Oracle® Real Application Clusters

Real Application Clusters Installation on Oracle Cloud Native Environment

19c Oracle Linux x86-64

G33091-01
October 2025

Oracle Real Application Clusters Real Application Clusters Installation on Oracle Cloud Native Environment, 19c Oracle Linux x86-64

G33091-01

Copyright © 2022, 2025, Oracle and/or its affiliates.

Primary Author: Douglas Williams

Contributing Authors: Param Saini

Contributors: Saurabh Ahuja, Gia-Khanh Nguyen, Jyoti Verma

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Copyright © 2022, 2025, Oracle and/or its affiliates.

Copyright © 2022, 2025, Oracle and/or its affiliates.

1

How to Install and Configure Oracle RAC on Oracle Cloud Native Environment

Use these instructions to install Oracle Real Application Clusters (Oracle RAC) on Oracle Cloud Native Environment (OCNE).

In this publication, the Linux servers hosting the Oracle Cloud Native Environment (OCNE) are referred to as the operator node, Kubernetes control plane and Worker nodes. The control plane node is responsible for Kubernetes cluster management and for providing the API that is used to configure and manage resources within the Kubernetes cluster. Worker nodes within the Kubernetes cluster are used to run Pods and handle networking to ensure that traffic between applications across the cluster and from outside of the cluster can be properly facilitated.

The Oracle RAC Pods will be deployed on Worker nodes.

- Overview of Oracle RAC on OCNE
  Learn how you can deploy Oracle Real Application Clusters (Oracle RAC) in OCNE environment for production deployments.
- Install and Configure the OCNE Environment
  To deploy Oracle RAC pods, install and configure Oracle Cloud Native Environment (OCNE).
- Provisioning the OCNE Operator Node, Kubernetes Control Plane and Worker Nodes
  You can provision the Linux server hosting the Kubernetes module either on a bare metal (physical) server, or on an Oracle Linux Virtual Machine (VM).
- Prerequisites for Oracle RAC on OCNE
  Before beginning to deploy Oracle Real Application Clusters (Oracle RAC) on Oracle Linux Cloud Native Environment (OCNE), ensure that you are prepared for the installation, and that your system meets software and storage requirements.
- OCNE Worker Node Overview and Preparations for Deployment of Oracle RAC
  Learn about OCNE configuration, and prepare the worker node and Oracle RAC Pods for deployment.
- Download Oracle Grid Infrastructure and Oracle Database Software
  Download the Oracle Database and Oracle Grid Infrastructure software from the Oracle Technology Network, and stage it.
- Create the Oracle RAC Pods
  To create the Oracle Real Application Clusters (Oracle RAC) StatefulSets, log in to the host that has the `kubectl` setup, and complete configuration steps.
- Deploy Oracle Grid Infrastructure in the Kubernetes Pods
  To deploy Oracle Grid Infrastructure in the Kubernetes Pods, complete this procedure.
- Deploy Oracle RAC Database in the Pods
  To deploy an Oracle RAC database in the OCNE Pods, complete this procedure.
- Check Oracle RAC Network Configuration
  To confirm IP addresses are functioning, check the public and private network configuration for RAC inside the pods using this procedure.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- How to Connect to the Database Using a Client
  Learn what is required to connect to the database with a client either inside or outside the Kubernetes cluster.
- Complete Pod Restart Configuration Tasks
- Options to Consider After Deployment
  After deployment of Oracle RAC in pods, you can choose to add or remove Oracle Real Application Clusters (Oracle RAC) nodes, or install different releases of Oracle RAC database.
- How to Clean Up a StatefulSet for Oracle RAC on OCNE
  If you need to remove a StatefulSet currently used for Oracle Real Application Clusters (Oracle RAC) on OCNE, then use this procedure.
- Known Issues for Oracle RAC on OCNE
  When you deploy Oracle Real Application Clusters (Oracle RAC) on OCNE, if you encounter an issue, check to see if it is a known issue.
- Additional Information for Oracle RAC on OCNE Configuration
  This information can help to resolve issues that can arise with Oracle Real Application Clusters (Oracle RAC) on OCNE.

Overview of Oracle RAC on OCNE

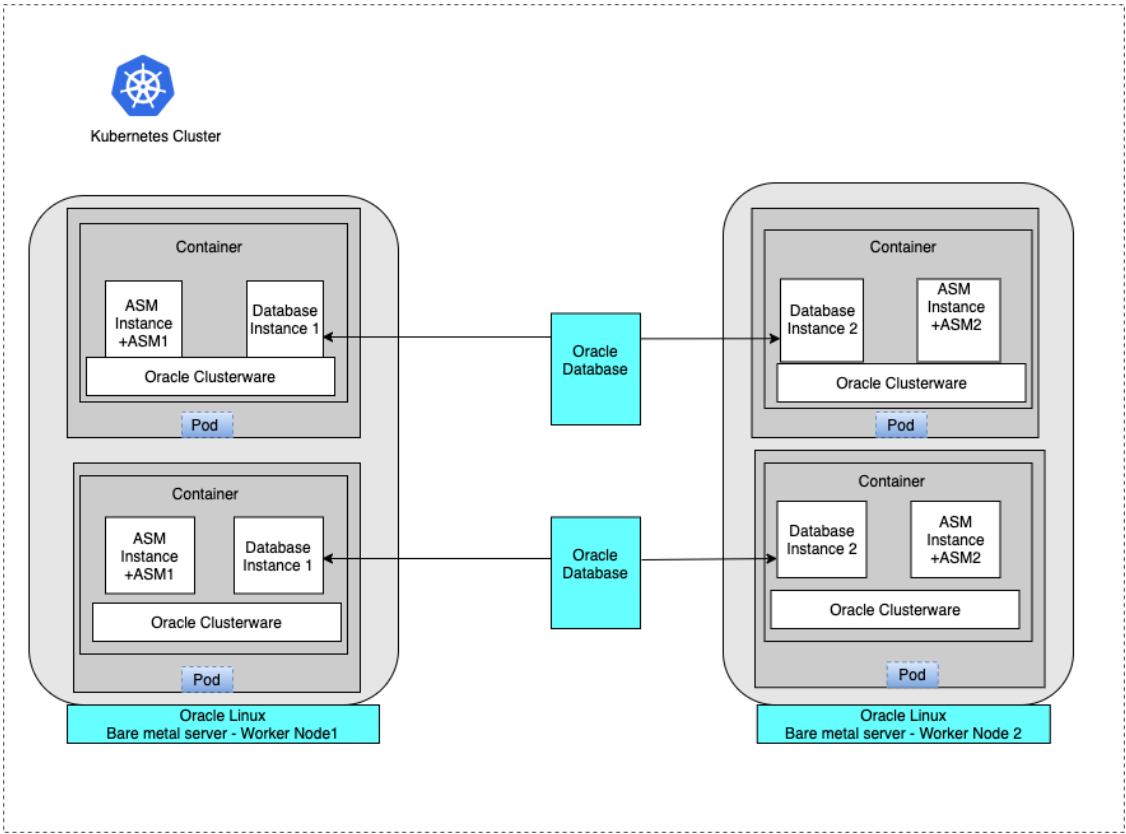
Learn how you can deploy Oracle Real Application Clusters (Oracle RAC) in OCNE environment for production deployments.

To prevent a single host causing complete downtime in a production environment, distribute Oracle RAC nodes across OCNE worker nodes running on different physical servers. It is possible to configure Oracle RAC on OCNE Worker nodes running on the same physical server for test and development environments. The procedures in this document are tested for a two-node Oracle RAC cluster, with each node running on a separate Linux OCNE Worker node, and using block devices for shared storage.

This figure shows a production deployment, in which containers are located in different OCNE Worker nodes on separate hardware servers, with a high availability storage and network configuration. Placing Oracle RAC Pods on different Worker node virtual machines on different hardware enhances high availability:

Copyright © 2022, 2025, Oracle and/or its affiliates.

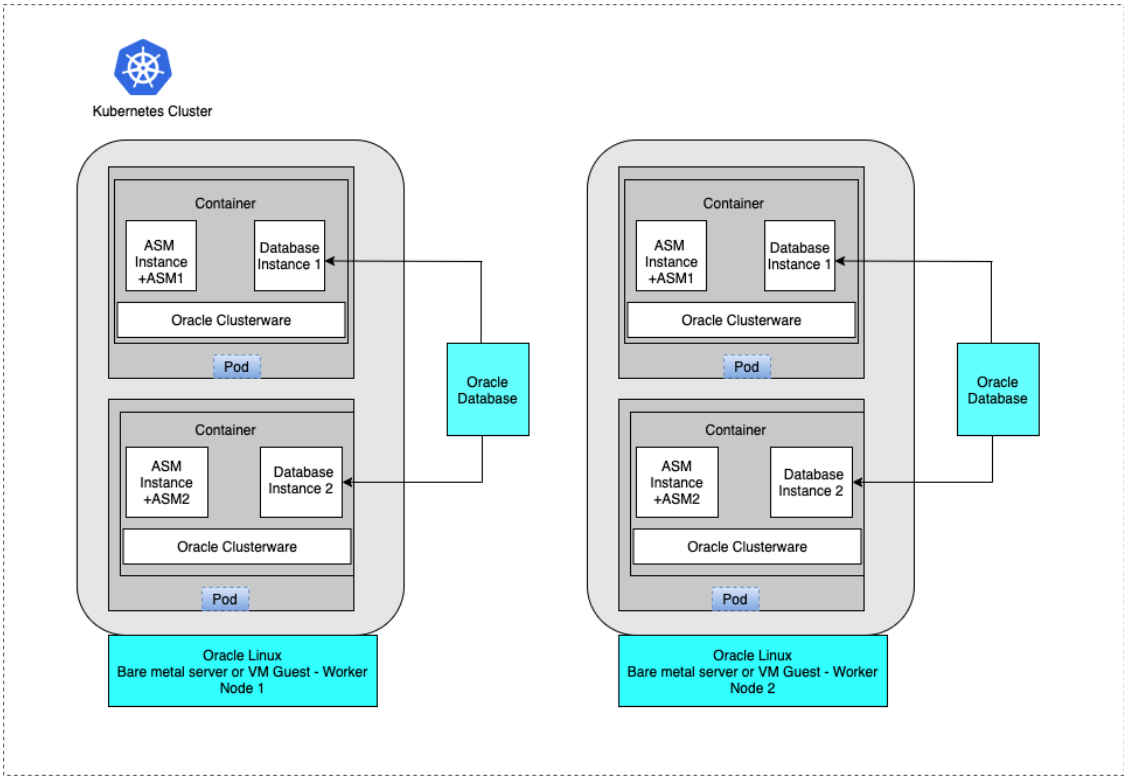
Figure 1-1 Bare Metal Server Production Deployment for Oracle RAC on OCNE



This figure shows another production configuration, in which there are two Oracle RAC pods on one host, in separate guest domains, and two RAC Pods on a second host. The second host also has two separate bare metal servers with a high availability storage and network configuration provided by Oracle Clusterware.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Figure 1-2 Oracle RAC Pods on Two Hosts Production Deployment for Oracle RAC on OCNE

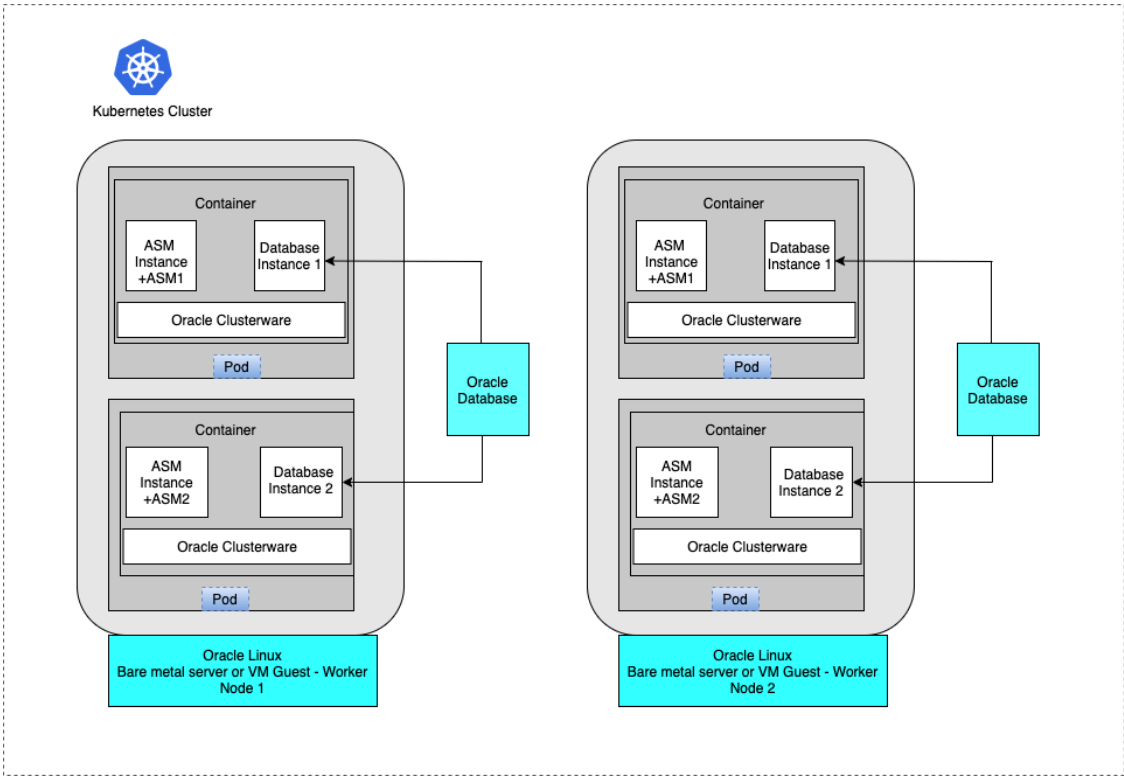


Test or Development Deployment Examples of Oracle RAC on OCNE

For virtual test and development clusters, you can use two or more Oracle RAC pods on the same cluster, running on only one Oracle Linux Worker node, because high availability is not required for testing.

Copyright © 2022, 2025, Oracle and/or its affiliates.

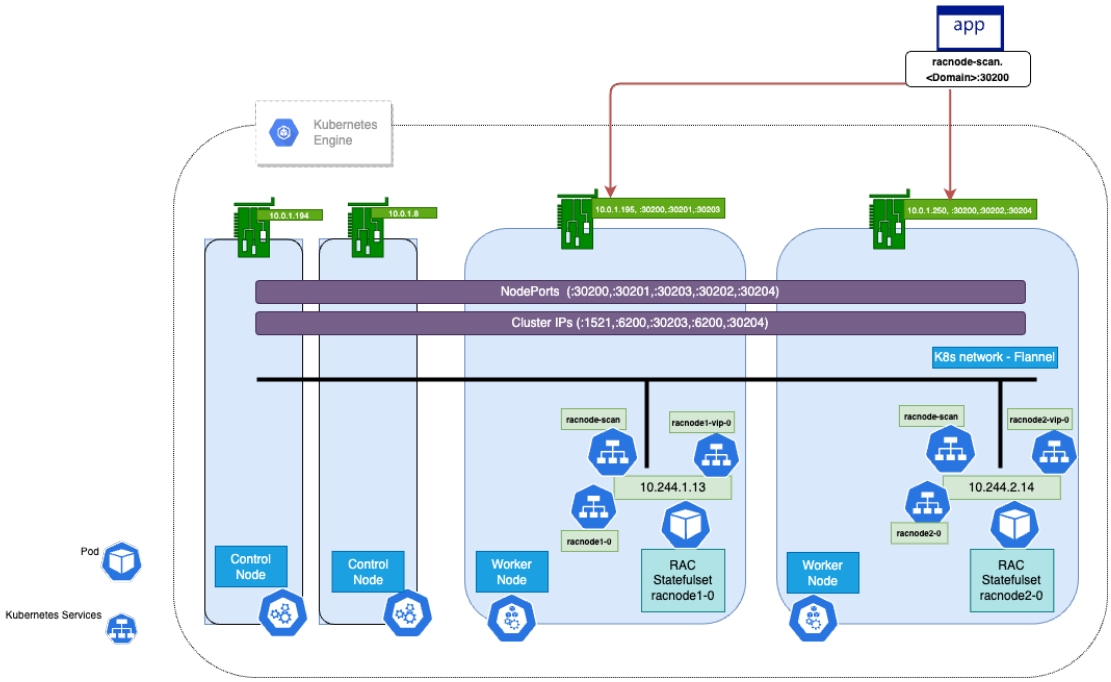
Figure 1-3 Oracle RAC Pods on the Same Cluster for Test or Development



Whole Deployment of Oracle RAC on OCNE

Based on these hardware configurations, the whole deployment of Oracle RAC on OCNE has the following configuration:

Figure 1-4 Configuration Diagram Oracle RAC on OCNE



Copyright © 2022, 2025, Oracle and/or its affiliates.

Install and Configure the OCNE Environment

To deploy Oracle RAC pods, install and configure Oracle Cloud Native Environment (OCNE).

To install your Oracle Cloud Native Environment, use the Oracle Cloud Native Environment Getting Started guide.

Each pod that you deploy as part of your cluster must satisfy the minimum hardware requirements of the Oracle Real Application Clusters (Oracle RAC) and Oracle Grid Infrastructure software. If you are planning to install Oracle Grid Infrastructure and Oracle RAC database software on data volumes exposed from your environment, then you must have at least 5 GB space allocated for the Oracle RAC on OCNE Pod.

Refer to the OCNE concept guide to look for the supported modules in the OCNE enviornment. You must install the OCNE cluster with the primary network plugin, which is Flannel. You also need to install and configure the Multus module.

For this deployment, the Multus module installation adds the Multus Container Networking Interface (CNI) into a Kubernetes cluster, which allows you to use it to create one or more L2 networks interconnecting the RAC Pods.

> **Note**
>
> During the installation of the OCNE Multus module, do not use the optional argument for a multus configuration file, as the actual configuration will be done later on. You must follow the network setup requirements to configure Multus for Oracle RAC, as describe below:
>
> Configure Multus

Also, as part of the Kubernetes module setup, after the Kubernetes cluster is created, you must set up the kubectl command-line interface either on a Kubernetes control plane node (recommended), or on a non-cluster node. For information about setting up the kubectl command-line interface, see:

Oracle Cloud Native Environment 1.9 documentation

In this guide, when you see a `kubectl` command in a step, the command is run from the host with the setup for the Kubernetes `kubectl` command line tool.

Related Topics

- Oracle Cloud Native Environment Documentation

Provisioning the OCNE Operator Node, Kubernetes Control Plane and Worker Nodes

You can provision the Linux server hosting the Kubernetes module either on a bare metal (physical) server, or on an Oracle Linux Virtual Machine (VM).

In addition to the standard memory (RAM) required for Oracle Linux (Linux-x86-64) and the Oracle Grid Infrastructure and Oracle Real Application Clusters (Oracle RAC) instances, Oracle recommends that you provide an additional 2 GB of RAM to each Kubernetes node for the control plane.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Prerequisites for Oracle RAC on OCNE

Before beginning to deploy Oracle Real Application Clusters (Oracle RAC) on Oracle Linux Cloud Native Environment (OCNE), ensure that you are prepared for the installation, and that your system meets software and storage requirements.

- Preparing to Install Oracle RAC on OCNE
  To use these instructions, you should have background knowledge of the technology and operating system.
- Software and Storage Requirements for Oracle RAC on OCNE
  Review which Oracle software releases are supported for deployment with Oracle RAC on Kubernetes, and what storage options you can use.
- Network Setup Requirements for Oracle RAC on OCNE
  Complete the Multus and Flannel network requirements for OCNE.

Preparing to Install Oracle RAC on OCNE

To use these instructions, you should have background knowledge of the technology and operating system.

You should be familiar with the following technologies:

- Linux
- Kubernetes
- Oracle Cloud Native Environment (OCNE)
- Oracle Real Application Clusters (Oracle RAC) installation
- Oracle Grid Infrastructure installation
- Oracle Automatic Storage Management (Oracle ASM).

For information about standard steps and details of configuration, refer to *Oracle Grid Infrastructure Installation and Upgrade Guide for Linux*. Review the Oracle Grid Infrastructure Installation Checklist before starting the installation.

Related Topics

- Oracle Grid Infrastructure Installation Checklist

Software and Storage Requirements for Oracle RAC on OCNE

Review which Oracle software releases are supported for deployment with Oracle RAC on Kubernetes, and what storage options you can use.

Software Requirements for Oracle Linux 8

- Oracle Grid Infrastructure Release 19c (minimum 19.3.0) , starting with Release Update 19.28.0
- Oracle Database Release 19c, (minimum 19.3.0), starting with Release Update 19.28.0
- Oracle Cloud Native Environment (OCNE) 1.9 or later
- OPatch utility version 12.2.0.1.46 or later. Refer to the 19.28 Release Update Readme for details
- Oracle Linux 8 Container image (`oraclelinux:8`)

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Oracle Linux for Operator, control plane, and Worker nodes on Oracle Linux 8 (Linux-x86-64) Update 5 or later updates
- Unbreakable Enterprise Kernel Release 7 UEKR7 (Kernel Release `5.15.0-208.159.3.2.el9uek.x86_64`) or later updates.

> **Note**
>
> In this documentation, we apply Release Update 19.28 on top of the Oracle Grid Infrastructure 19.3.0 and Oracle Database 19.3.0 homes.

Software requirements for Oracle Linux 9

- Oracle Grid Infrastructure Release 19c (minimum 19.3.0) , starting with Release Update 19.28.0
- Oracle Database Release 19c, (minimum 19.3.0), starting with Release Update 19.28.0
- Oracle Cloud Native Environment (OCNE) 1.9 or later
- OPatch utility version 12.2.0.1.46 or later. Refer to the 19.28 Release Update Readme for details
- Oracle Linux 9 Container image (`oraclelinux:9`)
- Oracle Linux for Operator, control plane, and Worker nodes on Oracle Linux 9 (Linux-x86-64) Update 3 or later updates.
- Unbreakable Enterprise Kernel Release 7 UEKR7 (Kernel Release 5.15.0-208.159.3.2.el8uek.x86_64) or later updates.

> **Tip**
>
> https://docs.oracle.com/en/operating-systems/
> Oracle Cloud Native Environment 1.9
>
> Oracle Cloud Native Environment

Deployment using Gold Images

(Optional) You can build Gold Images of Oracle Grid Infrastructure and Oracle RAC Database using the base software (19.3.0) and Release Update 19.28. For more information, see:

Gold Image How To (Doc ID 2965269.1)

> **See Also**
>
> https://docs.oracle.com/en/operating-systems/
> Oracle Cloud Native Environment 1.9
>
> Oracle Cloud Native Environment

Required Patches

In this documentation, we apply Release Update 19.28 on top of the Oracle Grid Infrastructure 19.3.0 and Oracle Database 19.3.0 homes.

Copyright © 2022, 2025, Oracle and/or its affiliates.

For this scenario you must download and apply the following patches

- Bug 38336965 - MERGE ON OCW RU 19.28.0.0.0 OF 37531398 37805442 (Apply to the Oracle Grid Infrastructure home)
- Bug 34436514 - DBCA REPORTS INCORRECT MEMORY IN PODMAN CONTAINERS (Apply to the Oracle Database home)

You can download patches using following procedure:

1. Log in to My Oracle Support (support.oracle.com) with your Oracle account.
2. Navigate to the “Patches & Updates” section.
3. Enter your product information or patch number to search for the desired patches.
4. Follow the prompts to download the patch files directly from the site.

> **Note**
>
> For the 19.28 Release Update, you must use the OPatch utility version 12.2.0.1.46 or later to apply this patch. Oracle recommends that you use the latest released OPatch 12.2.0.1.xx version for DB 19.0.0.0.0, which is available for download from My Oracle Support patch 6880880 by selecting "OPatch for DB 19.0.0.0.0" from the Select a Release dropdown. Oracle recommends that you download the OPatch utility and the patch to a shared location so that you can access them from any node in the cluster for the patch application on each node.

Storage Requirements

Database storage for Oracle RAC on OCNE must use Oracle Automatic Storage Management (Oracle ASM) configured either on block storage, or on a network file system (NFS).

> **Caution**
>
> Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Filter Driver (Oracle ASMFD) are not supported.

Related Topics

- Oracle RAC Technologies Certification Matrix for UNIX Platforms

Network Setup Requirements for Oracle RAC on OCNE

Complete the Multus and Flannel network requirements for OCNE.

An Oracle Clusterware configuration requires at least two interfaces:

- A **public** network interface, on which users and application servers connect to access data on the database server.
- A **private** network interface, on which internode communication between cluster member nodes takes place.

To create these required interfaces, you can use the following pod networking technologies with your Kubernetes cluster:

Copyright © 2022, 2025, Oracle and/or its affiliates.

- **Flannel**: This is the default networking option when you create a Kubernetes module. We use Flannel for the Oracle RAC public network. For Oracle RAC on OCNE, we support default pod networking with Flannel.
- **Multus**: You can set up the Multus module after the Kubernetes module is installed. Multus is installed as a module on top of Flannel. By default in Kubernetes, a pod is configured with single interface (and loopback)-based selected pod networking. To enable Kubernetes' multi-networking capability, Multus creates Custom Resources Definition (CRD)-based network objects, and creates a multi-container networking interface (CNI) plug-in. The Kubernetes application programming interface (API) can be expanded through CRD itself. For the Oracle RAC private network, you must configure Multus to have multiple network cards on different subnets, as described in this document.

Related Topics

- Configure Multus
  To create an Oracle RAC private network, configure Multus, using either the example configuration file provided here, or your own configuration file.
- Setting the Kubernetes Network - Multus Networking

OCNE Worker Node Overview and Preparations for Deployment of Oracle RAC

Learn about OCNE configuration, and prepare the worker node and Oracle RAC Pods for deployment.

- Worker Node Preparation for Oracle RAC on OCNE
  The procedures in this documentation have been tested on a two-node Oracle RAC cluster and deployed on two RAC pods. These Oracle RAC pods are running on two distinct worker nodes.
- Oracle RAC Pod Node Preparation
  Before you can install Oracle Grid Infrastructure and Oracle Real Application Clusters, you must configure the Oracle RAC Pods. .

Worker Node Preparation for Oracle RAC on OCNE

The procedures in this documentation have been tested on a two-node Oracle RAC cluster and deployed on two RAC pods. These Oracle RAC pods are running on two distinct worker nodes.

- OCNE Worker Node Configuration
  When configuring your Worker node servers, follow these guidelines, and see the configuration Oracle used for testing.
- Configure Multus
  To create an Oracle RAC private network, configure Multus, using either the example configuration file provided here, or your own configuration file.
- Oracle RAC StatefulSets
  Learn about the configuration used in this document, so that you can understand the procedures we follow.
- Set Clock Source on the Worker Node
  Oracle recommends that you set the clock source to `tsc` for better performance in virtual environments (VM) on Linux x86-64.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Setting up a Network Time Service
  You must set up the `chronyd` time service for Oracle Real Application Clusters (Oracle RAC).
- Configuring HugePages for Oracle RAC
  HugePages is a feature integrated into the Linux kernel. For Oracle Database, using HugePages reduces the operating system maintenance of page states and increases Translation Lookaside Buffer (TLB) hit ratio.
- Restrictions for HugePages and Transparent HugePages Configurations
  Review the HugePages and Transparent HugePages guidelines discussed in this section.

OCNE Worker Node Configuration

When configuring your Worker node servers, follow these guidelines, and see the configuration Oracle used for testing.

Each OCNE worker node must have sufficient resources to support the intended number of Oracle RAC Pods, each of which must meet at least the minimum requirements for Oracle Grid Infrastructure servers hosting an Oracle Real Application Clusters node.

The Oracle RAC Pods in this example configuration were created on the machines `worker-1` and `worker-2` for Oracle Real Application Clusters (Oracle RAC):

- Oracle RAC Node 1
  - Worker node: worker-1
  - Pod: `racnode1-0`
- Oracle RAC Node 2
  - Worker node: `worker-2`
  - Pod: `racnode2-0`

Each worker node has the following configuration:

- RAM: 60GB
- Operating system disk:
  You can use any supported storage options for Oracle Grid Infrastructure. Ensure that your storage has at least the following available space:
  - Root (/): 40 GB
  - `/scratch`: 80 GB (the Worker node directory, which will be used for `/u01` to store Oracle Grid Infrastructure and Oracle Database homes)
  - `/var/lib/containers`: 100 GB `xfs`
  - `/scratch/software/stage/`:(the stage directory for Oracle software)
- Oracle Linux 9.3 with Unbreakable Enterprise Kernel Release UEKR7 (Kernel Release `5.15.0-208.159.3.2.el9uek.x86_64`) or later
- Oracle recommends that you provide redundant private networks for the Oracle RAC cluster. Each worker node has two interfaces for this purpose. In this document example, we use `ens5` and `ens6` for the private networks. Each private network interface is connected to a separate and isolated network. If the interface and the underlying network support Jumbo Frame, then Oracle recommends that you configure the `ens5` and `ens6` interfaces with Jumbo Frame MTU 9000 in the worker node, without configuring any IP addresses on these interfaces. In this case, Multus will expose two network cards inside

Copyright © 2022, 2025, Oracle and/or its affiliates.

the pod as `eth1` and `eth2`, based on the `ens5` and `ens6` network interfaces configured in the worker node.

Network Cards:

`ens3`: Default network interface. It will be used by Flannel, and inside the pod, it will be used for the cluster public network.

`ens5`: First private network interface. It will be used by Multus for the cluster first private network.

`ens6`: Second private network interface. It will be used by Multus for the cluster second private network.

Block devices, shared by both Worker nodes

You can use any supported storage options for Oracle Grid Infrastructure. Ensure that your storage has at least the following available space:

- `/dev/sdd` (50 GB)
- `/dev/sde` (50 GB)

Related Topics

- Supported Storage Options for Oracle Grid Infrastructure

Configure Multus

To create an Oracle RAC private network, configure Multus, using either the example configuration file provided here, or your own configuration file.

Oracle RAC requires multiple network cards on different subnets. You obtain this by using Multus. The Multus module was installed without the optional argument for a configuration file to give you the opportunity to configure it. Configure Multus using the following configuration example, or write a configuration file to suit your own requirements based on your environment. Refer to "Oracle Cloud Native Environment Concepts" to understand the different supported modules. Also refer to "Oracle Cloud Native Environment Container Orchestration for Release 1.9" for step-by-step procedures to create, validate and deploy the Multus module.

To configure the Multus module in the Kubernetes cluster, create a configuration file for your private networks with your choice of parameter values that meet your requirements. Oracle supports using Multus config types `macvlan` and `ipvlan`.

1. This Multus file, `multus-rac-conf.yaml`, has been tested on our systems. You can use this file, and modify it based on your environment. We are using the `multus-rac` namesapace. However, you can use a different namespace if required.

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: macvlan-conf1
  namespace: multus-rac
spec:
  config: '{
      "cniVersion": "0.3.0",
      "type": "macvlan",
      "master": "ens5",
      "mode": "bridge",
      "ipam": {
        "type": "host-local",
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
        "subnet": "10.0.11.0/24",
        "rangeStart": "10.0.11.140",
        "rangeEnd": "10.0.11.150"
      }
    }'
---
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: macvlan-conf2
  namespace: multus-rac
spec:
  config: '{
      "cniVersion": "0.3.0",
      "type": "macvlan",
      "master": "ens6",
      "mode": "bridge",
      "ipam": {
        "type": "host-local",
        "subnet": "10.0.12.0/24",
        "rangeStart": "10.0.12.140",
        "rangeEnd": "10.0.12.150"
      }
    }'
```

> **Note**
>
> If you want to use Multus with `ipvlan`, then set types to `"ipvlan"` and mode to `"l2"`.

2. After you create your Multus configuration file, apply it to the system. For example, to apply `multus-rac-conf.yaml`, enter the following command:

   ```
   kubectl apply -f multus-rac-conf.yaml
   ```

3. Check the Multus configuration on OCNE cluster using `kubectl get all -n kube-system -l app=multus`. For example:

   ```
   # kubectl get all -n kube-system -l app=multus
   NAME READY STATUS RESTARTS AGE
   pod/kube-multus-ds-68k86 1/1 Running 0 90d
   pod/kube-multus-ds-lkjzf 1/1 Running 0 90d
   pod/kube-multus-ds-rstbf 1/1 Running 0 90d

   NAME DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE
   daemonset.apps/kube-multus-ds 3 3 3 3 3 <none> 90d
   ```

   You must confirm that all Multus pods are up and running.

4. Check the network attach definitions. For example:

   ```
   # Network attachment definition Check
   kubectl get Network-Attachment-Definition -n multus-rac
   NAME            AGE
   ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
macvlan-conf1   70m
macvlan-conf2   70m

# Check the resource macvlan-conf1 by describing the
kubectl describe Network-Attachment-Definition macvlan-conf1 -n multus-rac
Name:         macvlan-conf1
Namespace:    multus-rac
Labels:       <none>
Annotations:  <none>
API Version:  k8s.cni.cncf.io/v1
Kind:         NetworkAttachmentDefinition
Metadata:
  Creation Timestamp:  2023-08-22T01:03:02Z
  Generation:          1
  Managed Fields:
    API Version:  k8s.cni.cncf.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:metadata:
        f:annotations:
          .:
          f:kubectl.kubernetes.io/last-applied-configuration:
      f:spec:
        .:
        f:config:
    Manager:         kubectl-client-side-apply
    Operation:       Update
    Time:            2023-08-22T01:03:02Z
  Resource Version:  70063
  UID:               d092f33e-fdd2-4640-bc07-b22c18dd7e35
Spec:
  Config:  { "cniVersion": "0.3.0", "type": "macvlan", "control": "ens5",
"mode": "bridge", "ipam": { "type": "host-local", "subnet":
"10.0.11.0/24", "rangeStart": "10.0.11.140", "rangeEnd": "10.0.11.150" } }
Events:    <none>

# Check the resource macvlan-conf2 by describing the
kubectl describe Network-Attachment-Definition macvlan-conf2 -n multus-rac
Name:         macvlan-conf2
Namespace:    multus-rac
Labels:       <none>
Annotations:  <none>
API Version:  k8s.cni.cncf.io/v1
Kind:         NetworkAttachmentDefinition
Metadata:
  Creation Timestamp:  2023-08-22T01:03:02Z
  Generation:          1
  Managed Fields:
    API Version:  k8s.cni.cncf.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:metadata:
        f:annotations:
          .:
          f:kubectl.kubernetes.io/last-applied-configuration:
      f:spec:
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
        .:
        f:config:
    Manager:         kubectl-client-side-apply
    Operation:       Update
    Time:            2023-08-22T01:03:02Z
  Resource Version:  70064
  UID:               6dd1b858-a98c-44ac-81bd-aa72e89c3361
Spec:
  Config:  { "cniVersion": "0.3.0", "type": "macvlan", "control": "ens6",
"mode": "bridge", "ipam": { "type": "host-local", "subnet":
"10.0.12.0/24", "rangeStart": "10.0.12.140", "rangeEnd": "10.0.12.150" } }
Events:    <none>
```

Related Topics

- Oracle Native Cloud Environment

Oracle RAC StatefulSets

Learn about the configuration used in this document, so that you can understand the procedures we follow.

We are using StatefulSets to deploy Oracle RAC on OCNE. StatefulSets represent a set of Pods with unique, persistent identities, and stable host names that OCNE maintains regardless of where they are scheduled. The state information and other resilient data for any given StatefulSet Pod is maintained in persistent volumes associated with each Pod in the StatefulSet. The Statefulset will have a single Pod, which will also have a single container that will be used as the Oracle RAC node.

StatefulSets use an ordinal index for the identity and ordering of their Pods. By default, StatefulSet Pods are deployed in sequential order and are terminated in reverse ordinal order. For example, a StatefulSet named `racnode1` has its Pod named `racnode1-0`. In this guide, we will have two StatefulSets for Oracle RAC: `racnode1` and `racnode2`.

- Headless Service for OCNE
  A **headless service** is a service with a service IP. However, in our use case, its purpose is to obtain the IPs of the associated Pods from CoreDNS, which is the default DNS server for the Kubernetes cluster.
- Oracle RAC Pods
  The Oracle RAC cluster configuration in this document uses a two-node configuration as an example.

Headless Service for OCNE

A **headless service** is a service with a service IP. However, in our use case, its purpose is to obtain the IPs of the associated Pods from CoreDNS, which is the default DNS server for the Kubernetes cluster.

In the headless services created for the Oracle RAC node, SCAN and node VIP host names, the property `.spec.clusterIP` is set to `None` so that CoreDNS service will resolve these host names into the IPs of the corresponding Pods. The client that is running in the same Kubernetes cluster can connect to the database service by using the SCAN host name. The clients from outside of the Kubernetes cluster need additional services to expose the Oracle RAC cluster end points on the external networks.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Oracle RAC Pods

The Oracle RAC cluster configuration in this document uses a two-node configuration as an example.

The configuration information that follows is a reference for that configuration..

Oracle RAC Node 1

- Statefulset: `racnode1`
- Pod: `racnode1-0`
- CPU count: 4
- Memory
  - RAM memory: 16 GB
- Disk storage for Oracle Grid and Oracle RAC software: Exposed as a host directory inside the Pod (`/u01`) from the Worker nodes (`/scratch/`)
- Oracle Automatic Storage Management (Oracle ASM) Disks are exposed to the Pod as a persistent volume claim.
  - `/dev/asm-disk1`
  - `/dev/asm-disk2`
- Host names (IP addresses are allocated by the flannel network plug-in to the Pods.)
  - `racnode1-0`
  - `racnode1-0-vip`
  - `racnode-scan`
  - Domain: `rac.svc.cluster.local`
  - Namespace: Kubernetes namespace called `rac`

  The following Kubernetes services need to be created:

  ```
  racnode1-0
  racnode1-0-vip
  racnode-scan
  ```

  The endpoint of these services will be pointing to Oracle RAC statefulset pods.
- The `racnode1-0` Pod volumes are mounted using the Worker node host directory path, and appropriate permissions.

  Each of the following volumes are mounted:
  - `/stage`: read-write (`/scratch/software/stage`)
  - `/u01`: read-write (`/scratch/rac/cluster01/node1`)
  - `/mnt/.ssh`: read-only (the Kubernetes Secret location)

  Note that `/mnt/.ssh` must be `read-only` inside the container.
- Oracle Database configuration
  - Release 19.28
  - CDB name: `orclcdb`

Copyright © 2022, 2025, Oracle and/or its affiliates.

  - PDB name: `orclpdb`
  - Instance: `orclcdb1`
  - SGA size: 3 GB
  - PGA size: 2 GB

Oracle RAC Node 2

- Statefulset: `racnode2`
- Pod: `racnode2-0`
- CPU count: 4
- Memory
  - RAM memory: 16 GB
- Disk storage for Oracle Grid and Oracle RAC software is exposed as a host directory inside the Pod (`/u01`) from the Worker nodes (`/scratch/`)
- Oracle Automatic Storage Management (Oracle ASM) Disks
  - `/dev/asm/disk1`
  - `/dev/asm-disk2`
- Host names (IPs will be allocated by the flannel network plug-in to the PODs.)
  - `racnode2-0`
  - `racnode2-0-vip`
  - `racnode-scan`
  - Domain: `rac.svc.cluster.local`
  - Namespace: Kubernetes namespace called `rac`

  The following Kubernetes services need to be created:

  ```
  racnode2-0
  racnode2-0-vip
  racnode-scan
  ```

  The endpoint of these services will point to Oracle RAC StatefulSet pods.
- The `racnode2-0` pod volumes are mounted using the Worker node host directory path, and appropriate permissions.

  Each of the following volumes are created:
  - `/u01` read-write (`/scratch/rac/cluster01/node2`)
  - `/mnt/.ssh` read-only (the Kubernetes Secret location)

  Note that `/mnt/.ssh` must be `readonly` inside the container.
- Oracle Database
  - Instance: `orclcdb2`

Copyright © 2022, 2025, Oracle and/or its affiliates.

Set Clock Source on the Worker Node

Oracle recommends that you set the clock source to `tsc` for better performance in virtual environments (VM) on Linux x86-64.

The worker node containers inherit the clock source of their Linux host. For better performance, and to provide the clock source expected for the Oracle Real Application Clusters (Oracle RAC) database installation, Oracle recommends that you change the clock source setting to TSC.

1. As the `root` user, check if the `tsc` clock source is available on your system.

   ```
   # cat /sys/devices/system/clocksource/clocksource0/available_clocksource
   kvm-clock tsc acpi_pm
   ```

2. If the `tsc` clock source is available, then set `tsc` as the current clock source.

   ```
   # echo "tsc">/sys/devices/system/clocksource/clocksource0/
   current_clocksource
   ```

3. Verify that the current clock source is set to `tsc`.

   ```
   # cat /sys/devices/system/clocksource/clocksource0/current_clocksource
   tsc
   ```

4. Using any text editor, append the `clocksource` directive to the `GRUB_CMDLINE_LINUX` line in the `/etc/default/grub` file to retain this clock source setting even after a restart.

   ```
   GRUB_CMDLINE_LINUX="rd.lvm.lv=ol/root rd.lvm.lv=ol/swap rhgb quiet
   numa=off transparent_hugepage=never clocksource=tsc"
   ```

Setting up a Network Time Service

You must set up the `chronyd` time service for Oracle Real Application Clusters (Oracle RAC).

As a clustering environment, Oracle Cloud Native Environment (OCNE) requires that the system time is synchronized across each Kubernetes control plane and worker node within the cluster. Typically, this can be achieved by installing and configuring a Network Time Protocol (NTP) daemon on each node. Oracle recommends installing and setting up the `chronyd` daemon for this purpose.

Refer to the OCNE guide to set up the `chronyd` time service.

Configuring HugePages for Oracle RAC

HugePages is a feature integrated into the Linux kernel. For Oracle Database, using HugePages reduces the operating system maintenance of page states and increases Translation Lookaside Buffer (TLB) hit ratio.

To configure HugePages on the Linux operating system on OCNE, complete the following steps on the worker nodes:

1. In the `/etc/sysctl.conf` file, add the following entry:

   ```
   vm.nr_hugepages=16384
   ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

2. As root, run the following commands:

```
sysctl -p
sysctl -a
```

3. Run the following command to display the value of the `Hugepagesize` variable:

```
$ grep Hugepagesize /proc/meminfo
```

4. To check the configuration, as root run `grep HugePages_Total` and `grep Hugepagesize` on `/proc/meminfo`.

   For example:

```
# grep HugePages_Total /proc/meminfo

HugePages_Total:   16384

# grep Hugepagesize: /proc/meminfo

Hugepagesize:       2048 kB
```

5. Run the following command to check the available hugepages:

```
$ grep HugePages_Total /proc/meminfo

HugePages_Total: 16384
HugePages_Free: 16384
HugePages_Rsvd: 0
HugePages_Surp: 0
```

Restrictions for HugePages and Transparent HugePages Configurations

Review the HugePages and Transparent HugePages guidelines discussed in this section.

Transparent HugePages memory is enabled by default with Oracle Linux. Oracle continues to recommend using standard HugePages for Linux. However, for optimal performance, Oracle recommends that you set Transparent HugePages to `madvise` on all Oracle Database servers running UEK7 and later kernels. On Oracle Database servers running UEK release earlier than UEK7, Oracle recommends that you disable Transparent HugePages before you start the installation.

Transparent Hugepages are similar to standard HugePages. However, while standard HugePages allocate memory at startup, Transparent HugePages memory uses the `khugepaged` thread in the kernel to allocate memory dynamically during runtime, using swappable HugePages.

Oracle recommends that you review and modify the Transparent HugePages setting, because Transparent HugePages can cause delays in accessing memory that can result in node restarts in Oracle RAC environments, or performance issues or delays for Oracle Database single instances. For this reason, Oracle recommends that you explicitly use an `madvise` system call to inform the kernel about the memory requirements for the database, and the allocation of HugePages memory to optimize the performance of the database.

Copyright © 2022, 2025, Oracle and/or its affiliates.

To check to see if Transparent HugePages are enabled, enter the following command:

```
# cat /sys/kernel/mm/transparent_hugepage/enabled
always madvise [never]
```

Your current setting is returned in brackets. In this example, it is `[never]`.

If enabled, then set `madvise` for Transparent HugePages using the following commands:

1. Edit the file `/etc/default/grub` to add the entry `transparent_hugepage=madvise` at the end of the following line.

   ```
   GRUB_CMDLINE_LINUX="crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M
   LANG=en_US.UTF-8 console=tty0 console=ttyS0,115200 rd.luks=0 rd.md=0 rd.dm=0
   rd.lvm.vg=ocivolume rd.lvm.lv=ocivolume/root rd.net.timeout.dhcp=10
   rd.net.timeout.carrier=5
   netroot=iscsi:169.254.0.2:::1:iqn.2015-02.oracle.boot:uefi
   rd.iscsi.param=node.session.timeo.replacement_timeout=6000 net.ifnames=1
   nvme_core.shutdown_timeout=10 ipmi_si.tryacpi=0 ipmi_si.trydmi=0
   libiscsi.debug_libiscsi_eh=1 loglevel=4 crash_kexec_post_notifiers
   amd_iommu=on transparent_hugepage=madvise"
   ```

2. Generate the Grub configuration file:

   ```
   # grub2-mkconfig -o /boot/grub2/grub.cfg --update-bls-cmdline

   Generating grub configuration file ...
   Adding boot menu entry for UEFI Firmware Settings ...
   done
   ```

3. Write the `madvise` update to the `/sys/kernel/mm/transparent_hugepage/enabled` file and then confirm the change by reading the file back.

   ```
   # echo madvise > /sys/kernel/mm/transparent_hugepage/enabled
   # cat /sys/kernel/mm/transparent_hugepage/enabled
   always [madvise] never
   ```

Related Topics

- Setting Transparent HugePages to madvise

Oracle RAC Pod Node Preparation

Before you can install Oracle Grid Infrastructure and Oracle Real Application Clusters, you must configure the Oracle RAC Pods. .

- Prepare the Worker Node for Oracle RAC Deployment
  Before you install Oracle RAC inside the OCNE Pods, you must update the system configuration.
- Set up SELinux Module on Worker Nodes
  To enable command access, create an SELinux policy package on the Worker nodes.
- Using CVU to Validate Readiness for Worker Node
  Oracle recommends that you use Cluster Verification Utility (CVU) for Container hosts on your Worker Node to help to ensure that the Worker Node is configured correctly.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Create the Oracle RAC Container Image
  To create the image, set your Oracle Linux environment as required, create a Containerfile, and then build the image and make it available on the other Worker node.

Prepare the Worker Node for Oracle RAC Deployment

Before you install Oracle RAC inside the OCNE Pods, you must update the system configuration.

1. Log in as `root`
2. Use the `vim` editor to update `/etc/sysctl.conf` parameters to the following values:

   ```
   fs.file-max = 6815744
   net.core.rmem_default = 262144
   net.core.rmem_max = 4194304
   net.core.wmem_default = 262144
   net.core.wmem_max = 1048576
   fs.aio-max-nr = 1048576
   ```

3. Run the following commands:

   ```
   # sysctl -a
   # sysctl -p
   ```

4. Disable the swap memory

   ```
   [root@worker-1~]# free -m
                 total        used        free      shared  buff/cache
   available
   Mem:          63792        1487       43372         503       18931
   61078
   Swap:             0           0           0
   ```

   Set the swap memory off on the Worker node and control plane node, as described in the OCNE documentation. Swap must be disabled, because Kubernetes doesn't allow kubelet to come up if swap is enabled. PODs should be pinned with CPU/memory limits.

5. We need to enable kernel parameters at the Kubelet level, so that kernel parameters can be set at the Pod level. This is a one-time activity.

   In the `/etc/sysconfig/kubelet` file, add or replace `--allowed-unsafe-sysctls` under the `KUBELET_EXTRA_ARGS` environment variable as shown below:

   ```
   KUBELET_EXTRA_ARGS="--fail-swap-on=false --allowed-unsafe-
   sysctls='kernel.shm*,net.*,kernel.sem'"
   ```

   Restart Kubelet:

   ```
   # systemctl restart kubelet
   ```

   Check the Kubelet status:

   ```
   # systemctl check kubelet
   ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

6. Create the Oracle RAC and Oracle Grid Infrastructure mount points

   On node 1:

   ```
   # mkdir -p /scratch/rac/cluster01/node1
   # mkdir -p /scratch/software/stage
   ```

   On node 2:

   ```
   # mkdir -p /scratch/rac/cluster01/node2
   ```

Set up SELinux Module on Worker Nodes

To enable command access, create an SELinux policy package on the Worker nodes.

Traditional Unix security uses discretionary access control (DAC). SELinux is an example of mandatory access control. SELinux restricts many commands from the RAC Pods that are not allowed to run, which results in permission denied errors. To avoid such errors, you must create an SELinux policy package to allow certain commands.

Log in as `root`, and complete the following steps:

1. Verify that SELinux is enabled on the Worker nodes. For example:

   ```
   [root@worker-1]# getenforce
   Enforcing
   ```

2. Install the SELinux `devel` package on the Worker nodes:

   ```
   [root@worker-1]# dnf install selinux-policy-devel
   ```

3. Create a file `rac-ocne.te` under `/var/opt` on the Worker nodes:

   ```
   [root@worker-1]#
   module rac-ocne 1.0;

   require {
           type kernel_t;
           class system syslog_read;
           type container_runtime_t;
           type container_init_t;
           class file getattr;
           type container_file_t;
           type lib_t;
           type textrel_shlib_t;
           type bin_t;
           class file { execmod execute map setattr };
   }

   #============= container_init_t ==============
   allow container_init_t container_runtime_t:file getattr;
   allow container_init_t bin_t:file map;
   allow container_init_t bin_t:file execute;
   allow container_init_t container_file_t:file execmod;
   allow container_init_t lib_t:file execmod;
   ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
allow container_init_t textrel_shlib_t:file setattr;
allow container_init_t kernel_t:system syslog_read;
```

4. Create a policy package

```
[root@worker-1]# cd /var/opt
make -f /usr/share/selinux/devel/Makefile rac-ocne.pp
semodule -i rac-ocne.pp
semodule -l | grep rac-ocne
```

5. Repeat steps 1 through 4 on worker node 2.
6. Configure the worker node directory file context for the SELinux module:

```
worker-1 # semanage fcontext -a -t container_file_t /scratch/rac/cluster01/
node1
worker-1 # sudo restorecon -vF /scratch/rac/cluster01/node1
worker-1 # semanage fcontext -a -t container_file_t  /scratch/software/
stage/
worker-1 # sudo restorecon -vF /scratch/software/stage/
worker-2 # semanage fcontext -a -t container_file_t /scratch/rac/cluster01/
node2
worker-2 # sudo restorecon -vF /scratch/rac/cluster01/node2
```

Using CVU to Validate Readiness for Worker Node

Oracle recommends that you use Cluster Verification Utility (CVU) for Container hosts on your Worker Node to help to ensure that the Worker Node is configured correctly.

You can use CVU to assist you with system checks in preparation for creating a container for Oracle Real Application Clusters (Oracle RAC), and installing Oracle RAC inside the containers. CVU runs the appropriate system checks automatically, and prompts you to fix problems that it detects. To obtain the benefits of system checks, you must run the utility on all the Worker Nodes that you want to configure to host the Oracle RAC containers.

Related Topics

- Use the latest Cluster Verification Utility (CVU) (Doc ID 2731675.1)

Create the Oracle RAC Container Image

To create the image, set your Oracle Linux environment as required, create a Containerfile, and then build the image and make it available on the other Worker node.

1. Log in as the `root` user on the first Worker node, and move to the directory for image creation that you have previously prepared:

```
# cd /scratch/image
```

2. Prepare the container setup script.

   To maintain your configuration between OCNE RAC Pod restarts, use this procedure to build a script that you can configure to run on each container to set up the environment after restarts. In this example, the script name is `setupContainerEnv.sh`:

```
setupContainerEnv.sh
====
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
# cat setupContainerEnv.sh
#!/bin/bash
chown grid:asmadmin /dev/asm-disk1
chmod 660 /dev/asm-disk1
chown grid:asmadmin /dev/asm-disk2
chmod 660 /dev/asm-disk2
systemctl disable dnf-makecache.service
systemctl stop dnf-makecache.service
systemctl reset-failed
```

3. To set up the Oracle Real Application Clusters (Oracle RAC) container images, you must use a base Oracle Linux 9 image. In the following example, we create a file called `Containerfile`.

   `Containerfile`:

```
# Pull base image
# ---------------
FROM oraclelinux:9
# Environment variables required for this build (do NOT change)
# -------------------------------------------------------------
## Environment Variables
## ---
ENV SCRIPT_DIR=/opt/scripts/startup \
    SETUPCONTAINERENV="setupContainerEnv.sh"
### Copy Files
# ----
COPY $SETUPCONTAINERENV $SCRIPT_DIR/
### RUN Commands
# -----
#COPY $HOSTFILEENV $RESOLVCONFENV $SCRIPT_DIR/
RUN dnf install -y oracle-database-preinstall-19c systemd vim openssh-
server hostname policycoreutils-python-utils libxcrypt-compat && \
groupadd -g 54334 asmadmin && \
groupadd -g 54335 asmdba && \
groupadd -g 54336 asmoper && \
useradd -u 54332 -g oinstall -G oinstall,asmadmin,asmdba,asmoper,racdba
grid && \
usermod -g oinstall -G
oinstall,dba,oper,backupdba,dgdba,kmdba,asmdba,racdba oracle && \
cp /etc/security/limits.d/oracle-database-preinstall-19c.conf /etc/
security/limits.d/grid-database-preinstall-19c.conf && \
sed -i 's/oracle/grid/g' /etc/security/limits.d/grid-database-
preinstall-19c.conf && \
rm -f /etc/systemd/system/oracle-database-preinstall-19c-firstboot.service
&& \
sed -i 's/^TasksMax\S*/TasksMax=80%/g' /usr/lib/systemd/system/
user-.slice.d/10-defaults.conf && \
rm -f /usr/lib/systemd/system/dnf-makecache.service && \
echo "$SCRIPT_DIR/$SETUPCONTAINERENV" >> /etc/rc.local && \
setcap 'cap_net_admin,cap_net_raw+ep' /usr/bin/ping && \
rm -f /etc/sysctl.com && \
chmod +x $SCRIPT_DIR/$SETUPCONTAINERENV && \
chmod +x /etc/rc.d/rc.local && \
sync
USER root
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
WORKDIR /root
CMD ["/usr/sbin/init"]
# End of the containerfile
```

> **Note**
>
> To create the Oracle RAC Container image for Oracle Linux 8, the base container image is `oraclelinux:8`, and the `RUN dnf install` line is as follows:
> `RUN dnf install -y oracle-database-preinstall-19c systemd vim openssh-server hostname policycoreutils-python-utils`

4. Run the procedure for your use case:
   - Your server is behind a proxy firewall:
     - Run the following commands to set up your environment, where:
       - *`localhost-domain`* is the local host and domain for the internet gateway in your network
       - *`http-proxy`* is the HTTP proxy server for your network environment
       - *`https-proxy`* is the HTTPS proxy server for your network environment
       - `dbversion` (defined in this case with the `19.28` RU) is the Oracle Database release and RU that you are planning to install inside the container.
       - `olversion` (defined in this case as `ol9`, for Oracle Linux 9) is the Oracle Linux release.
       - `version` is defined as the values of variables `$dbversion$dbversion-$olversion`, which in this case `19.28-ol9-slim`

     ```
     # export NO_PROXY=localhost-domain
     # export http_proxy=http-proxy
     # export https_proxy=https-proxy
     # export dbversion=19.28
     # export olversion=ol9
     # export version=$dbversion-$olversion
     # podman build --force-rm=true --no-cache=true --build-arg \
     http_proxy=${http_proxy} --build-arg https_proxy=${https_proxy} \
     -t oracle/database-rac:$version-slim  -f Containerfile .
     ```

   - Your server is not behind a proxy firewall. With these environment variables set, you can now run the build command:

     ```
     # podman build --force-rm=true --no-cache=true -t oracle/database-
     rac:$version-slim -f Containerfile .
     ```

5. After the image builds successfully, you should see the 19c image created on your Worker node:

```
[root@worker-1 image]# podman images
REPOSITORY                                  TAG             IMAGE
ID      CREATED        SIZE
localhost/oracle/database-rac               19.28-ol9-slim
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
c0249037d5ad  8 minutes ago  472 MB
container-registry.oracle.com/os/oraclelinux  9
ad7ddf550f3d  4 weeks ago    246 MB
```

6. Save the image into a `tar` file, and transfer it to the other Worker node:

```
# podman image save -o /var/tmp/database-rac.tar localhost/oracle/database-
rac:19.28-ol9-slim
# scp -i ssh-key-for-host-worker-2 /var/tmp/database-rac.tar
worker-2:/var/tmp/database-rac.tar
```

7. On the other Worker node, load the image from the tar file and check that it is loaded:

```
# podman image load -i /var/tmp/database-rac.tar
# podman images
REPOSITORY                                    TAG            IMAGE ID
CREATED        SIZE
localhost/oracle/database-rac                 19.28-ol9-slim c0249037d5ad
13 minutes ago 472 MB
container-registry.oracle.com/os/oraclelinux  9              c0a711f33e08
9 days ago     246 MB
```

Download Oracle Grid Infrastructure and Oracle Database Software

Download the Oracle Database and Oracle Grid Infrastructure software from the Oracle Technology Network, and stage it.

The path `/scratch/software/stage` exists on worker node 1, where the Pod of Oracle RAC Node 1 will be deployed. This path is mounted inside the Pod under `/stage/software`.

Download the software to Worker node 1 using the following link, and stage it to the folder `/scratch/software/stage` so that in the Pod for Oracle RAC node 1, those files are are accessible under `/stage/software`.

https://www.oracle.com/database/technologies/

The software includes the following:

- Base 19.3.0 Release Software for Oracle Grid Infrastructure home and Oracle Database homes copied under `/scratch/software/stage` on the worker node 1
- The 19.28 GI RU file `p37957391_190000_Linux-x86-64.zip` copied to location `/scratch/software/stage` on the worker node 1
- The compatible OPatch software copied to location `/scratch/software/stage` on the worker node 1
- The Merge Patch 38336965 file `p38336965_19280000CWRU_Linux-x86-64.zip` copied to location `/scratch/software/stage` on the worker node 1
- The DBCA Patch 34436514 file `p34436514_1928000DBRU_Generic.zip` copied to location `/scratch/software/stage` on the worker node 1

Copyright © 2022, 2025, Oracle and/or its affiliates.

Create the Oracle RAC Pods

To create the Oracle Real Application Clusters (Oracle RAC) StatefulSets, log in to the host that has the `kubectl` setup, and complete configuration steps.

In Kubernetes, namespaces provides a mechanism for isolating groups of resources within a single cluster. Names of resources need to be unique within a namespace, but not across namespaces. Namespace-based scoping is applicable only for namespaced objects (for example, StatefulSets, Services, and so on), but not for clusterwide objects (for example, StorageClass, Nodes, PersistentVolumes, and so on).

In this procedure, we will create a namespace called `rac` to isolate the Oracle RAC Pods.

1. Create the Kubernetes namespace `rac`.

```
$ kubectl create ns rac

$ kubectl get ns
NAME                           STATUS   AGE
default                        Active   2h
externalip-validation-system   Active   2h
kube-node-lease                Active   2h
kube-public                    Active   2h
multus-rac                     Active   2h
kubernetes-dashboard           Active   2h
rac                            Active   2h
```

2. For this test environment, OCNE is set to `OCNE 1.9`, and Kubernetes is version `v1.29.3+3.el9`. From the operator node, check the package version for OCNE:

```
$ rpm -qa | grep olcne
oracle-olcne-release-el9-1.0-4.el9.x86_64
olcne-selinux-1.0.0-9.el9.x86_64
olcne-agent-1.9.0-4.el9.x86_64
olcne-utils-1.9.0-4.el9.x86_64
olcne-nginx-1.9.0-4.el9.x86_64
olcne-multus-chart-1.9.0-4.el9.x86_64


$ rpm -qa | grep olcne
oracle-olcne-release-el9-1.0-3.el9.x86_64
olcne-selinux-1.0.0-9.el9.x86_64
olcne-agent-1.8.0-2.el9.x86_64
olcne-utils-1.8.0-2.el9.x86_64
olcne-nginx-1.8.0-2.el9.x86_64
olcne-multus-chart-1.8.0-2.el9.x86_64
```

   From the host with `kubectl` setup, check the Kubernetes and worker node environment of the Kubernetes cluster:

```
$ kubectl get nodes -o wide

NAME       STATUS ROLES         AGE VERSION       INTERNAL-IP EXTERNAL-IP
OS-IMAGE            KERNEL-VERSION                    CONTAINER-RUNTIME
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
control-1  Ready  control-plane 2h  v1.29.3+3.el9 10.0.1.170  <none>
Oracle Linux Server 9.4 5.15.0-207.156.6.el9uek.x86_64 cri-o://1.29.1
control-2  Ready  control-plane 2h  v1.29.3+3.el9 10.0.1.8    <none>
Oracle Linux Server 9.4 5.15.0-207.156.6.el9uek.x86_64 cri-o://1.29.1
worker-1   Ready  <none>        2h  v1.29.3+3.el9 10.0.1.60   <none>
Oracle Linux Server 9.4 5.15.0-207.156.6.el9uek.x86_64 cri-o://1.29.1
worker-2   Ready  <none>        2h  v1.29.3+3.el9 10.0.1.124  <none>
Oracle Linux Server 9.4 5.15.0-207.156.6.el9uek.x86_64 cri-o://1.29.1
```

3. Create a Kubernetes Secret:

```
-- Generate an SSH key pair:
ssh-keygen -N "" -C "Key_Pair_for_ssh_connectivity_for_rac_setup"
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /tmp/id_rsa
Your identification has been saved in /tmp/id_rsa.
Your public key has been saved in /tmp/id_rsa.pub.
The key fingerprint is:
SHA256:jGEsDMUGAYczxn1BqJBWRtr3Na8m59DAMo0h9cnlIb8
Key_Pair_for_ssh_connectivity_for_rac_setup
The key's randomart image is:
+---[RSA 3072]----+
|o+OO++.. o       |
|*=+=+oo * .      |
|++.++o++ =       |
| .  oo*+. +      |
|     +.=SE .     |
|      o o .      |
|       o =       |
|        *        |
|         .       |
+----[SHA256]-----+

-- Use this key pair to create a Kubernetes secret named "ssh-key-secret":
kubectl create secret generic ssh-key-secret -n rac --from-file=/tmp/
id_rsa --from-file=/tmp/id_rsa.pub
Example:
[root@oper01-820919 tmp]# kubectl describe secret  ssh-key-secret -n rac

Name:         ssh-key-secret
Namespace:    rac
Labels:       <none>
Annotations:  <none>
Type:  Opaque
Data
====
id_rsa.pub:      565 bytes
id_rsa:  2590 bytes
```

A Secret is an object that contains a small amount of sensitive data, such as a password, a token, or a key. Secrets can be created independently of the Pods that use them. In this way, there is less risk of the Secret (and its data) being exposed during the workflow of creating, viewing, and editing Pods. Oracle RAC requires that SSH connectivity is set up between the Oracle RAC nodes for the Grid user (`grid`) and Oracle Database user (`oracle`). We will create an SSH key secret, and populate the SSH keys with the Secret

Copyright © 2022, 2025, Oracle and/or its affiliates.

inside the Oracle RAC Pod by mounting the Secret. Besides the method using Kubernetes secret, you can also use the traditional SSH equivalence setup as documented in *Oracle Grid Infrastructure Grid Infrastructure Installation and Upgrade Guide*.

4. From the host with `kubectl` set up, create the headless services by copying and pasting the `yaml` strings `racnode1-svc.yaml` and `racnode2-svc.yaml` into the directory `/root/yaml-rac` on the OCNE Operator node:

```
racnode1-svc.yaml


---
apiVersion: v1
kind: Service
metadata:
  labels:
   statefulset.kubernetes.io/pod-name: racnode1-0
  name: racnode1-0
  namespace: rac
spec:
  clusterIP: None
  selector:
   statefulset.kubernetes.io/pod-name: racnode1-0
  type: ClusterIP
  publishNotReadyAddresses: true
---
apiVersion: v1
kind: Service
metadata:
  labels:
   statefulset.kubernetes.io/pod-name: racnode1-0
  name: racnode1-0-vip
  namespace: rac
spec:
  clusterIP: None
  selector:
   statefulset.kubernetes.io/pod-name: racnode1-0
  type: ClusterIP
  publishNotReadyAddresses: true


racnode2-svc.yaml


---
apiVersion: v1
kind: Service
metadata:
  labels:
   statefulset.kubernetes.io/pod-name: racnode2-0
  name: racnode2-0
  namespace: rac
spec:
  clusterIP: None
  selector:
   statefulset.kubernetes.io/pod-name: racnode2-0
  type: ClusterIP
  publishNotReadyAddresses: true
---
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
apiVersion: v1
kind: Service
metadata:
  labels:
   statefulset.kubernetes.io/pod-name: racnode2-0
  name: racnode2-0-vip
  namespace: rac
spec:
  clusterIP: None
  selector:
   statefulset.kubernetes.io/pod-name: racnode2-0
  type: ClusterIP
  publishNotReadyAddresses: true
```

5. Create a service using the file `racnode-scan.yaml`:

```
---
apiVersion: v1
kind: Service
metadata:
  labels:
   cluster: racnode
  name: racnode-scan
  namespace: rac
spec:
  clusterIP: None
  selector:
   cluster: racnode
  type: ClusterIP
  publishNotReadyAddresses: true
```

6. Identify the Kubernetes node names using `kubectl get nodes --show-labels`. For example:

```
$ kubectl get nodes --show-labels

worker-1
worker-2
```

7. Create ASM persistent volume claims using `asm_pv.yaml`.

> **Note**
>
> The hostnames (in this example, `worker-1` and `worker-2`) should match with the statefult set volume.

```
---
kind: PersistentVolume
apiVersion: v1
metadata:
  name: block-asm-disk1-pv-vol1
  labels:
    asm_vol: "block-asm-pv-vol1"
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
  namespace: rac
spec:
  volumeMode: Block
  capacity:
    storage: 100Gi
  local:
    path: /dev/sdd
  accessModes:
    - ReadWriteMany
  nodeAffinity:
    required:
      nodeSelectorTerms:
        - matchExpressions:
            - key: kubernetes.io/hostname
              operator: In
              values:
                - worker-1
                - worker-2
---
kind: PersistentVolume
apiVersion: v1
metadata:
  name: block-asm-disk1-pv-vol2
  labels:
    asm_vol: "block-asm-pv-vol2"
  namespace: rac
spec:
  volumeMode: Block
  capacity:
    storage: 100Gi
  local:
    path: /dev/sde
  accessModes:
    - ReadWriteMany
  nodeAffinity:
    required:
      nodeSelectorTerms:
        - matchExpressions:
            - key: kubernetes.io/hostname
              operator: In
              values:
                - worker-1
                - worker-2
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: block-asm-pvc-vol1
  namespace: rac
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
  selector:
     matchLabels:
       asm_vol: "block-asm-pv-vol1"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: block-asm-pvc-vol2
  namespace: rac
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  selector:
     matchLabels:
       asm_vol: "block-asm-pv-vol2"
```

8. Use `kubectl apply` to apply the `yaml` file that you have created for ASM storage, and then check the persistent volume claims in the namespace with `kubectl get pvc -n rac`. For example:

```
$ kubectl apply -f asm_pv.yaml

$ kubectl get pvc -n rac

NAME                      STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS   AGE
block-asm-pvc-vol1        Bound    block-asm-disk1-pv-vol1
100Gi      RWX                           2h
block-asm-pvc-vol2        Bound    block-asm-disk1-pv-vol2
100Gi      RWX                           2h
```

9. Create an Oracle RAC StatefulSet for node one called `rac1_sts_sc.yaml`: This example uses standard memory and CPU resources. You can tune your memory parameters to your requirements.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  labels:
    cluster: racnode
    hostname: racnode1-0
  name: racnode1
  namespace: rac
spec:
  selector:
    matchLabels:
      cluster: racnode
  serviceName: racnode
  template:
    metadata:
      annotations:
        k8s.v1.cni.cncf.io/networks: '[
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
            {"name":"macvlan-conf1","namespace":"multus-rac",
"interface":"eth1",
              "ips":["10.0.11.141"]},
            {"name":"macvlan-conf2","namespace":"multus-rac",
"interface":"eth2",
               "ips":["10.0.12.141"]}
        ]'
      labels:
        cluster: racnode
        hostname: racnode1-0
    spec:
     securityContext:
       sysctls:
        - name: kernel.shmall
          value: "2097152"
        - name: kernel.sem
          value: "250 32000 100 128"
        - name: kernel.shmmax
          value: "8589934592"
        - name: kernel.shmmni
          value: "4096"
        - name: net.ipv4.conf.all.rp_filter
          value: "2"
     affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                - key: kubernetes.io/hostname
                  operator: In
                  values:
                    - worker-1
     volumes:
       - name: myshm
         emptyDir:
          medium: Memory
       - name: stagevol
         hostPath:
          path: /scratch/software/stage
       - name: swvol-racnode1
         hostPath:
          path: /scratch/rac/cluster01/node1
       - name: asmvol1
         persistentVolumeClaim:
          claimName: block-asm-pvc-vol1
       - name: asmvol2
         persistentVolumeClaim:
          claimName: block-asm-pvc-vol2
       - name: sshkeys
         secret:
          secretName: ssh-key-secret
     containers:
       - name: racnode1-0
         image: localhost/oracle/database-rac:19.28-ol9-slim
         imagePullPolicy: Always
         resources:
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
           requests:
            memory: "16Gi"
            cpu: "4"
           limits:
            hugepages-2Mi: "8Gi"
            memory: "16Gi"
            cpu: "4"
          securityContext:
           privileged: false
           capabilities:
            add: ["NET_ADMIN", "NET_RAW", "SYS_CHROOT", "SYS_NICE",
"SYS_RESOURCE", "AUDIT_WRITE", "AUDIT_CONTROL"]
          lifecycle:
           preStop:
            exec:
             command: ["/bin/bash","-c","kill -SIGRTMIN+3 1"]
          command: [/usr/sbin/init]
          volumeDevices:
            - name: asmvol1
              devicePath: /dev/asm-disk1
            - name: asmvol2
              devicePath: /dev/asm-disk2
          readinessProbe:
             tcpSocket:
               port: 1521
             initialDelaySeconds: 10
             periodSeconds: 30
             failureThreshold: 1
          volumeMounts:
          - mountPath: "/dev/shm"
            name: myshm
          - mountPath: /u01
            name: swvol-racnode1
          - mountPath: "/stage"
            name: stagevol
          - mountPath: /mnt/.ssh
            name: sshkeys
```

10. Create an Oracle RAC StatefulSet for node two called `rac2_sts_sc.yaml`, with parameter resource allocations identical to node one.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  labels:
    cluster: racnode
    hostname: racnode2-0
  name: racnode2
  namespace: rac
spec:
  selector:
    matchLabels:
      cluster: racnode
  serviceName: racnode
  template:
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
    metadata:
       annotations:
        k8s.v1.cni.cncf.io/networks: '[
            {"name":"macvlan-conf1","namespace":"multus-rac",
"interface":"eth1",
              "ips":["10.0.11.142"]},
            {"name":"macvlan-conf2","namespace":"multus-rac",
"interface":"eth2",
               "ips":["10.0.12.142"]}
        ]'
      labels:
        cluster: racnode
        hostname: racnode2-0
    spec:
     securityContext:
       sysctls:
        - name: kernel.shmall
          value: "2097152"
        - name: kernel.sem
          value: "250 32000 100 128"
        - name: kernel.shmmax
          value: "8589934592"
        - name: kernel.shmmni
          value: "4096"
        - name: net.ipv4.conf.all.rp_filter
          value: "2"
     affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                - key: kubernetes.io/hostname
                  operator: In
                  values:
                    - worker-2
     volumes:
       - name: myshm
         emptyDir:
          medium: Memory
       - name: swvol-racnode2
         hostPath:
          path: /scratch/rac/cluster01/node2
       - name: asmvol1
         persistentVolumeClaim:
          claimName: block-asm-pvc-vol1
       - name: asmvol2
         persistentVolumeClaim:
          claimName: block-asm-pvc-vol2
       - name: sshkeys
         secret:
          secretName: ssh-key-secret
     containers:
       - name: racnode2-0
         image: localhost/oracle/database-rac:19.28-ol9-slim
         imagePullPolicy: Always
         resources:
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
            requests:
             memory: "16Gi"
             cpu: "4"
            limits:
             hugepages-2Mi: "8Gi"
             memory: "16Gi"
             cpu: "4"
           securityContext:
            privileged: false
            capabilities:
             add: ["NET_ADMIN", "NET_RAW", "SYS_CHROOT", "SYS_NICE",
"SYS_RESOURCE", "AUDIT_WRITE", "AUDIT_CONTROL"]
           lifecycle:
            preStop:
             exec:
              command: ["/bin/bash","-c","kill -SIGRTMIN+3 1"]
           command: [/usr/sbin/init]
           volumeDevices:
             - name: asmvol1
               devicePath: /dev/asm-disk1
             - name: asmvol2
               devicePath: /dev/asm-disk2
           readinessProbe:
              tcpSocket:
                port: 1521
              initialDelaySeconds: 10
              periodSeconds: 30
              failureThreshold: 1
           volumeMounts:
           - mountPath: "/dev/shm"
             name: myshm
           - mountPath: /u01
             name: swvol-racnode2
           - mountPath: /mnt/.ssh
             name: sshkeys
```

11. Create the Kubernetes headless services using `kubectl apply` to apply the `yaml` files you have created, and then check the services and their endpoints in the namespace with `kubectl get svc,ep -n rac`:

```
$ kubectl apply -f racnode-scan.yaml

$ kubectl apply -f racnode1-svc.yaml

$ kubectl apply -f racnode2-svc.yaml

$ kubectl get svc,ep -n rac

NAME           TYPE      CLUSTER-IP EXTERNAL-IP PORT(S)  AGE
racnode-scan   ClusterIP None       <none>      <none>   2s
racnode1-0     ClusterIP None       <none>      <none>   2s
racnode1-0-vip ClusterIP None       <none>      <none>   2s
racnode2-0     ClusterIP None       <none>      <none>   2s
racnode2-0-vip ClusterIP None       <none>      <none>   2s
NAME                     ENDPOINTS                AGE
endpoints/kubernetes     100.96.243.41:6443       2s
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
endpoints/racnode-scan   10.254.1.10,10.254.1.9,10.254.2.7 + 1 more...
endpoints/racnode1-0     10.254.1.10             2s
endpoints/racnode1-0-vip 10.254.1.10             2s
endpoints/racnode2-0     10.254.2.8              2s
endpoints/racnode2-0-vip 10.254.2.8              2s
```

12. Apply the StatefulSet files for `racnode1` and `racnode2`:

```
$ kubectl apply -f rac1_sts_sc.yaml

$ kubectl apply -f rac2_sts_sc.yaml
```

13. Confirm that the StatefulSets are configured:

```
$ kubectl get all -n rac

NAME           READY  STATUS    RESTARTS  AGE
pod/racnode1-0 0/1    Running   0         16s
pod/racnode2-0 0/1    Running   0         15s

NAME                    TYPE        CLUSTER-IP  EXTERNAL-IP  PORT(S)    AGE
service/racnode-scan    ClusterIP   None        <none>       <none>     19s
service/racnode1-0      ClusterIP   None        <none>       <none>     23s
service/racnode1-0-vip  ClusterIP   None        <none>       <none>     23s
service/racnode2-0      ClusterIP   None        <none>       <none>     22s
service/racnode2-0-vip  ClusterIP   None        <none>       <none>     22s

NAME                           READY      AGE
statefulset.apps/racnode1      0/1        16s
statefulset.apps/racnode2      0/1        15s
```

You can also show the endpoints with the Pod IP addresses using `kubectl describe ep racnode-scan -n rac`. For example:

```
$ kubectl describe ep racnode-scan -n rac

Name:         racnode-scan
Namespace:    rac
Labels:       cluster=racnode
              scan_svc=racnode-scan
              service.kubernetes.io/headless=
Annotations:  <none>
Subsets:
  Addresses:          10.244.1.7,10.244.2.11
  NotReadyAddresses:  <none>

Events:  <none>

$ kubectl describe ep racnode1-0-vip  -n rac

Name:         racnode1-0-vip
Namespace:    rac
Labels:       service.kubernetes.io/headless=
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
              statefulset.kubernetes.io/pod-name=racnode1-0
Annotations:  endpoints.kubernetes.io/last-change-trigger-time:
2022-10-18T01:59:01Z
Subsets:
  Addresses:          10.244.1.7
  NotReadyAddresses:  <none>

Events:  <none>

$ kubectl describe ep racnode2-0-vip  -n rac

Name:         racnode2-0-vip
Namespace:    rac
Labels:       service.kubernetes.io/headless=
              statefulset.kubernetes.io/pod-name=racnode2-0
Annotations:  endpoints.kubernetes.io/last-change-trigger-time:
2022-10-18T02:02:31Z
Subsets:
  Addresses:          10.244.2.11
  NotReadyAddresses:  <none>

Events:  <none>
```

Deploy Oracle Grid Infrastructure in the Kubernetes Pods

To deploy Oracle Grid Infrastructure in the Kubernetes Pods, complete this procedure.

> **Note**
>
> To avoid losing configuration work after pod restarts, complete the steps in "Complete Pod Restart Configuration Tasks."

1. Log in to `racnode1-0 Pod`

   ```
   kubectl exec -i -t pod/racnode1-0 -n rac -- /bin/bash
   ```

2. As `root`, set up SSH for the Grid and Oracle users

   Grid user:

   ```
   mkdir -p /home/grid/.ssh
   chmod 700 /home/grid/.ssh
   cat /mnt/.ssh/id_rsa > /home/grid/.ssh/id_rsa
   cat /mnt/.ssh/id_rsa.pub > /home/grid/.ssh/id_rsa.pub
   chmod 400 /home/grid/.ssh/id_rsa
   cat /mnt/.ssh/id_rsa.pub > /home/grid/.ssh/authorized_keys
   chmod 600 /home/grid/.ssh/authorized_keys
   chown -R grid:oinstall /home/grid/.ssh
   ```

   Oracle user:

   ```
   mkdir -p /home/oracle/.ssh
   chmod 700 /home/oracle/.ssh
   ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
cat /mnt/.ssh/id_rsa > /home/oracle/.ssh/id_rsa
cat /mnt/.ssh/id_rsa.pub > /home/oracle/.ssh/id_rsa.pub
chmod 400 /home/oracle/.ssh/id_rsa
cat /mnt/.ssh/id_rsa.pub > /home/oracle/.ssh/authorized_keys
chmod 600 /home/oracle/.ssh/authorized_keys
chown -R oracle:install /home/oracle/.ssh
```

3. Repeat steps 1 and 2 on the `racnode2-0` Pod.
4. Log in to `racnode1-0` and populate the `known_host` file with the SSH keys for the Grid and Oracle users:

   Grid user:

   ```
   su - grid -c "ssh-keyscan -H racnode1-0 >> /home/grid/.ssh/known_hosts"
   su - grid -c "ssh-keyscan -H racnode2-0 >> /home/grid/.ssh/known_hosts"
   ```

   Oracle user:

   ```
   su - oracle -c "ssh-keyscan -H racnode1-0 >> /home/oracle/.ssh/known_hosts"
   su - oracle -c "ssh-keyscan -H racnode2-0 >> /home/oracle/.ssh/known_hosts"
   ```

   Repeat this step on `racnode2-0`.
5. On `racnode1-0`, verify SSH is functioning between `racnode1-0` Pod and `racnode2-0` Pods for both the grid and oracle users::

   ```
   su - grid -c "ssh -o BatchMode=yes -o ConnectTimeout=5 grid@racnode1-0
   echo ok 2>&1"
   su - grid -c "ssh -o BatchMode=yes -o ConnectTimeout=5 grid@racnode2-0
   echo ok 2>&1"
   su - oracle -c "ssh -o BatchMode=yes -o ConnectTimeout=5 oracle@racnode1-0
   echo ok 2>&1"
   su - oracle -c "ssh -o BatchMode=yes -o ConnectTimeout=5 oracle@racnode2-0
   echo ok 2>&1"
   ```

6. Repeat this command on the `racnode2-0` Pod to verify that SSH is functioning between `racnode1-0` Pod and `racnode2-0` Pods for both the `grid` and `oracle` users:

   ```
   su - grid -c "ssh -o BatchMode=yes -o ConnectTimeout=5 grid@racnode1-0
   echo ok 2>&1"
   su - grid -c "ssh -o BatchMode=yes -o ConnectTimeout=5 grid@racnode2-0
   echo ok 2>&1"
   su - oracle -c "ssh -o BatchMode=yes -o ConnectTimeout=5 oracle@racnode1-0
   echo ok 2>&1"
   su - oracle -c "ssh -o BatchMode=yes -o ConnectTimeout=5 oracle@racnode2-0
   echo ok 2>&1"
   ```

7. As `root`, create the required Oracle Grid Infrastructure and Oracle RAC directories on `racnode1-0` and `racnode2-0`

   `racnode 1-0`:

   ```
   mkdir -p /u01/app/19c/grid
   mkdir -p /u01/app/grid
   mkdir -p /u01/app/oraInventory
   ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
mkdir -p /u01/app/oracle
mkdir -p /u01/app/oracle/product/19c/dbhome_1
chown -R grid:oinstall /u01/app
chown -R oracle:oinstall /u01/app/oracle
chmod -R 775 /u01/


racnode2-0:

mkdir -p /u01/app/19c/grid
mkdir -p /u01/app/grid
mkdir -p /u01/app/oraInventory
mkdir -p /u01/app/oracle
mkdir -p /u01/app/oracle/product/19c/dbhome_1
chown -R grid:oinstall /u01/app
chown -R oracle:oinstall /u01/app/oracle
chmod -R 775 /u01/
```

8. Add the following variables to the `.bashrc` file of the `grid` user on both the nodes:

```
export TMPDIR=/var/tmp
export ORACLE_HOME=/u01/app/19c/grid
export PATH=$ORACLE_HOME/bin:$PATH
```

9. Add the following variables to the `.bashrc` file of the `oracle` user on both the nodes:

```
export TMPDIR=/var/tmp
export ORACLE_HOME=/u01/app/oracle/product/19c/dbhome_1
export PATH=$ORACLE_HOME/bin:$PATH
```

10. As the `grid` user on `racnode1-0`, unzip the Oracle Grid Infrastructure software:

```
unzip -q /stage/grid_home.zip -d /u01/app/19c/grid
```

11. As the `grid` user on racnode1-0, create a directory called `RU1928` and unzip the 19.28 GIRU Patch 37957391 Software into it:

```
mkdir -p /stage/RU1928
unzip -q /stage/p37957391_190000_Linux-x86-64.zip -d /stage/RU1928
```

12. As the `grid` user on racnode1-0, create a directory called `mlr38336965` and unzip the Merge Patch 38336965 software into it:

```
mkdir -p /stage/mlr38336965
unzip -q /stage/p38336965_1928000OCWRU_Linux-x86-64.zip -d /stage/
mlr38336965
```

13. As the `grid` user on racnode1-0, replace the OPatch software in the Oracle Grid Infrastructure home to use the version downloaed for RU 19.28:

```
mv /u01/app/19c/grid/OPatch /u01/app/19c/grid/OPatch.old
unzip -q /stage/p6880880_190000_Linux-x86-64.zip -d /u01/app/19c/grid
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

14. To avoid a Swap space-related error in the Oracle Grid Infrastructure 19.3.0 Base Release software, you must edit out a line in `oraparam.ini`.

    As the `grid` user, edit the file `/u01/app/19c/grid/oui/oraparam.ini` to comment out the following line:

    ```
    =======
    [Generic Prereqs]
    #SWAP_SPACE=500
    =======
    ```

15. Edit the file `$ORACLE_HOME/cv/admin/cvu_config` and add the following line:

    ```
    CV_DESTLOC=/var/tmp
    ```

16. Prepare a response file called `gi.rsp` for the Oracle Grid Infrastructure installation, and replace the variables *`your-password`* with the password for your software:

    > **Note**
    >
    > Graphic user interface installations are not supported on OCNE.

    ```
    oracle.install.responseFileVersion=/oracle/install/
    rspfmt_crsinstall_response_schema_v19.0.0
    oracle.install.option=CRS_CONFIG
    ORACLE_BASE=/u01/app/grid
    oracle.install.asm.OSDBA=asmdba
    oracle.install.asm.OSOPER=asmoper
    oracle.install.asm.OSASM=asmadmin
    oracle.install.crs.config.scanType=LOCAL_SCAN
    oracle.install.crs.config.SCANClientDataFile=
    oracle.install.crs.config.gpnp.scanName=racnode-scan
    oracle.install.crs.config.gpnp.scanPort=1521
    oracle.install.crs.config.ClusterConfiguration=STANDALONE
    oracle.install.crs.config.configureAsExtendedCluster=false
    oracle.install.crs.config.memberClusterManifestFile=/common_scripts/
    oracle.install.crs.config.clusterName=racnode-c
    oracle.install.crs.config.gpnp.configureGNS=false
    oracle.install.crs.config.autoConfigureClusterNodeVIP=false
    oracle.install.crs.config.gpnp.gnsOption=
    oracle.install.crs.config.gpnp.gnsClientDataFile=
    oracle.install.crs.config.gpnp.gnsSubDomain=
    oracle.install.crs.config.gpnp.gnsVIPAddress=
    oracle.install.crs.config.sites=
    oracle.install.crs.config.clusterNodes=racnode1-0:racnode1-0-
    vip:HUB,racnode2-0:racnode2-0-vip:HUB
    oracle.install.crs.config.networkInterfaceList=eth0:10.244.0.0:1,eth1:10.0.
    11.0:5,eth2:10.0.12.0:5
    oracle.install.crs.configureGIMR=false
    oracle.install.asm.configureGIMRDataDG=
    oracle.install.crs.config.storageOption=
    oracle.install.crs.config.sharedFileSystemStorage.votingDiskLocations=
    oracle.install.crs.config.sharedFileSystemStorage.ocrLocations=
    oracle.install.crs.config.useIPMI=
    ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
oracle.install.crs.config.ipmi.bmcUsername=
oracle.install.crs.config.ipmi.bmcPassword=
oracle.install.asm.SYSASMPassword=your-password
oracle.install.asm.diskGroup.name=DATA
oracle.install.asm.diskGroup.redundancy=EXTERNAL
oracle.install.asm.diskGroup.AUSize=4
oracle.install.asm.diskGroup.FailureGroups=
oracle.install.asm.diskGroup.disksWithFailureGroupNames=/dev/asm-
disk1,,/dev/asm-disk2,
oracle.install.asm.diskGroup.disks=/dev/asm-disk1,/dev/asm-disk2
oracle.install.asm.diskGroup.quorumFailureGroupNames=
oracle.install.asm.diskGroup.diskDiscoveryString=/dev/asm*
oracle.install.asm.monitorPassword=your-password
oracle.install.asm.gimrDG.name=
oracle.install.asm.gimrDG.redundancy=
oracle.install.asm.gimrDG.AUSize=1
oracle.install.asm.gimrDG.FailureGroups=
oracle.install.asm.gimrDG.disksWithFailureGroupNames=
oracle.install.asm.gimrDG.disks=
oracle.install.asm.gimrDG.quorumFailureGroupNames=
oracle.install.asm.configureAFD=false
oracle.install.crs.configureRHPS=false
oracle.install.crs.config.ignoreDownNodes=false
oracle.install.config.managementOption=NONE
oracle.install.config.omsHost=
oracle.install.config.omsPort=0
oracle.install.config.emAdminUser=
oracle.install.config.emAdminPassword=
oracle.install.crs.rootconfig.executeRootScript=false
oracle.install.crs.rootconfig.configMethod=ROOT
oracle.install.crs.rootconfig.sudoPath=
oracle.install.crs.rootconfig.sudoUserName=
oracle.install.crs.config.batchinfo=
oracle.install.crs.app.applicationAddress=
oracle.install.crs.deleteNode.nodes=
```

In this document, we will place the response file in the path `/mnt/gridrsp/gi.rsp`.

17. As the grid user, patch the unzipped Oracle Grid Infrastructure home with the 19.3.0 base release software, using the 19.28 RU and MLR patch, by running the response file installation using the following command::

```
/u01/app/19c/grid/gridSetup.sh -applyRU /stage/RU1928/37957391 -
applyOneOffs \
/stage/mlr38336965/38336965 -waitforcompletion -silent \
-responseFile /mnt/gridrsp/gi.rsp \
oracle.install.crs.config.netmaskList=eth0:255.255.0.0,eth1:255.255.255.0,e
th2:255.255.255.0
```

18. When prompted, run the `orainstRoot.sh` and `root.sh` scripts:

```
Run /u01/app/oraInventory/orainstRoot.sh on the following nodes:
[racnode1-0, racnode2-0]
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
Run /u01/app/19c/grid/root.sh on the following nodes:
[racnode1-0, racnode2-0]
```

19. As the grid installation user, run the following command on node 1 to complete the configuration:

```
su - grid -c "/u01/app/19c/grid/gridSetup.sh -executeConfigTools -
responseFile /mnt/gridrsp/gi.rsp -silent"
```

20. After the message displays indicating that the installation has completed successfully, check the cluster configuration with `crsctl stat res -t`. The results should be similar to the following output:

```
[grid@racnode1-0 ~]$ /u01/app/19c/grid/bin/crsctl stat res -t
--------------------------------------------------------------------------------
-----
Name           Target  State        Server                   State details
--------------------------------------------------------------------------------
-----
Local Resources
--------------------------------------------------------------------------------
-----
ora.LISTENER.lsnr
               ONLINE ONLINE        racnode1-0               STABLE
               ONLINE ONLINE        racnode2-0               STABLE
ora.chad
               ONLINE ONLINE        racnode1-0               STABLE
               ONLINE ONLINE        racnode2-0               STABLE
ora.net1.network
               ONLINE ONLINE        racnode1-0               STABLE
               ONLINE ONLINE        racnode2-0               STABLE
ora.ons
               ONLINE ONLINE        racnode1-0               STABLE
               ONLINE ONLINE        racnode2-0               STABLE
--------------------------------------------------------------------------------
-----
Cluster Resources
--------------------------------------------------------------------------------
-----
ora.ASMNET1LSNR_ASM.lsnr(ora.asmgroup)
      1        ONLINE  ONLINE       racnode1-0               STABLE
      2        ONLINE  ONLINE       racnode2-0               STABLE
ora.ASMNET2LSNR_ASM.lsnr(ora.asmgroup)
      1        ONLINE  ONLINE       racnode1-0               STABLE
      2        ONLINE  ONLINE       racnode2-0               STABLE
ora.DATA.dg(ora.asmgroup)
      1        ONLINE  ONLINE       racnode1-0               STABLE
      2        ONLINE  ONLINE       racnode2-0               STABLE
ora.LISTENER_SCAN1.lsnr
      1        ONLINE  ONLINE       racnode1-0               STABLE
ora.LISTENER_SCAN2.lsnr
      1        ONLINE  ONLINE       racnode2-0               STABLE
ora.asm(ora.asmgroup)
      1        ONLINE  ONLINE       racnode1-0               Started,STABLE
      2        ONLINE  ONLINE       racnode2-0               Started,STABLE
ora.asmnet1.asmnetwork(ora.asmgroup)
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
      1        ONLINE  ONLINE       racnode1-0               STABLE
      2        ONLINE  ONLINE       racnode2-0               STABLE
ora.asmnet2.asmnetwork(ora.asmgroup)
      1        ONLINE  ONLINE       racnode1-0               STABLE
      2        ONLINE  ONLINE       racnode2-0               STABLE
ora.cvu
      1        ONLINE ONLINE        racnode1-0               STABLE
ora.qosmserver
      1        ONLINE ONLINE        racnode1-0               STABLE
ora.racnode1-0.vip
      1        ONLINE ONLINE        racnode1-0               STABLE
ora.racnode2-0.vip
      1        ONLINE ONLINE        racnode2-0               STABLE
ora.scan1.vip
      1        ONLINE ONLINE        racnode1-0               STABLE
ora.scan2.vip
      1        ONLINE ONLINE        racnode2-0               STABLE
--------------------------------------------------------------------------------
[grid@racnode1-0 ~]$
```

21. Check the patches installed in the Oracle Grid Infrastructure home:

```
[grid@racnode1-0 ~]$ $ORACLE_HOME/OPatch/opatch lspatches

38336965;OCW Interim patch for 38336965
38124772;TOMCAT RELEASE UPDATE 19.0.0.0.0 (38124772)
37962938;ACFS RELEASE UPDATE 19.28.0.0.0 (37962938)
37960098;Database Release Update : 19.28.0.0.250715 (37960098)
36758186;DBWLM RELEASE UPDATE 19.0.0.0.0 (36758186)

OPatch succeeded.
```

Deploy Oracle RAC Database in the Pods

To deploy an Oracle RAC database in the OCNE Pods, complete this procedure.

1. Log in to `racnode1-0 Pod`

```
kubectl exec -i -t pod/racnode1-0 -n rac -- /bin/bash
```

2. Unzip the Oracle Database software as the Oracle (`oracle`) user:

```
unzip -q /stage/db_home.zip -d /u01/app/oracle/product/19c/dbhome_1
```

3. As the `oracle` user, create a directory called `dbcapatch_34436514` and unzip the DBCA Patch 34436514 software into it:

```
mkdir /stage/dbcapatch_34436514
unzip -q /stage/p34436514_1928000DBRU_Generic.zip -d /stage/
dbcapatch_34436514
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

4. As the `oracle` user, replace the OPatch software in the Oracle home:

```
mv /u01/app/oracle/product/19c/dbhome_1/OPatch /u01/app/oracle/product/19c/
dbhome_1/OPatch.old
unzip -q /stage/p6880880_190000_Linux-x86-64.zip -d /u01/app/oracle/
product/19c/dbhome_1
```

5. To avoid a Swap space-related error in the Oracle Database 19.3.0 Base Release software, you must edit out a line in oraparam.ini.

   As the `oracle` user, edit the file `/u01/app/oracle/product/19c/dbhome_1/oui/oraparam.ini` to comment out the following line:

```
=======
[Generic Prereqs]
#SWAP_SPACE=500
=======
```

6. As the `oracle` user, apply the 19.28 RU on top of the unzipped 19.3.0 RDBMS binaries along with the DBCA one-off patch by running the installation using the following command:

```
/u01/app/oracle/product/19c/dbhome_1/runInstaller -waitforcompletion -
silent -applyRU /stage/RU1928/37957391/ -applyOneOffs /stage/
dbcapatch_34436514/34436514 \
oracle.install.option=INSTALL_DB_SWONLY \
ORACLE_HOSTNAME=racnode1-0 \
UNIX_GROUP_NAME=oinstall \
oracle.install.db.CLUSTER_NODES=racnode1-0,racnode2-0 \
INVENTORY_LOCATION=/u01/app/oraInventory \
SELECTED_LANGUAGES=en \
ORACLE_HOME=/u01/app/oracle/product/19c/dbhome_1 \
ORACLE_BASE=/u01/app/oracle \
oracle.install.db.InstallEdition=EE \
oracle.install.db.OSDBA_GROUP=dba \
oracle.install.db.OSBACKUPDBA_GROUP=backupdba \
oracle.install.db.OSDGDBA_GROUP=dgdba \
oracle.install.db.OSKMDBA_GROUP=kmdba \
oracle.install.db.OSRACDBA_GROUP=racdba \
SECURITY_UPDATES_VIA_MYORACLESUPPORT=false \
DECLINE_SECURITY_UPDATES=true
```

7. When prompted, run the `root.sh` script as the root user on both `racnode1-0` and `racnode2-0` nodes.
8. As the Oracle user, check the Patches installed in the RDBMS HOME:

```
oracle@racnode1-0 ~]$ $ORACLE_HOME/OPatch/opatch lspatches
34436514;DBCA REPORTS INCORRECT MEMORY IN PODMAN CONTAINERS
37962946;OCW RELEASE UPDATE 19.28.0.0.0 (37962946)
37960098;Database Release Update : 19.28.0.0.250715 (37960098)

OPatch succeeded.
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

9. As the Oracle user, run the following `dbca` command to create the Oracle RAC database, replacing the variable *`your-password`* with the passwords for your software:

```
/u01/app/oracle/product/19c/dbhome_1/bin/dbca -silent -createDatabase \
-templateName General_Purpose.dbc \
-gdbname ORCLCDB \
-createAsContainerDatabase true \
-sysPassword your-password \
-systemPassword your-password \
-datafileDestination +DATA \
-storageType ASM \
-characterSet AL32UTF8 \
-redoLogFileSize 1024 \
-databaseType OLTP \
-databaseConfigType RAC \
-nodelist racnode1-0,racnode2-0 \
-useOMF true \
-numberOfPDBs 1 \
-pdbAdminPassword your-password \
-pdbName ORCLPDB \
-memoryPercentage 30
```

10. After the message displays indicating that the installation has completed successfully, check the database configuration `srvctl status database`. The results should be similar to the following:

```
[oracle@racnode1-0 ~]$ $ORACLE_HOME/bin/srvctl status database -d ORCLCDB -
verbose
Instance ORCLCDB1 is running on node racnode1-0. Instance status: Open.
Instance ORCLCDB2 is running on node racnode2-0. Instance status: Open.
```

11. Create a non-default service with the PDB for use by applications. You can create the service with the attributes as required by your applications. For example, the service has at least the notification attribute set to true for the application clients to receive notifications of changes in the DB service.

```
$ORACLE_HOME/bin/srvctl add service -d ORCLCDB -pdb ORCLPDB -service
ORCLPDBSVC -notification true -preferred ORCLCDB1,ORCLCDB2

$ORACLE_HOME/bin/srvctl start service -d ORCLCDB -service ORCLPDBSVC

$ORACLE_HOME/bin/srvctl status service -d ORCLCDB

Service orclpdbsvc is running on instance(s) ORCLCDB1,ORCLCDB2
```

Related Topics

- "Default and User-Defined Services" in *Oracle Multitenant Administrator's Guide*
- Setup Wizard Installation Options for Creating Images in *Oracle Database Installation Guide for Linux*
- Gold Image How To (Doc ID 2965269.1)

Copyright © 2022, 2025, Oracle and/or its affiliates.

Check Oracle RAC Network Configuration

To confirm IP addresses are functioning, check the public and private network configuration for RAC inside the pods using this procedure.

1. Log in as root to `racnode1-0`, and run `ip` to display the all the IPs on `racnode1-0`

```
[root@racnode1-0 ~]# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8950 qdisc noqueue
state UP group default
    link/ether 1e:36:ab:46:36:b9 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.244.1.7/24 brd 10.244.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::1c36:abff:fe46:36b9/64 scope link
       valid_lft forever preferred_lft forever
3: eth1@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default
    link/ether de:7b:25:59:5c:41 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.11.141/24 brd 10.0.11.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet 169.254.5.127/20 brd 169.254.15.255 scope global eth1:1
       valid_lft forever preferred_lft forever
    inet6 fe80::dc7b:25ff:fe59:5c41/64 scope link
       valid_lft forever preferred_lft forever
4: eth2@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default
    link/ether 16:63:1e:2a:28:c9 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.12.141/24 brd 10.0.12.255 scope global eth2
       valid_lft forever preferred_lft forever
    inet 169.254.30.184/20 brd 169.254.31.255 scope global eth2:1
       valid_lft forever preferred_lft forever
    inet6 fe80::1463:1eff:fe2a:28c9/64 scope link
       valid_lft forever preferred_lft forever
```

2. Log in as root to `racnode2-0`, and run `ip` to display the all the IPs on `racnode2-0`

```
[root@racnode2-0 ~]# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0@if20: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8950 qdisc noqueue
state UP group default
    link/ether 8a:22:ec:70:54:2f brd ff:ff:ff:ff:ff:ff link-netnsid 0
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
    inet 10.244.2.11/24 brd 10.244.2.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::8822:ecff:fe70:542f/64 scope link
       valid_lft forever preferred_lft forever
3: eth1@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default
    link/ether ca:a6:95:03:7a:00 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.11.142/24 brd 10.0.11.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet 169.254.6.38/20 brd 169.254.15.255 scope global eth1:1
       valid_lft forever preferred_lft forever
    inet6 fe80::c8a6:95ff:fe03:7a00/64 scope link
       valid_lft forever preferred_lft forever
4: eth2@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default
    link/ether 72:9a:a5:01:e4:fc brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.12.142/24 brd 10.0.12.255 scope global eth2
       valid_lft forever preferred_lft forever
    inet 169.254.23.15/20 brd 169.254.31.255 scope global eth2:1
       valid_lft forever preferred_lft forever
    inet6 fe80::709a:a5ff:fe01:e4fc/64 scope link
       valid_lft forever preferred_lft forever
```

3. As the `grid` user, run the Oracle Interface Configuration Tool (`OIFCFG`) command-line interface using `iflist` to list the network interfaces and addresses:

```
$ORACLE_HOME/bin/oifcfg iflist -n
eth0  10.244.2.0  255.255.255.0
eth1  10.0.11.0  255.255.255.0
eth1  169.254.0.0  255.255.240.0
eth2  10.0.12.0  255.255.255.0
eth2  169.254.16.0  255.255.240.0

$ORACLE_HOME/bin/oifcfg getif
eth0 10.244.0.0 global public
eth1 10.0.11.0 global cluster_interconnect,asm
eth2 10.0.12.0 global cluster_interconnect,asm
```

4. Check the ASM network configuration using `srvctl config asmnetwork`:

```
$ORACLE_HOME/bin/srvctl config asmnetwork
ASM network 1 exists
Subnet IPv4: 10.0.11.0/255.255.255.0/
Subnet IPv6:
Network is enabled
Network is individually enabled on nodes:
Network is individually disabled on nodes:
ASM network 2 exists
Subnet IPv4: 10.0.12.0/255.255.255.0/
Subnet IPv6:
Network is enabled
Network is individually enabled on nodes:
Network is individually disabled on nodes:
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

5. Confirm that the Highly Available IP (HAIP) resource is available using `crsctl stat res -init -t`:

```
$ORACLE_HOME/bin/crsctl stat res -init -t
--------------------------------------------------------------------------------
-----
Name           Target  State        Server                   State details
--------------------------------------------------------------------------------
-----
Cluster Resources
--------------------------------------------------------------------------------
-----
ora.asm
      1        ONLINE  ONLINE       racnode1-0               Started,STABLE
ora.cluster_interconnect.haip
      1        ONLINE  ONLINE       racnode1-0               STABLE
ora.crf
      1        ONLINE  ONLINE       racnode1-0               STABLE
ora.crsd
      1        ONLINE  ONLINE       racnode1-0               STABLE
ora.cssd
      1        ONLINE  ONLINE       racnode1-0               STABLE
ora.cssdmonitor
      1        ONLINE  ONLINE       racnode1-0               STABLE
ora.ctssd
      1        ONLINE  ONLINE       racnode1-0
OBSERVER,STABLE
ora.diskmon
      1        OFFLINE OFFLINE                               STABLE
ora.evmd
      1        ONLINE  ONLINE       racnode1-0               STABLE
ora.gipcd
      1        ONLINE  ONLINE       racnode1-0               STABLE
ora.gpnpd
      1        ONLINE  ONLINE       racnode1-0               STABLE
ora.mdnsd
      1        ONLINE  ONLINE       racnode1-0               STABLE
ora.storage
      1        ONLINE  ONLINE       racnode1-0               STABLE
--------------------------------------------------------------------------------
-----
```

How to Connect to the Database Using a Client

Learn what is required to connect to the database with a client either inside or outside the Kubernetes cluster.

- Requirements to Connect to the Database with a Client Inside the Kubernetes Cluster
  Inside the cluster, the client must be configured in accordance with these requirements.
- Requirements to Connect to the Database with a Client Outside the Kubernetes Cluster
  Outside of the cluster, the client must be configured in accordance with these requirements.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Connect to the Database with a Client
  Use this procedure to configure a cluster node listener with endpoints that will be exposed to external clients using the Kubernetes nodePort services.

Requirements to Connect to the Database with a Client Inside the Kubernetes Cluster

Inside the cluster, the client must be configured in accordance with these requirements.

Oracle RAC databases running on OCNE can be accessed by clients or applications that are provisioned inside the Kubernetes cluster.

To access Oracle RAC on OCNE inside the Kubernetes cluster, you can do the following:

1. Install the Oracle client on a Pod. You can do this either on the same Kubernetes namespace, or on a different namespace.
2. Use SQL*Plus to connect with an Oracle RAC database running on the OCNE pod.

   ```
   sqlplus sys@//racnode-scan.rac.svc.cluster.local:1521/ORCLCDB as sysdba
   ```

Other Pods in the same Kubernetes cluster can access the Oracle Database services by connecting to the SCAN, using the full SCAN FQDN host name. For example: `racnode-scan.rac.svc.cluster.local`. That host name will be resolved by the Kubernetes core DNS service.

Requirements to Connect to the Database with a Client Outside the Kubernetes Cluster

Outside of the cluster, the client must be configured in accordance with these requirements.

To access Oracle RAC on OCNE outside of the Kubernetes cluster, you need the following:

- An endpoint with which to communicate
- A set of credentials and coordinates, including a user name and password, and a database service

To integrate with an external application, you need the following configured:

- A Kubernetes Service object to represent the Oracle Database
- One or more Endpoints for the Kubernetes service

Connect to the Database with a Client

Use this procedure to configure a cluster node listener with endpoints that will be exposed to external clients using the Kubernetes nodePort services.

1. Run the following commands in the `racnode1-0` pod to create a listener: (the values in italics are variables; you can use your own values):

   ```
   kubectl exec -i -t pod/racnode1-0 -n rac -- /bin/bash
   su - grid -c "/u01/app/19c/grid/bin/srvctl add listener -listener db-lsnr -
   endpoints 30203,30204"
   su - grid -c "/u01/app/19c/grid/bin/srvctl start listener -listener db-
   lsnr"
   su - grid -c "/u01/app/19c/grid/bin/srvctl status listener -listener db-
   ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

lsnr"

As root on racnode1-0 (this is an example):

```
[root@racnode1-0 ~]# su - grid -c "/u01/app/19c/grid/bin/srvctl config
listener"
Name: DB-LSNR
Type: Database Listener
Network: 1, Owner: grid
Home: <CRS home>
End points: TCP:30203,30204
Listener is enabled.
Listener is individually enabled on nodes:
Listener is individually disabled on nodes:
Name: LISTENER
Type: Database Listener
Network: 1, Owner: grid
Home: <CRS home>
End points: TCP:1522
Listener is enabled.
Listener is individually enabled on nodes:
Listener is individually disabled on nodes:
```

The database listener (DB-LSNR) status is as follows:

```
[root@racnode1-0 ~]# su - grid -c "/u01/app/19c/grid/bin/srvctl status
listener"
Listener DB-LSNR is enabled
Listener DB-LSNR is running on node(s): racnode1-0,racnode2-0
Listener LISTENER is enabled
Listener LISTENER is running on node(s): racnode1-0,racnode2-0
```

2. Configure each Oracle RAC database instance to register with one endpoint of the new listeners.

   In this example, we have virtual IP addresses (VIP) set to racnode1-0-vip.rac.svc.cluster.local and racnode2-0-vip.rac.svc.cluster.local. We need to configure the DB instances using SQL*Plus, based on the VIPs and the domain name for the database. On racnode1-0, as the Oracle user, connect to the database using the SYSDBA account to set the ORACLE_SID environment variable:

   ```
   export ORACLE_SID=ORCLCDB1; $ORACLE_HOME/bin/sqlplus "/as sysdba"
   ```

   Then enter the SQL statements to set up the virtual IP addresses:

   ```
   ALTER SYSTEM SET local_listener='(ADDRESS=(PROTOCOL=TCP)(HOST=racnode1-0-
   vip.rac.svc.cluster.local)(PORT=30203))' SCOPE=BOTH SID='ORCLCDB1';
   ALTER SYSTEM SET local_listener='(ADDRESS=(PROTOCOL=TCP)(HOST=racnode2-0-
   vip.rac.svc.cluster.local)(PORT=30204))' SCOPE=BOTH SID='ORCLCDB2';
   alter system register;
   ```

3. Create the Kubernetes NodePort services.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Note**
>
> In this example, the port definition for ONS allows for redundant ONS notification channels. For that reason, there should be a separate NodePort service for each of the ONS endpoints, and for each of the new node listener endpoints.
>
> Ensure that the node port and the listener port match each other. For example, if the node port is 30203, then the listener port must be 30203 for that particular VIP for the database.

```
# cat racnode-np-svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: racnode-scan-lsnr
  namespace: rac
spec:
  type: NodePort
  ports:
    - port: 1521
      nodePort: 30200
      targetPort: 1521
  selector:
    cluster: racnode
---
apiVersion: v1
kind: Service
metadata:
  labels:
   cluster: racnode
   scan_svc_np: racnode1-np
  name: racnode1-np
  namespace: rac
spec:
  type: NodePort
  ports:
  - name: ons-np-30201
    nodePort: 30201
    port: 6200
    protocol: TCP
    targetPort: 6200
  - name: nodelsnr-np-30203
    nodePort: 30203
    port: 30203
    protocol: TCP
    targetPort: 30203
  selector:
   hostname: racnode1-0
---
apiVersion: v1
kind: Service
metadata:
  labels:
   cluster: racnode
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
   scan_svc_np: racnode2-np
  name: racnode2-np
  namespace: rac
spec:
  type: NodePort
  ports:
  - name: ons-np-30202
    nodePort: 30202
    port: 6200
    protocol: TCP
    targetPort: 6200
  - name: nodelsnr-np-30204
    nodePort: 30204
    port: 30204
    protocol: TCP
    targetPort: 30204
  selector:
   hostname: racnode2-0

[root@oper01 yaml_files]# kubectl apply -f racnode-np-svc.yaml
```

The results of a `kubectl get all -n rac` command to show all the resources in the namespace should appear similar to the following:

```
[root@oper01 yaml_files]# kubectl get all -n rac
NAME             READY   STATUS    RESTARTS        AGE
pod/racnode1-0   1/1     Running   0               3h
pod/racnode2-0   1/1     Running   0               3h

NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP
PORT(S)                            AGE
service/racnode-scan          ClusterIP   None            <none>
<none>                             3h
service/racnode-scan-lsnr     NodePort    10.105.240.58   <none>
1521:30200/TCP                     3h
service/racnode1-0            ClusterIP   None            <none>
<none>                             3h
service/racnode1-0-vip        ClusterIP   None            <none>
<none>                             3h
service/racnode1-np           NodePort    10.108.48.205   <none>
6200:30201/TCP,30203:30203/TCP   3h
service/racnode2-0            ClusterIP   None            <none>
<none>                             3h
service/racnode2-0-vip        ClusterIP   None            <none>
<none>                             3h
service/racnode2-np           NodePort    10.97.225.157   <none>
6200:30202/TCP,30204:30204/TCP   3h

NAME                          READY   AGE
statefulset.apps/racnode1     1/1     3h
statefulset.apps/racnode2     1/1     3h
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

4. Obtain the IP addresses of the worker nodes. For example:

```
# getent hosts worker-1 worker-2
10.0.1.60 worker-1
10.0.1.124 worker-2
```

5. As `root`, edit `/etc/hosts` on the client. Add the Worker node IP addresses and domain name of the VIP and the Scan IP addresses set up in the OCNE RAC Pods:

```
10.0.1.60     racnode1-0.rac.svc.cluster.local racnode1-0-
vip.rac.svc.cluster.local  racnode-scan.rac.svc.cluster.local
10.0.1.124     racnode2-0.rac.svc.cluster.local racnode2-0-
vip.rac.svc.cluster.local  racnode-scan.rac.svc.cluster.local
```

6. Test the Connection that you have configured.

   After you have populated `/etc/hosts` with the Worker node entries, confirm that you can use the client to connect to the database. In our example, we are using SQL*Plus for testing the connection on port 30200.

```
sqlplus <username>/<Password>@racnode-scan.rac.svc.cluster.local:30200/
ORCLPDB
```

   If you want to use a Java application, then for ONS notification, ensure that you use ONS ports exposed at the Worker node level. In our example configuration, ONS ports are set at 30201,30202 at the Worker node level.

Complete Pod Restart Configuration Tasks

To bring up the Oracle Clusterware stack and Oracle Database Instance on the Oracle RAC Node after a host Pod restart, complete these steps. A Pod that is hosting an Oracle RAC node loses its `systemd` configuration for Oracle Grid Infrastructure after a restart, which prevents starting Oracle Grid Infrastructure and the database instances that were previously running in the Pod. For example, after the worker node where the Pod was running has been restarted, Oracle Grid Infrastructure and the database instances it supported fail to restart. You must restore that configuration after a Pod restart.

> **Note**
>
> Oracle recommends that you back up the user home directories in a location under `/u01` that is in persistent storage, so that you do not lose any additional application- or operational-specific steps that populate other files in the grid and oracle user home directories when a Pod is restarted. If you have those backups, then you can restore the grid and oracle home directories from those backups, and avoid having to do these manual steps to restore SSH keys and the `.bashrc` content. For the same reason, if you have additional content in the root file system, then Oracle recommends you also back that content up so that it can be restored after recreating the host pod.

If you do not have backups in persistent storage, then recreate the Pod configuration after a Pod restart. The high-level overview of this procedure is as follows:

Copyright © 2022, 2025, Oracle and/or its affiliates.

1. Restore the default AHF installation by running as root user in the RAC Pod:

   ```
   # /u01/app/19c/grid/crs/install/ahf_setup -silent -crshome /u01/app/19c/
   grid
   ```

   Confirm that the `oracle-tfa` service is running using the following command:

   ```
   # systemctl status oracle-tfa --no-pager
   ```

2. Recover the Clusterware `systemd` services and its other locations in the root file system by running `rootcrs.sh -updateosfiles`.

   Before you run `rootcrs.sh -updateosfiles`, you can see that no files are in `/etc/oracle`:

   ```
   [root@racnode1-0 ~]# ls -rlt /etc/oracle
   ls: cannot access '/etc/oracle': No such file or directory
   ```

   Run the command `rootcrs.sh -updateosfiles` to create the required files. For example, if the Grid home is `/u01/app/19c/grid`, then run the following command:

   ```
   [root@racnode1-0 ~]# /u01/app/19c/grid/crs/install/rootcrs.sh -
   updateosfiles
   ```

   Check to ensure that the files are now populated under `/etc/oracle location`:

   ```
   [root@racnode1-0 ~]# ls -rlt /etc/oracle
   total 4352
   drwxrwx---. 2 root oinstall       6 Dec 21 21:48 lastgasp
   drwxrwxr-x. 5 root oinstall      44 Dec 21 21:48 oprocd
   drwxr-xr-x. 3 root oinstall      24 Dec 21 21:48 scls_scr
   drwxrwxrwt. 2 root oinstall       6 Dec 21 21:48 maps
   -rws--x---. 1 root oinstall 4446656 Dec 21 21:48 setasmgid
   -rw-rw-rw-. 1 root root           0 Dec 21 21:48 olr.loc.orig
   -rw-r--r--. 1 root oinstall      96 Dec 21 21:48 olr.loc
   -rw-rw-rw-. 1 root root           0 Dec 21 21:48 ocr.loc.orig
   -rw-r--r--. 1 root oinstall      37 Dec 21 21:48 ocr.loc
   ```

3. Set up SSH equivalence for the Oracle Clusterware (`grid`) user and Oracle (`oracle`) user accounts

4. Start the Oracle Grid Infrastructure stack as the `root` user. For example, if the Grid home is `/u01/app/19c/grid`, then run `start crs -wait`. For example:

   ```
   [root@racnode1-0 ~]# /u01/app/19c/grid/bin/crsctl start crs -wait

   CRS-4123: Starting Oracle High Availability Services-managed resources
   ...
   CRS-6024: Completed start of Oracle Cluster Ready Services-managed
   resources
   CRS-4123: Oracle High Availability Services has been started.
   ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

5. Wait for the return of the command to start the Oracle Grid Infrastructure stack. When it is started, check the Oracle Grid Infrastructure status. For example:

```
root@racnode1-0 ~]# /u01/app/19c/grid/bin/crsctl stat res -t
--------------------------------------------------------------------------------
-----
Name           Target  State   Server       State details
--------------------------------------------------------------------------------
-----
Local Resources
--------------------------------------------------------------------------------
-----
ora.DB-LSNR.lsnr
               ONLINE ONLINE    racnode1-0 STABLE
               ONLINE ONLINE    racnode2-0 STABLE
ora.LISTENER.lsnr
               ONLINE ONLINE    racnode1-0 STABLE
               ONLINE ONLINE    racnode2-0 STABLE
ora.chad
               ONLINE ONLINE    racnode1-0  STABLE
               ONLINE ONLINE    racnode2-0  STABLE
ora.net1.network
               ONLINE ONLINE    racnode1-0  STABLE
               ONLINE ONLINE    racnode2-0  STABLE
ora.ons
               ONLINE ONLINE    racnode1-0  STABLE
               ONLINE ONLINE    racnode2-0  STABLE
--------------------------------------------------------------------------------
-----
Cluster Resources
--------------------------------------------------------------------------------
-----
ora.ASMNET1LSNR_ASM.lsnr(ora.asmgroup)
      1       ONLINE ONLINE    racnode1-0  STABLE
      2       ONLINE ONLINE    racnode2-0  STABLE
ora.ASMNET2LSNR_ASM.lsnr(ora.asmgroup)
      1       ONLINE ONLINE    racnode1-0   STABLE
      2       ONLINE ONLINE    racnode2-0   STABLE
ora.DATA.dg(ora.asmgroup)
      1       ONLINE ONLINE    racnode1-0   STABLE
      2       ONLINE ONLINE    racnode2-0   STABLE
ora.LISTENER_SCAN1.lsnr
      1       ONLINE ONLINE    racnode1-0   STABLE
ora.LISTENER_SCAN2.lsnr
      1       ONLINE ONLINE    racnode2-0   STABLE
ora.asm(ora.asmgroup)
      1       ONLINE ONLINE    racnode1-0   Started,STABLE
      2       ONLINE ONLINE    racnode2-0   Started,STABLE
ora.asmnet1.asmnetwork(ora.asmgroup)
      1      ONLINE ONLINE     racnode1-0   STABLE
      2      ONLINE ONLINE     racnode2-0   STABLE
ora.asmnet2.asmnetwork(ora.asmgroup)
      1      ONLINE ONLINE     racnode1-0   STABLE
      2      ONLINE ONLINE     racnode2-0   STABLE
ora.cvu
      1      ONLINE ONLINE     racnode1-0   STABLE
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
ora.orclcdb.db
      1      ONLINE ONLINE     racnode1-0   Open,HOME=/u01/app/oracle/
product/19c/dbhome_1,STABLE
      2      ONLINE ONLINE     racnode2-0   Open,HOME=/u01/app/oracle/
product/19c/dbhome_1,STABLE
ora.orclcdb.orclpdbsvc.svc
      1      ONLINE ONLINE     racnode1-0   STABLE
      2      ONLINE ONLINE     racnode2-0   STABLE
ora.qosmserver
      1      ONLINE ONLINE     racnode1-0   STABLE
ora.racnode1-0.vip
      1      ONLINE ONLINE     racnode1-0   STABLE
ora.racnode2-0.vip
      1      ONLINE ONLINE     racnode2-0   STABLE
ora.scan1.vip
      1     ONLINE ONLINE      racnode1-0   STABLE
ora.scan2.vip
      1     ONLINE ONLINE      racnode2-0   STABLE
--------------------------------------------------------------------------------
-----
```

6. Add environment variables for the Oracle Clusterware (`grid`) user and Oracle (`oracle`) user accounts on the recreated Pod.

> **Note**
>
> If a database PDB is in `MOUNT` status after Oracle Grid Infrastructure is up and running due to the PDB startup policy, then you can start the PDB in open mode

In the following scenario, the Pod `racnode1-0` was recreated. To restore the ssh equivalence setup and user environment variables in the Pod, each of these steps must be completed in the order given.

1. Log in to `racnode1-0` Pod:

```
# kubectl exec -i -t  pod/racnode1-0 --/bin/bash -n rac
```

2. As `root`, set up SSH for the Grid and Oracle users

   Grid user:

```
mkdir -p /home/grid/.ssh
chmod 700 /home/grid/.ssh
cat /mnt/.ssh/id_rsa > /home/grid/.ssh/id_rsa
cat /mnt/.ssh/id_rsa.pub > /home/grid/.ssh/id_rsa.pub
chmod 400 /home/grid/.ssh/id_rsa
cat /mnt/.ssh/id_rsa.pub > /home/grid/.ssh/authorized_keys
chmod 600 /home/grid/.ssh/authorized_keys
chown -R grid:oinstall /home/grid/.ssh
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

Oracle user:

```
mkdir -p /home/oracle/.ssh
chmod 700 /home/oracle/.ssh
cat /mnt/.ssh/id_rsa > /home/oracle/.ssh/id_rsa
cat /mnt/.ssh/id_rsa.pub > /home/oracle/.ssh/id_rsa.pub
chmod 400 /home/oracle/.ssh/id_rsa
cat /mnt/.ssh/id_rsa.pub > /home/oracle/.ssh/authorized_keys
chmod 600 /home/oracle/.ssh/authorized_keys
chown -R oracle:oinstall /home/oracle/.ssh
```

3. Log in to `racnode1-0` and populate the `known_host` file with the SSH keys for the Grid and Oracle users:

   Grid user:

   ```
   su - grid -c "ssh-keyscan -H racnode1-0 >> /home/grid/.ssh/known_hosts"
   su - grid -c "ssh-keyscan -H racnode2-0 >> /home/grid/.ssh/known_hosts"
   ```

   Oracle user:

   ```
   su - oracle -c "ssh-keyscan -H racnode1-0 >> /home/oracle/.ssh/known_hosts"
   su - oracle -c "ssh-keyscan -H racnode2-0 >> /home/oracle/.ssh/known_hosts"
   ```

   Repeat this step on `racnode2-0`.

4. On `racnode1-0`, verify SSH is functioning between `racnode1-0` Pod and `racnode2-0` Pods for both the grid and oracle users::

   ```
   su - grid -c "ssh -o BatchMode=yes -o ConnectTimeout=5 grid@racnode1-0
   echo ok 2>&1"
   su - grid -c "ssh -o BatchMode=yes -o ConnectTimeout=5 grid@racnode2-0
   echo ok 2>&1"
   su - oracle -c "ssh -o BatchMode=yes -o ConnectTimeout=5 oracle@racnode1-0
   echo ok 2>&1"
   su - oracle -c "ssh -o BatchMode=yes -o ConnectTimeout=5 oracle@racnode2-0
   echo ok 2>&1"
   ```

5. Repeat this command on the `racnode2-0` Pod to verify that SSH is functioning between `racnode1-0` Pod and `racnode2-0` Pods for both the grid and oracle users::

   ```
   su - grid -c "ssh -o BatchMode=yes -o ConnectTimeout=5 grid@racnode1-0
   echo ok 2>&1"
   su - grid -c "ssh -o BatchMode=yes -o ConnectTimeout=5 grid@racnode2-0
   echo ok 2>&1"
   su - oracle -c "ssh -o BatchMode=yes -o ConnectTimeout=5 oracle@racnode1-0
   echo ok 2>&1"
   su - oracle -c "ssh -o BatchMode=yes -o ConnectTimeout=5 oracle@racnode2-0
   echo ok 2>&1"
   ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

6. If you do not have a backup of the `.bashrc` file available in persistent storage, then add the following variables to the `.bashrc` file of the `grid` user on the recreated Pod:

```
export TMPDIR=/var/tmp
export ORACLE_HOME=/u01/app/19c/grid
export PATH=$ORACLE_HOME/bin:$PATH
```

7. Add the following variables to the `.bashrc` file of the `oracle` user on the recreated `racnode1-0` Pod:

```
export TMPDIR=/var/tmp
export ORACLE_HOME=/u01/app/oracle/product/19c/dbhome_1
export PATH=$ORACLE_HOME/bin:$PATH
```

Options to Consider After Deployment

After deployment of Oracle RAC in pods, you can choose to add or remove Oracle Real Application Clusters (Oracle RAC) nodes, or install different releases of Oracle RAC database.

> **Note**
>
> After completing your deployment, you can make changes to your Oracle RAC cluster:

Recreating the environment: Same or Different Releases

To install other releases of Oracle databases, refer to the documentation to ensure that the database release that you want to install is supported for your container software release.

How to Clean Up a StatefulSet for Oracle RAC on OCNE

If you need to remove a StatefulSet currently used for Oracle Real Application Clusters (Oracle RAC) on OCNE, then use this procedure.

> **Note**
>
> The procedures are different, depending on whether you intend to delete the StatefulSet, or recreate it at another time.

When you delete a StatefulSet for an Oracle RAC Pod, and if you intend to remove the Oracle RAC node, then you should first complete the node deletion procedure for the Oracle RAC instance and for the Oracle Grid Infrastructure node. After you delete the Oracle RAC and Oracle Grid Infrastructure software, you can then delete the StatefulSet and its associated Pod.

In the following example, a StatefulSet is removed permanently.

1. On the container where you want to remove the StatefulSet, stop the cluster, delete the Oracle RAC instance from the cluster, and delete the Oracle Clusterware node.

   See:

Copyright © 2022, 2025, Oracle and/or its affiliates.

Adding and Deleting Oracle RAC from Nodes on Linux and UNIX Systems in *Oracle Grid Infrastructure Installation and Upgrade Guide for Linux*

Adding and Deleting Cluster Nodes on Linux and UNIX Systems in *Oracle Real Application Clusters Administration and Deployment Guide*

2. Deinstall Oracle Grid Infrastructure and Oracle RAC software.

   See:

   "Removing Oracle Database Software" in *Oracle Grid Infrastructure Installation and Upgrade Guide for Linux*

3. Delete the StatefulSet.

   If you intend to delete and remove the StatefulSet, then you can delete it without shutting down the Pod gracefully. For example:

   ```
   kubectl delete statfulset statefulset-name
   ```

> **Note**
>
> If you intend to recreate the StatefulSet, and perhaps intend to change parts of the StatefulSet definition, then do not use this procedure. To reuse a StatefulSet, the Pod must be gracefully deleted. For example:
>
> ```
> kubectl delete statefuset statefulset-name --cascade=orphan
> kubectl delete pod pod-name --grace-period=600 --wait=true
> ```
>
> After you recreate the Pod, you must then apply the steps in "Complete Pod Restart Configuration Tasks"

Related Topics

- Oracle Cloud Native Environment
- Podman Documentation
- Adding and Deleting Cluster Nodes on Linux and UNIX Systems
- Adding and Deleting Oracle RAC from Nodes on Linux and UNIX Systems

Known Issues for Oracle RAC on OCNE

When you deploy Oracle Real Application Clusters (Oracle RAC) on OCNE, if you encounter an issue, check to see if it is a known issue.

For issues specific to Oracle RAC on OCNE deployments, refer to My Oracle Support Doc ID 2969928.1.

Related Topics

- Oracle RAC on OCNE - Released Versions and Known Issues (Doc ID 2969928.1)

Copyright © 2022, 2025, Oracle and/or its affiliates.

Additional Information for Oracle RAC on OCNE Configuration

This information can help to resolve issues that can arise with Oracle Real Application Clusters (Oracle RAC) on OCNE.

Copyright © 2022, 2025, Oracle and/or its affiliates.