configurations using existing instances as a template, and uses the API or CLI to do this, add the following statements to the policy:

```
Allow group InstancePoolAdmins to read instance-family in tenancy
Allow group InstancePoolAdmins to inspect volumes in tenancy
```

If a particular group needs to start, stop, or reset the instances in existing instance pools, but not create or delete instance pools, use this statement:

```
Allow group InstancePoolUsers to use instance-pools in tenancy
```

If resources used by the instance pool contain default tags, add the following statement to the policy to give the group permission to the tag namespace "oracle-tags":

```
Allow group InstancePoolUsers to use tag-namespaces in tenancy where target.tag-
namespace.name = 'oracle-tags'
```

Let volume admins manage block volumes, backups, and volume groups

Type of access: Ability to do all things with block storage volumes, volume backups, and volume groups in all compartments with the exception of copying volume backups across regions. This makes sense if you want to have a single set of volume admins manage all the volumes, volume backups, and volume groups in all the compartments. The second statement is required in order to attach/detach the volumes from instances.

Where to create the policy: In the tenancy, so that the access is easily granted to all compartments by way of policy inheritance. To reduce the scope of access to just the volumes/backups and instances in a particular compartment, specify that compartment instead of the tenancy.

```
Allow group VolumeAdmins to manage volume-family in tenancy
Allow group VolumeAdmins to use instance-family in tenancy
```

Let volume backup admins manage only backups

Type of access: Ability to do all things with volume backups, but not create and manage volumes themselves. This makes sense if you want to have a single set of volume backup admins manage all the volume backups in all the compartments. The first statement gives the required access to the volume that is being backed up; the second statement enables creation of the backup and the ability to delete backups. The third statement enables the creation and management of user defined backup policies; the fourth statement enables assignment and removal of assignment of backup policies.

Where to create the policy: In the tenancy, so that the access is easily granted to all compartments by way of policy inheritance. To reduce the scope of access to just the volumes and backups in a particular compartment, specify that compartment instead of the tenancy.

```
Allow group VolumeBackupAdmins to use volumes in tenancy
Allow group VolumeBackupAdmins to manage volume-backups in tenancy
Allow group VolumeBackupAdmins to manage backup-policies in tenancy
Allow group VolumeBackupAdmins to manage backup-policy-assignments in tenancy
```

If the group uses the Compute Web UI, extend the policy as shown below for a better user experience.

```
Allow group VolumeBackupAdmins to use volumes in tenancy
Allow group VolumeBackupAdmins to manage volume-backups in tenancy
Allow group VolumeBackupAdmins to inspect volume-attachments in tenancy
Allow group VolumeBackupAdmins to inspect instances in tenancy
Allow group VolumeBackupAdmins to manage backup-policies in tenancy
Allow group VolumeBackupAdmins to manage backup-policy-assignments in tenancy
```

The last two statements are not strictly required. They enable the display of all information about a particular volume and available backup policies.

Let boot volume backup admins manage only backups

Type of access: Ability to do all things with boot volume backups, but not create and manage boot volumes themselves. This makes sense if you want to have a single set of boot volume backup admins manage all the boot volume backups in all the compartments. The first statement gives the required access to the boot volume that is being backed up; the second statement enables creation of the backup and the ability to delete backups. The third statement enables the creation and management of user defined backup policies; the fourth statement enables assignment and removal of assignment of backup policies.

Where to create the policy: In the tenancy, so that the access is easily granted to all compartments by way of policy inheritance. To reduce the scope of access to just the boot volumes and backups in a particular compartment, specify that compartment instead of the tenancy.

```
Allow group BootVolumeBackupAdmins to use volumes in tenancy
Allow group BootVolumeBackupAdmins to manage boot-volume-backups in tenancy
Allow group BootVolumeBackupAdmins to manage backup-policies in tenancy
Allow group BootVolumeBackupAdmins to manage backup-policy-assignments in tenancy
```

If the group uses the Compute Web UI, extend the policy as shown below for a better user experience.

```
Allow group BootVolumeBackupAdmins to use volumes in tenancy
Allow group BootVolumeBackupAdmins to manage boot-volume-backups in tenancy
Allow group BootVolumeBackupAdmins to inspect instances in tenancy
Allow group BootVolumeBackupAdmins to manage backup-policies in tenancy
Allow group BootVolumeBackupAdmins to manage backup-policy-assignments in tenancy
```

The last two statements are not strictly required. They enable the display of all information about a particular volume and available backup policies.

Let users create a volume group

Type of access: Ability to create a volume group from a set of volumes.

Where to create the policy: In the tenancy, so that the access is easily granted to all compartments by way of policy inheritance. To reduce the scope of access to just the volumes and volume groups in a particular compartment, specify that compartment instead of the tenancy.

```
Allow group VolumeGroupCreators to inspect volumes in tenancy
Allow group VolumeGroupCreators to manage volume-groups in tenancy
```

Let users clone a volume group

Type of access: Ability to clone a volume group from an existing volume group.

Where to create the policy: In the tenancy, so that the access is easily granted to all compartments by way of policy inheritance. To reduce the scope of access to just the volumes and volume groups in a particular compartment, specify that compartment instead of the tenancy.

```
Allow group VolumeGroupCloners to inspect volumes in tenancy
Allow group VolumeGroupCloners to manage volume-groups in tenancy
Allow group VolumeGroupCloners to manage volumes in tenancy
```

Let users create a volume group backup

Type of access: Ability to create a volume group backup.

Where to create the policy: In the tenancy, so that the access is easily granted to all compartments by way of policy inheritance. To reduce the scope of access to just the volumes/backups and volume groups/volume group backups in a particular compartment, specify that compartment instead of the tenancy.

```
Allow group VolumeGroupBackupAdmins to inspect volume-groups in tenancy
Allow group VolumeGroupBackupAdmins to manage volumes in tenancy
Allow group VolumeGroupBackupAdmins to manage volume-group-backups in tenancy
Allow group VolumeGroupBackupAdmins to manage volume-backups in tenancy
```

Let users restore a volume group backup

Type of access: Ability to create a volume group by restoring a volume group backup.

Where to create the policy: In the tenancy, so that the access is easily granted to all compartments by way of policy inheritance. To reduce the scope of access to just the volumes/backups and volume groups/volume group backups in a particular compartment, specify that compartment instead of the tenancy.

```
Allow group VolumeGroupBackupAdmins to inspect volume-group-backups in tenancy
Allow group VolumeGroupBackupAdmins to read volume-backups in tenancy
Allow group VolumeGroupBackupAdmins to manage volume-groups in tenancy
Allow group VolumeGroupBackupAdmins to manage volumes in tenancy
```

Let users create, manage, and delete file systems

Type of access: Ability to create, manage, or delete a file system. Administrative functions for a file system include the ability to rename or delete it or disconnect from it.

Where to create the policy: In the tenancy, so that the ability to create, manage, or delete a file system is easily granted to all compartments by way of policy inheritance. To reduce the scope of these administrative functions to file systems in a particular compartment, specify that compartment instead of the tenancy.

```
Allow group StorageAdmins to manage file-family in tenancy
```

Let users create file systems

Type of access: Ability to create a file system.

Where to create the policy: In the tenancy, so that the ability to create a file system is easily granted to all compartments by way of policy inheritance. To reduce the scope of these administrative functions to file systems in a particular compartment, specify that compartment instead of the tenancy.

```
Allow group Managers to manage file-systems in tenancy
Allow group Managers to read mount-targets in tenancy
```

The second statement is required when users create a file system through the Compute Web UI. It enables the UI to display a list of mount targets that the new file system can be associated with.

Let object storage admins manage buckets and objects

Type of access: Ability to do all things with Object Storage buckets and objects in all compartments.

Where to create the policy: In the tenancy, so that the access is easily granted to all compartments by way of policy inheritance. To reduce the scope of access to just the buckets and objects in a particular compartment, specify that compartment instead of the tenancy.

```
Allow group ObjectAdmins to manage buckets in tenancy
Allow group ObjectAdmins to manage objects in tenancy
```

Let users write objects to object storage buckets

Type of access: Ability to write objects to any object storage bucket in compartment ABC. Consider a situation where a client needs to regularly write log files to a bucket. This includes the ability to list the buckets in the compartment, list the objects in a bucket, and create a new object in a bucket. Although the second statement gives broad access with the manage verb, that access is then scoped down to only the OBJECT_INSPECT and OBJECT_CREATE permissions with the condition at the end of the statement.

Where to create the policy: The easiest approach is to put this policy in the tenancy. If you want the admins of compartment ABC to have control over the policy, it needs to be attached to that compartment.

```
Allow group ObjectWriters to read buckets in compartment ABC
Allow group ObjectWriters to manage objects in compartment ABC where any
{request.permission='OBJECT_CREATE', request.permission='OBJECT_INSPECT'}
```

To limit access to a specific bucket in a particular compartment, add the condition `where target.bucket.name='<bucket_name>'`. The following policy allows the user to list all the buckets in a particular compartment, but they can only list the objects in and upload objects to BucketA:

```
Allow group ObjectWriters to read buckets in compartment ABC
Allow group ObjectWriters to manage objects in compartment ABC
  where all {target.bucket.name='BucketA', any {request.permission='OBJECT_CREATE',
request.permission='OBJECT_INSPECT'}}
```

Let users download objects from object storage buckets

Type of access: Ability to download objects from any Object Storage bucket in compartment ABC. This consists of the ability to list the buckets in the compartment, list the objects in a bucket, and read existing objects in a bucket.

Where to create the policy: The easiest approach is to put this policy in the tenancy. If you want the admins of compartment ABC to have control over the policy, it needs to be attached to that compartment.

```
Allow group ObjectReaders to read buckets in compartment ABC
Allow group ObjectReaders to read objects in compartment ABC
```

To limit access to a specific bucket in a particular compartment, add the condition `where target.bucket.name='<bucket_name>'`. The following policy allows the user to list all buckets in a particular compartment, but they can only read the objects in and download from BucketA:

```
Allow group ObjectReaders to read buckets in compartment ABC
Allow group ObjectReaders to read objects in compartment ABC where
target.bucket.name='BucketA'
```

Let users manage their own credentials

No policy is required to let users manage their own credentials. All users have the ability to change and reset their own passwords and manage their own API keys.

Let a compartment admin manage the compartment

Type of access: Ability to manage all aspects of a particular compartment. For example, a group called A-Admins could manage all aspects of a compartment called Project-A, including writing additional policies that affect the compartment.

Where to create the policy: In the tenancy.

```
Allow group A-Admins to manage all-resources in compartment Project-A
```

Advanced Policy Features

This section describes policy language features that let you grant more granular access.

Conditions

As part of a policy statement, you can specify one or more conditions that must be met in order for access to be granted. Each condition consists of one or more predefined variables that you specify values for in the policy statement. When someone requests access to the resource type in question, and the condition in the policy is met, it evaluates to *true* and the request is allowed.

There are two types of variables: those that are relevant to the request itself, and those relevant to the resource being acted upon in the request, also known as the target. The name of the variable is prefixed accordingly with either `request` or `target` followed by a period. For example, the request variable called `request.operation` represents the API operation being requested. This variable lets you write a broad policy statement, but add a condition based on the specific API operation.

> **Caution:**
>
> Condition matching is case insensitive. This is important to remember when writing conditions for resource types that allow case-sensitive naming. For example, the Object Storage service allows you to create both a bucket named "BucketA" and a bucket named "bucketA" in the same compartment. If you write a condition that specifies "BucketA", it will also apply to "bucketA", because the condition matching is case insensitive.

Non-Applicable Variables

As a general rule, if a variable is not applicable to the incoming request, the condition evaluates to false and the request is declined. This means that a request normally allowed by the combination of verb and resource type in a policy statement, is declined because it does not specify a value for the condition variable. If you want to grant the access associated with the policy statement without the condition, you need to include an additional statement.

For example, the policy statements below allow someone to add and remove users from any group, as long as they are not members of the Administrators group.

```
Allow group GroupAdmins to use users in tenancy where target.group.name !=
'Administrators'
Allow group GroupAdmins to use groups in tenancy where target.group.name !=
'Administrators'
```

If a user in GroupAdmins calls a general API operation such as `ListUsers` or `UpdateUser`, the request is declined even though the operations are covered by `use users`. This is because the `list` and `update` commands do not involve specifying a group, which means there is no value to match the `target.group.name` variable in the condition of the policy statement. The variable is not applicable to the incoming request, therefore the condition evaluates to *false* and the request is declined.

To allow the GroupAdmins to list users, you need to add another policy statement, but without the condition. In this example, the verb `inspect` is required to allow the `list` command.

```
Allow group GroupAdmins to use users in tenancy where target.group.name !=
'Administrators'
Allow group GroupAdmins to use groups in tenancy where target.group.name !=
'Administrators'
Allow group GroupAdmins to inspect users in tenancy
```

This general concept also applies to groups, and any other resource type with target variables.

Tag-Based Access Control

Using conditions and a set of tag variables, you can write policy to scope access based on the tags that have been applied to a resource. More specifically, access can be controlled based on the value of a tag that exists on the group to which the requesting user belongs. Tag-based access control provides additional flexibility to your policies by allowing you to define access that spans compartments, groups, and resources.

For details about how to write policies to scope access by tags, refer to the section "Tag-Based Access Control" in the chapter Tagging Overview.

Permissions

Permissions are the atomic units of authorization that control a user's ability to perform operations on resources. All the permissions are defined in the policy language. When you write a policy giving a group access to a particular verb and resource type, you are actually giving that group access to one or more predefined permissions. The purpose of verbs is to simplify the process of granting multiple related permissions that cover a broad set of access or a particular operational scenario.

Relation to Verbs

To understand the relationship between permissions and verbs, consider the following example. A policy statement that allows a group to `inspect volumes` actually provides access to a permission called VOLUME_INSPECT. Permissions are always written with all capital letters and underscores. In general, that permission enables the user to get information about block volumes.

As you go from `inspect` > `read` > `use` > `manage`, the level of access generally increases, and the permissions granted are cumulative, as shown in the table below. Note that in this case no additional permissions are granted going from `inspect` to `read`.

Inspect Volumes	Read Volumes	Use Volumes	Manage Volumes
VOLUME_INSPECT	VOLUME_INSPECT	VOLUME_INSPECT VOLUME_UPDATE VOLUME_WRITE	VOLUME_INSPECT VOLUME_UPDATE VOLUME_WRITE VOLUME_CREATE VOLUME_DELETE

For detailed information about permissions covered by each verb for each given resource type, see the Policy Reference.

Relation to API Operations

Each API operation requires the caller to have access to one or more permissions. For example:

- To use either `ListVolumes` or `GetVolume`, you must have access to a single permission: VOLUME_INSPECT.
- To attach a volume to an instance, you must have access to multiple permissions, related to different resource types: volumes, volume-attachments and instances. Those permissions are, respectively: VOLUME_WRITE, VOLUME_ATTACHMENT_CREATE , and INSTANCE_ATTACH_VOLUME.

The Policy Reference lists which permissions are required for each API operation.

Understanding a User's Access

The policy language is designed to let you write simple statements involving only verbs and resource types, without having to state the desired permissions in the statement. However, there may be situations where a security team member or auditor wants to understand the specific permissions a particular user has. The Policy Reference lists the permissions associated with each verb. You can look at the groups the user is in and the policies applicable to those groups, and from there compile a list of the permissions granted.

However, having a list of the permissions is not the complete picture. Conditions in a policy statement can scope a user's access beyond individual permissions. Also, each policy statement specifies a particular compartment and can have conditions that further scope the access to only certain resources in that compartment.

Scoping Access with Permissions or API Operations

In a policy statement, you can use conditions combined with permissions or API operations to reduce the scope of access granted by a particular verb. For example, you want group XYZ to be able to list, get, create, or update groups, but not delete them. To list, get, create, and update groups, you need a policy with `manage groups` as the verb and resource type, but this would include the permission to delete groups.

To restrict access to only the desired permissions, you could add a condition that explicitly states the permissions you want to allow:

```
Allow group XYZ to manage groups in tenancy
where any {request.permission='GROUP_INSPECT',
           request.permission='GROUP_CREATE',
           request.permission='GROUP_UPDATE'}
```

An alternative would be a policy that allows all permissions except GROUP_DELETE:

```
Allow group XYZ to manage groups in tenancy where request.permission != 'GROUP_DELETE'
```

However, with this approach, any future new permissions would automatically be granted to group XYZ. Only GROUP_DELETE would be omitted.

Another alternative would be to write a condition based on the specific API operations:

```
Allow group XYZ to manage groups in tenancy
where any {request.operation='ListGroups',
           request.operation='GetGroup',
           request.operation='CreateGroup',
           request.operation='UpdateGroup'}
```

It can be beneficial to use permissions instead of API operations in conditions. In the future, if a new API operation is added that requires one of the permissions listed in the permissions-based policy above, that policy already controls the XYZ group's access to that new API operation.

A user's access to a permission can be scoped even further by also specifying a condition based on API operation. For example, you could give a user access to GROUP_INSPECT, but then only to `ListGroups`.

```
Allow group XYZ to manage groups in tenancy
where all {request.permission='GROUP_INSPECT',request.operation='ListGroups'}
```

Cross-Tenancy Policies

Before You Begin

You can write policies to allow tenancy access from other tenancies so you can share resources across tenancies. The administrators of both tenancies need to create special policy statements that explicitly state which resources can be accessed and shared. These special statements use the following special verbs:

Verb	Use in a Policy Statement
`endorse`	Describes what work a group in a *source tenancy* can perform in other tenancies. You write the `endorse` statement for the tenancy that contains the group of users who need to work with another tenancy's resources.
`admit`	Describes what work a group from other tenancies can perform in a *destination tenancy*. You write the `admit` statement for the tenancy that is granting permission to access its resources. The `admit` statement identifies the group of users from the source tenancy that requires resource access in the destination tenancy.
`define`	Assigns an alias for a *source tenancy* OCID, a *source group* OCID, and a *destination tenancy* OCID. You define a *source tenancy* alias and a *source group* alias for use in `admit` policy statements. You define a *destination tenancy* alias for use in `endorse` policy statements. You must include a `define` statement in the same policy entity as the `endorse` or `admit` statement.

The `endorse` and `admit` statements work together. An `endorse` statement resides in the source tenancy while an `admit` statement resides in the destination tenancy. Without a corresponding statement that specifies access, a particular `endorse` or `admit` statement grants no access. *Both tenancies must agree on access and have policies that allow for access.*

In the source tenancy, you write `define` and `endorse` policy statements using the following syntax:

```
define tenancy destination-tenancy-alias as tenancy_ocid

endorse group group-name to verb resource in tenancy destination-tenancy-alias
```

In the destination tenancy, you write two `define` policy statemens and an `admit` policy statement using the following syntax:

```
define tenancy source-tenancy-alias as tenancy_ocid

define group source-group-alias as group_ocid

admit group source-group-alias of tenancy source-tenancy-alias to verb resource
in compartment/tenancy
```

For more information and examples of common statements, see "Writing Policies to Access Resources Across Tenancies" in the Identity and Access Management in the *Oracle Private Cloud Appliance User Guide*.

Policy Reference

Use this section as a source of information to help you write policies for access control in your tenancy. The table provides reference information as follows:

- It lists all resource types for which policy statements can be written.
- For each resource type, it lists the API operations that can be allowed or denied through policy statements.
- For each API operation, it lists the required permissions and the associated verb/resource combination to be used in policy statements.

> **Note:**
>
> For some API operations the table displays no permission or verb/resource combination. These empty cells indicate that either no explicit permission is required for the operation, or the operation is dependent on other API operations and the permissions associated with those.
>
> The IAM service is only aware of permissions directly associated with an API operation; it is not aware of further permission dependencies or conditions defined by other services for their specific resources.
>
> The table may contain resource types and API operations that are not yet supported by the services available in your tenancy. Those rows can be ignored.

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
users	CreateUser	USER_CREATE	manage users
	CreateOrResetUIPassword	USER_UIPASS_SET	manage users
	GetUser	USER_INSPECT	inspect users
	ListUsers	USER_INSPECT	inspect users
	ListApiKeys	USER_READ	read users
	UpdateUser	USER_UPDATE	use users
	UpdateUserState	USER_UNBLOCK	manage users
	UploadApiKey	USER_APIKEY_ADD	manage users
	DeleteUser	USER_DELETE	manage users
	DeleteApiKey	USER_APIKEY_REMOVE	manage users
	AddUserToGroup	USER_UPDATE	use users
	RemoveUserFromGroup	USER_UPDATE	use users
	GetUserGroupMembership	USER_INSPECT	inspect users
	ListUserGroupMemberships	USER_INSPECT	inspect users
groups	CreateGroup	GROUP_CREATE	manage groups
	GetGroup	GROUP_INSPECT	inspect groups
	ListGroups	GROUP_INSPECT	inspect groups
	UpdateGroup	GROUP_UPDATE	use groups
	DeleteGroup	GROUP_DELETE	manage groups
	AddUserToGroup	GROUP_UPDATE	use groups
	RemoveUserFromGroup	GROUP_UPDATE	use groups
	GetUserGroupMembership	GROUP_INSPECT	inspect groups
	ListUserGroupMemberships	GROUP_INSPECT	inspect groups
	ListIdpGroupMappings	GROUP_INSPECT	inspect groups
	CreateIdpGroupMapping	GROUP_UPDATE	use groups
	GetIdpGroupMapping	GROUP_INSPECT	inspect groups
	UpdateIdpGroupMapping	GROUP_UPDATE	use groups
	DeleteIdpGroupMapping	GROUP_UPDATE	use groups

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
compartments	ListCompartments	COMPARTMENT_INSPECT	inspect compartments
	GetCompartment	COMPARTMENT_INSPECT	inspect compartments
	ListAvailabilityDomains	COMPARTMENT_INSPECT	inspect compartments
	ListFaultDomains	COMPARTMENT_INSPECT	inspect compartments
	UpdateCompartment	COMPARTMENT_UPDATE	use compartments
	CreateCompartment	COMPARTMENT_CREATE	manage compartments
	DeleteCompartment	COMPARTMENT_DELETE	manage compartments
	RecoverCompartment	COMPARTMENT_RECOVER	manage compartments
	MoveCompartment	MANAGE_ALL_RESOURCES	manage all-resources
policies	ListPolicies	POLICY_READ	inspect policies
	GetPolicy	POLICY_READ	inspect policies
	UpdatePolicy	POLICY_UPDATE	manage policies
	CreatePolicy	POLICY_CREATE	manage policies
	DeletePolicy	POLICY_DELETE	manage policies
tag-defaults	ListTagDefaults	TAG_DEFAULT_INSPECT	inspect tag-defaults
	GetTagDefault	TAG_DEFAULT_INSPECT	inspect tag-defaults
	AssembleEffectiveTagSet	TAG_DEFAULT_INSPECT	inspect tag-defaults
	CreateTagDefault	TAG_DEFAULT_CREATE	manage tag-defaults
	UpdateTagDefault	TAG_DEFAULT_UPDATE	manage tag-defaults
	DeleteTagDefault	TAG_DEFAULT_DELETE	manage tag-defaults
tag-namespaces	ListTagNamespaces	TAG_NAMESPACE_INSPECT	inspect tag-namespaces
	GetTagNamespace	TAG_NAMESPACE_INSPECT	inspect tag-namespaces
	ListTags	TAG_NAMESPACE_INSPECT	inspect tag-namespaces
	ListCostTrackingTags	TAG_NAMESPACE_INSPECT	inspect tag-namespaces
	GetTag	TAG_NAMESPACE_INSPECT	inspect tag-namespaces
	GetTaggingWorkRequest	TAG_NAMESPACE_INSPECT	inspect tag-namespaces

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	ListTaggingWorkRequests	TAG_NAMESPACE_INSPECT	inspect tag-namespaces
	ListTaggingWorkRequestErrors	TAG_NAMESPACE_INSPECT	inspect tag-namespaces
	ListTaggingWorkRequestLog	TAG_NAMESPACE_INSPECT	inspect tag-namespaces
	CreateTag	TAG_NAMESPACE_USE	use tag-namespaces
	UpdateTag	TAG_NAMESPACE_USE	use tag-namespaces
	UpdateTagNamespace	TAG_NAMESPACE_UPDATE	manage tag-namespaces
	CreateTagNamespace	TAG_NAMESPACE_CREATE	manage tag-namespaces
	ChangeTagNamespaceCompartment	TAG_NAMESPACE_MOVE	manage tag-namespaces
	DeleteTagNamespace	TAG_NAMESPACE_DELETE	manage tag-namespaces
	DeleteTag	TAG_NAMESPACE_DELETE	manage tag-namespaces
tenancies	ListRegionSubscriptions	TENANCY_INSPECT	inspect tenancies
	GetTenancy	TENANCY_INSPECT	inspect tenancies
	ListRegions	TENANCY_INSPECT	inspect tenancies
	CreateRegionSubscription	TENANCY_UPDATE	use tenancies
identity-providers	ListIdentityProviders	IDENTITY_PROVIDER_INSPECT	inspect identity-providers
	GetIdentityProvider	IDENTITY_PROVIDER_INSPECT	inspect identity-providers
	UpdateIdentityProvider	IDENTITY_PROVIDER_UPDATE	manage identity-providers
	CreateIdentityProvider	IDENTITY_PROVIDER_CREATE	manage identity-providers
	DeleteIdentityProvider	IDENTITY_PROVIDER_DELETE	manage identity-providers
	ListIdpGroupMappings	IDENTITY_PROVIDER_INSPECT	inspect identity-providers
	CreateIdpGroupMapping	IDENTITY_PROVIDER_UPDATE	manage identity-providers
	GetIdpGroupMapping	IDENTITY_PROVIDER_INSPECT	inspect identity-providers
	UpdateIdpGroupMapping	IDENTITY_PROVIDER_UPDATE	manage identity-providers
	DeleteIdpGroupMapping	IDENTITY_PROVIDER_UPDATE	manage identity-providers

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
work-requests	ListWorkRequests	WORKREQUEST_INSPECT	inspect work-requests
	GetWorkRequest	WORKREQUEST_INSPECT	inspect work-requests
	ListWorkRequestErrors	WORKREQUEST_INSPECT	inspect work-requests
	ListWorkRequestLogs	WORKREQUEST_INSPECT	inspect work-requests
instances	ListInstances	INSTANCE_READ	read instances
	GetInstance	INSTANCE_READ	read instances
	UpdateInstance	INSTANCE_UPDATE	use instances
	InstanceAction	INSTANCE_POWER_ACTIONS	use instances
	AttachVolume	INSTANCE_ATTACH_VOLUME	use instances
	DetachVolume	INSTANCE_DETACH_VOLUME	use instances
	ChangeInstanceCompartment	INSTANCE_MOVE	manage instances
	LaunchInstance	INSTANCE_CREATE	manage instances
	TerminateInstance	INSTANCE_DELETE	manage instances
	AttachVnic	INSTANCE_ATTACH_SECONDARY_VNIC	manage instances
	DetachVnic	INSTANCE_DETACH_SECONDARY_VNIC	manage instances
	ListVnicAttachments	INSTANCE_INSPECT	inspect instances
	ListShapes	INSTANCE_INSPECT	inspect instances
	CreateImage	INSTANCE_CREATE_IMAGE	use instances
	ListInstanceConsoleConnections	INSTANCE_INSPECT	inspect instances
		INSTANCE_READ	read instances
	GetInstanceConsoleConnection	INSTANCE_READ	read instances
	CreateInstanceConsoleConnection	INSTANCE_READ	read instances
	ListVolumeAttachments	INSTANCE_INSPECT	inspect instances
	ListBootVolumeAttachments	INSTANCE_INSPECT	inspect instances
	GetVolumeAttachment	INSTANCE_INSPECT	inspect instances
	GetBootVolumeAttachment	INSTANCE_INSPECT	inspect instances
	CreateInstancePool	INSTANCE_CREATE	manage instances

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	TerminateInstancePool	INSTANCE_DELETE	manage instances
	ListConsoleHistories	INSTANCE_INSPECT	inspect instances
	CreateInstanceConfiguration	INSTANCE_READ	read instances
console-histories	ListConsoleHistories	CONSOLE_HISTORY_INSPECT	inspect console-histories
	GetConsoleHistory	CONSOLE_HISTORY_INSPECT	inspect console-histories
	ShowConsoleHistoryData	CONSOLE_HISTORY_READ	read console-histories
	DeleteConsoleHistory	CONSOLE_HISTORY_DELETE	manage console-histories
	CaptureConsoleHistory	CONSOLE_HISTORY_CREATE	manage console-histories
instance-console-connection	ListInstanceConsoleConnections	INSTANCE_CONSOLE_CONNECTION_INSPECT	inspect instance-console-connection
	GetInstanceConsoleConnection	INSTANCE_CONSOLE_CONNECTION_READ	read instance-console-connection
	DeleteInstanceConsoleConnection	INSTANCE_CONSOLE_CONNECTION_DELETE	manage instance-console-connection
	CreateInstanceConsoleConnection	INSTANCE_CONSOLE_CONNECTION_CREATE	manage instance-console-connection
	UpdateInstanceConsoleConnection	INSTANCE_CONSOLE_CONNECTION_CREATE	manage instance-console-connection
		INSTANCE_CONSOLE_CONNECTION_DELETE	manage instance-console-connection
instance-images	ListImages	INSTANCE_IMAGE_READ	read instance-images
	GetImage	INSTANCE_IMAGE_READ	read instance-images
	LaunchInstance	INSTANCE_IMAGE_READ	read instance-images
	UpdateImage	INSTANCE_IMAGE_UPDATE	use instance-images
	DeleteImage	INSTANCE_IMAGE_DELETE	manage instance-images
	ChangeImageCompartment	INSTANCE_IMAGE_MOVE	manage instance-images
	CreateImage	INSTANCE_IMAGE_CREATE	manage instance-images
	CreateInstancePool	INSTANCE_IMAGE_READ	read instance-images
	ExportImage		
app-catalog-listing	ListAppCatalogSubscriptions	APP_CATALOG_LISTING_INSPECT	inspect app-catalog-listing

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	CreateAppCatalogSubscription	APP_CATALOG_LISTING_SUBSCRIBE	manage app-catalog-listing
	DeleteAppCatalogSubscription	APP_CATALOG_LISTING_SUBSCRIBE	manage app-catalog-listing
volume-attachments-partial	AttachVolume	VOLUME_ATTACHMENT_CREATE	manage volume-attachments-partial
	DetachVolume	VOLUME_ATTACHMENT_DELETE	manage volume-attachments-partial
instance-configurations	ListInstanceConfigurations	INSTANCE_CONFIGURATION_INSPECT	inspect instance-configurations
	GetInstanceConfiguration	INSTANCE_CONFIGURATION_READ	read instance-configurations
	CreateInstanceConfiguration	INSTANCE_CONFIGURATION_CREATE	manage instance-configurations
	UpdateInstanceConfiguration	INSTANCE_CONFIGURATION_UPDATE	manage instance-configurations
	LaunchInstanceConfiguration	INSTANCE_CONFIGURATION_LAUNCH	manage instance-configurations
	DeleteInstanceConfiguration	INSTANCE_CONFIGURATION_DELETE	manage instance-configurations
	ChangeInstanceConfigurationCompartment	INSTANCE_CONFIGURATION_MOVE	manage instance-configurations
instance-pools	ListInstancePools	INSTANCE_POOL_INSPECT	inspect instance-pools
	GetInstancePool	INSTANCE_POOL_READ	read instance-pools
	ListInstancePoolInstances	INSTANCE_POOL_READ	read instance-pools
	ResetInstancePool	INSTANCE_POOL_POWER_ACTIONS	use instance-pools
	SoftresetInstancePool	INSTANCE_POOL_POWER_ACTIONS	use instance-pools
	StartInstancePool	INSTANCE_POOL_POWER_ACTIONS	use instance-pools
	StopInstancePool	INSTANCE_POOL_POWER_ACTIONS	use instance-pools
	UpdateInstancePool	INSTANCE_POOL_UPDATE	manage instance-pools
	ChangeInstancePoolCompartment	INSTANCE_POOL_MOVE	manage instance-pools
	CreateInstancePool	INSTANCE_POOL_CREATE	manage instance-pools
	TerminateInstancePool	INSTANCE_POOL_DELETE	manage instance-pools

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
auto-scaling-configurations	ListAutoScalingConfigurations	AUTO_SCALING_CONFIGURATION_INSPECT	inspect auto-scaling-configurations
	ListAutoScalingPolicies	AUTO_SCALING_CONFIGURATION_INSPECT	inspect auto-scaling-configurations
	GetAutoScalingConfiguration	AUTO_SCALING_CONFIGURATION_READ	read auto-scaling-configurations
	GetAutoScalingPolicy	AUTO_SCALING_CONFIGURATION_READ	read auto-scaling-configurations
	ChangeAutoScalingConfigurationCompartment	AUTO_SCALING_CONFIGURATION_MOVE	manage auto-scaling-configurations
	CreateAutoScalingConfiguration	AUTO_SCALING_CONFIGURATION_CREATE	manage auto-scaling-configurations
	UpdateAutoScalingConfiguration	AUTO_SCALING_CONFIGURATION_UPDATE	manage auto-scaling-configurations
	DeleteAutoScalingConfiguration	AUTO_SCALING_CONFIGURATION_DELETE	manage auto-scaling-configurations
	CreateAutoScalingPolicy	AUTO_SCALING_CONFIGURATION_CREATE	manage auto-scaling-configurations
	UpdateAutoScalingPolicy	AUTO_SCALING_CONFIGURATION_UPDATE	manage auto-scaling-configurations
	DeleteAutoScalingPolicy	AUTO_SCALING_CONFIGURATION_DELETE	manage auto-scaling-configurations
dedicated-vm-hosts	ListDedicatedVmHosts	DEDICATED_VM_HOST_INSPECT	inspect dedicated-vm-hosts
	GetDedicatedVmHost	DEDICATED_VM_HOST_READ	read dedicated-vm-hosts
	ListDedicatedVmHostInstances	DEDICATED_VM_HOST_READ	read dedicated-vm-hosts
	UpdateDedicatedVmHost	DEDICATED_VM_HOST_UPDATE	use dedicated-vm-hosts
	CreateDedicatedVmHost	DEDICATED_VM_HOST_CREATE	manage dedicated-vm-hosts
	DeleteDedicatedVmHost	DEDICATED_VM_HOST_DELETE	manage dedicated-vm-hosts
	ChangeDedicatedVmHostCompartment	DEDICATED_VM_HOST_MOVE	manage dedicated-vm-hosts
vcns	ListVcns	VCN_READ	inspect vcns
	GetVcn	VCN_READ	inspect vcns
	CreateVcn	VCN_CREATE	manage vcns
	UpdateVcn	VCN_UPDATE	manage vcns
	DeleteVcn	VCN_DELETE	manage vcns

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	ChangeVcnCompartment	VCN_MOVE	manage vcns
	CreateDhcpOptions	VCN_ATTACH	manage vcns
	DeleteDhcpOptions	VCN_DETACH	manage vcns
	CreateInternetGateway	VCN_ATTACH	manage vcns
	DeleteInternetGateway	VCN_DETACH	manage vcns
	CreateLocalPeeringGateway	VCN_ATTACH	manage vcns
	DeleteLocalPeeringGateway	VCN_DETACH	manage vcns
	CreateNatGateway	VCN_READ	inspect vcns
		VCN_ATTACH	manage vcns
	DeleteNatGateway	VCN_READ	inspect vcns
		VCN_DETACH	manage vcns
	CreateNetworkSecurityGroup	VCN_ATTACH	manage vcns
	DeleteNetworkSecurityGroup	VCN_DETACH	manage vcns
	DeleteSubnet	VCN_DETACH	manage vcns
	CreateSubnet	VCN_ATTACH	manage vcns
	CreateServiceGateway	VCN_READ	inspect vcns
		VCN_ATTACH	manage vcns
	DeleteServiceGateway	VCN_READ	inspect vcns
		VCN_DETACH	manage vcns
	CreateRouteTable	VCN_ATTACH	manage vcns
	DeleteRouteTable	VCN_DETACH	manage vcns
	UpdateRouteTable	VCN_ATTACH	manage vcns
		VCN_DETACH	manage vcns
	CreateDrgAttachment	VCN_ATTACH	manage vcns
	DeleteDrgAttachment	VCN_DETACH	manage vcns
subnets	ListSubnets	SUBNET_READ	inspect subnets
	GetSubnet	SUBNET_READ	inspect subnets
	ChangeSubnetCompartment	SUBNET_MOVE	manage subnets
	CreateSubnet	SUBNET_CREATE	manage subnets

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	DeleteSubnet	SUBNET_DELETE	manage subnets
	UpdateSubnet	SUBNET_UPDATE	manage subnets
	LaunchInstance	SUBNET_ATTACH	use subnets
	TerminateInstance	SUBNET_DETACH	use subnets
	AttachVnic	SUBNET_ATTACH	use subnets
	DetachVnic	SUBNET_DETACH	use subnets
	CreateInstancePool	SUBNET_ATTACH	use subnets
	TerminateInstancePool	SUBNET_DETACH	use subnets
	CreatePrivateIp	SUBNET_ATTACH	use subnets
	CreateMountTarget	SUBNET_ATTACH	use subnets
	DeleteMountTarget	SUBNET_DETACH	use subnets
route-tables	ListRouteTables	ROUTE_TABLE_READ	inspect route-tables
	GetRouteTable	ROUTE_TABLE_READ	inspect route-tables
	ChangeRouteTableCompartment	ROUTE_TABLE_MOVE	manage route-tables
	CreateRouteTable	ROUTE_TABLE_CREATE	manage route-tables
	DeleteRouteTable	ROUTE_TABLE_DELETE	manage route-tables
	UpdateRouteTable	ROUTE_TABLE_UPDATE	manage route-tables
	CreateDrgAttachment	ROUTE_TABLE_ATTACH	manage route-tables
	UpdateDrgAttachment	ROUTE_TABLE_ATTACH	manage route-tables
	CreateLocalPeeringGateway	ROUTE_TABLE_ATTACH	manage route-tables
	UpdateLocalPeeringGateway	ROUTE_TABLE_ATTACH	manage route-tables
	DeleteSubnet	ROUTE_TABLE_DETACH	manage route-tables
	CreateSubnet	ROUTE_TABLE_ATTACH	manage route-tables
	UpdateSubnet	ROUTE_TABLE_ATTACH	manage route-tables
		ROUTE_TABLE_DETACH	manage route-tables
	CreateServiceGateway	ROUTE_TABLE_ATTACH	manage route-tables
	UpdateServiceGateway	ROUTE_TABLE_ATTACH	manage route-tables

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
network-security-groups	CreateNetworkSecurityGroup	NETWORK_SECURITY_GROUP_CREATE	manage network-security-groups
	GetNetworkSecurityGroup	NETWORK_SECURITY_GROUP_INSPECT	inspect network-security-groups
	ListNetworkSecurityGroups	NETWORK_SECURITY_GROUP_INSPECT	inspect network-security-groups
	UpdateNetworkSecurityGroup	NETWORK_SECURITY_GROUP_UPDATE	manage network-security-groups
	DeleteNetworkSecurityGroup	NETWORK_SECURITY_GROUP_DELETE	manage network-security-groups
	ListNetworkSecurityGroupVnics	NETWORK_SECURITY_GROUP_LIST_MEMBERS	use network-security-groups
	ChangeNetworkSecurityGroupCompartment	NETWORK_SECURITY_GROUP_MOVE	manage network-security-groups
	ListNetworkSecurityGroupSecurityRules	NETWORK_SECURITY_GROUP_LIST_SECURITY_RULES	use network-security-groups
	AddNetworkSecurityGroupSecurityRules	NETWORK_SECURITY_GROUP_UPDATE_SECURITY_RULES	manage network-security-groups
	UpdateNetworkSecurityGroupSecurityRules	NETWORK_SECURITY_GROUP_UPDATE_SECURITY_RULES	manage network-security-groups
	RemoveNetworkSecurityGroupSecurityRules	NETWORK_SECURITY_GROUP_UPDATE_SECURITY_RULES	manage network-security-groups
	LaunchInstance	NETWORK_SECURITY_GROUP_UPDATE_MEMBERS	use network-security-groups
	AttachVnic	NETWORK_SECURITY_GROUP_UPDATE_MEMBERS	use network-security-groups
	UpdateVnic	NETWORK_SECURITY_GROUP_UPDATE_MEMBERS	use network-security-groups
security-lists	ListSecurityLists	SECURITY_LIST_READ	inspect security-lists
	GetSecurityList	SECURITY_LIST_READ	inspect security-lists
	UpdateSecurityList	SECURITY_LIST_UPDATE	manage security-lists
	ChangeSecurityListCompartment	SECURITY_LIST_MOVE	manage security-lists
	CreateSecurityList	SECURITY_LIST_CREATE	manage security-lists
	DeleteSecurityList	SECURITY_LIST_DELETE	manage security-lists
	DeleteSubnet	SECURITY_LIST_DETACH	manage security-lists

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	CreateSubnet	SECURITY_LIST_ATTACH	manage security-lists
	UpdateSubnet	SECURITY_LIST_ATTACH	manage security-lists
		SECURITY_LIST_DETACH	manage security-lists
dhcp-options	CreateDhcpOptions	DHCP_CREATE	manage dhcp-options
	GetDhcpOptions	DHCP_READ	inspect dhcp-options
	ListDhcpOptions	DHCP_READ	inspect dhcp-options
	UpdateDhcpOptions	DHCP_UPDATE	manage dhcp-options
	DeleteDhcpOptions	DHCP_DELETE	manage dhcp-options
	ChangeDhcpOptionsCompartment	DHCP_MOVE	manage dhcp-options
	DeleteSubnet	DHCP_DETACH	manage dhcp-options
	CreateSubnet	DHCP_ATTACH	manage dhcp-options
	UpdateSubnet	DHCP_ATTACH	manage dhcp-options
		DHCP_DETACH	manage dhcp-options
private-ips	GetPrivateIp	PRIVATE_IP_READ	inspect private-ips
	ListPrivateIps	PRIVATE_IP_READ	inspect private-ips
	ListPublicIps	PRIVATE_IP_READ	inspect private-ips
	GetPublicIp	PRIVATE_IP_READ	inspect private-ips
	GetPublicIpByPrivateIpId	PRIVATE_IP_READ	inspect private-ips
	UpdatePrivateIp	PRIVATE_IP_UPDATE	use private-ips
	CreatePrivateIp	PRIVATE_IP_CREATE	use private-ips
		PRIVATE_IP_ASSIGN	use private-ips
	DeletePrivateIp	PRIVATE_IP_DELETE	use private-ips
		PRIVATE_IP_UNASSIGN	use private-ips
	CreateRouteTable	PRIVATE_IP_ROUTE_TABLE_ATTACH	manage private-ips
	DeleteRouteTable	PRIVATE_IP_ROUTE_TABLE_DETACH	manage private-ips
	UpdateRouteTable	PRIVATE_IP_ROUTE_TABLE_ATTACH	manage private-ips
		PRIVATE_IP_ROUTE_TABLE_DETACH	manage private-ips
	CreateMountTarget	PRIVATE_IP_CREATE	use private-ips
		PRIVATE_IP_ASSIGN	use private-ips
	DeleteMountTarget	PRIVATE_IP_DELETE	use private-ips
		PRIVATE_IP_UNASSIGN	use private-ips

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
public-ips	GetPublicIp	PUBLIC_IP_READ	read public-ips
	ListPublicIps	PUBLIC_IP_READ	read public-ips
	GetPublicIpByPrivateIpId	PUBLIC_IP_READ	read public-ips
	GetPublicIpByIpAddress	PUBLIC_IP_READ	read public-ips
	UpdatePublicIp	PUBLIC_IP_UPDATE	manage public-ips
	CreatePublicIp	PUBLIC_IP_CREATE	manage public-ips
	DeletePublicIp	PUBLIC_IP_DELETE	manage public-ips
ipv6s	GetIpv6	IPV6_READ	read ipv6s
	ListIpv6s	IPV6_READ	read ipv6s
	UpdateIpv6	IPV6_UPDATE	manage ipv6s
	CreateIpv6	IPV6_CREATE	manage ipv6s
	DeleteIpv6	IPV6_DELETE	manage ipv6s
internet-gateways	ListInternetGateways	INTERNET_GATEWAY_READ	inspect internet-gateways
	GetInternetGateway	INTERNET_GATEWAY_READ	inspect internet-gateways
	UpdateInternetGateway	INTERNET_GATEWAY_UPDATE	manage internet-gateways
	ChangeInternetGatewayCompartment	INTERNET_GATEWAY_MOVE	manage internet-gateways
	CreateInternetGateway	INTERNET_GATEWAY_CREATE	manage internet-gateways
	DeleteInternetGateway	INTERNET_GATEWAY_DELETE	manage internet-gateways
	CreateRouteTable	INTERNET_GATEWAY_ATTACH	manage internet-gateways
	DeleteRouteTable	INTERNET_GATEWAY_DETACH	manage internet-gateways
	UpdateRouteTable	INTERNET_GATEWAY_ATTACH	manage internet-gateways
		INTERNET_GATEWAY_DETACH	manage internet-gateways
nat-gateways	ListNatGateways	NAT_GATEWAY_READ	read nat-gateways
	GetNatGateway	NAT_GATEWAY_READ	read nat-gateways
	UpdateNatGateway	NAT_GATEWAY_UPDATE	manage nat-gateways
	ChangeNatGatewayCompartment	NAT_GATEWAY_MOVE	manage nat-gateways
	CreateNatGateway	NAT_GATEWAY_CREATE	manage nat-gateways

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	DeleteNatGateway	NAT_GATEWAY_DELETE	manage nat-gateways
	CreateRouteTable	NAT_GATEWAY_ATTACH	use nat-gateways
	DeleteRouteTable	NAT_GATEWAY_DETACH	use nat-gateways
	UpdateRouteTable	NAT_GATEWAY_ATTACH	use nat-gateways
		NAT_GATEWAY_DETACH	use nat-gateways
service-gateways	ListServiceGateways	SERVICE_GATEWAY_READ	inspect service-gateways
	GetServiceGateway	SERVICE_GATEWAY_READ	inspect service-gateways
	ChangeServiceGatewayCompartment	SERVICE_GATEWAY_MOVE	manage service-gateways
	AttachServiceId	SERVICE_GATEWAY_ADD_SERVICE	manage service-gateways
	DetachServiceId	SERVICE_GATEWAY_DELETE_SERVICE	manage service-gateways
	CreateServiceGateway	SERVICE_GATEWAY_CREATE	manage service-gateways
	UpdateServiceGateway	SERVICE_GATEWAY_UPDATE	manage service-gateways
	DeleteServiceGateway	SERVICE_GATEWAY_DELETE	manage service-gateways
	CreateRouteTable	SERVICE_GATEWAY_ATTACH	use service-gateways
	DeleteRouteTable	SERVICE_GATEWAY_DETACH	use service-gateways
	UpdateRouteTable	SERVICE_GATEWAY_ATTACH	use service-gateways
		SERVICE_GATEWAY_DETACH	use service-gateways
local-peering-gateways	ListLocalPeeringGateways	LOCAL_PEERING_GATEWAY_READ	inspect local-peering-gateways
	GetLocalPeeringGateway	LOCAL_PEERING_GATEWAY_READ	inspect local-peering-gateways
	CreateLocalPeeringGateway	LOCAL_PEERING_GATEWAY_CREATE	manage local-peering-gateways
	UpdateLocalPeeringGateway	LOCAL_PEERING_GATEWAY_UPDATE	manage local-peering-gateways
	DeleteLocalPeeringGateway	LOCAL_PEERING_GATEWAY_DELETE	manage local-peering-gateways
	ChangeLocalPeeringGatewayCompartment	LOCAL_PEERING_GATEWAY_MOVE	manage local-peering-gateways
	CreateRouteTable	LOCAL_PEERING_GATEWAY_ATTACH	manage local-peering-gateways

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	DeleteRouteTable	LOCAL_PEERING_GATEWAY_DETACH	manage local-peering-gateways
	UpdateRouteTable	LOCAL_PEERING_GATEWAY_ATTACH	manage local-peering-gateways
		LOCAL_PEERING_GATEWAY_DETACH	manage local-peering-gateways
local-peering-from	ConnectLocalPeeringGateways	LOCAL_PEERING_GATEWAY_CONNECT_FROM	manage local-peering-from
local-peering-to	ConnectLocalPeeringGateways	LOCAL_PEERING_GATEWAY_CONNECT_TO	manage local-peering-to
remote-peering-connections	ListRemotePeeringConnections	REMOTE_PEERING_CONNECTION_READ	inspect remote-peering-connections
	GetRemotePeeringConnection	REMOTE_PEERING_CONNECTION_READ	inspect remote-peering-connections
	UpdateRemotePeeringConnection	REMOTE_PEERING_CONNECTION_UPDATE	manage remote-peering-connections
	CreateRemotePeeringConnection	REMOTE_PEERING_CONNECTION_CREATE	manage remote-peering-connections
	DeleteRemotePeeringConnection	REMOTE_PEERING_CONNECTION_DELETE	manage remote-peering-connections
	ChangeRemotePeeringConnectionCompartment	REMOTE_PEERING_CONNECTION_RESOURCE_MOVE	manage remote-peering-connections
remote-peering-from	ConnectRemotePeeringConnections	REMOTE_PEERING_CONNECTION_CONNECT_FROM	manage remote-peering-from
remote-peering-to	ConnectRemotePeeringConnections	REMOTE_PEERING_CONNECTION_CONNECT_TO	manage remote-peering-to
drgs	ListDrgs	DRG_READ	inspect drgs
	GetDrg	DRG_READ	inspect drgs
	CreateDrg	DRG_CREATE	manage drgs
	UpdateDrg	DRG_UPDATE	manage drgs
	DeleteDrg	DRG_DELETE	manage drgs
	ChangeDrgCompartment	DRG_MOVE	manage drgs
	CreateDrgAttachment	DRG_ATTACH	manage drgs
	DeleteDrgAttachment	DRG_DETACH	manage drgs
	CreateRouteTable	DRG_ATTACH	manage drgs

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	DeleteRouteTable	DRG_DETACH	manage drgs
	UpdateRouteTable	DRG_ATTACH	manage drgs
		DRG_DETACH	manage drgs
drg-attachments	CreateDrgAttachment		
	DeleteDrgAttachment		
	ListDrgAttachments	DRG_ATTACHMENT_READ	inspect drg-attachments
	GetDrgAttachment	DRG_ATTACHMENT_READ	inspect drg-attachments
	UpdateDrgAttachment	DRG_ATTACHMENT_UPDATE	manage drg-attachments
cpes	ListCpes	CPE_READ	inspect cpes
	GetCpe	CPE_READ	inspect cpes
	CreateCpe	CPE_CREATE	manage cpes
	UpdateCpe	CPE_UPDATE	manage cpes
	DeleteCpe	CPE_DELETE	manage cpes
	ChangeCpeCompartment	CPE_RESOURCE_MOVE	manage cpes
ipsec	ListIPSecConnections	IPSEC_CONNECTION_READ	inspect ipsec
	GetIPSecConnection	IPSEC_CONNECTION_READ	inspect ipsec
	GetIPSecConnectionStatus	IPSEC_CONNECTION_READ	inspect ipsec
	ListIPSecConnectionTunnels	IPSEC_CONNECTION_READ	inspect ipsec
	GetIPSecConnectionTunnel	IPSEC_CONNECTION_READ	inspect ipsec
	GetTunnelCpeDeviceConfig	IPSEC_CONNECTION_READ	inspect ipsec
	GetTunnelCpeDeviceTemplateContent	IPSEC_CONNECTION_READ	inspect ipsec
	GetCpeDeviceTemplateContent	IPSEC_CONNECTION_READ	inspect ipsec
	GetIpsecCpeDeviceTemplateContent	IPSEC_CONNECTION_READ	inspect ipsec
	GetIPSecConnectionDeviceConfig	IPSEC_CONNECTION_DEVICE_CONFIG_READ	read ipsec

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	GetIPSecConnectionTunnelSharedSecret	IPSEC_CONNECTION_DEVICE_CONFIG_READ	read ipsec
	UpdateIPSecConnection	IPSEC_CONNECTION_UPDATE	manage ipsec
	UpdateTunnelCpeDeviceConfig	IPSEC_CONNECTION_UPDATE	manage ipsec
	UpdateIPSecConnectionTunnel	IPSEC_CONNECTION_UPDATE	manage ipsec
	CreateIPSecConnection	IPSEC_CONNECTION_CREATE	manage ipsec
	DeleteIPSecConnection	IPSEC_CONNECTION_DELETE	manage ipsec
cross-connects	ListCrossConnects	CROSS_CONNECT_READ	inspect cross-connects
	GetCrossConnect	CROSS_CONNECT_READ	inspect cross-connects
	UpdateCrossConnect	CROSS_CONNECT_UPDATE	manage cross-connects
	CreateCrossConnect	CROSS_CONNECT_CREATE	manage cross-connects
	DeleteCrossConnect	CROSS_CONNECT_DELETE	manage cross-connects
	ChangeCrossConnectCompartment	CROSS_CONNECT_RESOURCE_MOVE	manage cross-connects
cross-connect-groups	ListCrossConnectGroups	CROSS_CONNECT_GROUP_READ	inspect cross-connect-groups
	GetCrossConnectGroup	CROSS_CONNECT_GROUP_READ	inspect cross-connect-groups
	UpdateCrossConnectGroup	CROSS_CONNECT_GROUP_UPDATE	manage cross-connect-groups
	CreateCrossConnectGroup	CROSS_CONNECT_GROUP_CREATE	manage cross-connect-groups
	DeleteCrossConnectGroup	CROSS_CONNECT_GROUP_DELETE	manage cross-connect-groups
	ChangeCrossConnectGroupCompartment	CROSS_CONNECT_GROUP_RESOURCE_MOVE	manage cross-connect-groups
virtual-circuits	ListVirtualCircuits	VIRTUAL_CIRCUIT_READ	inspect virtual-circuits
	GetVirtualCircuit	VIRTUAL_CIRCUIT_READ	inspect virtual-circuits
	ChangeVirtualCircuitCompartment	VIRTUAL_CIRCUIT_RESOURCE_MOVE	manage virtual-circuits

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	CreateVirtualCircuit	VIRTUAL_CIRCUIT_CREATE	manage virtual-circuits
	DeleteVirtualCircuit	VIRTUAL_CIRCUIT_DELETE	manage virtual-circuits
vnics	GetVnic	VNIC_READ	inspect vnics
	AttachVnic	VNIC_ATTACH	use vnics
		VNIC_CREATE	use vnics
	UpdateVnic	VNIC_UPDATE	use vnics
	DetachVnic	VNIC_DETACH	use vnics
		VNIC_DELETE	use vnics
	LaunchInstance	VNIC_ATTACH	use vnics
		VNIC_CREATE	use vnics
	TerminateInstance	VNIC_DELETE	use vnics
	CreateInstancePool	VNIC_CREATE	use vnics
	TerminateInstancePool	VNIC_DELETE	use vnics
	CreateInstanceConfiguration	VNIC_READ	inspect vnics
	CreatePrivateIp	VNIC_ASSIGN	use vnics
	CreateMountTarget	VNIC_ASSIGN	use vnics
		VNIC_CREATE	use vnics
		VNIC_ATTACH	use vnics
	DeleteMountTarget	VNIC_UNASSIGN	use vnics
		VNIC_DELETE	use vnics
		VNIC_DETACH	use vnics
vnic-attachments	GetVnicAttachment	VNIC_ATTACHMENT_READ	inspect vnic-attachments
	ListVnicAttachments	VNIC_ATTACHMENT_READ	inspect vnic-attachments
	TerminateInstance		
	CreateInstanceConfiguration	VNIC_ATTACHMENT_READ	inspect vnic-attachments
cluster-networks	ListClusterNetworks	CLUSTER_NETWORK_INSPECT	inspect cluster-networks
	GetClusterNetwork	CLUSTER_NETWORK_READ	read cluster-networks
	ListClusterNetworkInstances	CLUSTER_NETWORK_READ	read cluster-networks
	UpdateClusterNetwork	CLUSTER_NETWORK_UPDATE	manage cluster-networks

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	ChangeClusterNetworkCompartment	CLUSTER_NETWORK_MOVE	manage cluster-networks
	CreateClusterNetwork	CLUSTER_NETWORK_CREATE	manage cluster-networks
	TerminateClusterNetwork	CLUSTER_NETWORK_DELETE	manage cluster-networks
dns-zones	ListZones	DNS_ZONE_INSPECT	inspect dns-zones
	CreateZone	DNS_ZONE_CREATE	manage dns-zones
	CreateChildZone	DNS_ZONE_CREATE	manage dns-zones
	InspectParentZone	DNS_ZONE_INSPECT	inspect dns-zones
	DeleteZone	DNS_ZONE_DELETE	manage dns-zones
	GetZone	DNS_ZONE_READ	read dns-zones
	UpdateZone	DNS_ZONE_UPDATE	use dns-zones
	ChangeZoneCompartment	DNS_ZONE_MOVE	manage dns-zones
	CreateSteeringPolicyAttachment	DNS_ZONE_UPDATE	use dns-zones
	UpdateSteeringPolicyAttachment	DNS_ZONE_UPDATE	use dns-zones
	DeleteSteeringPolicyAttachment	DNS_ZONE_UPDATE	use dns-zones
	GetZoneRecords	DNS_ZONE_READ	read dns-zones
	PatchZoneRecords	DNS_ZONE_UPDATE	use dns-zones
	UpdateZoneRecords	DNS_ZONE_UPDATE	use dns-zones
dns-records	GetZoneRecords	DNS_RECORD_READ	read dns-records
	PatchZoneRecords	DNS_RECORD_UPDATE	use dns-records
	UpdateZoneRecords	DNS_RECORD_UPDATE	use dns-records
	GetDomainRecords	DNS_RECORD_READ	read dns-records
	DeleteDomainRecords	DNS_RECORD_DELETE	manage dns-records
	PatchDomainRecords	DNS_RECORD_UPDATE	use dns-records

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	UpdateDomainRecords	DNS_RECORD_UPDATE	use dns-records
	DeleteRRSet	DNS_RECORD_UPDATE	use dns-records
	GetRRSet	DNS_RECORD_READ	read dns-records
	PatchRRSet	DNS_RECORD_UPDATE	use dns-records
	UpdateRRSet	DNS_RECORD_UPDATE	use dns-records
dns-steering-policies	ListSteeringPolicies	DNS_STEERING_POLICY_INSPECT	inspect dns-steering-policies
	CreateSteeringPolicy	DNS_STEERING_POLICY_CREATE	manage dns-steering-policies
	GetSteeringPolicy	DNS_STEERING_POLICY_READ	read dns-steering-policies
	UpdateSteeringPolicy	DNS_STEERING_POLICY_UPDATE	use dns-steering-policies
	DeleteSteeringPolicy	DNS_STEERING_POLICY_DELETE	manage dns-steering-policies
	ChangeSteeringPolicyCompartment	DNS_STEERING_POLICY_MOVE	manage dns-steering-policies
	CreateSteeringPolicyAttachment	DNS_STEERING_POLICY_READ	read dns-steering-policies
	UpdateSteeringPolicyAttachment	DNS_STEERING_POLICY_READ	read dns-steering-policies
	DeleteSteeringPolicyAttachment	DNS_STEERING_POLICY_READ	read dns-steering-policies
dns-steering-policy-attachments	ListSteeringPolicyAttachments	DNS_STEERING_ATTACHMENT_INSPECT	inspect dns-steering-policy-attachments
	CreateSteeringPolicyAttachment		
	GetSteeringPolicyAttachment	DNS_STEERING_ATTACHMENT_READ	read dns-steering-policy-attachments
	UpdateSteeringPolicyAttachment		
	DeleteSteeringPolicyAttachment		
dns-tsig-keys	ListTsigKeys	DNS_TSIG_KEY_INSPECT	inspect dns-tsig-keys
	CreateTsigKey	DNS_TSIG_KEY_CREATE	manage dns-tsig-keys
	GetTsigKey	DNS_TSIG_KEY_READ	read dns-tsig-keys
	UpdateTsigKey	DNS_TSIG_KEY_UPDATE	use dns-tsig-keys
	DeleteTsigKey	DNS_TSIG_KEY_DELETE	manage dns-tsig-keys

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	ChangeTsigKeyCompartment	DNS_TSIG_KEY_MOVE	manage dns-tsig-keys
dns-views	ListViews	DNS_VIEW_INSPECT	inspect dns-views
	CreateView	DNS_VIEW_CREATE	manage dns-views
	GetView	DNS_VIEW_READ	read dns-views
	UpdateView	DNS_VIEW_UPDATE	use dns-views
	DeleteView	DNS_VIEW_DELETE	manage dns-views
	ChangeViewCompartment	DNS_VIEW_MOVE	manage dns-views
dns-resolvers	ListResolvers	DNS_RESOLVER_INSPECT	inspect dns-resolvers
	GetResolver	DNS_RESOLVER_READ	read dns-resolvers
	UpdateResolver	DNS_RESOLVER_UPDATE	use dns-resolvers
	ChangeResolverCompartment	DNS_RESOLVER_MOVE	manage dns-resolvers
dns-resolver-endpoint	ListResolverEndpoints	DNS_RESOLVER_ENDPOINT_INSPECT	inspect dns-resolver-endpoint
	CreateResolverEndpoint	DNS_RESOLVER_ENDPOINT_CREATE	manage dns-resolver-endpoint
	GetResolverEndpoint	DNS_RESOLVER_ENDPOINT_READ	read dns-resolver-endpoint
	UpdateResolverEndpoint	DNS_RESOLVER_ENDPOINT_UPDATE	use dns-resolver-endpoint
	DeleteResolverEndpoint	DNS_RESOLVER_ENDPOINT_DELETE	manage dns-resolver-endpoint
objectstorage-namespaces	GetNamespace		
	GetNamespaceMetadata	OBJECTSTORAGE_NAMESPACE_READ	read objectstorage-namespaces
	UpdateNamespaceMetadata	OBJECTSTORAGE_NAMESPACE_UPDATE	manage objectstorage-namespaces
buckets	HeadBucket	BUCKET_INSPECT	inspect buckets
	ListBuckets	BUCKET_INSPECT	inspect buckets
	GetBucket	BUCKET_READ	read buckets
	ListMultipartUploads	BUCKET_READ	read buckets
	GetObjectLifecyclePolicy	BUCKET_READ	read buckets
	GetRetentionRule	BUCKET_READ	read buckets
	ListRetentionRules	BUCKET_READ	read buckets
	GetReplicationPolicy	BUCKET_READ	read buckets
	ListReplicationPolicies	BUCKET_READ	read buckets

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	ListReplicationSources	BUCKET_READ	read buckets
	UpdateBucket	BUCKET_UPDATE	use buckets
	DeleteObjectLifecyclePolicy	BUCKET_UPDATE	use buckets
	ReencryptBucket	BUCKET_UPDATE	use buckets
	CreateBucket	BUCKET_CREATE	manage buckets
	DeleteBucket	BUCKET_DELETE	manage buckets
	CreatePar	PAR_MANAGE	manage buckets
	GetPar	PAR_MANAGE	manage buckets
	ListPars	PAR_MANAGE	manage buckets
	DeletePar	PAR_MANAGE	manage buckets
	CreateRetentionRule	RETENTION_RULE_LOCK	manage buckets
	UpdateRetentionRule	RETENTION_RULE_LOCK	manage buckets
	DeleteRetentionRule	RETENTION_RULE_LOCK	manage buckets
	MakeBucketWritable	BUCKET_READ	read buckets
		BUCKET_UPDATE	use buckets
	CreateReplicationPolicy	BUCKET_READ	read buckets
		BUCKET_UPDATE	use buckets
	DeleteReplicationPolicy	BUCKET_READ	read buckets
		BUCKET_UPDATE	use buckets
	PutObjectLifecyclePolicy	BUCKET_UPDATE	use buckets
objects	HeadObject	OBJECT_INSPECT	inspect objects
	ListObjects	OBJECT_INSPECT	inspect objects
	ListMultipartUploadParts	OBJECT_INSPECT	inspect objects
	CreateObject	OBJECT_CREATE	manage objects
	GetObject	OBJECT_READ	read objects
	ReencryptObject	OBJECT_OVERWRITE	use objects
	RenameObject	OBJECT_CREATE	manage objects
		OBJECT_OVERWRITE	use objects
	RestoreObject	OBJECT_RESTORE	manage objects
	DeleteObject	OBJECT_DELETE	manage objects
	DeleteObjectVersion	OBJECT_VERSION_DELETE	manage objects

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	CreateMultipartUpload	OBJECT_CREATE	manage objects
		OBJECT_OVERWRITE	use objects
	UploadPart	OBJECT_CREATE	manage objects
		OBJECT_OVERWRITE	use objects
	CommitMultipartUpload	OBJECT_CREATE	manage objects
		OBJECT_OVERWRITE	use objects
	AbortMultipartUpload	OBJECT_DELETE	manage objects
	PutObject	OBJECT_CREATE	manage objects
	('PutObject', 'overwrite')	OBJECT_OVERWRITE	use objects
	CreateCopyRequest	OBJECT_READ	read objects
		OBJECT_CREATE	manage objects
		OBJECT_OVERWRITE	use objects
		OBJECT_INSPECT	inspect objects
	CopyObject	OBJECT_READ	read objects
		OBJECT_CREATE	manage objects
		OBJECT_OVERWRITE	use objects
		OBJECT_INSPECT	inspect objects
export-sets	CreateExport	EXPORT_SET_UPDATE	manage export-sets
	GetExport	EXPORT_SET_READ	read export-sets
	ListExports	EXPORT_SET_READ	read export-sets
	UpdateExport	EXPORT_SET_UPDATE	manage export-sets
	DeleteExport	EXPORT_SET_UPDATE	manage export-sets
	CreateExportSet	EXPORT_SET_CREATE	manage export-sets
	GetExportSet	EXPORT_SET_READ	read export-sets
	ListExportSets	EXPORT_SET_INSPECT	inspect export-sets
	UpdateExportSet	EXPORT_SET_UPDATE	manage export-sets
	DeleteExportSet	EXPORT_SET_DELETE	manage export-sets
file-systems	ListFileSystems	FILE_SYSTEM_INSPECT	inspect file-systems
	GetFileSystem	FILE_SYSTEM_READ	read file-systems
	CreateFileSystem	FILE_SYSTEM_CREATE	manage file-systems
	UpdateFileSystem	FILE_SYSTEM_UPDATE	manage file-systems
	DeleteFileSystem	FILE_SYSTEM_DELETE	manage file-systems
	ChangeFileSystemCompartment	FILE_SYSTEM_MOVE	manage file-systems
	CreateSnapshot	FILE_SYSTEM_CREATE_SNAPSHOT	manage file-systems

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	DeleteSnapshot	FILE_SYSTEM_DELETE_SNAPSHOT	manage file-systems
	GetSnapshot	FILE_SYSTEM_READ	read file-systems
	ListSnapshots	FILE_SYSTEM_READ	read file-systems
	UpdateSnapshot	FILE_SYSTEM_UPDATE	manage file-systems
mount-targets	ListMountTargets	MOUNT_TARGET_INSPECT	inspect mount-targets
	GetMountTarget	MOUNT_TARGET_READ	read mount-targets
	UpdateMountTarget	MOUNT_TARGET_UPDATE	manage mount-targets
	ChangeMountTargetCompartment	MOUNT_TARGET_MOVE	manage mount-targets
	CreateMountTarget	MOUNT_TARGET_CREATE	manage mount-targets
	DeleteMountTarget	MOUNT_TARGET_DELETE	manage mount-targets
volumes	ListVolumes	VOLUME_INSPECT	inspect volumes
	GetVolume	VOLUME_INSPECT	inspect volumes
	UpdateVolume	VOLUME_UPDATE	use volumes
	GetBootVolume	VOLUME_INSPECT	inspect volumes
	ListBootVolumes	VOLUME_INSPECT	inspect volumes
	UpdateBootVolume	VOLUME_UPDATE	use volumes
	DeleteBootVolume	VOLUME_DELETE	manage volumes
	CreateVolume	VOLUME_CREATE	manage volumes
	CreateBootVolume	VOLUME_CREATE	manage volumes
	DeleteVolume	VOLUME_DELETE	manage volumes
	AttachVolume	VOLUME_WRITE	use volumes
	DetachVolume	VOLUME_WRITE	use volumes
	TerminateInstance	VOLUME_WRITE	use volumes
	ListVolumeAttachments	VOLUME_INSPECT	inspect volumes
	ListBootVolumeAttachments	VOLUME_INSPECT	inspect volumes
	GetVolumeAttachment	VOLUME_INSPECT	inspect volumes
	GetBootVolumeAttachment	VOLUME_INSPECT	inspect volumes

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	ChangeVolumeCompartment	VOLUME_MOVE	manage volumes
	ChangeBootVolumeCompartment	BOOT_VOLUME_MOVE	manage volumes
	TerminateInstancePool	VOLUME_WRITE	use volumes
	CreateInstanceConfiguration	VOLUME_INSPECT	inspect volumes
	CreateBootVolumeBackup	VOLUME_WRITE	use volumes
	UpdateVolumeBackup	VOLUME_INSPECT	inspect volumes
	UpdateBootVolumeBackup	VOLUME_INSPECT	inspect volumes
	ListVolumeBackups	VOLUME_INSPECT	inspect volumes
	CreateVolumeGroupBackup	VOLUME_WRITE	use volumes
	CreateVolumeGroup	VOLUME_INSPECT	inspect volumes
		VOLUME_CREATE	manage volumes
		VOLUME_WRITE	use volumes
	UpdateVolumeGroup	VOLUME_INSPECT	inspect volumes
	DeleteVolumeBackup	VOLUME_INSPECT	inspect volumes
	GetVolumeBackupPolicyAssetAssignment	VOLUME_INSPECT	inspect volumes
	ChangeVolumeGroupCompartment	VOLUME_MOVE	manage volumes
		BOOT_VOLUME_MOVE	manage volumes
volume-attachments	ListVolumeAttachments	VOLUME_ATTACHMENT_INSPECT	inspect volume-attachments
	ListBootVolumeAttachments	VOLUME_ATTACHMENT_INSPECT	inspect volume-attachments
	GetVolumeAttachment	VOLUME_ATTACHMENT_INSPECT	inspect volume-attachments
	GetBootVolumeAttachment	VOLUME_ATTACHMENT_INSPECT	inspect volume-attachments
	AttachVolume	VOLUME_ATTACHMENT_CREATE	manage volume-attachments
	AttachBootVolume	VOLUME_ATTACHMENT_CREATE	manage volume-attachments
	DetachVolume	VOLUME_ATTACHMENT_DELETE	manage volume-attachments

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	DetachBootVolume	VOLUME_ATTACHMENT_DELETE	manage volume-attachments
	TerminateInstance	VOLUME_ATTACHMENT_DELETE	manage volume-attachments
	TerminateInstancePool	VOLUME_ATTACHMENT_DELETE	manage volume-attachments
	CreateInstanceConfiguration	VOLUME_ATTACHMENT_INSPECT	inspect volume-attachments
volume-backups	ListVolumeBackups	VOLUME_BACKUP_INSPECT	inspect volume-backups
	GetVolumeBackup	VOLUME_BACKUP_INSPECT	inspect volume-backups
	UpdateVolumeBackup	VOLUME_BACKUP_UPDATE	use volume-backups
	CopyVolumeBackup	VOLUME_BACKUP_COPY	use volume-backups
	CreateVolumeBackup	VOLUME_BACKUP_CREATE	manage volume-backups
	DeleteVolumeBackup	VOLUME_BACKUP_DELETE	manage volume-backups
	CreateVolume	VOLUME_BACKUP_READ	read volume-backups
	CreateVolumeGroupBackup	VOLUME_BACKUP_CREATE	manage volume-backups
	CreateVolumeGroup	VOLUME_BACKUP_READ	read volume-backups
	DeleteVolumeGroupBackup	VOLUME_BACKUP_DELETE	manage volume-backups
	ChangeVolumeBackupCompartment	VOLUME_BACKUP_MOVE	manage volume-backups
	ChangeVolumeGroupBackupCompartment	VOLUME_BACKUP_MOVE	manage volume-backups
boot-volume-backups	ListBootVolumeBackups	BOOT_VOLUME_BACKUP_INSPECT	inspect boot-volume-backups
	GetBootVolumeBackup	BOOT_VOLUME_BACKUP_INSPECT	inspect boot-volume-backups
	CreateBootVolume	BOOT_VOLUME_BACKUP_READ	read boot-volume-backups
	UpdateBootVolumeBackup	BOOT_VOLUME_BACKUP_UPDATE	use boot-volume-backups
	CopyBootVolumeBackup	BOOT_VOLUME_BACKUP_COPY	use boot-volume-backups
	CreateBootVolumeBackup	BOOT_VOLUME_BACKUP_CREATE	manage boot-volume-backups

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	DeleteBootVolumeBackup	BOOT_VOLUME_BACKUP_DELETE	manage boot-volume-backups
	CreateVolumeGroupBackup	BOOT_VOLUME_BACKUP_CREATE	manage boot-volume-backups
	CreateVolumeGroup	BOOT_VOLUME_BACKUP_READ	read boot-volume-backups
	DeleteVolumeGroupBackup	BOOT_VOLUME_BACKUP_DELETE	manage boot-volume-backups
	ChangeVolumeBackupCompartment	BOOT_VOLUME_BACKUP_MOVE	manage boot-volume-backups
	ChangeBootVolumeBackupCompartment	BOOT_VOLUME_BACKUP_MOVE	manage boot-volume-backups
	ChangeVolumeGroupBackupCompartment	BOOT_VOLUME_BACKUP_MOVE	manage boot-volume-backups
backup-policies	ListVolumeBackupPolicies	BACKUP_POLICIES_INSPECT	inspect backup-policies
	GetVolumeBackupPolicy	BACKUP_POLICIES_INSPECT	inspect backup-policies
	UpdateVolumeBackupPolicy	BACKUP_POLICIES_UPDATE	use backup-policies
	CreateVolumeBackupPolicy	BACKUP_POLICIES_CREATE	manage backup-policies
	DeleteVolumeBackupPolicy	BACKUP_POLICIES_DELETE	manage backup-policies
backup-policy-assignments	GetVolumeBackupPolicyAssignment	BACKUP_POLICY_ASSIGNMENT_INSPECT	inspect backup-policy-assignments
	GetVolumeBackupPolicyAssetAssignment	BACKUP_POLICY_ASSIGNMENT_INSPECT	inspect backup-policy-assignments
	CreateVolumeBackupPolicyAssignment	BACKUP_POLICY_ASSIGNMENT_CREATE	manage backup-policy-assignments
	DeleteVolumeBackupPolicyAssignment	BACKUP_POLICY_ASSIGNMENT_DELETE	manage backup-policy-assignments
volume-groups	ListVolumeGroups	VOLUME_GROUP_INSPECT	inspect volume-groups
	GetVolumeGroup	VOLUME_GROUP_INSPECT	inspect volume-groups
	DeleteVolumeGroup	VOLUME_GROUP_DELETE	manage volume-groups
	UpdateVolumeGroup	VOLUME_GROUP_UPDATE	manage volume-groups

Resource Type	API Operation	Required Permissions	Verb + Resource Combination
	CreateVolumeGroup	VOLUME_GROUP_CREATE	manage volume-groups
	CreateVolumeGroupBackup	VOLUME_GROUP_INSPECT	inspect volume-groups
	ChangeVolumeGroupCompartment	VOLUME_GROUP_MOVE	manage volume-groups
volume-group-backups	ListVolumeGroupBackups	VOLUME_GROUP_BACKUP_INSPECT	inspect volume-group-backups
	GetVolumeGroupBackup	VOLUME_GROUP_BACKUP_INSPECT	inspect volume-group-backups
	UpdateVolumeGroupBackup	VOLUME_GROUP_BACKUP_UPDATE	manage volume-group-backups
	CreateVolumeGroupBackup	VOLUME_GROUP_BACKUP_CREATE	manage volume-group-backups
	DeleteVolumeGroupBackup	VOLUME_GROUP_BACKUP_DELETE	manage volume-group-backups
	CreateVolumeGroup	VOLUME_GROUP_BACKUP_INSPECT	inspect volume-group-backups
	ChangeVolumeGroupBackupCompartment	VOLUME_GROUP_BACKUP_MOVE	manage volume-group-backups
clusters	ListClusters	CLUSTER_INSPECT	inspect clusters
	CreateCluster	CLUSTER_CREATE	manage clusters
	GetClusterKubeconfig	CLUSTER_USE	use clusters
	GetCluster	CLUSTER_READ	read clusters
	UpdateCluster	CLUSTER_UPDATE	manage clusters
	DeleteCluster	CLUSTER_DELETE	manage clusters
	AdministerK8s	CLUSTER_MANAGE	manage clusters
cluster-node-pools	ListNodePools	CLUSTER_NODE_POOL_INSPECT	inspect cluster-node-pools
	CreateNodePool	CLUSTER_NODE_POOL_CREATE	manage cluster-node-pools
	GetNodePool	CLUSTER_NODE_POOL_READ	read cluster-node-pools
	GetNodePoolOptions		
	UpdateNodePool	CLUSTER_NODE_POOL_UPDATE	manage cluster-node-pools
	DeleteNodePool	CLUSTER_NODE_POOL_DELETE	manage cluster-node-pools

5
Tagging Overview

Tagging is a way to add metadata to your resources. That metadata consists of keys and values that you can select and define in accordance with your business requirements. The tags allow you to categorize resources independently of compartment hierarchy, list them in an alternate way and track their usage. When you work in the Compute Web UI you also use tags to filter lists and search for resources.

Tag Types

Private Cloud Appliance supports two kinds of tags:

- **Free-form tags** are created and applied to resources by users. They contain unmanaged metadata.
- **Defined tags** offer a structured approach to tagging. Administrators create and manage the metadata, while users only have the ability to apply a tag to a resource, and in some cases add a value.

Most of the tagging features require defined tags. They ensure that the applied metadata is reliable for managing resources, collecting accurate information and automating certain operations. With defined tags, the following scenarios become possible:

- Creating default tags that are applied to all resources in compartments.
- Specifying that users must apply tags to resources to successfully create resources in compartments.
- If you make a typo using defined tags, correct it by editing or even deleting the tag. When you delete a defined tag, the key and any value for that tag are removed from all resources.
- Associating a list of predefined values with a defined tag.
- Using system variables to generate values for defined tags or tag defaults automatically.

Both free-form tags and defined tags can be used simultaneously within a tenancy. The table below shows a feature comparison of the two types.

Feature	Free-form Tags	Defined Tags
Hierarchy	Free-form tags consist of a key and a value, but do not belong to a namespace.	Defined tags consist of a tag namespace, a key, and a value.
Creation and application	Free-form tags can be applied during resource creation or to an existing resource.	The tag namespace and tag key definition must be set up in advance by an administrator. Defined tags can be applied during resource creation or to an existing resource.

Feature	Free-form Tags	Defined Tags
Predefined keys and values	Not supported.	When applying a defined tag, users select from the list of tag keys. Administrators can also create a list of predefined values to associate with a tag key. When applying a tag to a resource, users must select a value from that list.
Variables	Not supported.	Variables are used to set the value of a defined tag. When a user applies the tag, the variable is resolved and replaced with the data it represents.
Case sensitivity	Free-form tag keys and values are both case sensitive.	In defined tags, the key is *not* case sensitive; only the tag values are case sensitive.

Free-form tags allow users to quickly and conveniently include relevant information in the metadata of a resource. However, you should be aware of their limitations:

- When applying a free-form tag, you cannot see a list of existing free-form tags, so you do not know what tags and values have already been used. It is not possible to list the free-form tags used in your tenancy. The OCI CLI does allow you to list a set of resources and view the free-form tags applied to them.
- You cannot control access to resources based on free-form tags. They cannot be used in IAM policy statements.
- You cannot use predefined tag values or tag variables in free-form tags.

Categorization with Defined Tags

This section describes how defined tags work and which permissions are required.

Tag Namespaces and Keys

Defined tags provide more features and control than free-form tags. Before you create a defined tag key, you first set up a *tag namespace* for it. You can think of the tag namespace as a container for a set of *tag keys*. Within each namespace, the tag keys must be unique, but a tag key name can be repeated across namespaces. When you create the tag key definition, you must choose the type of value, which also determines how the user applying the tag adds the value:

- You can leave it empty so that a user can fill in the value.
- You can create a list of values so that the user must choose from those values.

To apply a defined tag to a resource, a user first selects the tag namespace, then the tag key within the namespace, and then they can assign the value. If the tag key contains a blank value, the user can type in a value or leave it blank. If the tag key contains a list, the user must select a value from the list.

Defined tags support policy to allow you to control who can apply your defined tags. The tag namespace is the entity to which you can apply policy. Administrators can control which groups of users are allowed to use each namespace.

Permissions for Working with Defined Tags

Setting up and managing tag namespaces and key definitions is an administrator responsibility. Full management permissions for tag namespaces are required.

To apply, update, or remove defined tags for a resource, users must be granted `use` access for the tag namespaces they need to work with. Users must also have the permission to update the resource to which they apply a tag. For many resources, the update permission is granted with the `use` verb. For example, users who can `use` instances in CompartmentA can also apply, update, or remove defined tags for those instances. For resources that do not include the update permission with the `use` verb, you can create a policy statement to grant only the update permission from the `manage` verb.

Here are a few examples of policy statements related to tagging:

- Administrators must be allowed to define tag namespaces and keys.

  ```
  Allow group TagAdmins to manage tag-namespaces in tenancy
  ```

- Users are granted permission to apply tags from all namespaces in the tenancy, or a subset.

  ```
  Allow group GroupA to use tag-namespaces in tenancy
  ```

  or

  ```
  Allow group GroupA to use tag-namespaces in tenancy where target.tag-
  namespace.name='Operations'
  ```

  or

  ```
  Allow group GroupA to use tag-namespaces in tenancy where any {target.tag-
  namespace.name='Operations', target.tag-namespace.name='Support'}
  ```

- Users are granted permission to update resources in order to apply tags.

  ```
  Allow group GroupA to use instance-family in compartment CompartmentA
  ```

  or

  ```
  Allow group GroupA to use vcns in compartment CompartmentA
  Allow group GroupA to manage vcns in compartment CompartmentA where
  request.permission='VCN_UDPATE'
  ```

Tag Management Features

This section describes the operations available to an administrator who sets up and manages tag namespaces and key definitions.

Retiring and Reactivating Tag Definitions

You can retire a tag key definition or a tag namespace definition.

When you retire a tag key definition, you can no longer apply it to resources. However, the tag is not removed from the resources that it was applied to. The tag still exists as metadata on those resources and you can still call the retired tag in operations such as searching, listing, sorting, or reporting.

You can reactivate retired tag key definitions and tag namespace definitions.

- When you reactivate a tag key, it is again available for users to apply to resources.
- When you reactivate a tag namespace, you can create new tag key definitions in that namespace. However, if you want to use any of the tag key definitions that were retired with the namespace, you must explicitly reactivate each tag key definition.

Moving Tag Namespaces

You can move a tag namespace to a different compartment. The tag namespace can be active or retired when you move it. When you move the tag namespace, all its tag key definitions are moved along with it.

This functionality is useful if you need to reorganize your compartment hierarchy, or if you need to delete a compartment that contains a retired tag namespace. Remember that you cannot delete a compartment that contains resources. A retired tag namespace, even though it is retired, is still an existing resource. Moving the retired tag namespace to a different compartment can enable you to delete its original containing compartment.

To move a tag namespace, you must be allowed to `manage tag-namespaces` in both compartments.

Deleting Tag Definitions

You can delete tag key definitions and tag namespaces.

When you delete a tag key definition, you begin a process that removes the tag from all resources in your tenancy. The delete action is asynchronous and initiates a work request. Once you start the delete operation, the state of the tag changes to deleting, and tag removal from resources begins. This process can take several hours depending on the number of resources that were tagged. When all tags have been removed, the state changes to deleted. You cannot restore a deleted tag. After the tag state changes to "deleted", you can use the same tag name again.

To delete a tag key definition, you must first retire it. To delete a tag namespace, you must first retire the tag namespace. When you retire a tag namespace that contains tag key definitions, all the tag keys in the namespace are retired, allowing you to delete the tag keys. The tag namespace can be deleted only after all its tag key definitions have been deleted.

Using Predefined Values

You can create a list of values and associate that list with a tag key definition. When users then apply the tag to a resource, they must select a value from the list of predefined values. Use lists of predefined values to impose limits on the values that users can apply to tags.

In the OCI CLI and Compute API, predefined values are specified through what is known as a *validator*. The validator is a parameter of a defined tag, used to supply the predefined values in JSON format. The Compute Web UI does not mention the term "validator", though it follows the same principle.

You can update existing tags to use predefined values. Every list of predefined values that you create must contain at least one value. Lists cannot contain duplicate values or blank entries. With predefined values, users applying tags cannot set the value of a tag to `null`.

You can use predefined values with defined tags and default tags. In combination with a default tag, users are required to select a tag value from the predefined list for each resource they create in the compartment where the default tag is used. This ensures that new resources contain metadata with values that you expect and can trust.

Using Tag Variables

You can use a variable to set the value of a defined tag. When users add the tag to a resource, the variable resolves to the data it represents. You can use tag variables in defined tags and default tags.

Consider the following example:

Operations.CostCenter="`${iam.principal.name}` at `${oci.datetime}`"

Operations is the namespace, CostCenter is the tag key, and the tag value contains two tag variables `${iam.principal.name}` and `${oci.datetime}`. When you add this tag to a resource, the variables resolve to your user name – the name of the principal who applied the tag – and a time stamp for when you added the tag.

```
user-name at 2021-04-18T14:32:57.604Z
```

The variable is replaced with data at the moment the tag is applied to a given resource. If you later edit the tag, the variable is gone and only the data remains. You can edit the tag value in all the ways you would edit any other tag value.

To create a tag variable, you must use a specific format. The following tag variables are supported:

Variable	Description
`${iam.principal.name}`	The name of the principal applying the tag to the resource.
`${iam.principal.type}`	The type of principal applying the tag to the resource.
`${oci.datetime}`	The date and time when the tag was created.

Tag Defaults

Tag defaults let you specify tags that are applied automatically to all resources at the time of creation in a specific compartment. This feature allows you to ensure that appropriate tags are applied at resource creation without requiring the user creating the resource to have access to the tag namespaces.

Tag defaults allow tenancy administrators to create secure permissions boundaries between users concerned with governance and users who need to create and manage resources. Tag defaults are defined for a specific compartment. A default tag is also applied to child compartments and the resources created within them. In the Compute Web UI, you manage tag defaults on the Compartment Details page.

To create or edit a tag default for a compartment, you must be granted the following combination of permissions:

- `manage tag-defaults` access for the compartment where you are adding the tag default
- `use tag-namespaces` access for the compartment where the tag namespace resides
- `inspect tag-namespaces` access for the tenancy

Tag defaults must include a tag value. If you use a default value, then you must create it as part of the tag default. This value is applied to all resources. Using a tag variable is allowed. If you specify that a user-applied value is required, then the user creating the resource must enter the value for the tag at the time of resource creation. Users cannot create resources without entering a value for the tag.

Tag defaults can be overridden at the time of resource creation by users who have the appropriate permissions to both create the resource and to use the tag namespace. Users with these permissions can also modify the default tags that were applied at resource creation at any later time.

You can define up to 5 tag defaults per compartment. After a tag default is created in a compartment, the default tag is applied to any new resources created in that compartment. Previously existing resources in the compartment are not tagged retro-actively. If you change the default value of the tag default, existing occurrences are not updated.

If you delete the tag default from the compartment, existing occurrences of the tag are not removed from resources. When you delete a tag key definition, existing tag defaults based on that tag key definition are not removed from the compartment. Until you delete the tag default in the compartment, it continues to count against your limit of 5 tag defaults per compartment.

Retired tags cannot be applied to new resources. Therefore, if the tag namespace or tag key specified in a tag default is retired, when new resources are created, the retired tag is not applied. As a best practice, you should delete the tag default that specifies the retired tag.

Tag-Based Access Control

Using conditions and a set of tag variables, you can write policy to scope access based on the tags that have been applied to a resource. Access can be controlled based on a tag that exists on the requesting resource or on the target resource or compartment of the request. Tag-based access control provides additional flexibility to your policies by allowing you to define access policies with tags that span compartments, groups, and resources.

> **Caution:**
>
> If your organization chooses to create policies that use tags to manage access, then ensure that you have appropriate controls in place to govern who can apply tags. Also, after policies are in place, keep in mind that applying tags to a group, user, or resource has the potential to confer access to resources.
>
> Before you create such a policy, ensure that you are aware of all the potential requestors and target resources that carry the tag.
>
> Before you apply a tag to a resource, ensure that you are aware of any policies in place that include the tag and could impact who has access to the resource.

You can control access based on the value of a tag applied to a requesting resource. More specifically, you can write policy statements to grant a user access based on the value of a tag that is assigned to the group to which the requesting user belongs.

You can also control access based on the value of a tag applied to a target resource, or a compartment that the target resource resides in.

By using tags in your policy statements to scope access, you can define access for multiple groups through a single policy statement. You can also confer access and revoke access for groups by applying or removing tags, without changing the original policies.

The Oracle Private Cloud Appliance User Guide provides a detailed description and examples of how to use tag variables and operators in your policy syntax. Refer to the section "Writing Policy Statements" in the chapter Identity and Access Management.

Important Notes and Limitations

This section describes a number of items to take into account when you start writing policies to control permissions based on resource tags.

Item	Description
Permissions to list resources must be granted separately	Policies that scope access based on the tag applied to the target resource cannot allow the permissions that enable you to return a list of resources. Therefore, permissions to allow listing a resource must be granted through an additional policy statement. This approach improves the tag-based policy because it allows users to see the resource they want to manage, but still limits the permissions to only inspect. Another approach you can use to avoid this limitation is to tag the compartments that contains the resources you want to grant access to.
Policies requiring a tag on the target resource cannot grant create permissions	When you write a policy to scope access based on the value of a tag on the resource, keep in mind that the policy cannot grant create permissions. For example, a request to create an instance would fail because the target resource has not been created yet and therefore does not have the appropriate tag to be evaluated. Using and deleting instances with that tag would be allowed, but not creating instances.

Item	Description
Limitations on characters in tag namespaces and tag key definitions used in policy variables	Tag namespaces and tag key definitions support a broader set of characters than are allowed in policy variables. Therefore, to you use tag namespaces and tag key definitions in variables, ensure that they only include the characters also supported by policy variables. Supported characters are: `a-z, A-Z, 0-9, _, @, -, :` If your tag namespaces or tag keys have characters other than these, you cannot use them in policy variables. Tag namespaces and tag keys cannot be renamed, so you will have to create new tag namespaces and tag key definitions.
Considerations for case	When working with tags in policies, be aware that tag values are case insensitive.
Compartment hierarchies	When you write a condition to allow access based on the tag applied to a target compartment, remember that this policy also allows access to all compartments nested inside the tagged compartment. All subcompartments in the tagged compartment are resources in the tagged compartment, and therefore the policy grants access.
Supported services	All services support the `request.principal.group`, and `target.resource.compartment.tag` policy variables. Not all services support the `target.resource.tag` policy variable. The supported services are: Compute, Networking, IAM, DNS, File Storage, Block Storage.

Example Using Tags Applied to a Group

Following is an example that demonstrates how you can use tags applied to user groups to manage access to resources in a compartment.

Assume you have three compartments: ProjectA, ProjectB, and ProjectC. For each compartment, there is an admin group set up: A-Admins, B-Admins, and C-Admins. Each admin group has full access over their compartment through the following policies:

```
allow group A-Admins to manage all-resources in compartment ProjectA
allow group B-Admins to manage all-resources in compartment ProjectB
allow group C-Admins to manage all-resources in compartment ProjectC
```

Your organization has set up the following tag namespace and key to tag each group by its role: `EmployeeGroup.Role`. Some values for this tag are 'Admin', 'Developer', 'Test-Engineer'. All your admin groups are tagged with `EmployeeGroup.Role='Admin'`.

Now you want to set up a Test compartment for members of the three projects to share. You want to give Admin access to all three of your existing admin groups. To accomplish this, you can write a policy like:

```
allow any-group to manage all-resources in compartment Test where
request.principal.group.tag.EmployeeGroup.Role='Admin'
```

With this policy, all of your existing admin groups with the tag will have access to the Test compartment. Also, any new or existing groups that you tag with `EmployeeGroup.Role='Admin'` in the future will have access to the Test compartment without having to update the policy statements.

Example Using Tags Applied to a Target Resource's Compartment

In this example, each of your organization's project compartments contains two child compartments: Test and Prod. You want to give your test engineers access to the test compartments across all three projects. To accomplish this, you create a tag, `ResourceGroup.Role='Test'`, and apply this to the Test compartments in each of your project compartments.

To allow your group Testers to access the resources across all three test compartments, you can then use a policy like:

```
allow group Testers to use all-resources in tenancy where
target.resource.compartment.tag.ResourceGroup.Role='Test'
```

6
Virtual Networking Overview

When building a private cloud, one of the first steps is to design and configure a virtual cloud network for your cloud resources. This chapter provides an overview of the components made available to you by the Networking service. Several of them are virtual versions of the traditional network components you might already be familiar with.

The sections in this chapter explain how the Networking service works and how you use it to design your virtual cloud network, connect it with your on-premises network, control traffic that flows through it, and so on. The detailed descriptions of typical network scenarios might provide a good starting point for your own configuration.

Virtual Cloud Network

A virtual cloud network, or VCN, is a software-defined equivalent of a traditional network, with firewall rules and various types of communication gateways. A VCN resides in the single region of your Private Cloud Appliance installation and covers one contiguous CIDR block of your choice.

The size of a VCN is /16 to /30. The CIDR block can **NOT** be changed after the VCN is created. The maximum number of private IPs within a VCN is 64,000. Oracle recommends using a private IP address range as specified in RFC 1918 (10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16). This documentation uses the term private IP address when referring to IP addresses in your VCN's CIDR.

You can privately connect a VCN to another VCN so that traffic does not leave the secure network environment of the appliance. However, the CIDRs of the two VCNs must not overlap. The concept of privately connecting VCNs is called *peering*. It involves setting up a type of virtual router known as a Local Peering Gateway. For more information about the use of gateways in your VCNs, see Network Gateways.

Subnets

A VCN is subdivided into subnets. Each subnet in a VCN consists of a contiguous range of IPv4 addresses that do not overlap with other subnets in the VCN. Example: 172.16.1.0/24. The first two IPv4 addresses and the last in the subnet's CIDR are reserved by the Networking service. You cannot change the size of the subnet after creation.

Subnets act as a unit of configuration: all instances in a given subnet use the same route table, security lists, and DHCP options. Subnets can be either public or private. This is defined when the subnet is created, and cannot be changed later. In a private subnet, instances cannot be assigned a public IP address.

You can think of a compute instance as residing in a subnet. However, to be precise, each instance is attached to a virtual network interface card (VNIC), which in turn resides in the subnet and enables a network connection for that instance.

IPv6 addressing is currently not supported in Private Cloud Appliance.

Route Tables

Each VCN automatically comes with an empty default route table. Subnets use the default route table of the parent VCN, unless you explicitly assign them a different route table. When you add route rules to your VCN, you can simply add them to the default table if that suits your needs. However, if you need both a public subnet and a private subnet, you instead create a separate custom route table for each subnet.

The primary routing scenario is for sending a subnet's traffic to destinations outside the VCN. A subnet has a single route table of your choice associated with it at the time of creation. All VNICs in that subnet are subject to the rules in the route table. Traffic within the VCN is automatically handled by the VCN internal routing. No route rules are required to enable that traffic. You can change which route table the subnet uses at any time. You can also edit a route table's rules, or remove all the rules from the table.

A route rule specifies a destination CIDR block and the target, or the next hop, for any traffic that matches that CIDR. When selecting the target you also specify its compartment. Supported target types for a route rule are the different VCN gateways.

Route rules must be configured carefully to ensure that the network traffic reaches the intended destination and is not dropped. Moving route rules between compartments is not supported.

Public Network in Private Cloud

Oracle Private Cloud Appliance is a *private* cloud solution. The infrastructure that provides the necessary services to deploy your cloud workloads, is configured to operate within the network environment of your data center. During initialization, the appliance core network components are integrated with your existing data center network design. Uplink ports in the appliance switches connect to your next-level data center switches to provide a redundant high-speed and high-bandwidth physical connection that carries all traffic into and out of the appliance.

This is a critical difference with a *public* cloud environment, where the infrastructure is managed by your service provider, who grants you restricted access to systems in their data center. Because your cloud resources are not inside your own network, you access them over the internet, using a secure tunneling connection. In this context, network traffic between your public cloud and locations external to it, is sent over the internet by definition. In other words, from the perspective of your cloud environment, the public network is effectively the internet. Compare this with the private cloud environment, where the public network is your on-premises network, and internet access is always indirect through your data center edge router.

Private Cloud Appliance is modeled after Oracle's public cloud offering: Oracle Cloud Infrastructure. This applies not only to the Networking service but to all the major infrastructure services: they have been optimized for the smaller scale of a single rack, while offering the same core functionality. One of the main objectives is to maintain maximum compatibility between both cloud platforms: this provides a very similar user experience and allows you to efficiently migrate workloads between your private cloud environment and your Oracle Cloud Infrastructure account.

Because the public cloud network model is applied to a private cloud environment, where the public network is actually the data center network, certain network resources or components present a different behavior. You need to be aware of these

differences when designing and configuring the virtual cloud networks in Private Cloud Appliance.

- **Access to the on-premises network**

  The appliance rack is located at your premises: it is set up inside your data center and connected directly to your on-premises network. There is no need for a secure tunnel over the internet to allow your cloud resources and your on-premises network to communicate with each other. Access is enabled through a gateway between your virtual cloud network (VCN) and your on-premises network.

- **Access to the internet**

  Resources in your cloud environment have no direct internet access. In contrast with a public cloud environment, no gateway is capable of enabling direct internet connectivity for a VCN. Public connectivity implies that resources have access to the data center network. The configuration of the networking components in the data center determines how cloud resources can connect to the internet and whether they can be reached from the internet.

- **VCN gateways**

  To manage network traffic into and out of your VCNs several types of gateways are used. Resources in a VCN can connect to external hosts through either a dynamic routing gateway (DRG), a NAT gateway or an internet gateway. Technically, each of these gateways provides a path to your on-premises network. However, a DRG has an important limitation in that it performs no address translation and thus cannot handle any overlap between VCNs and the on-premises network.

  Although these gateways seem interchangeable to a certain extent, it is important to configure and use them strictly for their intended purpose. It helps to ensure that your private cloud network configuration remains compatible with the public cloud network model. If you move a workload into Oracle Cloud Infrastructure, the data center network is removed from the equation, meaning the NAT and internet gateways will provide direct internet access, and a DRG will only provide access to your on-premises network.

- **Public IP addresses**

  From the perspective of a virtual cloud network, the public network is effectively the data center network, so a public IP address must be within its CIDR. To accommodate this, you must reserve an address range within the data center CIDR to be used exclusively by Private Cloud Appliance. You typically configure the public IP range during initial system setup. You can extend the range at a later time, but removing IPs is not allowed. The Networking service uses this range as a pool, from which it assigns public IP addresses to cloud resources that require them. A public IP makes a resource reachable from outside the VCN it resides in. In a private cloud context, it allows interaction with other resources in the data center or the on-premises network. The IPs that are considered "public" are really part of the data center's private range.

  The use of public IPs has different implications when you move a particular workload into Oracle Cloud Infrastructure, where public IPs are truly unique and publicly routable. True public IPs are scarce and expensive resources. Take this into account when designing applications and services in a private cloud environment: use private IPs to enable connectivity between components and assign public IPs only where true public connectivity is required.

Instance Connectivity

This section discusses different aspects of instance connectivity.

Virtual Network Interface Cards (VNICs)

The compute nodes in the Private Cloud Appliance have physical network interface cards (NICs). When you create and launch a compute instance on one of the servers, the Networking service ensures that a VNIC is created on top of a physical interface, so that the instance can communicate over the network. Each instance has a primary VNIC that is automatically created and attached during launch. The primary VNIC resides in the subnet you specify when creating the instance. It cannot be removed from the instance.

A VNIC enables an instance to connect to a VCN and determines how the instance communicates with endpoints inside and outside of the VCN. Each VNIC resides in a subnet within a VCN and includes these items:

- One primary private IPv4 address from the subnet the VNIC is in.
- Up to 31 optional private IPv4 addresses from the subnet the VNIC is in.
- An optional public IPv4 address for each private IP, assigned at your discretion.
- An optional host name for DNS for each private IP address.
- A MAC address.
- A flag to enable or disable the source/destination check on the VNIC's network traffic.
- Optional membership of one or more network security groups (NSGs).
- An Oracle-assigned identifier (OCID).
- An optional friendly name you can choose and assign.

You can add secondary VNICs to an instance after it is launched. To be able to use a secondary VNIC, you must also configure the instance OS for it. The maximum number of VNICs for an instance varies by shape. Each secondary VNIC can be in a different subnet than the primary VNIC, either within the same VCN or a different one. A secondary VNIC can also be in the same subnet as the primary VNIC. However, attaching multiple VNICs from the same subnet CIDR block to an instance can introduce asymmetric routing, especially on Linux instances.

> **Note:**
>
> To avoid asymmetric routing in a configuration with multiple IP addresses from the same subnet, Oracle recommends assigning multiple private IP addresses to one VNIC, or using policy-based routing.
>
> If traffic comes in to a service on the instance through a secondary VNIC and the service replies, the reply packets automatically have that VNIC's IP address as the source IP address. Policy-based routing is required for that reply to go back out on the same interface and find the correct default gateway.

Secondary VNICs must always be attached to an instance and cannot be moved. The process of creating a secondary VNIC automatically attaches it to the instance. The process of detaching a secondary VNIC automatically deletes it. They are

automatically detached and deleted when you terminate the instance. An instance's bandwidth is fixed regardless of the number of VNICs attached. You cannot specify a bandwidth limit for a particular VNIC on an instance.

By default, every VNIC performs the source/destination check on its network traffic. The VNIC looks at the source and destination listed in the header of each network packet. If the VNIC is not the source or destination, then the packet is dropped. If the VNIC needs to forward traffic – for example, if it needs to perform Network Address Translation (NAT) – you must disable the source/destination check on the VNIC.

VNICs reside in a subnet but attach to an instance. The VNIC's attachment to the instance is a separate object from the VNIC or the instance itself. The VNIC and subnet always exist together in the same compartment, but the VNIC's attachment to the instance always exists in the instance's compartment. This distinction affects your IAM policies if you set up an access control scenario where network administrators manage the network and other users manage instances.

VNICs are rate limited to prevent very active applications from reducing the available bandwidth of other applications to unacceptable levels. Rate limiting applies to each instance’s VNIC interface.

The rate is limited to a value below the maximum network bandwidth associated with each instance shape. Generally, the bandwidth limit increases with the maximum number of VNICs a shape uses, but there is no maximum bandwidth guarantee.

Users cannot specify the bandwidth limits directly. Rate limiting is a system function. The rate applied to each VNIC is based on the shape option during instance launch.

IP Addressing

Instances use IP addresses for communication. Each instance has at least one private IP address and optionally one or more public IP addresses. A private IP address enables the instance to communicate with other instances inside the VCN, or with hosts in your on-premises network. A public IP address enables the instance to communicate with hosts outside of the Private Cloud Appliance network environment.

Certain types of resources in your tenancy are designed to be directly reachable from outside the secure appliance network environment, and therefore automatically come with a public IP address. For example: a NAT gateway. Other types of resources are directly reachable only if you configure them to be. For example: specific instances in your VCN. Direct public connectivity also requires that the VCN has an internet gateway and that the public subnet has correctly configured route tables and security lists.

Private IPs

The Networking service defines a private IP object, identified by an Oracle-assigned OCID, which consists of a private IPv4 address and optional host name for DNS. Each instance receives a primary private IP object during launch. The Networking service uses DHCP to assign a private IP address. This address does not change during the instance's lifetime and cannot be removed from the instance. The private IP object is terminated when the instance is terminated.

If an instance has any secondary VNICs attached, each of those VNICs also has a primary private IP. A private IP can have a public IP assigned to it at your discretion. It is not supported to use a private IP as the target of a route rule in your VCN.

About Secondary Private IPs

After an instance is launched, you can add a *secondary private IP* to one of its VNICs. You can add it to either the primary VNIC or a secondary VNIC on the instance. The secondary private IP address must be in the subnet to which the VNIC belongs. You can move a secondary private IP from a VNIC on one instance to a VNIC on another instance on condition that both VNICs belong to the same subnet.

Typical use cases for a secondary private IP include:

- **Instance failover:** You assign a secondary private IP to an instance. If a problem occurs with the instance, you can easily reassign its secondary private IP to a standby instance in the same subnet. In additon, if the secondary private IP has an associated public IP, that public IP moves along with the private IP.
- **Running multiple services or endpoints on a single instance:** For example, you could have an instance that runs multiple container pods, where each pod uses its own IP address from the VCN's CIDR. The containers have direct connectivity to other instances and services in the VCN. Another example: you could run multiple SSL web servers with each one using its own IP address.

Secondary private IP addresses have the following properties:

- They are supported for all shapes and OS types.
- They can be assigned only after the instance is launched, or the secondary VNIC is created and attached.
- They are automatically deleted when you terminate the instance or detach and delete the secondary VNIC.
- Deleting a secondary private IP from a VNIC returns the address to the pool of available addresses in the subnet.
- A VNIC can have a maximum of 31 secondary private IPs.
- The instance's bandwidth is fixed regardless of the number of private IP addresses attached. You cannot specify a bandwidth limit for a particular IP address on an instance.
- A secondary private IP can have a reserved public IP associated with it at your discretion.

> **Note:**
>
> After assigning a secondary private IP to a VNIC, you must configure the instance OS to use it. For Linux and Microsoft Windows instructions, refer to the Oracle Private Cloud Appliance User Guide.

Public IPs

In a private cloud model, public IP addresses are not true unique and publicly routable internet IPs; they are just addresses external to the VCNs in the cloud environment. During initial setup of the appliance, a pool of addresses from the on-premises network is reserved for use as public IPs. A small set of these public IPs is required for system use.

You can assign a public IP address to an instance to enable external communication. The instance is assigned a public IP address from an address pool. Technically, the assignment is to a private IP object on the instance, and the VNIC that the private IP is assigned to must reside in a public subnet. A given instance can have multiple secondary VNICs; each with a (primary) private IP. Consequently, you can assign a given instance multiple public IPs across their VNICs.

The Networking service defines a public IP object, identified by an Oracle-assigned OCID, which consists of a private IPv4 address and additional properties that further define the public IP's type and behavior. There are two types of public IPs:

- An **ephemeral** public IP is temporary and exists for the lifetime of the instance.
- A **reserved** public IP is persistent and exists beyond the lifetime of the instance it is assigned to. It can be unassigned and then reassigned to another instance.

The following table summarizes the differences between both types:

Characteristics	Ephemeral Public IP	Reserved Public IP
Allowed assignment	to a VNIC's primary private IP limits: one per VNIC, two per instance	to a VNIC's primary private IP limit: one per VNIC
Creation	Optionally created and assigned during instance launch or secondary VNIC creation. You can create and assign one later if the VNIC doesn't already have one.	You create one at any time. You can then assign it when you like.
Unassignment	You can unassign it at any time, which deletes it. You might do this if whoever launched the instance included a public IP, but you do not want the instance to have one. When you stop an instance, its ephemeral public IPs remain assigned to the instance.	You can unassign it at any time, which returns it to your tenancy's pool of reserved public IPs.
Moving to a different resource	You cannot move an ephemeral public IP to a different private IP.	You can move it at any time by unassigning and then reassigning it to another private IP. Can be in a different VCN or availability domain.
Automatic deletion	Its lifetime is tied to the private IP's lifetime. Automatically unassigned and deleted when: • its private IP is deleted • its VNIC is detached or terminated • its instance is terminated	Never. Exists until you delete it.
Scope	Availability domain	Regional (can be assigned to a private IP in any availability domain in the region)
Compartment and availability domain	Same as the private IPs	Can be different from the private IPs

When you launch an instance in a public subnet, the instance gets a public IP by default, unless you specify otherwise. After you create a given public IP, you cannot change which

type it is. For example, if you launch an instance that is assigned an ephemeral public IP with address 203.0.113.2, you cannot convert the ephemeral public IP to a reserved public IP with address 203.0.113.2.

Resources that are designed to be directly publicly reachable automatically get a public IP address assigned from a pool upon creation. For NAT gateways, the assigned address is a regional ephemeral public IP. Like other ephemeral public IPs, it is automatically unassigned and deleted when you terminate its assigned resource. However, unlike other ephemeral public IPs, you cannot edit it or unassign it yourself.

DHCP Options

The Networking service uses DHCP to automatically provide configuration information to instances when they boot up. Although DHCP lets you change some settings dynamically, others are static and never change. For example, when you launch an instance, either you specify the instance's private IP address or the system chooses one for you. Each time the instance boots up or you restart the instance's DHCP client, DHCP passes that same private IP address to the instance. The address never changes during the instance's lifetime.

The Networking service provides DHCP options to let you control certain types of configuration on the instances in your VCN. You can change the values of these options at your discretion, and the changes take effect the next time you restart a given instance's DHCP client or reboot the instance.

Each subnet in a VCN can have a single set of DHCP options associated with it. That set of options applies to all instances in the subnet. Each VCN comes with a default set of DHCP options with initial values that you can change. Unless you create and assign a different set of DHCP options, every subnet uses the VCN's default set.

These are the available DHCP options you can configure:

- **Domain Name Server**

  The default setting is DNS Type = Internet and VCN Resolver.

  If instead you set DNS Type = Custom Resolver, you can specify up to three DNS servers of your choice.

- **Search Domain**

  If you set up your VCN with a DNS label, the default value for the Search Domain option is the VCN domain name: ***<VCN_DNS_label>***`.oraclevcn.com`. Otherwise, the Search Domain option is not present in the default set of DHCP options.

  In general, when any set of DHCP options is initially created – the default set or a custom set –, the Networking service automatically adds the Search Domain option and sets it to the VCN domain name (***<VCN_DNS_label>***`.oraclevcn.com`) if all of these are true:

  – The VCN has a DNS label.

  – DNS Type = Internet and VCN Resolver.

  – You did NOT specify a search domain of your choice during creation of the set of DHCP options.

  After the set of DHCP options is created, you can always remove the Search Domain option or set it to a different value. You can specify only a single search domain in a set of DHCP options.

Important notes about your instances and DHCP options:

- Whenever you change the value of one of the DHCP options, you must either restart the DHCP client on the instance, or reboot the instance for the changes to take effect. This applies to all existing instances in the subnets associated with that set of DHCP options.
- Keep the DHCP client running so you can always access the instance. If you stop the DHCP client manually or disable NetworkManager, the instance cannot renew its DHCP lease and will become inaccessible when the lease expires; typically within 24 hours. Do not disable NetworkManager unless you use another method to ensure renewal of the lease.
- Stopping the DHCP client might remove the host route table when the lease expires. Also, loss of network connectivity to your iSCSI connections might result in loss of the boot drive.
- Any changes you make to the `/etc/resolv.conf` file are overwritten whenever the DHCP lease is renewed or the instance is rebooted.
- Changes you make to the `/etc/hosts` file are overwritten whenever the DHCP lease is renewed or the instance is rebooted. To persist your changes to the `/etc/hosts` file in Oracle Linux or CentOS instances, add the following line to `/etc/oci-hostname.conf`: `PRESERVE_HOSTINFO=2`. If the `/etc/oci-hostname.conf` file does not exist, create it.

Name Resolution

The Domain Name System (DNS) translates human-readable domain names to machine-readable IP addresses. For example, when you type a domain name into your browser, your operating system queries several DNS name servers until it finds the authoritative name server for that domain. The authoritative name server then responds with an IP address or other requested record data; the answer is relayed back to your browser and the DNS record is resolved to the web page.

For DNS name resolution within your VCN, there are two options. You choose an option for each subnet in the VCN, using the subnet's set of DHCP options.

The default choice is the **Internet and VCN Resolver**, an Oracle-provided solution. It consists of two parts: the Internet Resolver and the VCN Resolver. The Internet Resolver lets instances resolve host names that are publicly published on the internet, without requiring the instances to have internet access by way of either an internet gateway or a connection to the on-premises network. The VCN resolver lets instances resolve host names, which you can assign, of other instances in the same VCN.

Alternatively, you can use a **Custom Resolver**. It allows you to use up to three DNS servers of your choice for name resolution. These could be DNS servers that are available through the internet, in your VCN, or in your on-premises network, which is connected to your VCN through a DRG.

Domains and Host Names

When you create a VCN and subnets, you can specify DNS labels for each. Subnet DNS labels can only be set if the VCN itself is created with a DNS label. The labels, along with the parent domain of `oraclevcn.com` form the VCN domain name and subnet domain name. Next, when you launch an instance, you can assign a host name. It is assigned to the primary VNIC that is automatically created during instance launch. Along with the subnet domain name, the host name forms the instance's fully qualified domain name (FQDN).

- **VCN domain name:** ***<VCN_DNS_label>***`.oraclevcn.com`

- **Subnet domain name:** ***<subnet_DNS_label>.<VCN_DNS_label>***`.oraclevcn.com`
- **Instance FQDN:**
  <host_name>.<subnet_DNS_label>.<VCN_DNS_label>`.oraclevcn.com`

The FQDN resolves to the instance's private IP address. The Internet and VCN Resolver also enables reverse DNS lookup, which lets you determine the host name corresponding to the private IP address.

If you add a secondary VNIC to an instance, you can specify a host name. The resulting FQDN resolves to the VNIC's primary private IP address. The resulting FQDN resolves to that private IP address.

DNS labels and host names must meet these requirements:

- VCN and subnet labels must start with a letter and have a maximum length of 15 alphanumeric characters. Hyphens and underscores are not allowed. The value cannot be changed later.
- Host names can be up to 63 characters in length and must be compliant with RFCs 952 and 1123. The value can be changed later.
- VCN DNS labels must be unique across the VCNs in your tenancy. Subnet DNS labels must be unique within the VCN, and host names must be unique within the subnet.

Host Name Validation and Generation

If you set DNS labels for VCN and subnet, the host name is validated for compliance during instance launch. If you have not specified a host name, the system attempts to use the instance display name. If the display name does not pass validation, or if you did not specify one, the system generates a DNS-compliant host name. The generated host name is displayed on the instance’s detail page in the Compute Web UI.

If you launch an instance using the CLI or SDK, and you have not specified a host name or display name, the system does not generate them for you. This means the instance will not be resolvable using the Internet and VCN Resolver.

If you add a secondary VNIC to an instance, the system never generates a host name. You must provide a valid host name if you want the private IP address to be resolvable using the Internet and VCN Resolver.

Public DNS Zones

To enable DNS name resolution for instances deployed inside your VCNs, from outside the appliance network environment, Private Cloud Appliance provides configuration support for public DNS zones. Within your tenancy you create the DNS zones, or sections of the DNS namespace, you intend to manage through the Compute Web UI. The edge network of the appliance handles all DNS queries for the domain(s) in question.

When creating a DNS zone, you specify the name of the domain it manages – for example: *example.com*. You select whether the zone is primary or secondary. A primary zone contains its own DNS records, while a secondary zone retrieves its records from another zone. To access the external zone's records, the secondary zone needs at least one server IP address for the external zone. In addition, a TSIG (Transaction Signature) key may be required. TSIG keys are shared secrets used for

authentication of secondary DNS zones. You can store these keys in the compartment of your choice.

Each DNS zone you create automatically contains two essential records:

- The SOA (Start of Authority) record specifies authoritative information about the DNS zone. This information includes the primary name server, the domain administrator email address, the domain serial number, and several timers related to refreshing the zone. For more information about SOA records, see RFC 1035.
- The NS (Name Server) record lists the authoritative name servers for a zone. For more information about NS records, see RFC 1035.

You configure the DNS zone by adding specific domain information in the form of resource records. For example, using an address record, you make a domain name resolve to the public IP address of an instance in a public subnet of a VCN. The public DNS zones in Private Cloud Appliance support the resource record types described in the table below.

Resource Record Type	Description
A	An address record used to point a host name to an IPv4 address. For more information about A records, see RFC 1035.
AAAA	An address record used point a host name at an IPv6 address. For more information about AAAA records, see RFC 3596.
ALIAS	A private pseudo-record that allows CNAME functionality at the apex of a zone.
CAA	A Certification Authority Authorization record allows a domain name holder to specify one or more Certification Authorities authorized to issue certificates for that domain. For more information about CAA records, see RFC 6844.
CDNSKEY	A Child DNSKEY moves a CDNSSEC key from a child zone to a parent zone. The information provided in this record must match the CDNSKEY information for your domain at your other DNS provider. This record is automatically created if you enable DNSSEC on a primary zone in Private Cloud Appliance DNS. For more information about CDNSKEY, see RFC 7344.
CDS	A Child Delegation Signer record is a child copy of a DS record, for transfer to a parent zone. For more information about CDS records, see RFC 7344.
CERT	A Certificate record stores public key certificates and related certificate revocation lists in the DNS. For more information about CERT records, see RFC 2538 and RFC 4398.
CNAME	A Canonical Name record identifies the canonical name for a domain. For more information about CNAME records, see RFC 1035.
CSYNC	A Child-to-Parent Synchronization record syncs records from a child zone to a parent zone. For more information about CNAME records, see RFC 7477.
DHCID	A DHCP identifier record provides a way to store DHCP client identifiers in the DNS to eliminate potential host name conflicts within a zone. For more information about DHCID, see RFC 4701.
DKIM	A Domain Keys Identified Mail is a special TXT record set up specifically to supply a public key used to authenticate arriving mail for a domain. For more information about DKIM records, see RFC 6376.
DNAME	A Delegation Name record has similar behavior to a CNAME record, but allows you to map an entire subtree beneath a label to another domain. For more information about DNAME records, see RFC 6672.

Resource Record Type	Description
DNSKEY	A DNS Key record documents public keys used for DNSSEC. The information in this record must match the DNSKEY information for your domain at your other DNS provider. For more information about DNSKEY records, see RFC 4034.
DS	A Delegation Signer record resides at the top-level domain and points to a child zone's DNSKEY record. DS records are created when DNSSEC security authentication is added to the zone. For more information about DS records, see RFC 4034.
IPSECKEY	An IPSec Key record stores public keys for a host, network, or application to connect to IP security (IPSec) systems. For more information on IPSECKEY records, see RFC 4025.
KEY	A Key record stores a public key that is associated with a domain name. Currently only used by SIG and TKEY records. IPSECKEY and DNSKEY have replaced key for use in IPSec and DNSSEC, respectively. For more information about KEY records, see RFC 4025.
KX	A Key Exchanger record identifies a key management agent for the associated domain name with some cryptographic systems (not including DNSSEC). For more information about KX records, see RFC 2230.
LOC	A Location record stores geographic location data of computers, subnets, and networks within the DNS. For more information about LOC records, see RFC 1876.
MX	A Mail Exchanger record defines the mail server accepting mail for a domain. MX records must point to a host name. MX records must not point to a CNAME or IP address. For more information about MX records, see RFC 1035.
NS	A Nameserver record lists the authoritative nameservers for a zone. NS records are automatically generated at the apex of each new primary zone you create. For more information about NS records, see RFC 1035.
PTR	A Pointer record reverse maps an IP address to a hostname. This behavior is the opposite of an A Record, which forward maps a hostname to an IP address. PTR records are commonly found in reverse DNS zones. For more information about PTR records, see RFC 1035.
PX	A resource record used in X.400 mapping protocols. For more information about PX records, see RFC 822 and RFC 2163.
SOA	A Start of Authority record specifies authoritative information about a DNS zone, including: • the primary nameserver • the email of the domain administrator • the domain serial number • several timers relating to refreshing the zone An SOA record is generated automatically when a zone is created. For more information about SOA records, see RFC 1035.
SPF	A Sender Policy Framework record is a special TXT record used to store data designed to detect email spoofing. For more information about SPF records, see RFC 4408.
SRV	A Service Locator record allows administrators to use several servers for a single domain. For more information about SRV records, see RFC 2782.

Resource Record Type	Description
SSHFP	An SSH Public Key Fingerprint record publishes SSH public host key fingerprints using the DNS. For more information about SSHFP records, see RFC 6594.
TLSA	A Transport Layer Security Authentication record associates a TLS server certificate, or public key, with the domain name where the record is found. This relationship is called a TLSA certificate association. For more information about TLSA records, see RFC 6698.
TXT	A Text record holds descriptive, human readable text, and can also include non-human readable content for specific uses. It is commonly used for SPF records and DKIM records that require non-human readable text items. For more information about TXT records, see RFC 1035.

Traffic Management Steering Policies

Steering Policies enable you to configure policies to serve intelligent responses to DNS queries, meaning different answers (endpoints) may be served for the query depending on the logic defined in the policy. Private Cloud Appliance supports these types of traffic management steering policies:

Policy Type	Description
Load Balancer	Load Balancer policies allow distribution of traffic across multiple endpoints. Endpoints can be assigned equal weights to distribute traffic evenly across the endpoints, or custom weights may be assigned for ratio load balancing.
IP Prefix Steering	IP Prefix steering policies enable you to steer DNS traffic based on the IP Prefix of the originating query. You can divide users into groups based on the subnets from where requests originate, and steer them to specific resources based on the subdivision you made. For example, you can serve different responses for your internal users as opposed to external users.

A steering policy contains rules to answer DNS queries. You use those rules to filter answers based on the properties of the DNS request. When multiple answers are served in response to DNS queries, that group of answers is called an answer pool. The answers within a pool are sorted by priority and are marked eligible or ineligible. Ineligible answers are omitted from the response.

When attached to a DNS zone, a steering policy takes precedence over all resource records of the zone it covers, and causes DNS responses to be constructed from the steering policy rules instead. For example, if a steering policy attached to the *example.com* DNS zone contains a rule covering the domain *application.example.com* and an answer for the A (address) record type, then the steering policy responds with that answer regardless of any relevant A records that exist in the zone. If a steering policy has no answer for the record type being requested, the DNS request is passed on to the next steering policy or the base zone records.

Steering policies only support records of the types A, AAAA, CNAME. A domain can have at most one steering policy attachment covering a given record type. This means a DNS zone (*example.com*) may have multiple attached steering policies covering different record types

for a given domain – for example, one A record policy and one CNAME record policy for *application.example.com*. A DNS zone may also have multiple attached steering policies covering a given record type for different domains – for example, an A record policy for *application.example.com* and an A record policy for *database.example.com*. However, multiple steering policies for the same domain and record type are not supported, because the answer can be provided by only one policy.

Network Gateways

To manage traffic to and from a VCN, you can add optional virtual routers, which provide additional functions. You set up rules in route tables to route traffic from the subnets through a gateway to destinations outside the VCN. This section describes the role and usage of each gateway type.

Dynamic Routing Gateway

A dynamic routing gateway, or DRG, provides a path for private network traffic between your VCN and an on-premises network. In the context of Private Cloud Appliance, this traffic is routed to the data center network and on to its destination. The data center network is considered a public network because it is external to the appliance environment. However, no form of tunneling is required since traffic does not travel over the internet. This is a significant difference with public cloud environments.

For the purpose of access control, when creating a DRG, you must specify the compartment where you want the DRG to reside. If you are not sure which compartment to use, put the DRG in the same compartment as the VCN.

A DRG is a standalone object. To use it, you must attach it to a VCN. In the API, the process of attaching creates a DRG Attachment object with its own OCID. To detach the DRG, you delete the attachment object.

A VCN can be attached to only one DRG at a time, though a DRG can be attached to multiple VCNs. You can detach a DRG and reattach it at any time. After attaching a DRG, you must update the routing in the VCN to use the DRG. Otherwise, traffic from the VCN will not flow to the DRG.

The basic routing scenario sends traffic from a subnet in the VCN to the DRG. When sending traffic from the subnet to your on-premises network, you set up a rule in the subnet's route table. The rule's destination CIDR is the CIDR for the on-premises network or a subnet within, and the rule's target is the DRG.

NAT Gateway

A NAT gateway gives cloud resources without a public IP access to the on-premises network, which is an external public network from the point of view of a VCN, without exposing those resources. You create a NAT gateway in the context of a specific VCN, so the gateway is automatically attached to that VCN upon creation. The gateway allows hosts to initiate connections to the on-premises network and receive responses, but prevents them from receiving inbound connections initiated from the on-premises network. NAT gateways are highly available and support TCP, UDP, and ICMP ping traffic. The Networking service automatically assigns a public IP address to the NAT gateway. You cannot choose its public IP address.

When a host in the private network initiates a connection to the on-premises network, the NAT device's public IP address becomes the source IP address for the outbound

traffic. The response traffic from the on-premises network therefore uses that public IP address as the destination IP address. The NAT device then routes the response back to the private network, to the host that initiated the connection.

VCN routing is controlled at the subnet level, so you can specify which subnets use a NAT gateway. Private Cloud Appliance only supports one NAT gateway per VCN.

For the purpose of access control, when creating a NAT gateway, you must specify the compartment where you want the gateway to reside. If you are not sure which compartment to use, put the gateway in the same compartment as the VCN.

By default, a NAT gateway allows traffic at the time of creation. However, you can block or allow traffic through the gateway at any time. Blocking a NAT gateway prevents all traffic, regardless of any existing route rules or security rules in the VCN.

Internet Gateway

An internet gateway connects the edge of the VCN with the on-premises network. The ultimate target of the traffic routed through an internet gateway may be the internet. However, in a private cloud model, the internet gateway routes traffic to the on-premises network. Traffic to and from the internet is then managed by the routing configuration in the on-premises network.

You create an internet gateway in the context of a specific VCN, so the gateway is automatically attached to that VCN upon creation. To use the gateway, the hosts on both ends of the connection must have public IP addresses for routing. Connections that originate in your VCN and are destined for a public IP address, either inside or outside the VCN, go through the internet gateway. Connections that originate outside the VCN and are destined for a public IP address inside the VCN also go through the internet gateway.

A given VCN can have only one internet gateway. You control which public subnets in the VCN can use the gateway by configuring the subnet's associated route table. The public subnet's security list rules ultimately determine the specific types of traffic that are allowed to and from the resources in the subnet. An internet gateway can be disabled, meaning no traffic flows to or from the internet, regardless of any existing route rules that enable such traffic.

For the purpose of access control, when creating an internet gateway, you must specify the compartment where you want the gateway to reside. If you are not sure which compartment to use, put the gateway in the same compartment as the VCN.

Local Peering Gateway

VCN peering is the process of connecting multiple virtual cloud networks (VCNs) so that resources can communicate using private IP addresses. You can use VCN peering to divide your network into multiple VCNs, for example, based on departments or lines of business, with each VCN having direct private access to the others. You can also place shared resources into a single VCN that all the other VCNs can access privately. Two peered VCNs can be in the same tenancy or different ones.

The following components are required to set up a peering connection:

- Two VCNs with non-overlapping CIDRs
- A local peering gateway (LPG) on each VCN in the peering relationship
- A connection between the two LPGs

- Route rules to enable traffic over the peering connection to and from the desired subnets in the respective VCNs
- Security rules to control the types of traffic allowed to and from the instances in the subnets in question

Policies

Peering between two VCNs requires explicit agreement from both parties in the form of IAM policies that each party implements for their own VCN's compartment or tenancy. If the VCNs are in different tenancies, each administrator must provide their tenancy OCID and put in place special coordinated policy statements to enable the peering.

To implement the IAM policies required for peering, the two VCN administrators must designate one administrator as the requestor and the other as the acceptor. The requestor must be the one to initiate the request to connect the two LPGs. In turn, the acceptor must create a particular IAM policy that gives the requestor permission to connect to LPGs in the acceptor's compartment. Without that policy, the requestor's request to connect fails. Either VCN administrator can terminate a peering connection by deleting their LPG.

Routing and Traffic Control

As part of configuring the VCNs, each administrator must update the VCN's routing to enable traffic to flow between the VCNs. In practice, this looks just like routing you set up for any gateway, such as an internet gateway or dynamic routing gateway. For each subnet that needs to communicate with the other VCN, you update the subnet's route table. The route rule specifies the destination traffic's CIDR and your LPG as the target. Your LPG routes traffic that matches that rule to the other LPG, which in turn routes the traffic to the next hop in the other VCN.

You can control packet flow over the peering connection with route tables in your VCN. For example, you can restrict traffic to only specific subnets in the other VCN. Without terminating the peering, you can stop traffic flow to the other VCN by simply removing route rules that direct traffic from your VCN to the other VCN. You can also effectively stop the traffic by removing any security list rules that enable ingress or egress traffic with the other VCN. This does not stop traffic flowing over the peering connection, but stops it at the VNIC level.

Security Rules

Each VCN administrator must ensure that all outbound and inbound traffic with the other VCN is intended, expected and well-defined. In practice, this means implementing security list rules that explicitly state the types of traffic your VCN can send to the other and accept from the other. If your subnets use the default security list, there are two rules allowing SSH and ICMP ingress traffic from anywhere, thus also the other VCN. Evaluate these rules and whether you want to keep or update them.

In addition to security lists and firewalls, you should evaluate other OS-based configuration on the instances in your VCN. There could be default configurations that do not apply to your own VCN's CIDR, but inadvertently apply to the other VCN's CIDR.

Service Gateway

Certain services, such as the object storage service, are exposed internally over a conceptual *Services Network*. Normally, these services would use public IP addresses that can be reached over a public network – or in a public cloud model, over the internet. Instead, the purpose of a service gateway is to enable a VCN to privately access the services in the Services Network, meaning resources in a private subnet, within a tightened down VCN with no external access, can still connect to those services.

A VCN can have only one service gateway. You create a service gateway in the context of a specific VCN, so the gateway is automatically attached to that VCN upon creation. A service gateway allows traffic to and from all subnets at the time of creation; there is no mechanism to block or disable this traffic.

In Private Cloud Appliance, these services are implemented at the infrastructure level through the management node cluster. Technically, the service endpoints, which are fully qualified domain names, are reachable from the entire on-premises network by default. which means the service gateway has no real function. Specifically for private cloud use, there is no need to configure a service gateway and associated route rules to enable private access to the service endpoints. However, in order to maintain compatibility with Oracle Cloud Infrastructure, the concept of a service gateway does exist.

The service gateway uses a service CIDR label, which is a string that represents the endpoints for the service or group of services of interest. This means that you don't have to know the specific endpoints. If the service's endpoint changes in the future, you do not need to make any adjustments. You use the service CIDR label when you configure the service gateway. With Private Cloud Appliance you are allowed to create the service gateway and configure route rules involving a *"Service CIDR Block"*. However, remember they serve no purpose other than compatibility.

Because Private Cloud Appliance is set up within the safe boundaries of your data center network, securing private access to services is much less of a concern, compared with a public cloud environment like Oracle Cloud Infrastructure. Therefore, security rules are not implemented for service gateways. For details, see Security Rules.

Virtual Firewall

The Networking service offers two virtual firewall features that both use security rules to control traffic at the packet level. They offer different ways to apply security rules to a set of virtual network interface cards (VNICs).

- **Security lists:**

  A security list defines security rules at the subnet level, which means that all VNICs in a given subnet are subject to the same rules. Each VCN comes with a default security containing default rules for essential traffic. The default security list is automatically used with all subnets, unless a custom security list is specified. A subnet can have up to five associated security lists.

- **Network security groups (NSGs):**

  A network security group defines security rules based on membership. Its security rules apply to resources that are explicitly added to the NSG. A VNIC can be added to a maximum of five NSGs. An NSG is intended to provide a virtual firewall for a set of cloud resources with the same security posture. For example: a group of instances that perform the same tasks and thus need to use the same set of ports.

Oracle recommends using NSGs instead of security lists because NSGs let you separate the VCN's subnet architecture from your application security requirements. However, NSGs are only supported for specific services. It is possible to use both security lists and NSGs together, depending on your particular security needs.

If you have security rules that you want to enforce for all VNICs in a VCN, the easiest solution is to put the rules in one security list, and then associate that security list with all subnets in the VCN. This way you can ensure that the rules are applied, regardless of who in your organization creates a VNIC in the VCN. Alternatively, you can add the required security rules to the VCN's default security list.

If you choose to combine security lists and network security groups, the set of rules that applies to a given VNIC is the union of these items:

- The security rules in the security lists associated with the VNIC's subnet
- The security rules in all NSGs that the VNIC is in

A packet in question is allowed if *any rule* in any of the relevant lists and groups allows the traffic, or if the traffic is part of an existing connection being tracked because of a stateful rule.

Security Rules

This section explains important aspects of security rules that you need to understand in order to implement them as part of security lists or network security groups. How you create, manage, and apply security rules varies between security lists and network security groups. Related sections provide more detailed information about each implementation.

Parts of a Security Rule

A security rule allows a particular type of traffic into or out of a VNIC. For example, a commonly used security rule allows ingress TCP port 22 traffic for establishing SSH connections to the instance. Without security rules, no traffic is allowed into and out of VNICs in the VCN.

Each security rule specifies the following items:

- **Direction (ingress or egress):** Ingress is inbound traffic to the VNIC; egress is outbound traffic from the VNIC.

  The REST API model for security lists is different from network security groups. With security lists, there is an `IngressSecurityRule` object and a separate `EgressSecurityRule` object. With network security groups, there is only a `SecurityRule` object, and the object's `direction` attribute determines whether the rule is for ingress or egress traffic.
- **Stateful or stateless:** If stateful, connection tracking is used for traffic matching the rule. If stateless, no connection tracking is used. See Stateful and Stateless Rules in this section.
- **Source type and source:** For ingress rules only; the source you provide depends on the source type you choose. These source types are allowed:

Source Type	Allowed Source
CIDR	The CIDR block where the traffic originates from. Use 0.0.0.0/0 to indicate all IP addresses. The prefix is required. For example, include the /32 if specifying an individual IP address.

- **Destination type and destination:** For egress rules only; the destination you provide depends on the destination type you choose. These destination types are allowed:

Destination Type	Allowed Destination
CIDR	The CIDR block that the traffic is destined for. Use 0.0.0.0/0 to indicate all IP addresses. The prefix is required. For example, include the /32 if specifying an individual IP address.

- **IP Protocol:** Either a single IPv4 protocol or "all" to cover all protocols.
- **Source port:** The port where the traffic originates from. For TCP or UDP, you can specify all source ports, or optionally specify a single source port number, or a range.
- **Destination port:** The port where the traffic is destined to. For TCP or UDP, you can specify all destination ports, or optionally specify a single destination port number, or a range.
- **ICMP type and code:** For ICMP, you can specify all types and codes, or optionally specify a single type with an optional code. If the type has multiple codes, create a separate rule for each code you want to allow.
- **Description:** NSG security rules contain an optional attribute to include a description of the rule. This is currently not supported for security list rules.

Stateful and Stateless Rules

When you create a security rule, you choose whether it is stateful or stateless. The default is stateful. Stateless rules are recommended if you have a high-volume internet-facing website, for the HTTP/HTTPS traffic.

Marking a security rule as **stateful** indicates that you want to use connection tracking for any traffic that matches that rule. This means that when an instance receives traffic matching the stateful ingress rule, the response is tracked and automatically allowed back to the originating host, regardless of any egress rules applicable to the instance. And when an instance sends traffic that matches a stateful egress rule, the incoming response is automatically allowed, regardless of any ingress rules.

Marking a security rule as **stateless** indicates that you do NOT want to use connection tracking for any traffic that matches that rule. This means that response traffic is not automatically allowed. To allow the response traffic for a stateless ingress rule, you must create a corresponding stateless egress rule.

If you use both stateful and stateless rules, and there is traffic that matches both a stateful and stateless rule in a particular direction, the stateless rule takes precedence and the connection is not tracked. You would need a corresponding rule in the other direction, either stateless or stateful, for the response traffic to be allowed.

If you decide to use stateless security rules to allow traffic to/from endpoints outside the VCN, it is important to add a security rule that allows ingress ICMP traffic type 3 code 4 from source 0.0.0.0/0 and any source port. This rule enables your instances to receive Path MTU Discovery fragmentation messages. This rule is critical for establishing a connection to your instances. Without it, you can experience connectivity issues.

Best Practices for Security Rules

- **Use network security groups**

  Oracle recommends using NSGs for components that all have the same security posture. For example, in a multi-tier architecture, you would have a separate NSG for each tier. A given tier's VNICs would all belong to that tier's NSG.

  Within a given tier, you might have a particular subset of the tier's VNICs that have additional, special security requirements. Therefore you would create another NSG for those additional rules, and place that subset of VNICs into both the tier's NSG and the NSG with additional rules.

- **Understand default security list rules**

  Each VCN automatically comes with a default security list that contains several default security rules to help you get started using the Networking service. Those rules exist because they enable basic connectivity.

  Even if you choose not to use security lists or the default security list, get familiar with the rules so you better understand the types of traffic that your networked cloud resources require. You might want to use those rules in your NSGs or any custom security lists that you set up.

  There is no default rule to allow ping requests. If you want to ping an instance, add a stateful ingress rule to specifically allow ICMP traffic type 8 from the source network you plan to ping from. To allow ping access from the internet, use 0.0.0.0/0 for the source. Note that this rule for pinging is separate from the default ICMP-related rules in the default security list. Do not remove those rules.

- **Do not delete default security rules indiscriminately**

  Your VCN might have subnets that use the default security list by default. Do not delete any of the list's default security rules unless you have first confirmed that resources in your VCN do not require them. Otherwise, you might disrupt your VCN's connectivity.

- **If necessary, add rules to allow ping requests**

  There is no default rule to allow ping requests. If you want to ping an instance, add a stateful ingress rule to specifically allow ICMP traffic type 8 from the source network you plan to ping from. To allow ping access from the internet, use 0.0.0.0/0 for the source. Note that this rule for pinging is separate from the default ICMP-related rules in the default security list. Do not remove those rules.

- **If necessary, add rules to handle fragmented UDP packets**

  Instances can send or receive UDP traffic. If a UDP packet is too large for the connection, it is fragmented. However, only the first fragment from the packet contains the protocol and port information. If the security rules that allow this ingress or egress traffic specify a particular (source or destination) port number, the fragments after the first one are dropped. If you expect your instances to send or receive large UDP packets, set both the source and destination ports for the applicable security rules to ALL instead of a particular port number.

- **Align OS firewall rules with security rules**

  Your instances running images provided with Private Cloud Appliance also have OS firewall rules that control access to the instance. When troubleshooting access to an instance, make sure that all of the following items are set correctly:

- The rules in the network security groups that the instance is in
- The rules in the security lists associated with the instance's subnet
- The instance's OS firewall rules

Security Lists

A security list acts as a virtual firewall for an instance, with ingress and egress rules that specify the types of traffic allowed in and out. Each security list is enforced at the VNIC level. However, you configure your security lists at the subnet level, which means that all VNICs in a given subnet are subject to the same set of security lists. The security lists apply to a given VNIC whether it is communicating with another instance in the VCN or a host outside the VCN. Each subnet can have multiple security lists associated with it, and each list can have multiple rules.

Each VCN comes with a default security list. If you do not specify a custom security list for a subnet, the default security list is automatically used with that subnet. You can add and remove rules in the default security list. It has an initial set of stateful rules, which should in most cases be changed to only allow inbound traffic from authorized subnets. The default rules are:

- **Stateful ingress:** Allow TCP traffic on destination port 22 (SSH) from authorized source IP addresses and any source port.

  This rule makes it easy for you to create a new cloud network and public subnet, launch a Linux instance, and then immediately use SSH to connect to that instance without needing to write any security list rules yourself.

  The default security list does not include a rule to allow Remote Desktop Protocol (RDP) access. If you are using Microsoft Windows images, make sure to add a stateful ingress rule for TCP traffic on destination port 3389 from authorized source IP addresses and any source port.
- **Stateful ingress:** Allow ICMP traffic type 3 code 4 from authorized source IP addresses.

  This rule enables your instances to receive Path MTU Discovery fragmentation messages.
- **Stateful ingress:** Allow ICMP traffic type 3 (all codes) from your VCN's CIDR block.

  This rule makes it easy for your instances to receive connectivity error messages from other instances within the VCN.
- **Stateful egress:** Allow all traffic.

  This allows instances to initiate traffic of any kind to any destination. This implies that instances with public IP addresses can talk to any internet IP address if the VCN has a configured internet gateway. And because stateful security rules use connection tracking, the response traffic is automatically allowed regardless of any ingress rules.

The general process for working with security lists is as follows:

1. Create a security list.
2. Add security rules to the security list.
3. Associate the security list with one or more subnets.
4. Create resources, such as compute instances, in the subnet.

   The security rules apply to all the VNICs in that subnet.

When you create a subnet, you must associate at least one security list with it. It can be either the VCN's default security list or one or more other security lists that you already created. You can change which security lists the subnet uses at any time. You can add and remove rules in the security list. It is possible for a security list to contain no rules.

Network Security Groups

A network security group (NSG) provides a virtual firewall for a set of cloud resources, within a single VCN, that all have the same security posture. For example: a group of compute instances that all perform the same tasks and thus all need to use the same set of ports.

Rules in an NSG are enforced on VNICs, but their NSG membership is determined through their parent resources. Not all cloud services support NSGs. Currently, the following types of parent resources support the use of NSGs:

- **Compute instances:** When you create an instance, you can specify one or more NSGs for the instance's primary VNIC. If you add a secondary VNIC to an instance, you can specify one or more NSGs for that VNIC. You can also change the NSG membership of existing VNICs.
- **Mount targets:** When you create a mount target for a file system, you can specify one or more NSGs. You can also update an existing mount target to use one or more NSGs.

For resource types that do not yet support NSGs, continue to use security lists to control traffic to and from those parent resources.

An NSG contains two types of elements:

- **VNICs:** One or more VNICs; for example, the VNICs attached to the set of compute instances that all have the same security posture. All the VNICs must be in the VCN that the NSG belongs to. A VNIC can be in a maximum of five NSGs.
- **Security rules:** Rules that define the types of traffic allowed into and out of the VNICs in the group. For example: ingress TCP port 22 SSH traffic from a particular source.

The general process for working with NSGs is as follows:

1. Create an NSG.

   When you create an NSG, it is initially empty, without any security rules or VNICs. After the NSG is created, you can add or remove security rules to allow the types of ingress and egress traffic that the VNICs in the group require.

2. Add security rules to the NSG.
3. Add parent resources, or more specifically VNICs, to the NSG.

   When you manage an NSG's VNIC membership, you do it as part of working with the parent resource, not the NSG itself. You can do this when you create the parent resource, or you can update the parent resource and add it to one or more NSGs.

   When you create a compute instance and add it to an NSG, the instance's primary VNIC is added to the NSG. You can create secondary VNICs separately, and optionally add them to NSGs.

There are some differences in the REST API model for NSGs compared to security lists:

- With security lists, there is an `IngressSecurityRule` object and a separate `EgressSecurityRule` object. With network security groups, there is only a `SecurityRule` object, and the object's `direction` attribute determines whether the rule is for ingress or egress traffic.
- With security lists, the rules are part of the `SecurityList` object, and you work with the rules by calling the security list operations; for example: `UpdateSecurityList`. With NSGs, the rules are not part of the `NetworkSecurityGroup` object. Instead you use separate operations to work with the rules for a given NSG; for example: `UpdateNetworkSecurityGroupSecurityRules`.
- The model for updating existing security rules is different between security lists and NSGs. With NSGs, each rule in a given group has a unique identifier. When you call `UpdateNetworkSecurityGroupSecurityRules`, you provide the IDs of the specific rules that you want to update. With security lists, the rules have no unique identifier. When you call `UpdateSecurityList`, you must pass in the entire list of rules, including rules that are not being updated in the call.
- There is a limit of 25 rules when calling the operations to add, remove, or update security rules.

Network Security

This section provides information about controlling access and security in your cloud network. The firewall functionality inside a VCN is described in detail in Virtual Firewall. Additional topics about secure network access are discussed here.

Ways to Secure Your Network

There are several mechanisms to control security for your cloud network and compute instances:

- **Private subnets:** If instances do not require a public IP address, you can designate a subnet to be private, to prevent instances from obtaining a private IP.
- **Firewall rules:** To control packet-level traffic into and out of an instance, you can configure firewall rules directly on the instance itself. Images provided with Private Cloud Appliance that run Oracle Linux automatically include default rules that allow ingress on TCP port 22 for SSH traffic. Also, Microsoft Windows images may include default rules that allow ingress on TCP port 3389 for Remote Desktop access.
- **Gateways and route tables:** To control general traffic flow from your cloud network to outside destinations – the internet, your on-premises network, or another VCN – you configure your cloud network's gateways and route tables to allow only the connectivity that is effectively required.
- **IAM policies:** You can control who has access to the Private Cloud Appliance interfaces. You can control which cloud resources can be accessed and which type of access is allowed. For example, you can control who can set up your network and subnets, or who can update route tables, network security groups, or security lists.

Access Control

This topic provides basic information about using compartments and IAM policies to control access to your cloud network.

Using Compartments with Network Resources

Whenever you create a cloud resource such as a virtual cloud network (VCN) or compute instance, you must specify which IAM compartment you want the resource to be in. A compartment is a collection of related resources that can only be accessed by certain groups that have been given permission by an administrator in the tenancy. The administrator creates compartments and corresponding IAM policies to control which users in your organization access which compartments. Ultimately, the goal is to ensure that users can only access the resources they need.

In an enterprise production environment, you typically divide the tenancy into multiple compartments to more easily control access to certain types of resources. For example, your administrator could create Compartment_A for your VCN and other networking components. The administrator could then create Compartment_B for all the compute instances and block storage volumes that the HR department uses, and Compartment_C for all the instances and block storage volumes that the Marketing department uses.

The administrator would then create IAM policies that give users only the level of access they need in each compartment. For example, HR instance administrators are not entitled to modify the existing cloud network. So they would have full permissions for Compartment_B, but limited access to Compartment_A: only what is required to launch instances into the network. If they try to modify other resources in Compartment_A, the request is denied.

Network resources such as VCNs, subnets, route tables, security lists, service gateways, and NAT gateways can be moved from one compartment to another. When you move a resource to a new compartment, inherent policies apply immediately.

Detailed information about the use of compartments, and how to control access to your cloud resources, is provided in the chapter Identity and Access Management Overview. More specifically, refer to Organizing Resources in Compartments and How Policies Work.

Using IAM Policies for Networking

A convenient and common approach to granting access to Networking is to grant permission to a Network Administrators group to manage the VCN and all related networking components: subnets, security lists, route tables, gateways, and so on. It is useful to also allow the Network Administrators to launch instances in order to test network connectivity. Both policies – to manage network resources and to launch instances – are covered in Common Policies. The required policy statements should look similar to this example:

```
Allow group NetworkAdmins to manage virtual-network-family in tenancy
Allow group NetworkAdmins to manage instance-family in compartment ABC
Allow group NetworkAdmins to use volume-family in compartment ABC
```

If you are not yet familiar with IAM policies, refer to How Policies Work in the chapter Identity and Access Management Overview.

If you prefer a finer-grained approach to access control, you can write policies that focus on individual resource types. For example, the `virtual-network-family` resource type includes individual resource types such as `subnets`, `route-tables`, `security-lists` or `local-peering-gateways`. You can grant specific access to these by writing separate policies per resource type.

In addition, you can write policies that limit the level of access by using a different policy verb; for example: `manage` versus `use`, and so on. If you do, there are some nuances to understand about the policy verbs for Networking.

Be aware that the `inspect` verb not only returns general information about the network components; for example, the name and OCID of a security list or route table. It also includes the contents of the component; for example, the actual rules in the security list, the routes in the route table, and so on.

Also, the following operations are available only with the `manage` verb, not the `use` verb:

- Update (enable/disable) `internet-gateways`
- Update `security-lists`
- Update `route-tables`
- Update `dhcp-options`
- Attach a DRG to a VCN
- Peer two VCNs

Each VCN has various components that directly affect the behavior of the network: route tables, security lists, DHCP options, internet gateway, and so on. When you create one of these components, you establish a relationship between that component and the VCN, which means you must be allowed in a policy to both create the component and manage the VCN itself. However, the ability to update that component – to change the route rules, security list rules, and so on – does NOT require permission to manage the VCN itself, even though changing that component can directly affect the behavior of the network. This discrepancy is designed to give you flexibility in granting least privilege to users, and not require you to grant excessive access to the VCN just so the user can manage other components of the network. Be aware that by giving someone the ability to update a particular type of component, you are implicitly trusting them with controlling the network's behavior.

Networking Scenarios

This section describes a number of basic networking scenarios to help you understand the networking service and how networking components operate together.

Public Subnet

This section describes a setup consisting of a VCN and a public subnet. For external connectivity the VCN needs an internet gateway. Your on-premises network also uses this gateway to communicate with resources inside the VCN. The IP addresses used in this scenario must be public. In a private cloud context, this means a unique address directly reachable from the on-premises network.

The subnet uses the default security list, which has default rules that are designed to make it easy to get started. The rules enable typical required access; for example inbound SSH connections and any type of outbound connections. Remember that security list rules only allow traffic. Any traffic not explicitly covered by a security list rule is implicitly denied. In this

scenario, you add additional rules to the default security list. You could instead create a custom security list for those rules. You would then set up the subnet to use both the default security list and the custom security list.

The subnet uses the default route table, which contains no rules when the VCN is created. In this scenario, the table has only a single rule: to route traffic intended for all destinations (0.0.0.0/0) through the internet gateway.

To set up this networking scenario, you perform the following steps:

1. Create the VCN.

   Choose a compartment you have permission to work in. Specify one or more non-overlapping CIDR blocks for the VCN; for example: 172.16.0.0/16. Optionally, enable DNS and specify a DNS label for the VCN.

2. Create the public subnet.

   Specify a single, contiguous CIDR block within the VCN's CIDR block; for example: 172.16.10.0/24. Select the default route table. Make sure the subnet is a public subnet, so that instances can obtain public IP addresses. If you enabled DNS at the VCN level, you can choose to assign host names in the subnet and specify a subnet DNS label as well.

3. Create the internet gateway.

   When you create the internet gateway, it is enabled immediately. However, you must add a route rule to allow traffic to flow to the gateway.

4. Update the default route table to use the internet gateway.

   The default route table starts out with no rules. No route rule is required in order to route traffic within the VCN itself. You must add a rule that routes all traffic destined for addresses outside the VCN to the internet gateway. Enter these parameters:

   - Target Type: Internet Gateway
   - Destination CIDR block: 0.0.0.0/0

     This means that all non-intra-VCN traffic that is not already covered by other rules in the route table goes to the target specified in this rule.

   - Target: The internet gateway you created.

   Because the subnet was set up to use the default route table, the resources in the subnet can now use the internet gateway. The existence of this rule also enables inbound connections to the subnet, through the internet gateway. The next step is to specify the types of traffic you want to allow into and out of the instances you later create in the subnet.

5. Update the default security list.

   You set up the subnet to use the VCN's default security list. Now you add security list rules that allow the types of connections that the instances in the VCN will need.

   For example, if the instances in your subnet are web servers, they likely need to receive inbound HTTPS connections. To enable that traffic, add an ingress rule to the default security list using these parameters:

   - Source Type: CIDR
   - Source CIDR: 0.0.0.0/0

- IP Protocol: TCP
- Source Port Range: All
- Destination Port Range: 443

6. Create instances.

   Your next step is to create one or more instances in the subnet. Each instance automatically gets a private IP address. With the network setup in this scenario, you must give each instance a public IP address, otherwise you cannot access them through the internet gateway.

Private Subnet

This section describes a setup consisting of a VCN and a private subnet. For connectivity to your on-premises network, the VCN needs a dynamic routing gateway (DRG).

The subnet uses the default security list, which has default rules that are designed to make it easy to get started. The rules enable typical required access; for example inbound SSH connections and any type of outbound connections. Remember that security list rules only allow traffic. Any traffic not explicitly covered by a security list rule is implicitly denied. In this scenario, you add additional rules to the default security list. You could instead create a custom security list for those rules. You would then set up the subnet to use both the default security list and the custom security list.

The subnet uses the default route table, which contains no rules when the VCN is created. In this scenario, the table has only a single rule: to route traffic intended for all destinations (0.0.0.0/0) through the DRG.

To set up this networking scenario, you perform the following steps:

1. Create the VCN.

   Choose a compartment you have permission to work in. Specify one or more non-overlapping CIDR blocks for the VCN; for example: 172.16.0.0/16. Optionally, enable DNS and specify a DNS label for the VCN.

2. Create the private subnet.

   Specify a single, contiguous CIDR block within the VCN's CIDR block; for example: 172.16.10.0/24. Make the subnet private; the instances you create cannot obtain a public IP address. Select the default route table. If you enabled DNS at the VCN level, you can choose to assign host names in the subnet and specify a subnet DNS label as well.

3. Update the default security list.

   You set up the subnet to use the VCN's default security list. Now you add security list rules that allow the types of connections that the instances in the VCN will need.

   For example, if your subnet contains Microsoft Windows instances and you intend to access them using RDP, add an ingress rule to the default security list using these parameters:

   - Source Type: CIDR
   - Source CIDR: 0.0.0.0/0
   - IP Protocol: TCP
   - Source Port Range: All
   - Destination Port Range: 3389

4. Create a dynamic routing gateway (DRG) and attach it to your VCN.

   When you create the DRG, it is in "Provisioning" state for a short period. Make sure provisioning is done before continuing. Next, attach the DRG you just created to your VCN. For this scenario you can ignore the advanced attachment options. The DRG attachment will be in "Attaching" state for a short period before it is ready.

   To allow traffic to flow to the DRG, you must add a route rule.

5. Update the default route table to use the DRG.

   The default route table starts out with no rules. No route rule is required in order to route traffic within the VCN itself. You must add a rule that routes all traffic destined for addresses in your on-premises network to the DRG. Enter these parameters:

   - Target Type: Dynamic Routing Gateway.

     The VCN's attached DRG is automatically selected as the target.

   - Destination CIDR block: 0.0.0.0/0

     This means that all non-intra-VCN traffic that is not already covered by other rules in the route table goes to the target specified in this rule.

   Because the subnet was set up to use the default route table, the DRG now enables traffic between the resources in the subnet and in your on-premises network.

6. Create instances.

   Your next step is to create one or more instances in the subnet. Each instance automatically gets a private IP address. With the network setup in this scenario, no additional configuration is required in order to access the instances from your on-premises network.

Public and Private Subnets

This section describes a simple multi-tier setup consisting of a VCN with a public and a private subnet. The public subnet holds public instances such as web servers, and the private subnet holds private instances such as database servers. The VCN has a dynamic routing gateway (DRG) for connectivity to your on-premises network. Instances in the public subnet have external access through an internet gateway.

> **Note:**
>
> In a public cloud environment, instances in the private subnet could be allowed to initiate external connections using a NAT gateway, for example to get software updates. However, in Private Cloud Appliance the NAT gateway would provide access to the on-premises network, which the DRG already enables for those instances. The combination of a NAT gateway and DRG could cause issues due to non-deterministic routing.

Each subnet uses the default security list, which has default rules that are designed to make it easy to get started. The rules enable typical required access; for example inbound SSH connections and any type of outbound connections. Remember that

security list rules only allow traffic. Any traffic not explicitly covered by a security list rule is implicitly denied.

Each subnet also has its own custom security list and custom route table with rules specific to the needs of the subnet's instances. In this scenario, the VCN's default route table, which is always empty to start with, is not used.

To set up this networking scenario, you perform the following steps:

1. Create the VCN.

   Choose a compartment you have permission to work in. Specify one or more non-overlapping CIDR blocks for the VCN; for example: 172.16.0.0/16. Optionally, enable DNS and specify a DNS label for the VCN.

2. Add the necessary gateways to the VCN.

   The instances in the public subnet need an internet gateway for incoming and outgoing public traffic. The instances in the private subnet need a NAT gateway to be able to reach the data center network and the internet. These gateways are enabled immediately upon creation, but you must add route rules to allow traffic to flow to them.

   To be able to reach the private instances in the VCN from your on-premises network, you create a DRG and attach it to the VCN. When you create the DRG, it is in "Provisioning" state for a short period. Make sure provisioning is done before attaching it to the VCN. For this scenario you can ignore the advanced attachment options. The DRG attachment will be in "Attaching" state for a short period before it is ready. To allow traffic to flow to the DRG, you must also add a route rule.

3. Create custom route tables for the subnets you will create later.

   a. For the public subnet, create a route table and add a rule that routes all traffic destined for addresses outside the VCN to the internet gateway. Enter these parameters:

      - Target Type: Internet Gateway
      - Destination CIDR block: 0.0.0.0/0

        This means that all non-intra-VCN traffic that is not already covered by other rules in the route table goes to the target specified in this rule.

      - Target: The internet gateway you created.

   b. For the private subnet, create a route table and add two rules: one that routes traffic destined for the on-premises network to the DRG, and one that routes all other traffic leaving the VCN to the NAT gateway.

      Create the route rule for the NAT gateway with these parameters:

      - Target Type: NAT Gateway
      - Destination CIDR block: 0.0.0.0/0

        This means that all non-intra-VCN traffic that is not already covered by other rules in the route table goes to the target specified in this rule.

      - Target: The NAT gateway you created.

      Create the route rule for the DRG with these parameters:

      - Target Type: Dynamic Routing Gateway.

        The VCN's attached DRG is automatically selected as the target.

      - Destination CIDR block: 10.25.0.0/16

This means that traffic intended for an address in the on-premises network (10.25.x.y) goes to the DRG target specified in this rule.

4. Update the default security list.

   Add security list rules that allow the types of connections that the instances in the VCN will need.

   Edit each of the existing stateful ingress rules so that the Source CIDR is **not** 0.0.0.0/0, but the CIDR for your on-premises network; in this example: 10.25.0.0/16.

5. Create custom security lists for the subnets you will create later.

   a. Create a custom security list for the public subnet and add rules to allow the types of connections that the public instances will need. For example, web servers likely need to receive HTTP and HTTPS ingress traffic. For HTTP, use the settings below. For HTTPS, add another similar rule for TCP port 443.

      - Source Type: CIDR
      - Source CIDR: 0.0.0.0/0
      - IP Protocol: TCP
      - Source Port Range: All
      - Destination Port Range: 80

   b. Create a custom security list for the private subnet and add rules to allow the types of connections that the private instances will need. For example, database servers likely need to receive SQL*Net (TCP port 1521) ingress traffic from clients in the private and the public subnet. For clients in the public subnet, use the settings below. For clients in the private subnet, add another similar rule for the CIDR of the private subnet (172.16.1.0/24).

      - Source Type: CIDR
      - Source CIDR: 172.16.2.0/24
      - IP Protocol: TCP
      - Source Port Range: All
      - Destination Port Range: 1521

6. Create the subnets in the VCN.

   - Public subnet:

     Specify a single, contiguous CIDR block within the VCN's CIDR block; for example: 172.16.2.0/24. Make sure the subnet is a public subnet, so that instances can obtain public IP addresses. Select the custom public subnet route table you created earlier.

     Select two security lists: both the default security list and the public subnet security list you created earlier. If you enabled DNS at the VCN level, you can choose to assign host names in the subnet and specify a subnet DNS label as well.

   - Private subnet:

     Specify a single, contiguous CIDR block within the VCN's CIDR block; for example: 172.16.1.0/24. Make the subnet private; the instances you create in this subnet cannot obtain a public IP address. Select the custom private subnet route table you created earlier.

Select two security lists: both the default security list and the private subnet security list you created earlier. If you enabled DNS at the VCN level, you can choose to assign host names in the subnet and specify a subnet DNS label as well.

7. Create instances.

   Your next step is to create instances in the subnets. Each instance automatically gets a private IP address. For each instance in the public subnet, make sure to assign the instance a public IP address. Otherwise, you won't be able to reach the instance from your on-premises network. The DRG you set up allows you to reach the instances in the private subnet from your on-premises network without any additional configuration.

7

Compute Instance Concepts

Oracle Private Cloud Appliance lets you provision and manage compute instances.

On Private Cloud Appliance, a compute instance is a virtual machine (VM), which is an independent computing environment that runs on top of physical hardware. The virtualization makes it possible to run multiple compute instances that are isolated from each other.

When you launch a compute instance, you can select the most appropriate type of compute instance for your applications based on characteristics such as the number of CPUs, amount of memory, and network resources.

After you launch a compute instance, you can access it securely from your computer, restart it, attach and detach volumes, and terminate it when you're done with it.

For step-by-step instructions for managing the Compute service, refer to Compute Instance Deployment in the Oracle Private Cloud Appliance User Guide.

Components for Launching Instances

These components are required to launch a compute instance:

Tenancy

The root compartment that contains all of your organization's compartments and cloud resources. The Service Enclave administrator creates the tenancy in which compartments are created. A tenancy administrator creates compartments in a tenancy where the compute resources are created. You must have a tenancy to have compartments where instances are launched.

Compartment

A collection of related resources that are only accessible by certain groups that have been given permission by an administrator in your organization. Compute instances are created in compartments. All compartments exist in a tenancy, which is the root compartment.

Virtual Cloud Network (VCN)

A virtual version of a traditional network—including subnets, route tables, and gateways—on which your compute instance runs. At least one cloud network must be set up by a Compute Enclave administrator before you launch compute instances.

Key Pair

If the image that is used to launch the instance is configured to require Secure Shell (SSH) for authentication, then you need an RSA SSH key pair before launching the instance. This requirement applies to instances launched from images provided with Private Cloud Appliance and by most UNIX type images. If the image is configured to use passwords instead, you need the password instead of the key pair.

Image

A template of a virtual hard drive that determines the operating system and other software for a compute instance. You can also launch compute instances using these images:

- Images provided with Oracle Private Cloud Appliance
- Custom images created from other instances
- Import your own image

For more information about images, see Compute Images in the Oracle Private Cloud Appliance User Guide.

Shape

A template that determines the number of CPUs, amount of memory, and other resources allocated to a newly created compute instance. You choose the most appropriate shape when you launch a compute instance.

Compute Shapes

A shape is a template that determines the number of OCPUs, amount of memory, and number of VNICs that are allocated to a compute instance. You choose a shape when you create an instance.

Private Cloud Appliance supports two types of shapes:

- **Standard shapes:** Each shape has a fixed number of OCPUs and memory that are allocated to an instance when the instance is created.
- **Flexible shape:** Does not have a fixed number of OCPUs and memory. The flexible shape lets you choose the number of OCPUs and amount of memory that are allocated to an instance when the instance is created.

Standard Shapes

Standard shapes are designed for general-purpose workloads and suitable for a wide range of applications and use cases. Standard shapes provide a balance of cores, memory, and network resources. All standard shapes use block storage for the boot device.

The following table lists the standard shapes:

Shape	OCPUs	Memory (GB)	Maximum VNICs	Maximum Bandwidth (Gbps)
VM.PCAStandard1.1	1	16	2	24.6
VM.PCAStandard1.2	2	32	2	24.6
VM.PCAStandard1.4	4	64	4	24.6
VM.PCAStandard1.8	8	128	8	24.6
VM.PCAStandard1.16	16	256	16	24.6
VM.PCAStandard1.24	24	384	24	24.6
VM.PCAStandard1.32	32	512	24	32.0
VM.PCAStandard1.48	48	768	24	48.0
VM.PCAStandard1.60	60	960	24	100.0

Flexible Shape

A flexible shape lets you customize the number of OCPUs and the amount of memory when launching your instance. This flexibility lets you create instances that meet your workload requirements, while optimizing performance and using resources efficiently.

The images provided with Private Cloud Appliance support the flex shape.

Shape	OCPUs	Memory (GB)	Maximum VNICs	Maximum Bandwidth
VM.PCAStandard1.Flex	1–32	64 GB maximum per OCPU 512 GB maximum per instance	1 OCPU: 2 VNICs 2 to 24 OCPUs: 1 VNIC per OCPU 25 to 32 OCPUs: 24 VNICs	1 Gbps per OCPU

Storage for Compute Instances

You can expand the storage that's available for your compute instances with the following services:

- **Block Volume:** Lets you dynamically provision and manage block volumes that you can attach to one or more compute instances.
- **File Storage:** A durable, scalable, secure, enterprise-grade network file system that you can connect to from any compute instance in your virtual cloud network (VCN).
- **Object Storage:** An internet-scale, high-performance storage platform that lets you store a large amount of unstructured data of any content type. This storage not tied to any specific compute instance.

Compute Instance Lifecycle

This list describes the different lifecycle states for compute instances.

- **Launching:** Occurs when you create a compute instance. The instance is displayed in the Compute Web UI in a provisioning state. Expect provisioning to take several minutes before the state updates to running. After the instance is running, allow another few minutes for the operating system to boot before you attempt to connect.
- **Connecting:** You connect to a running Linux or Oracle Solaris instance using a Secure Shell (SSH) connection. Most Linux and UNIX-like operating systems include an SSH client by default.
- **Backing up the boot volume:** You can back up the boot volume using the Block Volume backup feature using one of these methods:
  - **Manual backups:** You manually perform create, get, list, rename, and delete backup commands.
  - **Automatic backups:** You create a backup policy and a backup policy assignment that specifies the time and frequency of the volume backups. The system automatically performs the commands that back up the volume.

- **Stopping:** You can stop an instance using the Compute Web UI, OCI CLI, Compute API, or using the commands available in the operating system when you are logged in to the instance.

  If the applications that run on the instance take more than 15 minutes to shut down, they could be improperly stopped. To avoid this situation, shut down the instance using the commands available in the OS before you stop the instance.
- **Starting or restarting:** You can start or restart an instance as needed using the Compute Web UI, OCI CLI, and Compute API.
- **Rebooting:** You can reboot an instance as needed using the Compute Web UI, OCI CLI, and Compute API. By default, a reboot gracefully restarts the instance by sending a shutdown command to the operating system. After waiting 15 minutes for the OS to shut down, the instance is powered off and then powered back on.
- **Terminating:** You can permanently terminate (delete) instances that you no longer need. Any attached VNICs and volumes are automatically detached when the instance terminates. Eventually, the instance's public and private IP addresses are released and become available for other instances.

  By default, the instance's boot volume is preserved when you terminate the instance. You can attach the boot volume to a different instance as a data volume, or use it to launch a new instance. If you no longer need the boot volume, you can permanently delete as described in Deleting a Boot Volume in the Block Volume Storage chapter in the Oracle Private Cloud Appliance User Guide.

For more information, see Managing the Lifecycle of an Instance in Compute Instance Deployment.

Compute Instance Connections

You can connect to a running compute instance using a Secure Shell (SSH) or Remote Desktop connection.

Most UNIX-style systems include an SSH client by default.

For step-by-step instructions for connecting to an instance, refer to the section titled Connecting to a Compute Instance in the Compute Instance Deployment chapter in the Oracle Private Cloud Appliance User Guide.

Compute Images

An image is a template of a virtual hard drive. The image determines the operating system and other software for a compute instance. You specify an image to use when you launch a compute instance.

These are the types of images you can use to launch a compute instance:

- **Images provided with Oracle Private Cloud Appliance:**Private Cloud Appliance includes some images such as Oracle Linux and Oracle Solaris images. These included images are called platform images. Platform images are available in every compartment of every tenancy. Platform images do not need to be downloaded or imported in order to access them to create an instance.
- **Custom images:** You can create a custom image of a compute instance's boot disk and use it to launch other compute instances. Instances you launch from your

image include the customizations, configuration, and software installed when you created the image. See Custom Images Created From Instances.

- **Bring your own image:** You can bring your own versions of operating systems to the cloud as long as the underlying hardware supports it. The Private Cloud Appliance services do not depend on the OS that you run. See Bring Your Own Image (BYOI).

> **Note:**
>
> Images for Private Cloud Appliance must have paravirtualized network devices and boot volumes. SR-IOV network devices and iSCSI boot volumes are not supported.

Custom Images Created From Instances

You can create a custom image of a compute instance's boot disk and use it to launch other compute instances. Instances you launch from your image include the customizations, configuration, and software installed when you created the image.

Custom images do not include the data from any attached block volumes.

Limitations and Considerations

- Certain IP addresses are reserved for Private Cloud Appliance use and might not be used in your address numbering scheme. For details, refer to Reserved Network Resources in this guide.
- When you create an image of an instance, the instance must be in the stopped state. After the custom image is created, you can restart the instance.
- You cannot create extra custom images of a compute instance while the compute instance is engaged in the image creation process. You can, however, create images of different compute instances at the same time.
- Custom images are available to all users authorized for the compartment in which the image was created.
- Custom images inherit the compatible shapes that are set by default from the base image.

Bring Your Own Image (BYOI)

The Bring Your Own Image (BYOI) feature enables you to import your own versions of operating systems into Private Cloud Appliance as long as the underlying hardware supports it. The services do not depend on the OS you run.

The BYOI feature provides these benefits:

- Enables cloud migration projects.
- Supports both old and new operating systems.
- Encourages experimentation.
- Increases infrastructure flexibility.

> **Important:**
>
> You must comply with all licensing requirements when you upload and start instances based on OS images that you supply.

A critical part of any lift-and-shift cloud migration project is the migration of on-premises virtual machines (VMs) to the cloud. You can import your on-premises virtualized root volumes into Private Cloud Appliance using the custom image import feature, and then launch compute instances using those images.

You can import Microsoft Windows and Linux-based custom images and use them to launch instances on Private Cloud Appliance.

Linux Source Image Requirements

Custom images must meet the following requirements:

- The maximum image size is 400 GB.
- The image must be set up for BIOS boot.
- Only one disk is supported, and it must be the boot drive with a valid master boot record (MBR) and boot loader. You can migrate additional data volumes after you import the image's boot volume.
- The boot process must not require more data volumes to be present for a successful boot.
- The boot loader must use LVM or a UUID to locate the boot volume.
- The disk image cannot be encrypted.
- The disk image must be a VMDK or QCOW2 file. These images can be converted to `.oci` type images.
  - Create the image file by cloning the source volume, not by creating a snapshot.
  - VMDK files must be either the "single growable" (monolithicSparse) type or the "stream optimized" (streamOptimized) type, both of which consist of a single VMDK file. All other VMDK formats, such as those that use multiple files, split volumes, or contain snapshots, are not supported.
- The network interface must use DHCP to discover the network settings. When you import a custom image, existing network interfaces are not re-created. Any existing network interfaces are replaced with a single NIC after the import process is complete. You can attach more VNICs after you launch the imported instance.
- The network configuration must not hard code the MAC address for the network interface.
- We recommend that you enable certificate-based SSH, however this recommendation is optional.

Microsoft Windows Source Image Requirements

- The maximum image size is 400 GB.

- The image must be set up for a BIOS boot.
- Only one disk is supported, and it must be the boot drive with a valid master boot record (MBR) and boot loader. You can migrate additional data volumes after you import the image's boot volume.
- The minimum boot volume size is 256 GB.
- The boot process must not require other data volumes to be present for a successful boot.
- The disk image cannot be encrypted.
- The disk image must be a VMDK or QCOW2 file. Create the image file by cloning the source volume, not by creating a snapshot. VMDK files must be either the "single growable" (monolithicSparse) type or the "stream optimized" (streamOptimized) type, both of which consist of a single VMDK file. All other VMDK formats, such as those that use multiple files, split volumes, or contain snapshots, are not supported.
- The network interface must use DHCP to discover the network settings. When you import a custom image, existing network interfaces are not re-created. Any existing network interfaces are replaced with a single NIC after the import process is complete. You can attach additional VNICs after you launch the imported instance.
- The network configuration must not hard code the MAC address for the network interface.

Boot Volumes

When you launch a compute instance based on an Oracle platform image or custom image, a new boot volume for the compute instance is created in the same compartment. That boot volume is associated with that compute instance until you terminate the compute instance.

When you terminate the compute instance, you can preserve the boot volume and its data. This feature gives you more control and management options for your compute instance boot volumes, and enables:

- **Instance scaling:** When you terminate your compute instance, you can keep the associated boot volume and use it to launch a new compute instance using a different compute instance type or shape. This flexibility enables you to easily scale up or down the number of cores for a compute instance.
- **Troubleshooting and repair:** If you think a boot volume issue is causing a compute instance problem, you can stop the compute instance and detach the boot volume. Then you can attach it to another compute instance as a data volume to troubleshoot it. After resolving the issue, you can then reattach it to the original compute instance or use it to launch a new compute instance.

Boot volume Encryption

Boot volumes are encrypted by default, the same as other block storage volumes.

> **Important:**
>
> Usually, encryption is not supported for compute instances launched from custom images imported for "bring your own image" (BYOI) scenarios.

Listing Boot Volumes

You can list all boot volumes in a specific compartment, or list detailed information on a single boot volume.

Listing Boot Volume Attachments

You can list all the boot volume attachments in a specific compartment. You can also view detailed information on a single boot volume attachment.

Detaching and Attaching a Boot Volume

If a boot volume has been detached from the associated compute instance, you can reattach it to the compute instance. If you want to restart a compute instance with a detached boot volume, you must reattach the boot volume.

If you think a boot volume issue is causing a compute instance problem, you can stop the compute instance and detach the boot volume. Then you can attach it to another compute instance as a data volume to troubleshoot it.

If a boot volume has been detached from the associated compute instance, or if the compute instance is stopped or terminated, you can attach the boot volume to another compute instance as a data volume.

Extending a Boot Volume Partition

You can extend the partition for a boot volume for an existing compute instance by resizing a volume. To take advantage of the larger size, you also need to extend the partition for the boot volume.

Deleting a Boot Volume

When you terminate a compute instance, you choose to delete or preserve the associated boot volume.

If a boot volume has been detached from the compute instance, you can delete the boot volume.

For step-by-step instructions for managing boot volumes, refer to the Block Volume Storage chapter in the Oracle Private Cloud Appliance User Guide.

Custom Boot Volume Sizes

When you launch a compute instance, you can choose whether to use the selected image's default boot volume size, or to specify a custom size up to 32 TB.

For Linux-based images, the custom boot volume size must be larger than the image's default boot volume size or 50 GB, whichever is higher.

For Microsoft Windows-based images, the custom boot volume size must be larger than the image's default boot volume size or 256 GB, whichever is higher. The minimum size requirement for Microsoft Windows images is to ensure that there is enough space available for patches and updates that can require a large amount of space.

If you specify a custom boot volume size, you need to extend the volume to take advantage of the larger size.

Boot Volume Backups

The backups feature of the Block Volume service lets you make a crash-consistent backup, which is a point-in-time snapshot of a boot volume without application interruption or downtime. You can make a backup of a boot volume while it is attached to a running compute instance, or while it is detached from the compute instance. The backup is a full backup.

You can restore a boot volume from any of your boot volume backups. You only need to keep the backups taken for the times you care about.

Boot Volume Tags

When a boot volume backup is created, the source boot volume's tags are automatically included in the boot volume backup.

When you create a compute instance from the boot volume backup, the compute instance includes the source boot volume's tags.

Boot Volume Backup Size

The boot volume backup size might be larger than the source boot volume size for the following reasons:

- Any part of the boot volume that has been written to is included in the boot volume backup.
- Many operating systems write or zero out the content, which results in these blocks marked as used. The Block Volume service considers these blocks updated and includes them in the volume backup.
- Boot volume backups also include metadata, which can be up to 1 GB in additional data.

Restoring a Boot Volume

You can use a boot volume backup to create a compute instance or you can attach it to another compute instance as a data volume. However before you can use a boot volume backup, you need to restore it to a boot volume.

You can restore a boot volume from any of your boot volume backups. You only need to keep the backups taken for the times you care about.

Cloning a Boot Volume

You can create a clone from a boot volume using the Block Volume service. Cloning enables you to make a copy of an existing boot volume without needing to go through the backup and restore process.

Any subsequent changes to the data on the source boot volume are not copied to the boot volume clone. The clone is the same size as the source boot volume unless you specify a larger volume size when you create the clone.

The clone operation occurs immediately and you can use the cloned boot volume when the state changes to available.

There is a single point-in-time reference for a source boot volume while it is being cloned. If you clone a boot volume while the associated compute instance is running, you need to wait

for the first clone operation to complete before creating more clones. You also need to wait for any backup operations to complete.

You can only create a clone for a boot volume within the same tenant. You can create a clone for a boot volume between compartments as long as you have the required access permissions for the operation.

For a comparison between backups and clones see .

Instance Backup and Restore

Oracle Private Cloud Appliance provides Compute Enclave APIs that enable you to back up instances. The commands are flexible to suit a variety of use cases.

Use Cases

- Back up instances and any attached block volumes.
- Store the backups on another server for safekeeping.
- Restore a faulty instance and any attached block volumes.
- Use the backup to create matching instances.
- Use the backup and restore feature to migrate instances to another tenancy, or to another appliance.

The following sections provide an overview of the backup and restore processes. For step-by-step instructions, refer to "Backing Up and Restoring an Instance" in the chapter Compute Instance Deployment of the Oracle Private Cloud Appliance User Guide.

Instance Backup Process

1. Use the Export operation to create an instance backup in an Object Storage bucket.

   The instance backup export operation performs these actions:

   - Creates the backup of the instance, which includes the source image, boot volume, and any attached block volumes.
   - Writes the backups into QCOW2 format files.
   - Creates an archive that includes all the backup files and metadata of the boot volume and block volumes.
   - Exports the backup to an Object Storage bucket of your choosing.
2. Transfer the backup object from the bucket to another system in your data center for safekeeping.

Instance Restore Process

1. Transfer the backup from the system in your data center to an Object Storage bucket in the appliance where you want to restore the instance.
2. Use the Import process to import the instance backup from the bucket to internal resources of the appliance.

   The import function performs these actions:

   - Extracts the tar archive to get source image, boot volume, and block volume backups with attachment data.

- Creates a source image.
- Adds the block volumes to the compartment (unattached).

3. Create an instance, and select the backup boot volume as the image source.
4. Attach any block volumes.

The restored instance has some of the same characteristics as the source instance. For example:

- The type and version of the OS.
- The OS configuration matches the OS configuration of the source instance. This includes things like OS user accounts, installed applications, and so on.
- The type of storage, high-performance or balanced-performance, matches the source instance.
- All the software on the block volumes is available after you attached the block volumes to the restored instance.

Some aspects of the restored instance differ from the source instance, such as:

- The restored instance and associated components, like boot and block volumes have unique OCIDs that don't match the source instance.
- The source instance user account SSH keys are not included in the restored instance.
- While creating the restored instance, you can configure the instance in a different compartment, with a different name, shape, subnet, and all the other attributes that you configure during the launch.

Simplifying Compute Instance Management

You can simplify the management of your compute instances using these features:

- **Instance Configurations:** Are templates that define the settings to use when creating compute instances.
- **Instance Pools:** are a group of compute instances that are created from the same compute instance configuration and managed as a group.

For step-by-step instructions for managing the Compute service, refer to the Compute Images chapter in the Oracle Private Cloud Appliance User Guide.

Instance Configurations

A compute instance configuration defines the settings to use when creating compute instances, including details such as the base image, shape, and metadata. You can also specify the associated resources for the compute instance, such as block volume attachments and network configuration.

Instance Pools

Instance pools let you create multiple compute instances from the same instance configuration. They also enable integration with other services, such as the IAM service, making it easier to manage groups of compute instances.

You create a compute instance pool using an existing compute instance configuration.

If you need to update the compute instance configuration, create a new compute instance configuration and then update the compute instance pool to use the new compute instance configuration.

You can delete a compute instance pool.

> **Caution:**
>
> When you delete a compute instance pool all of its resources are permanently deleted, including associated compute instances, attached boot volumes, and block volumes.

Instance Pool Lifecycle States

The following list describes the different lifecycle states for compute instance pools.

- **Provisioning:** When you create a compute instance pool, this is the first state the compute instance pool is in. Instances for the compute instance pool are being configured based on the specified compute instance configuration.
- **Starting:** The compute instances are being launched. At this point, the only action you can take is to terminate the compute instance pool.
- **Running:** The compute instances are created and running.
- **Stopping:** The compute instances are in the process of being shut down.
- **Stopped:** The compute instances are shut down.
- **Scaling:** When you update the compute instance pool size, the pool goes into this state while creating compute instances (for increases in pool size) or terminating compute instances (for decreases in pool size). At this point, the only action you can take is to terminate the compute instance pool.
- **Terminating:** The compute instances and associated resources are being terminated.
- **Terminated:** The compute instance pool, all its compute instances, and associated resources are terminated.

When working with compute instance configurations and compute instance pools, keep the following points in mind:

- You can't delete a compute instance configuration if it is associated with at least one compute instance pool.
- You can use the same compute instance configuration for multiple compute instance pools. However, a compute instance pool can have only one compute instance configuration associated with it.
- If you modify the compute instance configuration for a compute instance pool, existing compute instances that are part of that pool will not change. Any new compute instances that are created after you modify the compute instance configuration will use the new compute instance configuration. New compute instances will not be created unless you have increased the size of the compute instance pool or terminate existing compute instances.

- If you decrease the size of a compute instance pool, the oldest compute instances are terminated first.

Extending Compute Resources

The topics in this section describe the compute resources that you can extend and the implications.

Expanding Volumes

You can expand the size of block volumes and boot volumes. You cannot decrease the size.

You have several options to increase the size of your volumes:

- Expand an existing volume in place with online resizing.
- Restore from a volume backup to a larger volume.
- Clone an existing volume to a new, larger volume.
- Expand an existing volume in place with offline resizing. See Offline Resizing of Block Volumes Using the Compute Web UI.

> **Caution:**
>
> Before you resize a boot or block volume, create a backup of the volume.

After a volume has been resized, the first backup of the resized volume will be a full backup.

For step-by-step instructions for managing the Compute service, refer to the section titled Resizing Volumes in the Block Volume Storage chapter in the Oracle Private Cloud Appliance User Guide.

Offline Resizing of Block Volumes Using the Compute Web UI

With offline resizing, you detach the volume from a compute instance before you expand the volume size. After the volume is resized and reattached, you need to extend the partition, but you do not need to rescan the disk.

Before you resize a volume, create a full backup of the volume.

Whenever you detach and reattach volumes, there are complexities and risks for both Linux-based and Microsoft Windows-based compute instances. For more information, refer to the section titled *Resizing Volumes* in the Block Volume Storage chapter in the User Guide.

Rescanning the Disk for a Block Volume or Boot Volume

The Block Volume service lets you expand the size of block volumes and boot volumes while they are online and attached to compute instances.

After the volume is provisioned, you need to run commands to rescan the disk so that the operating system identifies the expanded volume size. You run different rescan commands depending on the operating system of the attached compute instance.

Adding Another Network Interface

You can add additional VNICs to a compute instance. Each additional VNIC can be in a subnet in the same VCN as the primary VNIC, or in a different subnet that is either in the same VCN or a different one.

You might add a VNIC to connect a compute instance to subnets in multiple VCNs. For example, you might set up your own firewall to protect traffic between VCNs, so the compute instance needs to connect to subnets in different VCNs.

Secondary VNICs are supported for these types of compute instances:

- **Linux**
- **Microsoft Windows**

Here are more details about additional VNICs:

- There's a limit to how many VNICs can be attached to a compute instance, and it varies by shape. See Compute Shapes.
- They can be added only after the compute instance is launched.
- They must always be attached to a compute instance and cannot be moved. The process of creating an additional VNIC automatically attaches it to the compute instance. The process of detaching a secondary VNIC automatically deletes it.
- They are automatically detached and deleted when you terminate the compute instance.
- The compute instance's bandwidth is fixed regardless of the number of VNICs attached. You can't specify a bandwidth limit for a particular VNIC on a compute instance.
- Attaching multiple VNICs from the same subnet CIDR block to a compute instance can introduce asymmetric routing, especially on instances using a variant of Linux. If you need this type of configuration, assign multiple private IP addresses to one VNIC, or using policy-based routing.

For step-by-step instructions for managing the VNICs, refer to the section titled *Configuring VNICs and IP Addressing* in the Networking chapter in the Oracle Private Cloud Appliance User Guide.

8

Block Volume Storage Overview

Block Volumes provide high-performance network storage capacity that supports a broad range of I/O intensive workloads.

You can use block volumes to expand the storage capacity of your compute instances, to provide durable and persistent data storage that can be migrated across compute instances, and to host large databases.

The Block Volume service enables you to group multiple volumes in a volume group. Volume groups simplify the process to create backups and clones. For more information, see Block Volume Groups.

For step-by-step instructions for managing block volumes, Refer to Block Volume Storage in the Oracle Private Cloud Appliance User Guide.

Types of Block Volumes

There are two types of volumes:

- **Boot volume**: A detachable boot volume device that contains the image that is used to boot a Compute instance. See Boot Volumes.
- **Block volume**: A detachable block storage device that allows you to dynamically expand the storage capacity of an instance.

The default size for both types of volumes is 50 GB.

You can create, attach, connect, and move volumes, as well as change volume performance to meet your storage, performance, and application requirements.

After you attach and connect a volume to a compute instance, you can use the volume like a regular hard drive. You can also disconnect a volume and attach it to another compute instance without the loss of data.

Required Components

These components are required to create a volume and attach it to a compute instance:

- **Compute Instance**: A virtual machine (VM) running in the cloud appliance.
- **Volume attachment**: A paravirtualized attachment that is available for compute instances.
- **Volume**: A block volume or boot volume.

Block Volume Performance Options

Elastic Performance

When you configure block storage, you can select one of these block storage types:

- **High performance**: The Higher Performance elastic performance option is recommended for workloads with the highest I/O requirements, requiring the best possible performance, such as large databases.
- **Balanced performance**: Suitable for most applications including boot volumes.

  This performance option provides a good balance between performance and cost savings for most workloads, including workloads that perform random I/O such as boot volumes.

Performance Limitations and Considerations

The following performance results are for unformatted data volumes.

- Throughput performance on compute instances depends on the network bandwidth that is available to the compute instance, and further limited by that bandwidth for the volume.
- If Microsoft Defender Advanced Threat Protection (Microsoft Defender ATP) is enabled in an instance, it has a significant negative impact on disk I/O performance. However, carefully consider the security implications of disabling Microsoft Defender ATP.
- Block volume performance is per volume, so when a block volume is attached to multiple compute instances, the performance is shared across all the attached compute instances.

You can change a volume's performance setting anytime by editing the volume. Refer to the Block Volume Storage in the Oracle Private Cloud Appliance User Guide.

Block Volume Scenarios

Adding Storage Capacity

A common use of a block volume is to add storage capacity to an instance. After you launch an instance and set up your cloud network, you can create a block storage volume. Then, you attach the volume to a compute instance using a volume attachment. The volume can then be mounted and used by your compute instance.

Moving a Volume to Another Compute Instance

A block volume can be detached from a compute instance and moved to a different compute instance without the loss of data. This data persistence enables you to migrate data between compute instances and ensures that your data is safely stored, even when it is not connected to a compute instance. Any data remains intact until you delete the volume.

To move your volume to another compute instance, unmount the drive from the initial compute instance, detach the block volume, then attach the volume to another compute instance. From there, you connect and mount the drive from that instance's guest OS to have access to the data.

Scaling a Compute Instance

When you terminate a compute instance, you can keep the associated boot volume and use it to launch a new compute instance with a different compute instance type or shape. This capability enables you to easily scale up or scale down the number of cores for a compute instance.

Block Volume Groups

The Block Volume service enables you to group multiple volumes in a volume group.

A volume group can include both types of volumes:

- **Boot Volumes**: The system disks for compute instances
- **Block Volumes**: Volumes for data storage

You can use volume groups to create volume group backups and clones that are point-in-time and crash-consistent.

This simplifies the process to create time-consistent backups of running enterprise applications that span multiple storage volumes across multiple compute instances. You can then restore an entire group of volumes from a volume group backup.

> **Note:**
>
> You can also clone an entire volume group in a time-consistent and crash-consistent manner. A deep disk-to-disk and fully isolated clone of a volume group, with all the volumes associated in it, becomes available for use within a matter of seconds. This action speeds up the process of creating environments for development, quality assurance, user acceptance testing, and troubleshooting.

This capability is available using the Compute Web UI, CLI, or API.

Volume groups and volume group backups are high-level constructs that allow you to group multiple volumes. When working with volume groups and volume group backups, keep the following points in mind:

- You can only add a volume to a volume group when the volume status is available.
- You can add up to 32 volumes in a volume group. The maximum Block Volumes aggregated size is 100 TB.
- Each volume can only be in one volume group.
- When you clone a volume group, a new group with new volumes are created. For example, if you clone a volume group containing three volumes, after this operation is complete, you have two separate volume groups and six different volumes with nothing shared between the volume groups.
- When you update a volume group using the CLI or API, you need to specify all the volumes to include in the volume group each time you use the update operation. If you do not include a volume ID in the update call, that volume is removed from the volume group.
- When you delete a volume group, the individual volumes in the group are not deleted, only the volume group is deleted.
- When you delete a volume that is part of a volume group, you must first remove it from the volume group before you can delete it.
- When you delete a volume group backup, all the volume backups in the volume group backup are deleted.

Volume Backups and Clones

Backup and Restore

Backup and restore activities are supported on boot volumes, data volumes, and volume groups.

To back up a boot volume, you can create a manual backup or clone the boot volume. See Cloning a Boot Volume.

There are two backup methods for block volumes and volume groups:

- **Manual backups**: You manually perform `create`, `get`, `list`, `rename`, and `delete backup` commands.
- **Automatic backups**: You create a backup policy and a backup policy assignment that specifies the time and frequency of the volume backups. The system automatically performs the commands that back up the volume.

Volume Group Back Up and Restore

You can perform most of the same backup operations and tasks with volume groups that you can perform with individual block volumes.

Volume group backups enable you to manage the backup settings for several volumes in one place. This feature simplifies the process to create time-consistent backups of running enterprise applications that span multiple storage volumes across multiple compute instances.

You can restore a volume group backup to a volume group, or you can restore individual volumes in the volume group from volume backups.

Clones

You can create a clone from a volume using the Block Volume service. Cloning enables you to make a copy of an existing block volume without needing to go through the backup and restore process.

The clone operation occurs immediately, and you can attach and use the cloned volume as a regular volume when the state changes to available. At this point, the volume data is being copied in the background, and can take up to thirty minutes depending on the size of the volume.

There is a single point-in-time reference for a source volume while it is being cloned. If the source volume is attached when a clone is created, you need to wait for the first clone operation to complete from the source volume before creating additional clones. If the source volume is detached, you can create up to 10 clones from the same source volume simultaneously.

You can only create a clone for a volume within the same tenant. You can create a clone for a volume between compartments as long as you have the required access permissions for the operation.

Differences Between Volume Clones and Backups

Consider the following criteria when you decide whether to create a backup or a clone of a volume.

Comparison	Volume Backup	Volume Clone
Description	Creates a point-in-time backup of data on a volume. You can restore multiple new volumes from the backup later in the future.	Creates an immediately usable copy of a block volume without having to go through the backup and restore process.
Use Case	Retain a backup of the data in a volume, so that you can duplicate an environment later or preserve the data for future use. Meet compliance and regulatory requirements, because the data in a backup remains unchanged over time. Support business continuity requirements. Reduce the risk of outages or data mutation over time.	Creates an immediately usable copy of a block volume without having to go through the backup and restore process.
Storage Location	Block Volume	Block Volume
Retention Policy	Policy-based backups expire, manual backups do not expire.	No expiration
Volume Groups	Supported. You can back up a volume group.	Supported. You can clone a volume group.

9

File Storage Overview

The File Storage service provides a durable, scalable, secure, enterprise-grade network file system.

The File Storage service supports these protocols:

- Network File System version 4.1 (NFSv4.1)
- Network File System version 4.0 (NFSv4)
- Network File System version 3.0 (NFSv3)
- Server Message Block (SMBv2 - SMBv3.1) – Requires Active Directory

For step-by-step instructions for managing File Storage, refer to File System Storage in the Oracle Private Cloud Appliance User Guide.

File Storage Connectivity

You can connect to a File Storage service file system from any instance in your Virtual Cloud Network (VCN).

Suitable Workloads

The File Storage service is designed to meet the needs of applications and users that need an enterprise file system across a wide range of use cases, including the following:

- **General Purpose File Storage:** Access to a pool of file systems to manage growth of structured and unstructured data.
- **Big Data and Analytics:** Run analytic workloads and use shared file systems to store persistent data.
- **Lift and Shift of Enterprise Applications:** Migrate existing Oracle applications that need NFS storage, such as Oracle E-Business Suite and PeopleSoft.
- **Databases and Transactional Applications:** Run test and development workloads with Oracle, MySQL, or other databases.
- **Backups, Business Continuity, and Disaster Recovery:** Host a secondary copy of relevant file systems from on premises to the cloud for backup and disaster recovery purposes.
- **Portable Operating System Interface (POSIX)-compliant file system**
- **MicroServices and Docker:** Deliver stateful persistence for containers. Easily scale as your container-based environments grow.

Data Protection with Snapshots

The File Storage service supports snapshots for data protection of your file system. Snapshots are a consistent, point-in-time view of your file systems. Snapshots are copy-on-write, and scoped to the entire file system. The File Storage service encrypts all file system and snapshot data at rest. You can take as many snapshots as you need.

For data protection, you can use a tool that supports NFS to copy your data to a different file system, object storage, or remote location.

For best performance, use the parallel tar (`partar`) and parallel copy (`parcp`) tools provided in the File Storage Parallel File Toolkit for this purpose. These tools work best with parallel workloads and requests. The Parallel File Toolkit is available for Oracle Linux, Red Hat Enterprise Linux, and CentOS. You can use `rsync` or regular `tar` for other operating system types. See Installing the Parallel File Tools for more information.

File Storage Objects

Mount Target

A mount target is an NFS endpoint in a subnet of your choice. The mount target provides the IP address that is used in the mount command when connecting NFS clients to a file system. File systems are exported (made available) through mount targets.

For an instance to mount a file system, the instance's Virtual Cloud Network (VCN) must have a mount target. A VCN can only have one mount target.

You can reuse the same mount target to make many file systems available. To reuse the same mount target for multiple file systems, create an export in the mount target for each file system.

Export

Exports control how NFS clients access file systems when they connect to a mount target. File systems are exported (made available) through mount targets. Each mount target maintains an export set which contains one or many exports. A file system must have at least one export in one mount target for compute instances to mount the file system.

Export Set

An export set is a collection of one or more exports that control what file systems the mount target exports and how those file systems are found using the NFS mount protocol. Each mount target has an export set. Each file system associated with the mount target has at least one export in the export set.

Export Path

The export path uniquely identifies the file system within the mount target. The export path is used by a compute instance to mount (logically attach to) the file system. For more information, see File Storage Paths.

Export Options

NFS export options are a set of parameters within the export that specify the level of access granted to NFS clients when they connect to a mount target. An NFS export options entry within an export defines access for a single IP address or CIDR block range.

File System

In Private Cloud Appliance, file system refers to a file system that is accessed by one or more clients over the network. File systems are associated with a single compartment. File systems must have at least one export in one mount target for any client to mount and use the file system. Data is added to a file system from the client that has mounted (has access to) the file system.

The total number of file systems is limited to 100 per tenancy.

Virtual Cloud Network (VCN)

A private network that you set up in the Private Cloud Appliance, with firewall rules and specific types of communication gateways that you can choose to use. A VCN covers a single, contiguous IPv4 CIDR block of your choice.

Subnet

Subnets are subdivisions you define in a VCN (for example, 10.0.0.0/24 and 10.0.1.0/24). Subnets contain virtual network interface cards (VNICs), which attach to compute instances. A subnet consists of a contiguous range of IP addresses that do not overlap with other subnets in the VCN.

Security Rules

Security rules are virtual firewall rules for your VCN. Your VCN comes with a default security list, and you can add more. These security lists provide ingress and egress rules that specify the types of traffic allowed in and out of the compute instances. You can choose whether a given rule is stateful or stateless. Security list rules must be set up so that clients can connect to file system mount targets.

Another method for applying security rules is to set them up in a network security group (NSG), and then add the mount target to the NSG. Unlike security list rules that apply to all VNICs in the subnet, NSGs apply only to resource VNICs you add to the NSG.

Snapshots

Snapshots provide a consistent, point-in-time view of your file system, and you can take as many snapshots as you need. Each snapshot reflects only data that changed from the previous snapshot.

File Storage Paths

The File Storage service uses these kinds of paths:

- **Export Paths** are part of the information contained in an export that makes a file system available through a mount target.

  The export path is automatically generated when you create an export, and it uniquely identifies the file system within the mount target.

  Note – When you create an export from the CLI, you must specify a `--path` ***<path>*** argument. The path you specify is recorded but not used for mounting file systems. The appliance auto-generates a path that is used to mount the file system.

  Export path syntax:

  `/export/`***`<file-system-OCID-unique-string>`***

  where:

  – /export/ – Is the beginning of the export path.

  – ***<file-system-OCID-unique-string>*** – Is the unique character string portion of the file system's OCID.

  For example, a file system with this OCID . . .

```
ocid1.filesystem.oc1.pca.d0v812zdp48onybubehhx1c67i4p3mjfth5avt3z2rkn50uqpbce
3fhsa8nm
```

. . . has an export path that looks like this:

```
/export/d0v812zdp48onybubehhx1c67i4p3mjfth5avt3z2rkn50uqpbce3fhsa8nm
```

The export path is used by a file system client to mount (logically attach to) the file system. This path is unrelated to any path within the file system or the client instance. It exists solely as a way to distinguish one file system from another within a single mount target.

Example of an export path in a client's mount command:

```
sudo mount -t nfs  \
 -o nfsvers=4.0 192.0.2.0:/export/
d0v812zdp48onybubehhx1c67i4p3mjfth5avt3z2rkn50uqpbce3fhsa8nm /mnt/fs
```

In this mount command example, `192.0.2.0` is the mount target IP address. `/export/d0v812zdp48onybubehhx1c67i4p3mjfth5avt3z2rkn50uqpbce3fhsa8nm` is the unique export path that was specified when the file system was associated with a mount target during creation.

Export paths cannot be edited after the export is created.

For more information about export paths and mounting file systems, refer to the File System Storage in the Oracle Private Cloud Appliance User Guide.

- **Mount Point Paths** are paths within a client instance to a locally accessible directory to which the remote file system is mounted.

  In this mount command example, `/mnt/fs` is the path to the directory on the client instance on which the external file system is mounted.

  ```
  sudo mount -t nfs  \
   -o nfsvers=4.0 192.0.2.0:/export/
  d0v812zdp48onybubehhx1c67i4p3mjfth5avt3z2rkn50uqpbce3fhsa8nm /mnt/fs
  ```

- **File System Paths** are paths to directories within the file system, and contain the contents of the file system. When the file system is mounted, you can create any directory structure within it.

VCN Security Rules for File Storage

Before you can mount a file system, you must configure security rules to allow traffic to the mount target's VNIC using specific protocols and ports. Security rules enable traffic for the following:

- Open Network Computing Remote Procedure Call (ONC RPC) rpcbind utility protocol
- Network File System (NFS) protocol
- Network File System (MOUNT) protocol
- Network Lock Manager (NLM) protocol

Ways to Enable Security Rules for File Storage

The Networking service offers two virtual firewall features that both use security rules to control traffic at the packet level. The two features are:

- **Security lists:** The original virtual firewall feature from the Networking service. When you create a VCN, a default security list is also created. Add the required rules to the security list for the subnet that contains the mount target.
- **Network security groups (NSGs):** A subsequent feature designed for application components that have different security postures. Create an NSG that contains the required rules, and then add the mount target to the NSG. Each mount target can belong to up to five (5) NSGs.

> **Important:**
>
> You can use security lists alone, network security groups alone, or both together. It depends on your particular security needs.
>
> If you choose to use both security lists and network security groups, the set of rules that applies to a given mount target VNIC is the combination of these items:
>
> - The security rules in the security lists associated with the VNIC's subnet
> - The security rules in all NSGs that the VNIC is in
>
> It doesn't matter which method you use to apply security rules to the mount target VNIC, as long as the ports for protocols necessary for File Storage are correctly configured in the rules applied.

For additional conceptual information, see Virtual Firewall.

For instructions on how to create security rules and NSGs for the File Storage service, refer to the section titled Controlling Access to File Storage in the File System Storage chapter in the Oracle Private Cloud Appliance User Guide.

For general instructions for creating security lists and NSGs, refer to the sections titled *Controlling Traffic with Security Lists* and *Controlling Traffic with Network Security Groups* in the Networking chapter in the Oracle Private Cloud Appliance User Guide.

File Storage Network Ports

This table lists the port numbers used for networking services that support File Storage.

When you configure security lists and network security groups for the File Storage service, use the port numbers for the particular service you are using. For example, for NFS configure ports 2048–2050 for TCP and UDP.

Service	Protocol	Port
RPC	TCP, UDP	111
NFS	TCP, UDP	2048, 2049, 2050
lockd	TCP, UDP	4045
mountd	TCP, UDP	20048
SMB	TCP, UDP	445
LDAP	TCP, UDP	389
Kerberos	TCP, UDP	88

Service	Protocol	Port
DNS	TCP, UDP	53

Export Options for File Storage

NFS export options enable you to create more granular access control than is possible using security list rules to limit VCN access. You can use NFS export options to specify access levels for IP addresses or CIDR blocks connecting to file systems through exports in a mount target. Access can be restricted so that each client's file system is inaccessible and invisible to the other, providing better security controls in multi-tenant environments.

Using NFS export option access controls, you can limit clients' ability to connect to the file system and view or write data. For example, if you want to allow clients to consume but not update resources in your file system, you can set access to read-only. You can also reduce client root access to your file systems and map specified User IDs (UIDs) and Group IDs (GIDs) to a single anonymous UID/GID of your choice.

Export Options

Exports control how NFS clients access file systems when they connect to a mount target. File systems are exported (made available) through mount targets.

NFS export options are a set of parameters within the export that specify the level of access granted to NFS clients when they connect to a mount target. An NFS export options entry within an export defines access for a single IP address or CIDR block range. You can have up to 100 options per file system.

Each separate client IP address or CIDR block you want to define access for needs a separate export options entry in the export. For example, if you want to set options for NFS client IP addresses 10.0.0.6, 10.0.0.08, and 10.0.0.10, you need to create three separate entries, one for each IP address.

When there is more than one export using the same file system and same mount target, the export options that are applied to an instance are the options with a source that most closely matches the instance IP address. The smallest (most specific) match takes precedence across all the exports. Therefore, you can determine which export options are applied to an instance by looking at source value of all the exports.

For example, consider the following two export options entries specifying access for an export:

Entry 1: Source: 10.0.0.8/32, Access: Read/Write

Entry 2: Source: 10.0.0.0/16, Access: Read-only

In this case, clients who connect to the export from IP address 10.0.0.8 have read/write access. When there are multiple export options, the most specific match is applied.

> **Important:**
>
> When more than one file system is exported to the same mount target, you must export to the mount target with the smallest network (largest CIDR number) first. For detailed information and instructions, refer to My Oracle Support Doc ID 2823994.1.

> **Important:**
>
> File systems can be associated with one or more exports, contained within one or more mount targets.
>
> If the client source IP address does not match any entry on the list for a single export, then that export is not visible to the client. However, the file system could be accessed through other exports on the same or other mount targets. To completely deny client access to a file system, be sure that the client source IP address or CIDR block is not included in any export for any mount target associated with the file system.

For information about how to configure export options for various file sharing scenarios, see NFS Access Control Scenarios.

For more information about configuring export options, refer to the section titled *Setting NFS Export Options* in the File System Storage chapter in the Oracle Private Cloud Appliance User Guide.

NFS Export Option Defaults

When you create a file system and export, the NFS export options for that file system are set to the following defaults, which allow full access for all NFS client source connections. These defaults must be changed if you want to restrict access:

Note – When using the CLI to set the export options, if you set the options with an empty array (no options specified), the export is not accessible to any clients.

Export Option in the Web UI	Export Option in the CLI	Default Value	Description
Source:	`source`	0.0.0.0/0	The IP address or CIDR block of a connecting NFS client.
Ports:	`require-privileged-source-port`	Any	Always set to: • Web UI: Any • CLI: `false`
Access:	`access`	Read/Write	Specifies the source NFS client access. Can be set to one of these values: • `READ_WRITE` • `READ_ONLY`

Export Option in the Web UI	Export Option in the CLI	Default Value	Description
Squash:	`identity-squash`	None	Determines whether the clients accessing the file system as root have their User ID (UID) and Group ID (GID) remapped to the squash UID/GID. These are the possible values: • Root – Only the root user is remapped. • None – No users are remapped.
Squash UID/GID:	`anonymous-uid`and `anonymous-gid`	65534	This setting is used along with the Squash option. When remapping a root user, you can use this setting to change the default anonymousUid and anonymousGid to any user ID of your choice.

NFS Access Control Scenarios

Learn different ways to control NFS access by reviewing these scenarios:

- **Scenario A:** Provides a managed environment for two clients. The clients share a mount target, but each has their own file system, and cannot access each other's data.
- **Scenario B:** Provides data to clients for consumption, but doesn't allow them to update the data.
- **Scenario C:** Increases security by limiting the root user's privileges when connecting to a file system.

Scenario A: Control Host Based Access

Provide a managed hosted environment for two clients. The clients share a mount target, but each has their own file system, and cannot access each other's data. For example:

- Client A is assigned to CIDR block 10.0.0.0/24, and requires read/write access to file system A, but not file system B.
- Client B is assigned to CIDR block 10.1.1.0/24, and requires read/write access to file system B, but not file system A.
- Client C is assigned to CIDR block 10.2.2.0/24, and has no access of any kind to file system A or file system B.
- Both file systems A and B are associated to a single mount target, MT1. Each file system has an export contained in the export set of MT1.

Because Client A and Client B access the mount target from different CIDR blocks, you can set the client options for both file system exports to allow access to only a

single CIDR block. Client C is denied access by not including its IP address or CIDR block in the NFS export options for any export of either file system.

Web UI Example

Set the export options for file system A to allow read/write access only to Client A, who is assigned to CIDR block 10.0.0.0/24. Client B and Client C are not included in this CIDR block, and cannot access the file system.

> **Note:**
>
> To learn how to access the NFS export options in the Compute Web UI refer the section titled Setting NFS Export Options in the File System Storage chapter of the Oracle Private Cloud Appliance User Guide.

Source	Ports	Access	Squash	Squash UID/GID
10.0.0.0/24	Any	Read/Write	None	(not used)

Set the export options for file system B to allow read/write access only to Client B, who is assigned to CIDR block 10.1.1.0/24. Client A and Client C are not included in this CIDR block, and cannot access the file system.

Source	Ports	Access	Squash	Squash UID/GID
10.1.1.0/24	Any	Read/Write	None	(not used)

CLI Example

> **Note:**
>
> To learn how to access the NFS export options in the OCI CLI, refer the section titled Setting NFS Export Options in the File System Storage chapter of the Oracle Private Cloud Appliance User Guide.

Set the export options for file system A to allow `Read_Write` access only to Client A, who is assigned to CIDR block 10.0.0.0/24. Client B and Client C are not included in this CIDR block, and cannot access the file system.

```
oci fs export update --export-id File_system_A_export_ID --export-options \
'[{"source":"10.0.0.0/24","require-privileged-source-
port":"false","access":"READ_WRITE","identity-squash":"NONE","anonymous-
uid":"65534","anonymous-gid":"65534"}]'
```

Set the export options for file system B to allow `Read_Write` access only to Client B, who is assigned to CIDR block 10.1.1.0/24. Client A and Client C are not included in this CIDR block, and cannot access the file system.

```
oci fs export update --export-id File_system_B_export_ID --export-options \
'[{"source":"10.1.1.0/24 ","require-privileged-source-
port":"false","access":"READ_WRITE","identity-squash":"NONE","anonymous-
uid":"65534","anonymous-gid":"65534"}]'
```

Scenario B: Limit the Ability to Write Data

Provide data to customers for consumption, but don't allow them to update the data.

For example, you'd like to publish a set of resources in file system A for an application to consume, but not change. The application connects from IP address 10.0.0.8.

Web UI Example

Set the source IP address 10.0.0.8 to read-only in the export for file system A.

> **Note:**
>
> To learn how to access the NFS export options in the Compute Web UI refer the section titled *Setting NFS Export Options* in the File System Storage chapter of the Oracle Private Cloud Appliance User Guide.

Source	Ports	Access	Squash	Squash UID/GID
10.0.0.8	Any	Read-only	None	(not used)

CLI Example

> **Note:**
>
> To learn how to access the NFS export options in the OCI CLI, refer the section titled Setting NFS Export Options in the File System Storage chapter of the Oracle Private Cloud Appliance User Guide.

Set the source IP address 10.0.0.8 to `READ_ONLY` in the export for file system A.

```
oci fs export update --export-id File_System_A_export_OCID --export-options \
'[{"source":"10.0.0.8","require-privileged-source-
port":"false","access":"READ_ONLY","identitysquash":"NONE","anonymousuid":"65534"
,"anonymousgid":"65534"}]'
```

Scenario C: Improve File System Security

To increase security, you'd like to limit the root user's privileges when connecting to File System A. Use Identity Squash to remap root users to UID/GID 65534.

In UNIX-like systems, this UID/GID combination is reserved for '*nobody*', a user with no system privileges.

Web UI Example

Set the source IP address 10.0.0.8 to read-only in the export for file system A.

> **Note:**
>
> To learn how to access the NFS export options in the Compute Web UI refer the section titled Setting NFS Export Options in the File System Storage chapter of the Oracle Private Cloud Appliance User Guide.

Source	Ports	Access	Squash	Squash UID/GID
0.0.0.0/0	Any	Read/Write	Root	65534

CLI Example

> **Note:**
>
> To learn how to access the NFS export options in the OCI CLI, refer the section titled Setting NFS Export Options in the File System Storage chapter of the Oracle Private Cloud Appliance User Guide.

```
oci fs export update --export-id File_System_A_export_OCID --export-options   \
'[{"source":"0.0.0.0/0","require-privileged-source-
port":"false","access":"READ_WRITE","identitysquash":"ROOT","anonymousuid":"65534","ano
nymousgid":"65534"}]'
```

File System Snapshots

The File Storage service supports snapshots for data protection of your file system. Snapshots are a consistent, point-in-time view of your file systems. Snapshots are copy-on-write, and scoped to the entire file system. The File Storage service encrypts all file system and snapshot data at rest. You can take as many snapshots as you need.

The snapshot is accessible at `.zfs/snapshot/` ***name***.

File System Clones

A clone is a new file system that is created from a snapshot of an existing file system. Snapshots preserve the state of the data of a file system at a particular point in time. If you take snapshots of a file system at regular intervals, you can create clones of the file system as it existed at multiple points in its lifetime.

Cloned file systems are managed in the same way that any other file system is managed. See File System Storage in the Oracle Private Cloud Appliance User Guide.

A snapshot provides the initial blueprint for a clone. You can clone a parent file system, or you can clone a clone, as long as there's at least one snapshot available. At the point of creation, the data included in the clone is identical to the data in the snapshot. After creation, data changes in the clone aren't included in the original file system. Conversely, any data changes to the original file system aren’t included in the clone. All file systems operate independently of each other, regardless of whether they are parent file systems, clones, or clones of clones.

Clones are space and time efficient because creating a clone doesn't replicate or move any data from the parent file system to the clone. Instead, the clone references the parent file system for any data they share. A file system that is a clone of a clone also references the original parent file system for any shared data.

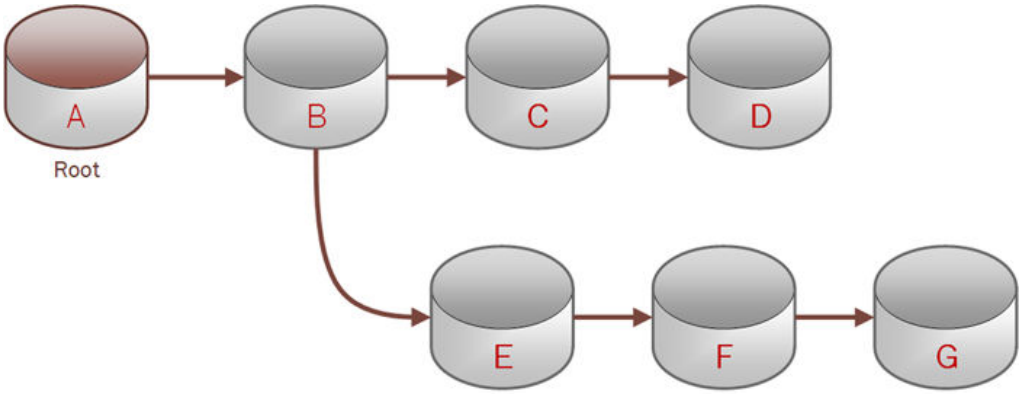
Parent File System
A parent file system is a file system that contains data referenced by one or many clones. When you create a clone, you must specify which file system snapshot is used as the blueprint for the clone directory hierarchy and file data. The file system that contains this snapshot is the parent of the clone. The clone continues to reference the parent file system for any data they share in common.

Source Snapshot
The snapshot used as a blueprint to create a clone. A snapshot is a point-in-time reference of a file system. You can take as many snapshots of a file system as you like, as often as you like. A parent file system can have snapshots available for many points along its lifetime. You can create a clone of your file system as it exists today, or as it existed in the past, as long as snapshots were taken of the file system at those times.

File System Clone
A clone is a new file system that is created based on a snapshot of existing file system. A clone automatically inherits the directory hierarchy and file data of the file system.

File system properties such as compartment, tags, display name, keys, and mount target export information are not copied over from the parent. These properties must be specified manually. You can access the clone by creating an export for it and mounting it to an instance in the same manner as any other file system.

Clone Tree
A clone tree is a group of clones that all descend from the same root file system. There is a transitive relationship between the root and the descendant clones. To delete the root of a clone tree, all its descendants must first be deleted.
In this diagram, B, C, D, E, F, G are all clones. A→ B→ C→D and A→ B→ E→ F→ G are all part of a clone tree. File system A is the root of this clone tree, and it is the parent of file system B.



Deleting File System Resources

File Systems
You can delete a file system if it is not a parent file system. If it is a parent file system, all descendant clones must first be deleted.

Snapshots
A parent snapshot can't be deleted. A snapshot that is not a parent can be deleted.

Clones
A parent clone can't be deleted. A clone that is not a parent can be deleted.

10 Object Storage Overview

The Private Cloud Appliance Object Storage service is a high-performance storage platform that offers reliable and cost-efficient data durability. The Object Storage service can store unstructured data of any content type, including analytic data and rich content, like images and videos.

Object Storage is not tied to any specific compute instance. You can access data from anywhere inside or outside the context of the appliance, as long you have internet connectivity and can access the Object Storage endpoint.

Object Storage offers multiple management interfaces that let you easily manage storage at scale.

For step-by-step instructions for managing Object Storage, refer to Object Storage in the Oracle Private Cloud Appliance User Guide.

Object Storage Resources

The Object Storage service uses these components to organize the data stored in object storage:

Objects

Any type of data, regardless of content type, is stored as an object.

An object is composed of the object itself and metadata about the object. Each object is stored in a bucket. The object is processed as a single entity.

Buckets

Buckets are logical containers for storing objects.

Users or systems create buckets as needed. A bucket is associated with a single compartment that has policies that determine what actions a user can perform on a bucket and on all the objects in the bucket. Buckets cannot be nested.

You determine the bucket names, but each bucket name must be unique within a namespace.

Namespace

An Object Storage namespace serves as the top-level container for all buckets and objects.

When your tenancy is provisioned, the tenancy is assigned one unique system-generated and immutable Object Storage namespace name. The namespace spans all compartments within the tenant.

Within a namespace, buckets and objects exist in flat hierarchy, but you can simulate a directory structure to help navigate a large set of objects.

Compartment

A compartment is the primary building block used to organize your cloud resources.

When your tenancy is provisioned, a root compartment is created for you. You can then create compartments under your root compartment to organize your resources. You control access by creating policies that specify what actions groups of users can take on the resources in those compartments. An Object Storage bucket can only exist in one compartment.

Object Naming Prefixes and Hierarchies

Within an Object Storage namespace, buckets and objects exist in a flat structure. However, you can simulate a directory structure by adding a prefix string that includes one or more forward slashes (/) to an object name. Doing so lets you list one directory at a time, which is helpful when navigating a large set of objects.

For example:

```
marathon/finish_line.jpg
marathon/participants/p_21.jpg
```

If you add prefixes to object names, you can:

- Use the OCI CLI or Compute API to perform bulk downloads and bulk deletes of all objects at a specified level of the hierarchy.
- Use the Compute Web UI to display a hierarchical view of your objects in virtual folders. In the previous example, `marathon` would be displayed as a folder containing an object named `finish_line.jpg` and `participants` would be a subfolder of `marathon`, containing an object named `p_21.jpg`. You can bulk upload objects to any level of the hierarchy and perform bulk deletes of all the objects in a bucket or folder.

Bulk operations at a specified level of the hierarchy do not affect objects in any level above.

When naming objects, you can also use prefix strings without a delimiter. No delimiters would allow search operations in the Compute Web UI and certain bulk operations in the OCI CLI orCompute API to match on the prefix portion of the object name. For example, in the object names below, the string `gloves_27_` can serve as a prefix for matching purposes when performing bulk operations:

```
gloves_27_dark_green.jpg
gloves_27_light_blue.jpg
```

When you perform bulk uploads with any of the UIs, you can add a prefix string to the names of the files you are uploading.

Object Names

Unlike other resources, objects do not have Oracle Cloud Identifiers (OCIDs). Instead, users define an object name when they upload an object.

Use the following guidelines when naming an object:

- The maximum length for an object and bucket name is 255 characters.
- Valid characters are letters (upper or lower case), numbers, and characters other than line feed, carriage return, and NULL.
- Bucket names and object names are case-sensitive.

- Use only Unicode characters for which the UTF-8 encoding does not exceed 1024 bytes. Clients are responsible for URL-encoding characters.
- Make the name unique within the bucket.
- Object names can include one or more forward slash (/) characters in the name. See Object Naming Prefixes and Hierarchies.

Optional Response Headers and Metadata

When you upload objects, you can provide optional response headers and user-defined metadata. Response headers are HTTP headers sent from Object Storage to Object Storage clients when objects are downloaded.

User-defined metadata are name-value pairs stored with an object.

You can use the Compute Web UI, Compute API, or OCI CLI to provide these optional attributes.

> **Important:**
>
> No validation is performed on the response headers or metadata you provide.

You can specify values for the following response headers:

- **Content-Disposition**

  Defines presentation only information for the object. Specifying values for this header has no effect on Object Storage behavior. Programs that read the object determine what to do based on the value provided. For example, you could use this header to let users download objects with custom file names in a browser. For Example:

  ```
  attachment; filename="fname.ext"
  ```

- **Cache-Control**

  Defines the caching behavior for the object. Specifying values for this header has no effect on Object Storage behavior. Programs that read the object determine what to do based on the value provided. For example, you could use this header to identify objects that require caching restrictions. For Example:

  ```
  no-cache, no-store
  ```

Metadata

You specify user-defined metadata in the form of name-value pairs. User-defined metadata names are stored and returned to Object Storage clients with the mandatory prefix of `opc-meta-`.

Multipart Uploads

The Object Storage service supports multipart uploads for more efficient and resilient uploads, especially for large objects.

You can perform multipart uploads using the Compute API or CLI. The Compute Web UI uses multipart uploads to upload objects larger than 64 MiB.

With multipart uploads, individual parts of an object can be uploaded in parallel to reduce the amount of time you spend uploading. Multipart uploads performed through the API can also minimize the impact of network failures by letting you retry a failed part upload instead of requiring you to retry an entire object upload.

Multipart uploads can accommodate objects that are too large for a single upload operation. For large uploads performed through the API, you have the flexibility of pausing between the uploads of individual parts, and resuming the upload when your schedule and resources allow.

Object Parts

With multipart upload, you split the object you want to upload into individual parts. Individual parts can be as large as 50 GiB. The maximum size for an uploaded object is 10 TiB.

Decide what part number you want to use for each part. Part numbers can range from 1 to 10,000. You do not need to assign contiguous numbers, but Object Storage constructs the object by ordering part numbers in ascending order.

Multipart Upload API

Before you use the multipart upload Compute API, you are responsible for creating the parts to upload. Object Storage provides API operations for the remaining steps.

A multipart upload performed using the API consists of the following steps:

1. Initiate an upload.
2. Upload object parts.
3. Commit the upload.

The service also provides API operations for listing in-progress multipart uploads, listing the object parts in an in-progress multipart upload, and aborting in-progress multipart uploads initiated through the API.

Multipart Upload CLI

When you perform a multipart upload using the OCI CLI, you do not need to split the object into parts as you are required to do by the Compute API. Instead, you specify the part size of your choice, and Object Storage splits the object into parts and performs the upload of all parts automatically. You can choose to set the maximum number of parts that can be uploaded in parallel. By default, the CLI limits the number of parts that can be uploaded in parallel to three. When using the CLI, you do not have to perform a commit when the upload is complete.

You can also use the CLI to list in-progress multipart uploads, and to abort multipart uploads initiated through the API.

Pre-Authenticated Requests

Pre-authenticated requests provide a way to let users access a bucket or an object without having their own credentials, as long as the request creator has permissions to access those objects.

For example, you can create a request that lets an operations support user upload backups to a bucket without owning API keys. Or, you can create a request that lets a business partner update shared data in a bucket without owning API keys.

When you create a pre-authenticated request, a unique URL is generated. Anyone you provide this URL to can access the Object Storage resources identified in the pre-authenticated request, using standard HTTP tools like curl and wget.

> **Important:**
>
> Assess the business requirements and the security ramifications of pre-authenticated access to a bucket or objects.
>
> A pre-authenticated request URL gives anyone who has the URL access to the targets identified in the request. Carefully manage the distribution of the URL.

Required Permissions

To Create a Pre-Authenticated Request

You need the `PAR_MANAGE` permission to the target bucket or object.

You must also have the appropriate permissions for the access type that you are granting. For example:

- If you are creating a pre-authenticated request for uploading objects to a bucket, you need the `OBJECT_CREATE` and `OBJECT_OVERWRITE` permissions.
- If you are creating a pre-authenticated request for read/write access to objects in a bucket, you need the `OBJECT_READ`, `OBJECT_CREATE`, and `OBJECT_OVERWRITE` permissions.

> **Important:**
>
> If the creator of a pre-authenticated request is deleted or loses the required permissions after they created the request, the request will no longer work.

To Use a Pre-Authenticated Request

Permissions of the pre-authenticated request creator are checked each time you use a pre-authenticated request.

The pre-authenticated request no longer works when any of the following occurs:

- Permissions of the pre-authenticated request creator have changed.
- The user who created the pre-authenticated request is deleted.
- A Federated user who created the pre-authenticated request has lost the user capabilities that they had when they created the request.
- Pre-authenticated request has expired.

Types of Pre-Authentication Requests

When creating a pre-authenticated request, you have the following options:

- You can specify the name of a bucket that a pre-authenticated request user has write access to and can upload one or more objects to.
- You can specify the name of an object that a pre-authenticated request user can read from, write to, or read from and write to.

Scope and Constraints

Understand the following scope and constraints regarding pre-authenticated requests:

- Users can't list bucket contents.
- You can create an unlimited number of pre-authenticated requests.
- There is no time limit to the expiration date that you can set.
- You can't edit a pre-authenticated request. If you want to change user access options in response to changing requirements, you must create a new pre-authenticated request.
- The target and actions for a pre-authenticated request are based on the creator's permissions. The request is not, however, bound to the creator's account login credentials. If the creator's login credentials change, a pre-authenticated request is not affected.
- You cannot delete a bucket that has a pre-authenticated request associated with that bucket or with an object in that bucket.

> **Important:**
>
> The unique URL provided by the system when you create a pre-authenticated request is the only way a user can access the bucket or object specified as the request target. Copy the URL to durable storage. The URL is displayed only at the time of creation and cannot be retrieved later.

Retention Rules

Retention rules provide immutable, WORM-compliant storage options for data written to Object Storage for data governance, regulatory compliance, and legal hold requirements.

Retention rules can also protect your data from accidental or malicious update, overwrite, or deletion. Retention rules can be locked to prevent rule modification and data deletion or modification even by administrators.

Retention rules are configured at the bucket level and are applied to all individual objects in the bucket.

Object Storage provides a flexible approach to data retention that supports the following use cases.

- **Regulatory Compliance**

  Your industry might require you to retain a certain class of data for a defined length of time. Your data retention regulations might also require that you lock the

retention settings. If you lock the settings, the only change you can make is to increase the retention duration.

For Object Storage regulatory compliance, you create a time-bound retention rule and specify a duration. Object modification and deletion are prevented for the duration specified. Duration is applied to each object individually, and is based on the object's Last Modified timestamp. Lock the rule as required.

- **Data Governance**

  You might need to protect certain data sets as a part of internal business process requirements. While retaining the data for a defined length of time is necessary, that time period could change.

  Create a time-bound retention rule and specify a duration. Object modification and deletion are prevented for the duration specified. Duration is applied to each object individually, and is based on the object's Last Modified timestamp. To be able to delete the rule and allow changes to the duration as required, do not lock the rule.

- **Legal Hold**

  You might need to preserve certain business data in response to potential or on-going lawsuits. A legal hold does not have a defined retention period and remains in effect until removed.

  For Object Storage legal holds, you create an indefinite retention rule. Object modification and deletion are prevented until you delete the rule. You cannot lock an indefinite retention rule because the rule has no duration.

It's important to understand retention duration for time-bound rules. Even though you are creating retention rules for a bucket, the duration of a rule is applied to each object in the bucket individually, and is based on the object's Last Modified timestamp. Let's say you have two objects in the bucket, ObjectX and ObjectY. ObjectX was last modified 14 months ago and ObjectY was last modified 3 months ago. You create a retention rule with a duration of 1 year. This rule prevents the modification or deletion of ObjectY for the next 9 months. The rule allows the modification or deletion of ObjectX because the retention rule duration (1 year) is less that the object's Last Modified timestamp (14 months). If ObjectX is overwritten some time in the coming year, modification and deletion would be prevented for the rule duration time remaining.

Locking a retention rule is an irreversible operation. Not even a tenancy administrator can delete a locked rule. There is a mandatory 14-day delay before a rule is locked. This delay lets you thoroughly test, modify, or delete the rule or the rule lock before the rule is permanently locked. A rule is active at the time of creation. The lock only controls whether the rule itself can be modified. After a rule is locked, only increases in the duration are allowed. Object modification is prevented and the rule can only be deleted by deleting the bucket. A bucket must be empty before it can be deleted.

Scope and Constraints

- Retention rules can be applied to a bucket in the Object Storage.
- The actions that you can perform on a bucket with active retention rules are limited. You cannot update, overwrite, or delete objects or object metadata until the retention rule is deleted (indefinite rule) or for the duration specified (time-bound rules). The duration for time-bound rules is applied to each object individually, and is based on the object's Last Modified timestamp.
- You can create multiple retention rules for a bucket. Indefinite retention rule is applied before any time-bound rule is considered.

- When a retention rule is locked, the rule can only be deleted by deleting the bucket. A bucket must be empty before it can be deleted.

Interaction Between Retention and Other Object Storage Features

Carefully review the policies and rules that you have in place for the other Object Storage features that you are using. Some of these policies and rules might not make sense with retention rules. This section describes some key things you need to know about the interaction between retention rules and other Object Storage features.

Multipart Uploads

Uncommitted (unfinished or failed) multipart uploads are not protected by retention rules and can be deleted at any time.

Versioning

- You cannot add retention rules to a bucket that has versioning enabled.
- You cannot enable versioning on a bucket with active retention rules.
- You can add retention rules to bucket that has versioning suspended. However, you cannot resume versioning with active retention rules.

Troubleshooting Retention Rules

Unable to Create a Retention Rule

If creating a retention rule fails, the most likely cause is missing or incomplete IAM permissions. Rule creation requires:

- User permissions that let you access the bucket and manage the objects in those buckets.
- Minimally, BUCKET_UPDATE and RETENTION_RULE_MANAGE permissions.
- Minimally, BUCKET_UPDATE and RETENTION_RULE_MANAGE permissions.

Unable to Lock a Retention Rule

If locking a retention rule fails, the most likely cause is missing or incomplete IAM permissions. Minimally, BUCKET_UPDATE, RETENTION_RULE_MANAGE, and RETENTION_RULE_LOCK permissions are required to lock retention rules.

Unable to Delete a Retention Rule

You cannot delete a time-bound retention rule that is locked. When a retention rule is locked, the rule can only be deleted by deleting the bucket. A bucket must be empty before it can be deleted.

If deleting an indefinite retention rule fails, the most likely cause is missing or incomplete IAM permissions. Rule deletion requires:

- User permissions that let you access the bucket and manage the objects in those buckets.
- Minimally, BUCKET_UPDATE and RETENTION_RULE_MANAGE permissions.

Object Versioning

Object versioning provides data protection against accidental or malicious object update, overwrite, or deletion.

Object versioning is enabled at the bucket level. Versioning directs Object Storage to automatically create an object version each time one of these actions takes place:

- A new object is uploaded.
- An existing object is overwritten.
- When an object is deleted.

You can enable object versioning at bucket creation time or later.

A bucket that is versioning-enabled can have many versions of an object. There is always one latest version of the object and zero or more previous versions.

Object Version Deletion

No object is physically deleted from a bucket that has versioning enabled until you take explicit action to do so.

When you delete an object without targeting a specific version, the latest object version becomes a previous object version and a special delete marker is created that marks the deletion point. A delete marker contains only minimal metadata. If you delete a folder, a delete marker is created for each object in the folder. You can simply delete the delete marker to make that deleted version become the latest object version.

When you upload an object with the same name as the delete marker, the uploaded object becomes the latest version of the object. The delete marker remains. There can be multiple delete markers for an object and you can recover any of the previous object versions.

Object version deletion is different. When you delete an object version, the version is permanently deleted. Permanent deletion also happens if you explicitly delete the latest version by version ID. All delete operations that target a specific object version ID permanently deletes the data.

Scope and Constraints

- Versioning can be enabled on a bucket in Object Storage.
- You can rename the latest version of an object, but you cannot rename a previous object version. Renaming an object creates a new object.

Interaction Between Versioning and Other Object Storage Features

This section describes some key things you need to know about the interaction between object versioning and other Object Storage features.

Copying Objects

If you copy the latest version of an object to a different bucket, only the object is copied. None of the object's previous versions are copied. You can copy a previous version of an object to another bucket, but that action creates either the latest version of a new object or a new object version in the destination bucket.

Retention Rules

- You cannot add retention rules to a bucket that has versioning enabled.
- You cannot enable versioning on a bucket with active retention rules.
- You can add retention rules to bucket that has versioning suspended. However, you cannot resume versioning with active retention rules.

Troubleshooting Versioning

Unable to Enable Versioning

If enabling versioning fails, the most likely cause is missing or incomplete IAM permissions. Enabling versioning requires:

- User permissions that let you use the bucket and manage the objects in that bucket.
- Minimally, BUCKET_UPDATE permissions.

Unable to Delete a Bucket

If deleting a bucket fails, the most likely cause is that the bucket is not empty.

You can permanently delete an empty bucket. You cannot delete a bucket that contains any of the following:

- Any objects
- Previous versions of an object
- A multipart upload in progress
- A pre-authenticated request

> **Tip:**
>
> When you delete an object in a version-enabled bucket, a previous version of that object is created. Select Show Deleted Objects to display the object versions that might prevent you from deleting the bucket.

Unable to Delete a Previous Version

If deleting a previous object version fails, the most likely cause is missing or incomplete IAM permissions. Object version deletion requires:

- User permissions that let you use the bucket and manage the objects in that bucket.
- Minimally, OBJECT_VERSION_DELETE permissions.