Oracle® Cloud

Using Oracle Cloud Infrastructure Container Service Classic

E75878-09
May 2020

Oracle Cloud Using Oracle Cloud Infrastructure Container Service Classic,

E75878-09

Primary Author: Andy Page

Contributing Authors: Jon Reeve, Mike Raab

Contributors: Oracle Cloud Infrastructure Container Service Classic Development, Product Management, and QA teams

Contents

Preface

Audience vii
Documentation Accessibility vii
Related Resources vii
Conventions viii

1 Getting Started with Oracle Container Cloud Service

About Oracle Container Cloud Service 1-1
About the Components and Interfaces to Oracle Container Cloud Service 1-1
Accessing Oracle Container Cloud Service 1-2
Before You Begin with Oracle Container Cloud Service 1-2
- About Docker 1-2
- About Oracle Cloud 1-3

2 Administering Oracle Container Cloud Service

Typical Workflow for Setting Up and Administering Oracle Container Cloud Service 2-2
Ordering a Subscription for Oracle Container Cloud Service 2-3
Administering Oracle Container Cloud Service Instances 2-4
- Accessing the Service Console for Oracle Container Cloud Service 2-4
- Creating Oracle Container Cloud Service Instances 2-5
- Viewing Information about Oracle Container Cloud Service Instances 2-7
- Stopping, Starting, and Restarting Oracle Container Cloud Service Instances 2-9
  - Stopping an Oracle Container Cloud Service Instance 2-9
  - Starting an Oracle Container Cloud Service Instance 2-9
  - Restarting an Oracle Container Cloud Service Instance 2-10
- Stopping, Starting, and Restarting Manager and Worker Nodes 2-10
  - Stopping, Starting, and Restarting Manager Nodes 2-10
  - Stopping, Starting, and Restarting Worker Nodes 2-11
- Enabling and Disabling Secure Shell (SSH) Access to Oracle Container Cloud Service Manager and Worker Nodes 2-12

About SSH Access to Oracle Container Cloud Service Manager and Worker Nodes 2-13
Connecting to Oracle Container Cloud Service Manager and Worker Nodes Through SSH 2-13
Adding Public SSH Keys to Oracle Container Cloud Service Instances 2-17
Removing Public SSH Keys from Oracle Container Cloud Service Manager and Worker Nodes Using SSH 2-17
Uploading Your Own SSL Certificates to a Manager Node Using SSH 2-18
Changing the Number of Worker Node Hosts in Oracle Container Cloud Service Instances 2-20
Adding Worker Node Hosts 2-20
Removing Worker Node Hosts 2-21
Managing Access Rules for Oracle Container Cloud Service Instances 2-22
Viewing Activity for Oracle Container Cloud Service Instances 2-24
Viewing Log Files on Oracle Cloud Container Service Manager and Worker Nodes Using SSH 2-24
Changing the Username or Password for an Oracle Container Cloud Service Instance Administrator 2-26
Resetting the Password for an Oracle Container Cloud Service Instance Administrator Using SSH 2-26
Backing Up and Restoring Oracle Container Cloud Service Instances 2-27
Backing Up an Oracle Container Cloud Service Instance 2-28
Restoring an Oracle Container Cloud Service Instance 2-28
Upgrading Oracle Container Cloud Service Instances 2-29
Deleting Oracle Container Cloud Service Instances 2-30

3 Managing Your Docker Environment with Oracle Container Cloud Service

Typical Workflow for Managing and Monitoring Your Docker Environment with Oracle Container Cloud Service 3-1
Accessing the Container Console for Oracle Container Cloud Service 3-3
A Quick Tour of the Container Console 3-4
Using the Oracle Container Cloud Service Dashboard 3-5
Using the Oracle Container Cloud Service Quick Start Wizard 3-6
Searching Oracle Container Cloud Service 3-9
Managing Resource Pools with Oracle Container Cloud Service 3-10
About Resource Pools 3-11
Managing Resource Pools 3-11
Creating a Resource Pool 3-13
Managing Hosts with Oracle Container Cloud Service 3-15
About Hosts 3-15
Managing Hosts 3-16

Moving Hosts Between Resource Pools 3-18
Managing Services with Oracle Container Cloud Service 3-18
About Services 3-19
Managing Services 3-19
Creating a Service with Oracle Container Cloud Service 3-21
Configuring the Default Orchestration Policy for a Service with Oracle Container Cloud Service 3-24
Deploying a Service with Oracle Container Cloud Service 3-25
Service Configuration Option Reference 3-26
Managing Application Stacks with Oracle Container Cloud Service 3-28
About Stacks 3-29
Managing Stacks 3-29
Creating a Stack 3-31
Deploying a Stack 3-34
Defining Dependencies between Services in a Stack using Phases 3-37
Managing Docker Containers with Oracle Container Cloud Service 3-39
About Docker Containers 3-39
Managing Docker Containers 3-40
Accessing an Application Running Inside a Deployed Container 3-42
Enabling Docker Containers to Communicate With Each Other 3-43
About Container Communication 3-43
Managing Entries in the Service Discovery Database to Enable Container Communication 3-44
Viewing the Log Files of a Running Container 3-46
Managing Docker Images with Oracle Container Cloud Service 3-47
About Docker Images 3-47
Managing Docker Images 3-48
Starting a Docker Container Directly from a Docker Image 3-50
Pushing a Docker Image to a Docker Repository 3-53
Managing Deployments with Oracle Container Cloud Service 3-55
About Deployments 3-55
Managing Deployments 3-56
Scaling a Running Deployment with Oracle Container Cloud Service 3-58
Monitoring the Health of a Deployment 3-58
Stopping and Re-starting Service and Stack Deployments 3-59
Stopping a Service or Stack Deployment 3-59
Re-starting a Service or Stack Deployment 3-60
Managing Docker Registries with Oracle Container Cloud Service 3-60
About Docker Registries and Registry Definitions 3-60
Managing Docker Registry Definitions 3-60
Creating a Docker Registry Definition 3-61
Managing Tags with Oracle Container Cloud Service 3-63

About Tags 3-63
Creating and Assigning Tags 3-63
Creating New Tags Using the Tags Page 3-64
Creating and Assigning New Tags When Defining Hosts and Resource Pools 3-65
Managing Tags Using the Tags Page 3-65
Monitoring Tasks, Events, and Status Updates with Oracle Container Cloud Service 3-66
About Tasks, Events, and Status Updates 3-67
Monitoring Tasks, Events, and Status Updates 3-67
Managing Profile Settings with Oracle Container Cloud Service 3-68
Changing and Viewing Profile Settings 3-69
Using Template Functions and Arguments with Oracle Container Cloud Service 3-70
About Template Functions and Arguments 3-70
Adding Template Functions and Arguments to Service Definitions 3-72
Adding Template Functions and Arguments to Stack Definitions 3-73
Template Function and Argument Reference 3-74
Using the Oracle Container Cloud Service REST API 3-78
About the Oracle Container Cloud Service REST API 3-78
Calling the Oracle Container Cloud Service REST API 3-79
Adding Keys and Values to the Key/Value Store Database 3-79
Example REST API Calls to Access the Key/Value Store Database 3-80

A Using Oracle Container Cloud Service Example Stacks

About the Example Stacks A-1
Exploring an Example Stack A-1

Preface

Learn how to configure, deploy, administer, monitor, and orchestrate services (and stacks of services) as Docker containers across multiple hosts using Oracle Container Cloud Service.

Topics:

- Audience
- Documentation Accessibility
- Related Resources
- Conventions

Audience

Using Oracle Container Cloud Service is intended for members of Development and Operations teams who want to setup, manage, and monitor Docker environments using the convenient features offered by Oracle Container Cloud Service.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Resources

For more information, see these Oracle resources:

- Oracle Public Cloud

  `http://cloud.oracle.com`

- *Getting Started with Oracle Cloud*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Getting Started with Oracle Container Cloud Service

Learn how to get started with Oracle Container Cloud Service.

Topics:

- About Oracle Container Cloud Service
- About the Components and Interfaces to Oracle Container Cloud Service
- Before You Begin with Oracle Container Cloud Service
- Accessing Oracle Container Cloud Service

About Oracle Container Cloud Service

Oracle Container Cloud Service (also known as Oracle Cloud Infrastructure Container Service Classic) offers Development and Operations teams the benefits of easy and secure Docker containerization when building and deploying applications.

Oracle Container Cloud Service:

- provides an easy-to-use interface to manage the Docker environment
- provides out-of-the-box examples of containerized services and application stacks that can be deployed in one click
- enables developers to easily connect to their private Docker registries (so they can ‘bring their own containers’)
- enables developers to focus on building containerized application images and Continuous Integration/Continuous Delivery (CI/CD) pipelines, not on learning complex orchestration technologies

Video

Find out more about Oracle Container Cloud Service at https://cloud.oracle.com/container.

Join the conversation about Oracle Container Cloud Service in the Product Forum.

About the Components and Interfaces to Oracle Container Cloud Service

You can set up, manage, and monitor your Docker environment using the following Oracle Container Cloud Service components and interfaces:

Component/Interface	Description	More Information
Oracle Container Cloud Service Container Console	Use graphical dashboards, editors, and views to manage the images, containers, and registries in your Docker environment.	Accessing the Container Console for Oracle Container Cloud Service
Oracle Container Cloud Service REST API	Code REST requests to call methods to programmatically manage the images, containers, and registries in your Docker environment.	Using the Oracle Container Cloud Service REST API
Oracle Container Cloud Service Console	Use dashboards and wizards to create and manage Oracle Container Cloud Service service instances.	Accessing the Service Console for Oracle Container Cloud Service

Accessing Oracle Container Cloud Service

After you or your organization have subscribed to Oracle Container Cloud Service, how you access Oracle Container Cloud Service will depend on your role and responsibilities.

If you're responsible for administering and monitoring Oracle Container Cloud Service instances, you'll be using the Service Console (see Accessing the Service Console for Oracle Container Cloud Service).

If you're responsible for managing your Docker environment, you'll be using the Container Console (see Accessing the Container Console for Oracle Container Cloud Service).

Before You Begin with Oracle Container Cloud Service

Before you sign in to Oracle Container Cloud Service, it's useful to be familiar with the technologies that it depends on and interacts with, namely Docker and Oracle Cloud.

Topics

- About Docker
- About Oracle Cloud

About Docker

Docker is an open platform for building, shipping, and running distributed applications. It gives programmers, development teams, and operations engineers the common toolbox they need to take advantage of the distributed and networked nature of modern applications.

To run an application using Docker, you:

- create a Dockerfile to describe the application and its dependencies
- use the Dockerfile to build a Docker image to hold the application
- associate the Docker image with a named repository in a Docker registry

- pull the image from the registry and load it into a Docker container
- run the Docker container

Docker Hub provides a public cloud-based Docker registry enabling you to link to code repositories, build your images and test them, and store manually pushed images.

Docker Compose is a command-line tool for defining and running multi-container Docker applications. You write a definition of the services that make up an application in YAML (a data-oriented language), which you include in a configuration file called docker-compose.yml. You then run Docker Compose, which reads the docker-compose.yml file and starts the entire application.

For more information about Docker, see the Docker documentation at www.docker.com.

Oracle Container Cloud Service provides an easy-to-use interface to manage the Docker environment, and much more besides. For more information, see About Oracle Container Cloud Service.

About Oracle Cloud

Oracle Cloud is an enterprise cloud for businesses. Oracle Cloud offers self-service business applications delivered on an integrated development and deployment platform with tools to extend and create new services rapidly.

With predictable subscription pricing, Oracle Cloud delivers instant value and productivity for users, administrators, and developers. Our fully managed environment is built using Oracle Exadata, Oracle Exalogic, Oracle Database, and Oracle WebLogic products.

In addition, the Oracle Cloud environment includes built-in identity management, high availability, elasticity, backup, and monitoring to enable secure and scalable applications. With open Java and SQL standards at the core, enterprises can finally leverage existing IT skill sets and avoid lock-in of their business applications in the cloud.

An Oracle Cloud user has one or more roles. These roles include privileges to order Oracle Cloud service subscriptions, administer services, and manage user accounts.

When ordering an Oracle Cloud service, you choose a subscription to suit the needs and budget of your organization. See Overview of Oracle Cloud Subscriptions in *Getting Started with Oracle Cloud*.

When you've set up and activated an Oracle Cloud service, Oracle Cloud sends designated administrators the following information required to access the My Services application:

- Sign-in credentials (a username, temporary password, identity domain, and data center where the service is located)
- My Services URL

As an administrator, you can then use the My Services application to administer your Oracle Cloud service, by using the service console and associated tools to:

- verify the service is up and running
- monitor utilization
- view service details

- create and administer service instances
- create and manage accounts for users who will be accessing the service

For more information about Oracle Cloud, see About Oracle Cloud in *Getting Started with Oracle Cloud*.

2

Administering Oracle Container Cloud Service

Learn how to order an Oracle Container Cloud Service subscription and how to access the Oracle Container Cloud Service Console to administer Oracle Container Cloud Service instances.

Topics

- Typical Workflow for Setting Up and Administering Oracle Container Cloud Service
- Ordering a Subscription for Oracle Container Cloud Service
- Administering Oracle Container Cloud Service Instances

Typical Workflow for Setting Up and Administering Oracle Container Cloud Service

Here's a typical workflow showing the tasks you'll usually perform to set up and administer Oracle Container Cloud Service.

Task	Description	More Information
Order and activate a subscription for Oracle Container Cloud Service	You choose whether to: • request a trial subscription • buy a nonmetered subscription • buy a metered subscription When you order a subscription for Oracle Container Cloud Service, you automatically get a subscription for Oracle Developer Cloud Service as an entitlement. You can also purchase other subscriptions, such as Oracle Database Cloud Service and Oracle Messaging Cloud Service, if needed for your application. **Note:** Oracle Container Cloud Service requires access to block and object storage in Oracle Cloud. You'll probably already have access to this storage through your subscriptions to other cloud services (for example, Oracle Storage Cloud Service, Oracle Compute Cloud Service, Oracle Database Cloud Service, Oracle Java Cloud Service). If not, you'll have to order a subscription to a service that does provide Oracle Cloud block and object storage.	Ordering a Subscription for Oracle Container Cloud Service
Create an Oracle Container Cloud Service service instance	Sign in to the My Services application to access the Oracle Container Cloud Service Console and create a service instance.	Accessing the Service Console for Oracle Container Cloud Service Creating Oracle Container Cloud Service Instances

Task	Description	More Information
Administer Oracle Container Cloud Service	Use the Oracle Container Cloud Service Console to administer Oracle Container Cloud Service and service instances, including to: • create additional service instances • manage SSH keys • manage access rules • view activity • delete unrequired service instances	Accessing the Service Console for Oracle Container Cloud Service Creating Oracle Container Cloud Service Instances Adding Public SSH Keys to Oracle Container Cloud Service Instances Managing Access Rules for Oracle Container Cloud Service Instances Viewing Activity for Oracle Container Cloud Service Instances Deleting Oracle Container Cloud Service Instances

Ordering a Subscription for Oracle Container Cloud Service

Before you can start using Oracle Container Cloud Service, you have to order a subscription.

To order a subscription for Oracle Container Cloud Service:

1. If you don't have one already, get an Oracle.com account (see Getting an Oracle.com Account in *Getting Started with Oracle Cloud*).
2. Decide the type of Oracle Container Cloud Service subscription that's most appropriate for your current requirement.

   If you're not sure which type of subscription is right for you, see Overview of Oracle Cloud Subscriptions in *Getting Started with Oracle Cloud*.
3. Order the Oracle Container Cloud Service subscription you've decided on by following the corresponding instructions in *Getting Started with Oracle Cloud*, as shown below:
   - to sign up for a free credit promotion, see Requesting and Managing Free Oracle Cloud Promotions
   - to sign up for a paid Oracle Cloud account, see Buying an Oracle Cloud Subscription
   - to sign up for traditional nonmetered and metered subscriptions, see Activating Other Types of Subscriptions
4. If you want other users to be able to create Oracle Container Cloud Service service instances, grant them the following roles:
   - Oracle Compute Cloud Service instance administrator (the 'Compute Compute Operations' role)
   - Oracle Container Cloud Service instance administrator (the 'CONTAINER Administrator' role)

   As the user who obtained the Oracle Container Cloud Service subscription, you're granted these roles automatically. See Creating a User and Assigning a Role in *Getting Started with Oracle Cloud*.

Oracle Cloud sends designated administrators the following information required to access the My Services application:

- Sign-in credentials (a username, temporary password, identity domain, and data center where the service is located)
- My Services URL

Administrators can now create Oracle Container Cloud Service service instances by following the instructions in Creating Oracle Container Cloud Service Instances.

Administering Oracle Container Cloud Service Instances

Learn how to administer Oracle Container Cloud Service instances.

Topics

- Accessing the Service Console for Oracle Container Cloud Service
- Creating Oracle Container Cloud Service Instances
- Viewing Information about Oracle Container Cloud Service Instances
- Stopping, Starting, and Restarting Oracle Container Cloud Service Instances
- Stopping, Starting, and Restarting Manager and Worker Nodes
- Enabling and Disabling Secure Shell (SSH) Access to Oracle Container Cloud Service Manager and Worker Nodes
- Uploading Your Own SSL Certificates to a Manager Node Using SSH
- Changing the Number of Worker Node Hosts in Oracle Container Cloud Service Instances
- Managing Access Rules for Oracle Container Cloud Service Instances
- Viewing Activity for Oracle Container Cloud Service Instances
- Viewing Log Files on Oracle Cloud Container Service Manager and Worker Nodes Using SSH
- Changing the Username or Password for an Oracle Container Cloud Service Instance Administrator
- Resetting the Password for an Oracle Container Cloud Service Instance Administrator Using SSH
- Backing Up and Restoring Oracle Container Cloud Service Instances
- Upgrading Oracle Container Cloud Service Instances
- Deleting Oracle Container Cloud Service Instances

Accessing the Service Console for Oracle Container Cloud Service

If you're responsible for administering and monitoring Oracle Container Cloud Service instances, you'll be using the Service Console.

To access the Oracle Container Cloud Service Console:

1. Sign in to the My Services application at the URL and using the credentials you've received, either from your administrator or in an email from Oracle Cloud.

2. In the My Services dashboard, navigate to the Oracle Container Cloud Service entry and select **Open Service Console** from the menu.
3. On the **Service Details** page, click **Open Service Console** to display the **Services** tab of the Oracle Container Cloud Service Console.
4. Administer Oracle Container Cloud Service instances by performing tasks such as:
   - Creating Oracle Container Cloud Service Instances
   - Adding Public SSH Keys to Oracle Container Cloud Service Instances
   - Managing Access Rules for Oracle Container Cloud Service Instances
   - Viewing Activity for Oracle Container Cloud Service Instances
   - Deleting Oracle Container Cloud Service Instances

Creating Oracle Container Cloud Service Instances

You can use a simple wizard to define and create Oracle Container Cloud Service instances, specifying (amongst other things) the number of worker nodes that can run Docker containers.

When you create an Oracle Container Cloud Service instance, you specify the username and password for an instance administrator. Use these credentials to launch the Oracle Container Cloud Service Container Console from the Oracle Container Cloud Service Console. You can change the instance administrator's username and/or password later:

- using the Container Console (see Changing the Username or Password for an Oracle Container Cloud Service Instance Administrator)
- using SSH (see Resetting the Password for an Oracle Container Cloud Service Instance Administrator Using SSH)

Every Oracle Container Cloud Service instance you create will always have one manager node, and the number of worker nodes that you specify. Oracle Container Cloud Service software running on the manager node orchestrates the deployment of Docker containers to the worker nodes in the instance.

Manager nodes and worker nodes are Oracle Compute virtual machines (VMs), also known as compute nodes or compute VMs. When you create an Oracle Container Cloud Service instance, you're billed for the total number of compute VMs you request for the instance (the number of worker nodes, plus one manager node).

To create a new Oracle Container Cloud Service service instance using the Oracle Container Cloud Service Console:

1. Navigate to the Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.
2. Click **Create Service** to display the Oracle Container Cloud Service Instance Creation Wizard.
3. Enter properties for the new service instance on the **Details** page of the Instance Creation Wizard as follows:

Option	Use to specify:
Service Name	A name for the service instance that is unique within the identity domain.
Service Description	An optional description to identify the service instance.
SSH Public Key	An existing public key (or a file containing the public key). Alternatively, you can generate a new public key.
Admin Username	The instance administrator's username to use when logging into the Container Console. The default username in this field is 'admin', but you can change it now before you create the instance. And you can change the instance administrator's username later if you need to, using the Container Console (see Changing the Username or Password for an Oracle Container Cloud Service Instance Administrator).
Admin Password and **Confirm Admin Password**	The instance administrator's password to use when logging into the Container Console. The password can be a minimum of 8 and a maximum of 32 characters with at least 1 uppercase letter, 1 lowercase letter, and 1 number. You can change the instance administrator's password later if you need to, using: • the Container Console (see Changing the Username or Password for an Oracle Container Cloud Service Instance Administrator) • SSH (see Resetting the Password for an Oracle Container Cloud Service Instance Administrator Using SSH)
Worker node Compute Shape	The shape, or resource profile, that determines the number of CPUs and amount of memory assigned to each worker node in the service instance.
Number of worker nodes	The number of worker nodes (between 1 and 999) to create with the service instance.
Worker node data volume size (GB)	The size of the data volume available to each worker node.
Manager node Compute Shape	The shape, or resource profile, that determines the number of CPUs and amount of memory assigned to the manager node in the service instance.

4. Click **Next** and review the details you've entered on the **Confirmation** page of the Oracle Container Cloud Service Instance Creation Wizard.
5. Click **Create** to create the Oracle Container Cloud Service instance. The message `Creating service ...` appears in the Status field.

> **Tip:**
>
> Click `Creating service ...` to see the progress and the messages output during the instance creation process.

Within a few minutes, the Oracle Container Cloud Service instance is created with the details you specified.

6. Click the name of the newly created instance to see the virtual machines that have been created on the **Service Details** page.

7. Optional: You can now administer the new Oracle Container Cloud Service instance by clicking the Menu icon (beside the service instance name at the top of the page) and selecting:
   - **Container Console** to access the Container Console to manage and monitor your Docker environment (see Accessing the Container Console for Oracle Container Cloud Service)
   - **SSH Access** to add public SSH keys to the VMs (see Adding Public SSH Keys to Oracle Container Cloud Service Instances)
   - **Access Rules** to create and manage access rules for selected sources and destinations (see Managing Access Rules for Oracle Container Cloud Service Instances)
8. Optional: If other users are going to be using Oracle Container Cloud Service to manage and monitor your Docker environment, notify them of the URL and credentials to use to access the Container Console.

   The URL is in the format `https://<manager-node-ip-address>/#/dashboard`, where `<manager-node-ip-address>` is the public IP address of the manager node. For example, `http://192.0.2.254/#/dashboard`. You can find out the Container Console's URL in either of the following ways:
   - On the **Service Details** page, click the Menu icon (beside the service instance name at the top of the page), select **Container Console**, and copy the URL that's used to launch the Container Console.
   - On the **Service Details** page, locate the manager node (the VM with the **Instance Type** set to MANAGER), copy the value in the **Public IP** field, and use that to construct the URL. For example, if the value in the **Public IP** field is `192.0.2.254`, the Container Console's URL is `http://192.0.2.254/#/dashboard`.

Viewing Information about Oracle Container Cloud Service Instances

Use the Oracle Container Cloud Service Console to see summary and detailed information about Oracle Container Cloud Service instances in the identity domain.

To view information about Oracle Container Cloud Service instances:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.
2. On the **Services** tab:
   a. View summary information for all the Oracle Container Cloud Service instances in the identity domain, including:
      - the total number of service instances
      - the total number of OCPUs allocated to all instances
      - the total amount of memory available to all instances
      - the total amount of storage available to all instances
      - the total number of public IP addresses
   b. View a selection of detailed information for all the Oracle Container Cloud Service instances in the identity domain. For each instance, you can see:

- the type of subscription
- the number of nodes and OCPUs allocated to the instance
- the Oracle Container Cloud Service version
- the amount of memory and storage available to the instance

c. View a log of service creation and deletion operations by expanding the **Service Create and Delete History** section.

3. On the **Services** tab, click the name of an Oracle Container Cloud Service instance about which you want to see more detailed information.
4. On the **Service Details** page:
   a. View summary information for the selected instance, including:
      - the number of nodes allocated to the instance
      - the total number of allocated OCPUs across all nodes in the instance
      - the total amount of allocated memory and storage across all nodes in the instance
   b. View detailed information for VMs in the selected instance. For each VM, you can see the VM's:
      - host name
      - number of OCPUs
      - public IP address
      - memory and storage
      - description and type
   c. View detailed metrics information about each VM in the selected instance by clicking **Healthcheck**. For each VM, you can see:
      - the CPU utilization
      - the available memory
      - the number of deployed applications (manager nodes only)
      - the status of the Docker daemon (worker nodes only)
      - the number of running Docker containers (worker nodes only)
   d. View additional detailed information about the selected instance by expanding the **More Information** section, including:
      - the type of subscription
      - the Oracle Container Cloud Service version
      - the data center location
      - the compute shape of manager and worker nodes
   e. View detailed information about operations on the instance by expanding the **Activity** section. For each operation, you can see:
      - the type of operation
      - when the operation started and finished
      - the status of the operation

Stopping, Starting, and Restarting Oracle Container Cloud Service Instances

Learn about how to stop, start, and restart Oracle Container Cloud Service instances.

Topics

- Stopping an Oracle Container Cloud Service Instance
- Starting an Oracle Container Cloud Service Instance
- Restarting an Oracle Container Cloud Service Instance

Stopping an Oracle Container Cloud Service Instance

When you stop an Oracle Container Cloud Service instance using the Oracle Container Cloud Service Console, the manager node and all worker nodes in the instance are stopped. You cannot perform management operations on a stopped instance except to start it or to delete it. When you stop an instance, its CPU and RAM are stopped.

To stop an Oracle Container Cloud Service instance:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.

2. Click the Menu icon beside the service instance that you want to stop, and select **Stop**.
3. When prompted, click **OK** to confirm that you want to stop the instance.

   The manager node and all worker nodes in the instance are stopped. In addition, the **OCPUs** and **Memory** fields indicate that the resources are not currently in use.

Starting an Oracle Container Cloud Service Instance

When you start a stopped Oracle Container Cloud Service instance using the Oracle Container Cloud Service Console, the manager node and all worker nodes in the instance are started. You can once again perform management operations such as changing the number of worker nodes and backing up the instance.

To start an Oracle Container Cloud Service instance:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.

2. Click the Menu icon beside the service instance that you want to stop, and select **Start**.
3. When prompted, click **OK** to confirm that you want to start the instance.

   The manager node and all worker nodes in the instance are started.

Restarting an Oracle Container Cloud Service Instance

When you restart an Oracle Container Cloud Service instance using the Oracle Container Cloud Service Console, the manager node and all worker nodes in the instance are stopped and then immediately started again.

To re-start an Oracle Container Cloud Service instance:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.

2. Click the Menu icon beside the service instance that you want to stop, and select **Restart**.
3. When prompted, click **OK** to confirm that you want to re-start the instance.

   The manager node and all worker nodes in the instance are stopped, and then started.

Stopping, Starting, and Restarting Manager and Worker Nodes

Learn about how to stop, start, and restart Oracle Container Cloud Service manager nodes and worker nodes.

Topics

- Stopping, Starting, and Restarting Manager Nodes
- Stopping, Starting, and Restarting Worker Nodes

Stopping, Starting, and Restarting Manager Nodes

You use the Oracle Container Cloud Service Console to restart the manager node in an Oracle Container Cloud Service instance. While the manager node is being restarted, the instance is not available for Oracle Container Cloud Service operations.

Manager nodes are implicitly stopped, started, and restarted when you stop, start, and restart Oracle Container Cloud Service instances. When you:

- Stop a running instance, the manager node and all worker nodes in the instance are stopped. You cannot start worker nodes while the manager node is stopped.
- Start a stopped instance, the manager node and all worker nodes in the instance are started.
- Restart a running instance, the manager node and all worker nodes in the instance are stopped, and then started.

What happens when you explicitly restart the manager node of an Oracle Container Cloud Service instance depends on whether the instance is running:

- If you restart the manager node of a running instance, the manager node and all running worker nodes are first stopped. Then the manager node is started, followed by all the worker nodes. The instance is returned to a running state.
- If you restart the manager node of a stopped instance (that is, an instance in which the manager node and all worker nodes are already stopped), the manager node

is started. However, note that worker nodes are not restarted. You have to restart the worker nodes individually, starting with the original worker nodes that were originally defined when the instance was initially created. When the original worker nodes have all been restarted, you can restart additional worker nodes that were added after the instance was initially created.

To explicitly restart the manager node in an Oracle Container Cloud Service instance:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.

2. On the **Services** tab, click the name of the Oracle Container Cloud Service instance in which you want to restart the manager node.

3. Click the Menu icon beside the manager node and select **Restart**.

4. When prompted, click **OK** to confirm that you want to restart the manager node.

   What happens next depends on whether the Oracle Container Cloud Service instance is running when you perform the operation:

   - If the instance is running, the manager node and all running worker nodes are first stopped. Then the manager node is started, followed by all the worker nodes. The instance is returned to a running state.
   - If the instance is not running, only the manager node is started. You have to start the worker nodes individually.

Stopping, Starting, and Restarting Worker Nodes

You use the Oracle Container Cloud Service Console to stop, start, and restart the worker nodes in an Oracle Container Cloud Service instance.

Worker nodes are implicitly stopped, started, and restarted when you stop, start, and restart Oracle Container Cloud Service instances. When you:

- Stop a running instance, the manager node and all worker nodes in the instance are stopped. You cannot start worker nodes while the manager node is stopped.
- Start a stopped instance, the manager node and all worker nodes in the instance are started.
- Restart a running instance, the manager node and all worker nodes in the instance are stopped, and then started.

Whether you can explicitly stop, start, and restart individual worker nodes depends on:

- Whether the worker node is the first of the original worker nodes defined and created when the instance itself was first created, or whether the worker node is a second (or subsequent) original worker node or an additional worker node that was added to the instance later.
- Whether the instance and/or manager node is currently running. If the instance and/or manager node is currently stopped, you cannot stop, start, or restart any worker nodes. If the instance and/or manager node is currently running:
  - you can restart the first of the original worker nodes (this node usually has a name that ends with "-occs-wkr-1")
  - you can stop, start, and restart other worker nodes, provided the first of the original worker nodes is already running

> **Note:**
>
> Before you stop a worker node, it is generally good practice to first use the Oracle Container Cloud Service Container Console to stop any deployments that are running on that worker node.
>
> In particular, note that when you stop a worker node, any deployments currently running on the worker node are restarted on the remaining nodes in the resource pool according to the service's orchestration policy (see Creating a Service with Oracle Container Cloud Service). If you don't want deployments restarted on other nodes in the resource pool, use the Oracle Container Cloud Service Container Console to stop deployments running on the worker node before you stop it.

To explicitly stop, start, or restart individual worker nodes in an Oracle Container Cloud Service instance:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.
2. On the **Services** tab, click the name of the Oracle Container Cloud Service instance in which you want to stop, start, or restart individual worker nodes.
3. Click the Menu icon beside the individual worker node and select **Stop**, **Start**, or **Restart** as appropriate.
4. When prompted, click **OK** to confirm that you want to stop, start, or restart the worker node.

   What happens next depends on whether the Oracle Container Cloud Service instance and/or manager node is running when you perform the operation:

   - If the instance and/or manager node is not running, all worker nodes are already stopped. You cannot start or restart worker nodes. If you attempt to start or restart worker nodes, an error message is shown.
   - If the instance and/or manager node is running:
     - you can restart the first of the original worker nodes (this node usually has a name that ends with "-occs-wkr-1")
     - you can stop, start, and restart other worker nodes, provided the first of the original worker nodes is already running

Enabling and Disabling Secure Shell (SSH) Access to Oracle Container Cloud Service Manager and Worker Nodes

Learn how to connect to Oracle Container Cloud Service manager and worker nodes using SSH, and how to enable and disable SSH access.

Topics

- About SSH Access to Oracle Container Cloud Service Manager and Worker Nodes

- Connecting to Oracle Container Cloud Service Manager and Worker Nodes Through SSH
- Adding Public SSH Keys to Oracle Container Cloud Service Instances
- Removing Public SSH Keys from Oracle Container Cloud Service Manager and Worker Nodes Using SSH

About SSH Access to Oracle Container Cloud Service Manager and Worker Nodes

You have SSH (Secure Shell) access to the manager and worker nodes in an Oracle Container Cloud Service instance to perform a number of administrative tasks.

When you create an Oracle Container Cloud Service instance, you're prompted to enter the public key of an SSH public/private key pair.

Later on, you might want to connect to a manager or worker node from an SSH client (for example, to reset the admin password, to retrieve support logs, or to upload your own signed SSL certificates). By default, port 22 on manager and worker nodes (the port used for SSH access) is open. If you want to connect to the node from an SSH client, you'll have to use the paired private key when logging in.

If you want to connect to a manager or worker node from a machine other than the one where you originally ran the Oracle Container Cloud Service Console to create the Oracle Container Cloud Service instance, the other machine must have access to the original private key (for example, by copying the private key to the other machine).

If the private key that you use to access the manager and worker nodes is lost or gets corrupted, you can add a new public key to the service instance. You might also want to add a new public key to a service instance to comply with your organization's security policies or regulations. When you add a new public key to a service instance:

- The new key is appended to any existing public keys in the /.ssh/authorized_keys file on the instance's manager and worker nodes. Existing public SSH keys can still be used to connect to the manager and worker nodes.
- All the VMs in the service instance are restarted.

To connect to a manager or worker node using SSH and the new public key, the machine you're connecting from must have access to the private key that is paired with the new public key.

To prevent a particular public SSH key from being used to gain SSH access to a manager or worker node, you remove the public key from the /.ssh/authorized_keys file on the node.

Connecting to Oracle Container Cloud Service Manager and Worker Nodes Through SSH

To perform administrative tasks (for example, to reset the admin password, to retrieve support logs, or to upload your own signed SSL certificates) on an Oracle Container Cloud Service manager or worker node, you use SSH client software to establish a secure connection and log in.

A number of SSH clients are freely available for different platforms, including:

- the ssh utility for UNIX and UNIX-like platforms

- the PuTTY program for Windows

Topics

- Connecting to Manager and Worker Nodes Using the ssh Utility on UNIX
- Connecting to Manager and Worker Nodes Using PuTTY on Windows

Connecting to Manager and Worker Nodes Using the ssh Utility on UNIX

On UNIX and UNIX-like platforms (including Solaris and Linux), you can connect through SSH to Oracle Container Cloud Service manager and worker nodes using the ssh utility (an SSH client) to perform administrative tasks.

Note the instructions below assume the UNIX machine you use to connect to the manager or worker node:

- Has the ssh utility installed.
- Has access to the SSH private key file paired with the SSH public key that was specified when the service instance was created.

To connect to a manager or worker node through SSH from a UNIX machine using the ssh utility:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.

2. On the **Services** tab, click the name of the Oracle Container Cloud Service instance to which you want to connect using the ssh utility.
3. Locate the manager node (the VM for which the **Instance Type** field is set to MANAGER) or worker node to which you want to connect and make a note of the IP address shown in its **Public IP** field. For example, 192.0.2.254.
4. On your UNIX machine, open a command line terminal window.
5. In the terminal window, type `ssh opc@<node_ip_address>` to connect to the manager or worker node, where `<node_ip_address>` is the IP address of the manager or worker node shown on the **Services** tab that you made a note of earlier. For example, `ssh opc@192.0.2.254`

> **Note:**
>
> If the SSH private key is not stored in the file or in the path that the ssh utility expects (for example, the ssh utility might expect the private key to be stored in ~/.ssh/id_rsa), you must explicitly specify the private key filename and location in one of two ways:
>
> - Use the `-i` option to specify the filename and location of the private key. For example, `ssh -i ~/.ssh/my_keys/my_occs_host_key_filename opc@192.0.2.254`
> - Add the private key filename and location to an SSH configuration file, either the client configuration file (~/.ssh/config) if it exists, or the system-wide client configuration file (/etc/ssh/ssh_config). For example, you might add the following:
>
> ```
> Host 192.0.2.254
>     IdentityFile ~/.ssh/my_keys/my_occs_host_key_filename
> ```
>
> For more about the ssh utility's configuration file, type `man ssh_config`
>
> Note also that permissions on the private key file must allow you read/write/execute access, but prevent other users from accessing the file. For example, to set appropriate permissions, you might type `chmod 600 ~/.ssh/my_keys/my_occs_host_key_filename`. If permissions are not set correctly and the private key file is accessible to other users, the ssh utility will simply ignore the private key file.

6. In the terminal window, perform administrative tasks on the manager or worker node using SSH.

   For example, see:

   - Viewing Log Files on Oracle Cloud Container Service Manager and Worker Nodes Using SSH
   - Resetting the Password for an Oracle Container Cloud Service Instance Administrator Using SSH
   - Uploading Your Own SSL Certificates to a Manager Node Using SSH

7. When you're finished, close the SSH connection by typing `exit` in the terminal window.

Connecting to Manager and Worker Nodes Using PuTTY on Windows

On Windows platforms, you can connect through SSH to Oracle Container Cloud Service manager and worker nodes using the PuTTY program (a freely available SSH client) to perform administrative tasks.

Note the instructions below assume the Windows machine you use to connect to the manager or worker node:

- Has the PuTTY program installed.

  If PuTTY is not installed, go to http://www.putty.org/ to download and install it.

- Has access to the SSH private key file paired with the SSH public key that was specified when the service instance was created.

The private key file must in the PuTTY .ppk format. If the private key file was originally created on the Linux platform, use the PuTTYgen program to convert it to the .ppk format.

To connect to a manager or worker node through SSH from a Windows machine using the PuTTY program:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.

2. On the **Services** tab, click the name of the Oracle Container Cloud Service instance to which you want to connect using the ssh utility.

3. Locate the manager node (the VM for which the **Instance Type** field is set to MANAGER) or worker node to which you want to connect and make a note of the IP address shown in its **Public IP** field. For example, 192.0.2.254.

4. On your Windows machine, run the PuTTY program.

   The PuTTY Configuration window is displayed, showing the Session panel.

5. In the **Host Name (or IP address)** box, enter the IP address of the manager or worker node.

6. Confirm that the **Connection type** option is set to **SSH**.

7. In the Category tree, expand **Connection** if necessary and then click **Data**.

   The Data panel is displayed.

8. In the **Auto-login username** box, type `opc`.

9. Confirm that the **When username is not specified** option is set to **Prompt**.

10. In the Category tree, expand **SSH** and then click **Auth**.

    The **Auth** panel is displayed.

11. Click the **Browse** button next to the **Private key file for authentication** box. Then, in the **Select private key file** window, navigate to and open the private key file that matches the public key.

12. In the **Category** tree, click **Session**.

    The **Session** panel is displayed.

13. In the **Saved Sessions** box, enter a name for this connection configuration. Then, click **Save**.

14. Click **Open** to open the connection.

    The PuTTY Configuration window is closed and the PuTTY terminal window is displayed.

    If this is the first time you're connecting to the manager or worker node, the PuTTY **Security Alert** window is displayed, prompting you to confirm the public key. Click **Yes** to continue connecting.

15. In the PuTTY terminal window, perform administrative tasks on the manager or worker node using SSH.

    For example, see:

    - Viewing Log Files on Oracle Cloud Container Service Manager and Worker Nodes Using SSH

- Resetting the Password for an Oracle Container Cloud Service Instance Administrator Using SSH
- Uploading Your Own SSL Certificates to a Manager Node Using SSH

16. When you're finished, close the SSH connection by typing `exit` in the PuTTY terminal window.

Adding Public SSH Keys to Oracle Container Cloud Service Instances

You can add additional public SSH keys to Oracle Container Cloud Service instances in your identity domain using the Oracle Container Cloud Service Console (for example, if you lose the original private key or it gets corrupted).

When you add a new public SSH key, it's appended to any existing public SSH keys in the /.ssh/authorized_keys file on the instance's manager and worker nodes (the existing public SSH keys can still be used). To connect to the manager node or worker nodes using the new public SSH key, the machine from which you're connecting must have access to the private key paired with the new SSH public key.

To add a new SSH public key to an Oracle Container Cloud Service instance:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.

2. Click the Menu icon beside the service instance to which you want to add a new public SSH key, and select **SSH Access**.

   The Add New Key dialog is displayed, showing the value of the most recent SSH public key.

3. Specify the new public key using one of the following methods:
   - Select **Upload a new SSH Public Key value** and click **Browse** to select a file that contains the public key.
   - Select **Key value**, delete the current key value, and paste the new public key into the text area. Make sure the value doesn't contain line breaks or end with a line break.

4. Click **Add New Key**.

   The new public key is added to the /.ssh/authorized_keys file on the manager and worker nodes.

5. When prompted, confirm that you want to restart the VMs for the Oracle Container Cloud Service instance.

Removing Public SSH Keys from Oracle Container Cloud Service Manager and Worker Nodes Using SSH

You can prevent a particular public SSH key from being used to gain SSH access to an Oracle Container Cloud Service instance's manager or worker node by removing the public SSH key from the /.ssh/authorized_keys file on the node.

Note the instructions below assume:

- you know the public SSH key that you want to prevent from accessing the manager or worker node

- the machine you use to connect to the manager or worker node:
  - has an SSH client installed
  - has access to the SSH private key file paired with the SSH public key that was specified when the service instance was created

To prevent a particular public SSH key from being used to gain SSH access to a manager or worker node:

1. Use SSH to connect to the manager or worker node.

   If you're not sure how to do this, see Connecting to Oracle Container Cloud Service Manager and Worker Nodes Through SSH.

2. In the terminal window, navigate to the /.ssh directory on the manager or worker node. For example, by typing:

   ```
   cd /.ssh
   ```

3. Open the authorized_keys file in a text editor.
4. Delete the public SSH key that you want to prevent from being used to gain SSH access to the manager or worker node.
5. Save and close the authorized_keys file.
6. Close the SSH connection by typing `exit` in the terminal window.

Uploading Your Own SSL Certificates to a Manager Node Using SSH

By default, the NGINX web server that runs on the Oracle Container Cloud Service manager node uses self-signed SSL certificates. If you prefer, you can upload your own SSL certificates that have been signed by a Certificate Authority for NGINX to use.

As with certificates signed by a Certificate Authority, self-signed SSL certificates securely encrypt user credentials. However, self-signed SSL certificates cause some browsers to display a connection warning message the first time users go to the Container Console. In the case of Oracle Container Cloud Service, it's fine for users to ignore the warning message. However, you might want to use your own signed SSL certificates with Oracle Container Cloud Service to:

- avoid users seeing the initial security warning
- discourage users from simply ignoring security warnings
- show the secure padlock icon in the browser url field (rather than an insecure icon)
- ensure the Oracle Container Cloud Service REST API is accessed via https

Note the instructions below assume you're using the Bourne shell on a UNIX machine to connect to the manager node, and the machine:

- has the ssh utility installed
- has access to the SSH private key file paired with the SSH public key that was specified when the service instance was created

To upload SSL certificates to a manager node from a UNIX machine using the ssh utility:

1. On the UNIX machine where the SSL certificates currently reside, open a terminal window.

2. Create a new local directory named `/certs` at the root level.
3. Copy the SSL certificate file to the local `/certs` directory. If the SSL certificate file you just copied into the `/certs` directory is not already named `cert.crt`, rename it to `cert.crt` now.
4. Copy the corresponding SSL key file to the local `/certs` directory. If the SSL key file you just copied into the `/certs` directory is not already named `cert.key`, rename it to `cert.key` now.
5. Set the values of two variables called CERT and KEY to the contents of the `/certs/cert.crt` and `/certs/cert.key` files respectively, by typing:

   ```
   export CERT=$(cat certs/cert.crt) && export KEY=$(cat certs/cert.key)
   ```

6. Connect to the manager node using the ssh utility as the default opc user, and create new files called `cert.crt` and `cert.key` in the `/certs` directory on the manager node from the contents of the CERT and KEY variables by typing:

   ```
   ssh -i <private_key> opc@<mgr_node_ip_address> "echo \"${CERT}\" > certs/cert.crt; echo \"${KEY}\" > certs/cert.key"
   ```

   where:

   - `<private_key>` is the full path and name of the file that contains the SSH private key corresponding to the SSH public key associated with the instance that you want to access
   - `<mgr_node_ip_address>` is the IP address of the manager node

   For example:

   ```
   ssh -i ~/.ssh/my_keys/my_private_key_file opc@192.0.2.254 "echo \"${CERT}\" > /certs/cert.crt; echo \"${KEY}\" > /certs/cert.key"
   ```

   For more information about connecting to the manager node using the ssh utility, see Connecting to Manager and Worker Nodes Using the ssh Utility on UNIX

   A script on the manager node regularly scans the /certs directory for new SSL certificate files (approximately once a minute). If it detects new SSL certificate files and determines they are valid, the script copies the certificate files from the root directory to the appropriate subdirectories for use by NGINX.

7. Optional: To confirm that the certificates have been copied successfully and that NGINX is now using them, you can view the messages written to the `certificator-nginx.log` file on the manager node as follows:

   a. Connect to the manager node using the ssh utility as the default opc user by typing:

      ```
      ssh -i <private_key> opc@<mgr_node_ip_address>
      ```

      where:

      - `<private_key>` is the full path and name of the file that contains the SSH private key corresponding to the SSH public key associated with the instance that you want to access
      - `<mgr_node_ip_address>` is the IP address of the manager node

      For example:

      ```
      ssh -i ~/.ssh/my_keys/my_private_key_file opc@192.0.2.254
      ```

b. From the root directory on the manager node, type `cat /log/certificator-nginx.log`

   Messages are written to the `certificator-nginx.log` file every minute. From the log file, you can see that:

   - before you copied your SSL certificates file, NGINX was regularly issuing warning messages like:

     ```
     issuer certificate not found for certificate "/etc/nginx/tls/certs/cert.crt"
     ```

   - Shortly after you copied your SSL certificates to the manager node, the log file shows messages like:

     ```
     copied /u01/data/opc/certs/cert.crt to /etc/nginx/tls/certs
     copied /u01/data/opc/certs/cert.key to /etc/nginx/tls/private
     ```

   - Subsequently, after NGINX has started using the SSL certificates you copied, the log file shows messages like:

     ```
     checking /u01/data/opc/certs
     checked - nginx certificates have not changed
     ```

c. When you're finished, close the SSH connection by typing `exit` in the terminal window.

Changing the Number of Worker Node Hosts in Oracle Container Cloud Service Instances

To improve the performance and efficiency of your Docker environment, you can optimize the number of worker node hosts that are available to run Docker containers in an Oracle Container Cloud Service instance.

You 'scale out' an Oracle Container Cloud Service instance by adding worker node hosts.

You 'scale in' an Oracle Container Cloud Service instance by removing worker node hosts.

Topics

- Adding Worker Node Hosts
- Removing Worker Node Hosts

Adding Worker Node Hosts

You can improve the performance and resilience of your Docker environment by increasing the number of worker node hosts available to run Docker containers in an Oracle Container Cloud Service instance.

To add a new worker node host to an Oracle Container Cloud Service instance:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.

2. On the **Services** tab, click the name of the Oracle Container Cloud Service instance to which you want to add worker node hosts.
3. In the header region of the **Service Overview** page, click the Menu icon beside the name of the service instance, and select **Scale Out**.
4. In the **Scale Out dialog**, specify the number of additional worker node hosts to add (between 1 and 50) and click **Scale Out**.

   A message confirms the service scale out request has been accepted, and a process begins to create the new worker node hosts you requested.
5. Refresh the Oracle Container Cloud Service Console page periodically, until the new worker node host appears in the **OCCS Worker Component - Resources** list.

   Typically, it takes around ten minutes to create a new worker node host.

   Worker node hosts that you create using the **Scale Out dialog** appear in the **OCCS Worker Component - Resources** list with a Menu icon beside them. Worker node hosts that were created when the Oracle Container Cloud Service instance was initially created don't have a Menu icon beside them

   When the new worker node host is shown in the **OCCS Worker Component - Resources** list, the host can be used to run deployed containers by adding it to a resource pool using the Oracle Container Cloud Service Container Console (see Managing Hosts).

Removing Worker Node Hosts

You can reduce your usage of Oracle Compute resources by decreasing the number of worker node hosts available to run Docker containers in an Oracle Container Cloud Service instance.

When you initially create an Oracle Container Cloud Service instance, you specify the number of worker node hosts to create. Later on, you can add more worker node hosts in addition to the worker node hosts you initially specified. If you subsequently decide you no longer need the additional worker node hosts, you can remove them. Note that you can only remove the additional worker node hosts. You can't remove the worker node hosts that were initially created.

> **Note:**
>
> When you remove a worker node host, any deployments currently running on the worker node host are restarted on the remaining hosts in the resource pool according to the service's orchestration policy (see Creating a Service with Oracle Container Cloud Service). If you don't want deployments restarted on other hosts in the resource pool, use the Oracle Container Cloud Service Container Console to stop deployments running on the worker node host before you remove it.

To remove a worker node host from an Oracle Container Cloud Service instance:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.

2. On the **Services** tab, click the name of the Oracle Container Cloud Service instance from which you want to remove worker node hosts.
3. On the **Service Overview** page, expand the **OCCS Worker Component** region to see the worker node hosts in the **Resources** list.
4. Click the Menu icon beside the worker node host that you want to remove, and select **Remove Node**.

   Note that the Menu icon only appears beside worker node hosts that have been added to the Oracle Container Cloud Service instance after it was initially created. Worker node hosts that do not have a Menu icon beside them were created when the Oracle Container Cloud Service instance was initially created and can't be removed.
5. When prompted, click **Remove Node** to confirm that you want to remove the node from the service instance.

   A message confirms the service scale in request has been accepted, and a process begins to remove the worker node host you selected. Any deployments currently running on the worker node host are redistributed between other available hosts in the resource pool according to the service's orchestration policy (see Creating a Service with Oracle Container Cloud Service).
6. Refresh the Oracle Container Cloud Service Console page periodically, until the worker node host no longer appears in the **Resources** list.

   Typically, it takes around ten minutes to remove a worker node host.

Managing Access Rules for Oracle Container Cloud Service Instances

You can control access to an Oracle Container Cloud Service instance by creating and managing access rules using the Oracle Container Cloud Service Console.

Access rules enable you to control access to the virtual machines (VMs) that make up a service instance. When you create a service instance, the system automatically creates and enables all the access rules you'll need for Oracle Container Cloud Service. For example:

- access from the public internet to the manager node VM on port 22
- access from the public internet to worker node VMs on all ports (ports 1 to 65535)

Since the necessary access rules have already been created for you, you probably won't need to change them. However, if you do want to change the access rules (for example, to explicitly restrict access to worker nodes to particular ports), you can use the Oracle Container Cloud Service Console to disable the default rules and create new rules.

To create a new access rule for an Oracle Container Cloud Service instance:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.

2. Click the Menu icon beside the service instance to which you want to add an access rule and select **Access Rules**. The **Access Rules** page is displayed, showing the list of all access rules.
3. Click **Create Rule** to display the **Create Access Rule** dialog.
4. Specify a unique name for the access rule.

   The name must begin with a letter, and can contain numbers, hyphens, or underscores. The name mustn't be longer than 50 characters. When you create a rule, you can't use the prefixes `ora_` or `sys_`.
5. Optional: Specify a description of the rule.
6. Specify a source for the rule:

Source option:	Use to permit access from:
MANAGER_ADMIN_HOST	The host running as the Oracle Container Cloud Service manager node.
MANAGER_MANAGER	The host running as the Oracle Container Cloud Service manager node.
PUBLIC-INTERNET	Any host on the internet
WORKER_ADMIN_HOST	The first host in the list of Oracle Container Cloud Service worker nodes shown on the **Service Details** page.
WORKER_WORKER	Any host running as an Oracle Container Cloud Service worker node.
<custom>	A list of IP addresses from which to permit traffic. In the field that displays when you select this option, enter a comma-separated list of the subnets (in CIDR format, such as 192.0.2.254/24) or IPv4 addresses from which you want to permit access.

7. Specify a destination for the rule:

Destination option:	Use to permit access to:
MANAGER_ADMIN_HOST	The host running as the Oracle Container Cloud Service manager node.
MANAGER_MANAGER	The host running as the Oracle Container Cloud Service manager node.
WORKER_ADMIN_HOST	The first host in the list of Oracle Container Cloud Service worker nodes shown on the **Service Details** page.
WORKER_WORKER	Host running as Oracle Container Cloud Service worker node.

   The source and the destination must be different.
8. Specify a port or ports through which the source will access the destination.

   You can specify a single port or a range of ports (for example, 7001–8001).
9. Click **Create** to create and enable the rule.
10. Optional: You can later disable, re-enable, or delete access rules on the **Access Rules** page, by clicking the Menu icon beside a rule and choosing the appropriate option. The options available depend on the type of rule and its current status:

Rule Type:	Can be enabled?	Can be disabled?	Can be deleted?
USER	Yes	Yes	Yes
DEFAULT	Yes	Yes	No
SYSTEM	No	No	No

Icons in the **Status** column indicate whether an access rule is enabled or disabled:

Icon	Indicates access rule is:
	Enabled
	Disabled

Viewing Activity for Oracle Container Cloud Service Instances

You can view the activities of Oracle Container Cloud Service instances in your identity domain using the Oracle Container Cloud Service Console **Activity** page.

To view activities of Oracle Container Cloud Service instances:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.

2. You can view activity for all instances in the identity domain, or just for a particular instance:

   - Click the **Activity** tab to view activity for all instances in the identity domain. Use the options in the **Search Activity Log** section to filter the results to meet your needs.
   - Click the name of the instance on the **Services** tab to view activity for a particular instance. On the **Instance Details** page, click the **Activity** link to see activities of that instance.

> **Tip:**
>
> If you only want to see information related specifically to instance creation or deletion, click the **Service Create and Delete History** link on the **Services** tab to see instance creation or deletion activities.

Viewing Log Files on Oracle Cloud Container Service Manager and Worker Nodes Using SSH

You can view the log files on the manager node and worker nodes in an Oracle Container Cloud Service instance using SSH (for example, for support purposes).

Three different Oracle Container Cloud Service components save log files on the manager node:

- the Cluster Manager (which handles communication between the manager node and worker nodes) saves log files named occs-cluster-manager.log

- the Service Manager (which handles API calls and the parsing of data structures) saves files named occs-data-manager.log
- the Data Manager (which manages the local Oracle Container Cloud Service database) saves files named occs-service-manager.log

The Oracle Container Cloud Service Cluster Agent component runs on worker nodes and saves files named occs-cluster-agent.log.

Note the instructions below assume the machine you use to connect to the manager or worker node:

- has an SSH client installed
- has access to the SSH private key paired with the SSH public key that was specified when the service instance was created

To view the log file on a manager or worker node using SSH:

1. Use SSH to connect to the manager or worker node of the Oracle Container Cloud Service instance for which you want to view the log files.

   If you're not sure how to do this, see Connecting to Oracle Container Cloud Service Manager and Worker Nodes Through SSH.

2. In the terminal window, navigate to the /var/log/occs directory on the manager or worker node. For example, by typing:

   ```
   cd /var/log/occs
   ```

   This directory contains the Oracle Container Cloud Service log files.

3. In the terminal window, list the available log files. For example, by typing:

   ```
   ls -al
   ```

   You'll see output similar to this:

   ```
   drwxr-xr-x  2 501 501     4096 Sep 12 07:46 .
   drwxr-xr-x. 7   0   0     4096 Sep 11 17:30 ..
   -rw-r--r--  1 501 501  4395961 Sep 12 16:19 occs-cluster-manager.log
   -rw-r--r--  1 501 501  9007241 Sep 12 16:19 occs-data-manager.log
   -rw-r--r--  1 501 501   390113 Sep 12 16:19 occs-service-manager.log
   ```

4. Decide the log file you want to look at, and open it using a file-viewing program. For example, to view the occs-cluster-manager.log file using the `more` command, type:

   ```
   more occs-cluster-manager.log
   ```

5. When you've finished looking at the log file, quit the file-viewing program without making any changes. For example, if you used the `more` command to view the file, type `q` to exit.
6. Close the SSH connection by typing `exit` in the terminal window.

Changing the Username or Password for an Oracle Container Cloud Service Instance Administrator

Having specified a username and password for the instance administrator when you created an Oracle Container Cloud Service instance, you can change either or both later if you need to.

The instance administrator's username and password are used to log into the Oracle Container Cloud Service Container Console.

To change the instance administrator's username and/or password using the Oracle Container Cloud Service Container Console:

1. Sign in to the Container Console as the instance administrator.

   If you're not sure how to do this, see Accessing the Container Console for Oracle Container Cloud Service .

2. In the Container Console, select **My Profile** from the **Settings** menu and enter either or both:

   - a new username in the **Username** field
   - a new password in the **Password** field

   > **Note:**
   >
   > If you change the username, note the following:
   >
   > - Make sure you take a note of the new username because there's no way to recover or reset it later.
   > - When you click **Save**, the Container Console session ends immediately and you're prompted to re-enter the username and password.
   > - A new API token value is generated, so you'll have to update any scripts that included the old API token.

3. Click **Save** to apply the changes.

   If you changed the username, the Container Console session ends immediately and you're prompted to re-enter the username and password.

Resetting the Password for an Oracle Container Cloud Service Instance Administrator Using SSH

You can change an Oracle Container Cloud Service instance administrator's password using SSH rather than using the Oracle Container Cloud Service Container Console.

Having specified an instance administrator's username and password when you created an Oracle Container Cloud Service instance, you can normally change either or both later by logging into the Container Console.

However, if you don't know the instance administrator's password (perhaps because another person has changed the password and subsequently left the company), you

won't be able to log into the Container Console to reset it. In this situation, provided you know the instance administrator's username, the solution is to use SSH to log into the manager node and change the password.

Note the instructions below assume:

- You know the instance administrator's current username. When creating a new instance, 'admin' is suggested as the administrator username, but a different username can be entered. Even if 'admin' was originally specified as the instance administrator's username, the username can also be changed later on the **My Profile** page of the Container Console. You must know the instance administrator's current username. If you don't, you won't be able to reset the administrator's password.
- The machine you use to connect to the manager node:
  - has an SSH client installed
  - has access to the SSH private key paired with the SSH public key that was specified when the service instance was created

To change the instance administrator's password using SSH:

1. Use SSH to connect to the manager node of the Oracle Container Cloud Service instance for which you want to change the instance administrator's password.

   If you're not sure how to do this, see Connecting to Oracle Container Cloud Service Manager and Worker Nodes Through SSH.
2. In the terminal window, type `change-admin-password.sh <admin-username>` where `<admin-username>` is the current username of the instance administrator.
3. In the terminal window, when prompted for a password, enter the new password.
4. Close the SSH connection by typing `exit` in the terminal window.
5. To verify the password has changed:
   a. Go back to the Oracle Container Cloud Service Console **Services** tab.
   b. Click the Menu icon beside the service instance, and choose **Container Console**.
   c. On the **Login** page, enter the instance administrator's username and the new password you've just specified.

   Assuming you successfully changed the instance administrator's password, the Container Console is displayed.

Backing Up and Restoring Oracle Container Cloud Service Instances

Learn about how to back up and restore Oracle Container Cloud Service instances.

Topics

- Backing Up an Oracle Container Cloud Service Instance
- Restoring an Oracle Container Cloud Service Instance

Backing Up an Oracle Container Cloud Service Instance

To avoid data loss as a result of hardware failure, file corruption, or accidental file deletion, it's always good practice to back up Oracle Container Cloud Service instances regularly.

When you back up an Oracle Container Cloud Service instance, you're taking a copy of configuration information about:

- deployments
- registries
- services
- stacks

You might back up an instance regularly as part of a disaster recovery policy. It's also good practice to take a backup of the current state of an Oracle Container Cloud Service instance before restoring from an earlier backup file, and especially before deleting an instance. And you can also use back up (and restore) as a way to preserve instance configuration information when moving from a trial subscription to a paid subscription.

In addition, backing up an instance is a mandatory step when you upgrade to a new version of Oracle Container Cloud Service.

To back up an Oracle Container Cloud Service instance using the Oracle Container Cloud Service Container Console:

1. Sign in to the Container Console.

   If you're not sure how to do this, see Accessing the Container Console for Oracle Container Cloud Service .
2. In the Container Console, select **Backup/Restore** from the **Settings** menu.
3. Click **Download a Backup Image**, select **Save File**, and click **OK**.
4. Specify a name and location for the backup file, and click **Save**.

The backup file is saved with the name and in the location that you specified.

If you're taking a backup as part of upgrading an instance, avoid making changes to the instance until you've completed the upgrade process. Any changes you do make will be lost. See Upgrading Oracle Container Cloud Service Instances.

Restoring an Oracle Container Cloud Service Instance

You can restore an Oracle Container Cloud Service instance to the state saved in a backup file.

When you back up an Oracle Container Cloud Service instance, you're taking a copy of configuration information about:

- deployments
- registries
- services
- stacks

See Backing Up an Oracle Container Cloud Service Instance.

You might restore an instance from a backup file to recover from hardware failure, file corruption, or accidental file deletion. And you can also use back up (and restore) as a way to preserve instance configuration information when moving from a trial subscription to a paid subscription.

In addition, restoring from a backup file into a new instance is a mandatory step when you upgrade to a new version of Oracle Container Cloud Service.

Note that when you restore an existing instance from a backup file, the current state of the instance is completely replaced by the contents of the backup file. Because there's no Undo option, it's therefore a good idea to take a backup of the current state of the instance immediately before restoring from the backup file. That way, you can roll back the changes if restoring the instance from the backup file doesn't progress as you expected.

To restore an Oracle Container Cloud Service instance from a backup file:

1. **Recommended:** Before restoring the instance from an existing backup file, save the current state of the instance to a new backup file (see Backing Up an Oracle Container Cloud Service Instance).
2. Sign in to the Oracle Container Cloud Service Container Console.

   If you're not sure how to do this, see Accessing the Container Console for Oracle Container Cloud Service .
3. In the Container Console, select **Backup/Restore** from the **Settings** menu.
4. On the **Backup/Restore** page, click **Choose File**, specify the name and location of the backup file from which you want to restore the instance, and click **Open**.

   > **Tip:**
   >
   > If you prefer, you can also drag and drop the backup file from another window onto the area indicated on the **Backup/Restore** page.
5. Click **Restore**.

The instance is restored to the state saved in the backup file, and the Container Console **Dashboard** page is displayed.

Upgrading Oracle Container Cloud Service Instances

When you're notified that a new version of Oracle Container Cloud Service has been released, you'll probably want to upgrade existing Oracle Container Cloud Service instances to take advantage of enhancements and bug fixes in the new version.

To upgrade an Oracle Container Cloud Service instance to a new version:

1. Save the current state of the existing instance to a new backup file (see Backing Up an Oracle Container Cloud Service Instance).

   Avoid making changes to the instance after taking the backup. Any changes you do make will be lost.
2. On the Oracle Container Cloud Service Console **Services** tab, create a new instance (see Creating Oracle Container Cloud Service Instances).

3. Sign in to the new instance using the Oracle Container Cloud Service Container Console.

   If you're not sure how to do this, see Accessing the Container Console for Oracle Container Cloud Service .

4. In the Container Console for the new instance, select **Backup/Restore** from the **Settings** menu.

5. On the **Backup/Restore** page, click **Choose File**, specify the name and location of the backup file that you want to restore into the new instance, and click **Open**.

   > **Tip:**
   >
   > If you prefer, you can also drag and drop the backup file from another window onto the area indicated on the **Backup/Restore** page.

6. Click **Restore**.

   The new instance is restored to the state saved in the backup file, and the Container Console **Dashboard** page is displayed.

7. Test the new instance to verify that the upgrade has progressed as expected.

8. When you're satisfied that the new instance is performing as expected, delete the old instance (see Deleting Oracle Container Cloud Service Instances).

Deleting Oracle Container Cloud Service Instances

When you no longer require an Oracle Container Cloud Service instance, you can delete it. Your account is no longer charged for the instance.

> **Tip:**
>
> When you delete an Oracle Container Cloud Service instance, all the configuration information held in the instance (for example, service and stack definitions, entries in the Service Discovery database) is permanently deleted. It's therefore a really good idea to take a backup of the instance before you delete it, just in case you need to retrieve the information later (see Backing Up an Oracle Container Cloud Service Instance).

To delete an Oracle Container Cloud Service instance:

1. Navigate to the Oracle Container Cloud Service Console **Services** tab.

   If you're not sure how to do this, see Accessing the Service Console for Oracle Container Cloud Service.

2. Click the Menu icon beside the service instance that you want to delete and select **Delete**.

3. When prompted, confirm that you want to delete the Oracle Container Cloud Service instance.

The instance you've deleted no longer appears in the Service Console.

3

Managing Your Docker Environment with Oracle Container Cloud Service

Learn how to manage and monitor your Docker environment using Oracle Container Cloud Service.

Topics

- Typical Workflow for Managing and Monitoring Your Docker Environment with Oracle Container Cloud Service
- Accessing the Container Console for Oracle Container Cloud Service
- A Quick Tour of the Container Console
- Using the Oracle Container Cloud Service Dashboard
- Using the Oracle Container Cloud Service Quick Start Wizard
- Searching Oracle Container Cloud Service
- Managing Resource Pools with Oracle Container Cloud Service
- Managing Hosts with Oracle Container Cloud Service
- Managing Services with Oracle Container Cloud Service
- Managing Application Stacks with Oracle Container Cloud Service
- Managing Docker Containers with Oracle Container Cloud Service
- Managing Docker Images with Oracle Container Cloud Service
- Managing Deployments with Oracle Container Cloud Service
- Managing Docker Registries with Oracle Container Cloud Service
- Managing Tags with Oracle Container Cloud Service
- Monitoring Tasks, Events, and Status Updates with Oracle Container Cloud Service
- Managing Profile Settings with Oracle Container Cloud Service
- Using Template Functions and Arguments with Oracle Container Cloud Service
- Using the Oracle Container Cloud Service REST API

Typical Workflow for Managing and Monitoring Your Docker Environment with Oracle Container Cloud Service

Here's a typical workflow showing the tasks you'll usually perform to manage your Docker environment using Oracle Container Cloud Service.

You can perform the tasks using the Oracle Container Cloud Service Container Console or the Oracle Container Cloud Service REST API.

Task	Description	More Information
Prepare Oracle Container Cloud Service to manage your Docker environment	• Organize hosts into resource pools • Define the services and stacks to deploy • (optional) Create tags • (optional) Link containers	Creating a Resource Pool Creating a Service with Oracle Container Cloud Service Creating a Stack Creating and Assigning Tags Managing Entries in the Service Discovery Database to Enable Container Communication
Launch containers in the Docker environment	• Deploy services and stacks as Docker containers • (optional) Start containers directly from images	Deploying a Stack Deploying a Service with Oracle Container Cloud Service Starting a Docker Container Directly from a Docker Image
(optional) Manage the Docker environment	• Manage hosts, resource pools, deployments, services, stacks, and containers • Stop and restart Docker containers • Move hosts between resource pools • Change the number of containers used by a running deployment	Managing Resource Pools Managing Hosts Managing Services with Oracle Container Cloud Service Managing Application Stacks with Oracle Container Cloud Service Managing Docker Containers Managing Docker Images Managing Deployments Managing Docker Registries with Oracle Container Cloud Service Stopping and Re-starting Service and Stack Deployments Moving Hosts Between Resource Pools Scaling a Running Deployment with Oracle Container Cloud Service
(optional) Monitor the Docker environment	• Monitor tasks, events, and status updates • View container logs • Monitor deployment health	Monitoring Tasks, Events, and Status Updates Viewing the Log Files of a Running Container Monitoring the Health of a Deployment

Accessing the Container Console for Oracle Container Cloud Service

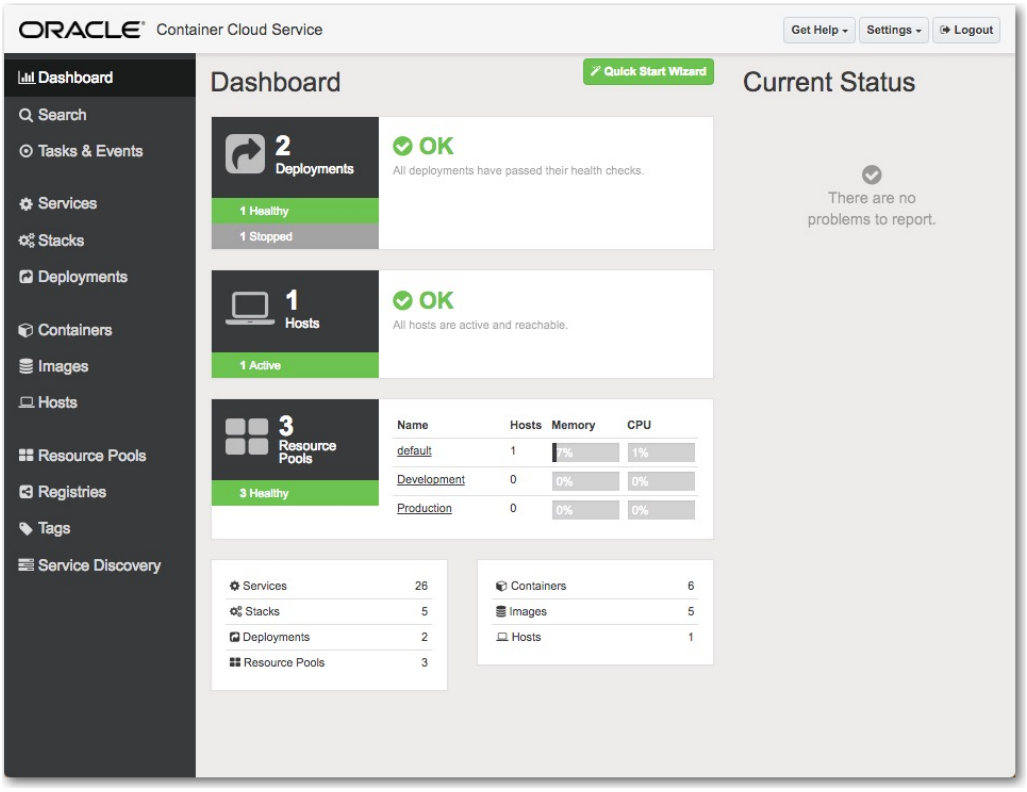
If you're responsible for managing your Docker environment using Oracle Container Cloud Service, you'll be using the Container Console.

To access the Oracle Container Cloud Service Container Console:

1. Sign in to the Container Console at the URL and using the credentials you've received from your administrator.

> **Tip:**
>
> If you're the service administrator, you can also access the Container Console straight from Oracle Container Cloud Service's Service Console. On the **Service Instance Summary** page or on a **Service Instance Details** page, click the Menu icon and choose **Container Console**.

When you sign in, you see the Container Console **Dashboard** page.



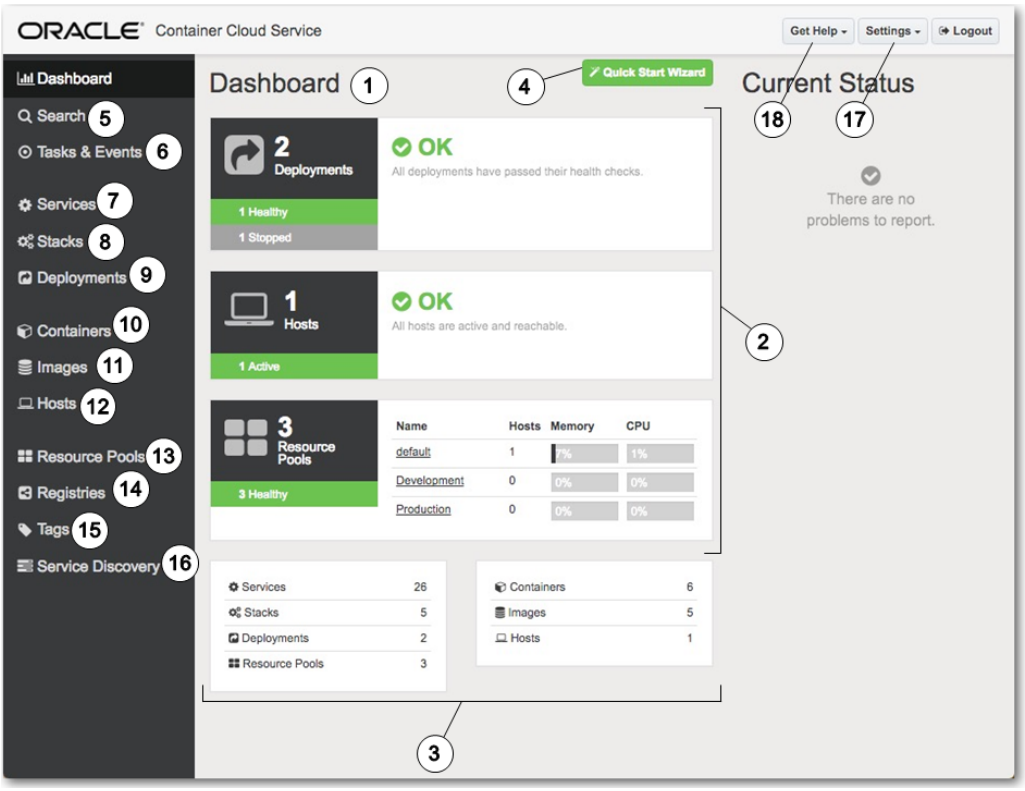
2. On the Dashboard page, review the status of deployments, hosts, resource pools, and containers in the Docker environment.

For more information, see Using the Oracle Container Cloud Service Dashboard

A Quick Tour of the Container Console

Oracle Container Cloud Service's Container Console provides graphical dashboards, editors, and views for you to manage the images, containers, and registries in your Docker environment. Take a minute to learn how and where to find what you need.

Here's what you see on your Container Console Dashboard page when you get started:



The **Dashboard page (1)** contains:

- Detailed health check panels for **Deployments, Hosts, and Resource Pools (2)** displaying their status.
- Summary panels for **Services, Stacks, Deployments, Resource Pools, Containers, Images, and Hosts (3)** showing the number of each type of object currently being managed by Oracle Container Cloud Service

Click **Quick Start Wizard (4)** for a shortcut way to set up and deploy services and stacks as Docker containers.

Options on the left-hand navigation panel get you access to the different Container Console pages:

- Use the **Search page (5)** to locate any of the objects managed by Oracle Container Cloud Service by searching on their name.

- Use the **Tasks & Events page (6)** to monitor the Docker environment by viewing tasks, events, and status updates.
- Use the **Services page (7)** and the **Stacks page (8)** to create new services and stacks of services, and to deploy and manage the services and stacks you've created as well as the pre-configured services and stacks that come with Oracle Container Cloud Service.
- Use the **Deployments page (9)** to manage and monitor deployed services and stacks, scale a running deployment, and stop and re-start deployments.
- Use the **Containers page (10)** to manage and monitor the Docker containers started when you deploy a service or stack, and to view the log files of running containers.
- Use the **Images page (11)** to manage Docker images that have been pulled from public or private Docker registries for a deployment, and to start containers directly from images.
- Use the **Hosts page (12)** to monitor the status of Oracle Compute virtual machines ('worker nodes') running containers for services and stacks you've deployed, and to move hosts between resource pools.
- Use the **Resource Pools page (13)** to organize hosts and combine them into isolated groups of compute resources.
- Use the **Registries page (14)** to define public or private Docker registries from which you want Oracle Container Cloud Service to pull Docker images.
- Use the **Tags page (15)** to categorize and organize resource pools and the hosts within them.
- Use the **Service Discovery page (16)** to view, update, and add service DNS information in the Service Discovery database to enable running containers to communicate with each other.

The branding bar appears at the top of every Container Console page:

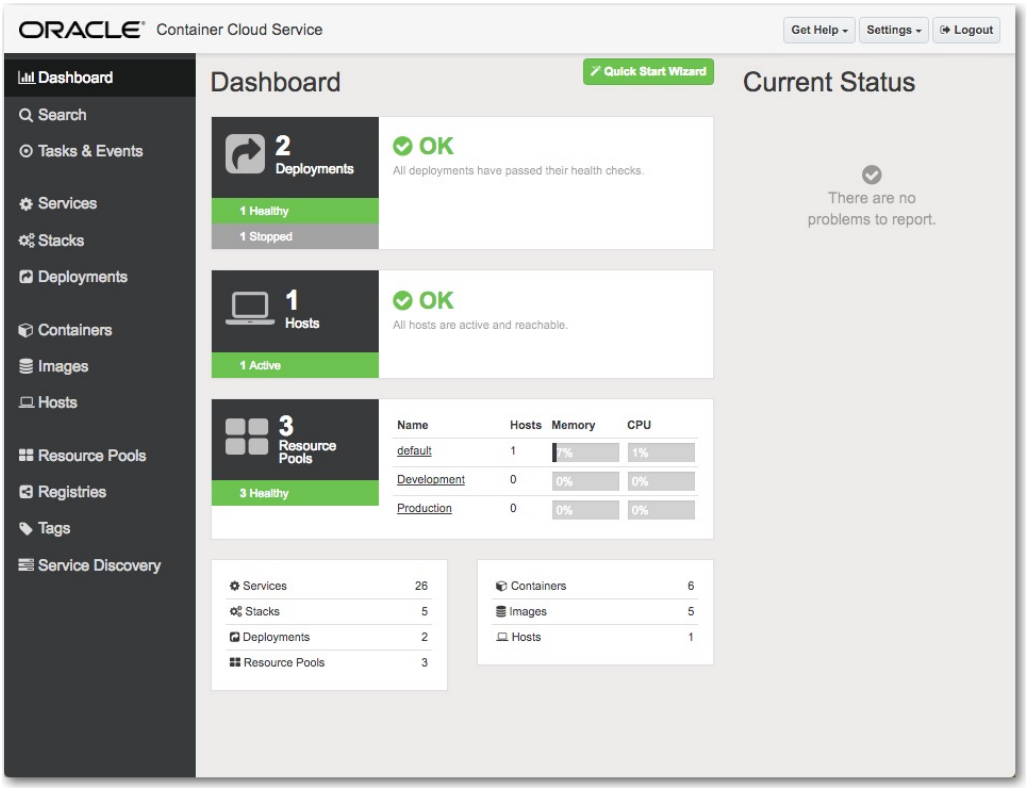
- Click **Settings (17)** to change your profile and password, backup the current configuration to a file, and restore the configuration from a previous backup file.
- Click **Get Help (18)** to access resources to assist you with your current task, including videos, step-by-step tutorials, and context-sensitive help on the current Container Console page.

Using the Oracle Container Cloud Service Dashboard

When you sign in to Oracle Container Cloud Service, you see the Container Console **Dashboard** page.

To get an overview of your Docker environment and manage it using Oracle Container Cloud Service:

1. On the **Dashboard** page of the Container Console , review the status of deployments, hosts, resource pools, and containers in the Docker environment.



2. Manage your Docker environment by performing tasks such as:
   - Managing Resource Pools
   - Managing Hosts
   - Managing Services
   - Managing Stacks
   - Managing Docker Containers
   - Managing Deployments

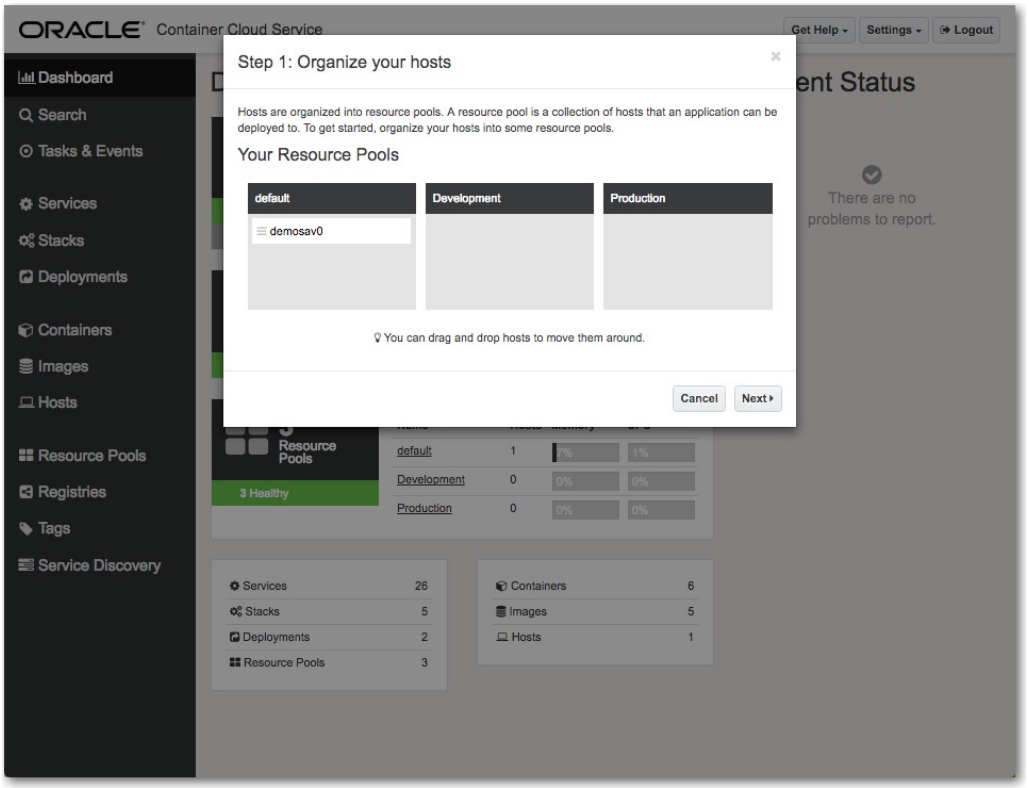
Using the Oracle Container Cloud Service Quick Start Wizard

Find out how to use the Oracle Container Cloud Service Quick Start Wizard to set up and deploy services (based on a single Docker image) and stacks of services (based on multiple different Docker images) as Docker containers.

To set up and deploy services and stacks as Docker containers using the Quick Start Wizard:

1. Click **Quick Start Wizard** on the **Dashboard** page, the **Services** page, or the **Stacks** page of the Oracle Container Cloud Service Container Console.

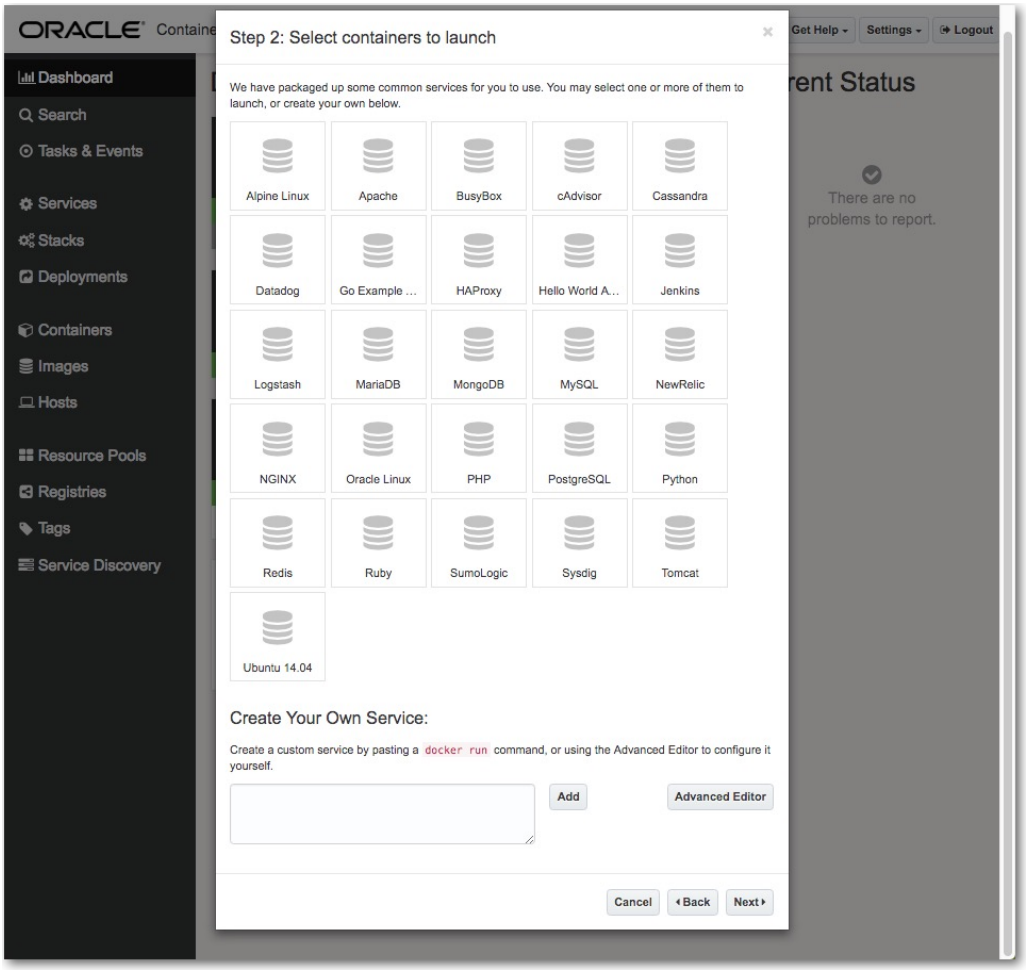
   Step 1 of the Quick Start Wizard enables you to organize the hosts on which to deploy services and stacks.



Grouping hosts into resource pools makes it easier to manage and distribute load between them. For example, you might want a small number of hosts in the resource pool that developers are going to use, a much larger number of hosts in a resource pool devoted to scalability testing, and even more hosts in a resource pool for the production system. See About Resource Pools and About Hosts.

2. Arrange hosts to meet your needs by dragging them from one resource pool to another, and click **Next**.

   Step 2 of the Quick Start Wizard enables you to specify the services and stacks that you want to deploy.

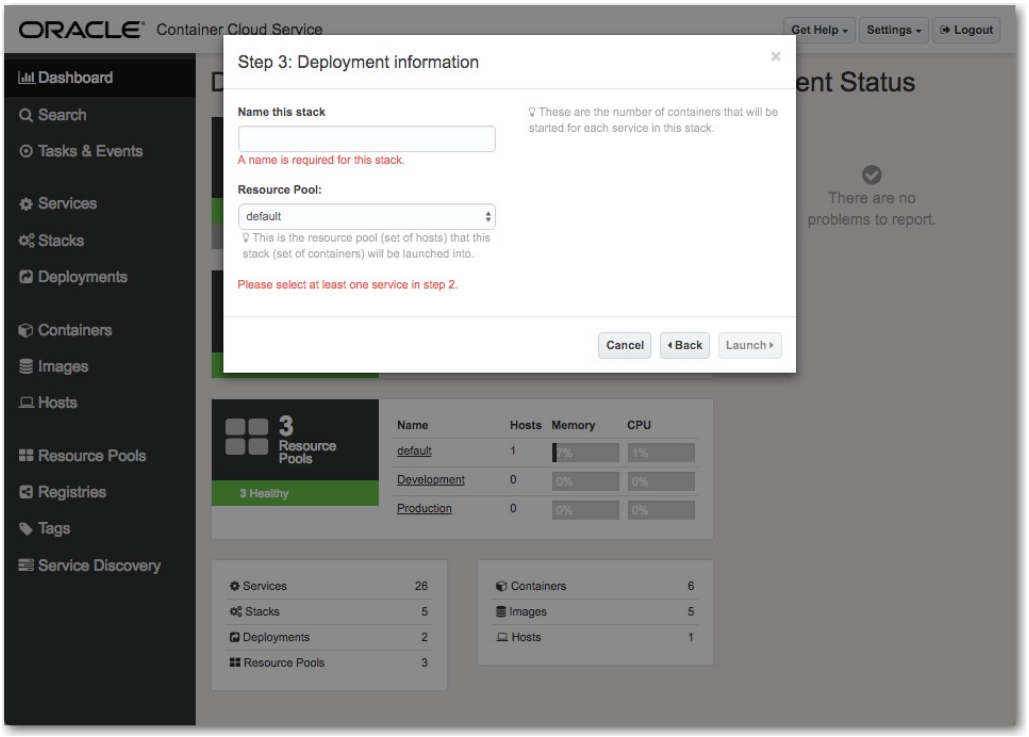


You can choose from services shipped with Oracle Container Cloud Service, paste a `docker run` command, or define a service. See About Services About Stacks

3. Specify the services and stacks to deploy in one of the following ways, and then click **Next**:

   - Select from the services and stacks shipped with Oracle Container Cloud Service that you want to deploy.
   - Paste a `docker run` command in the text box to create a service, and click **Add**.
   - Click **Advanced Editor** and define a new service in the Service Editor (to find out more about the options in the Service Editor, see Creating a Service with Oracle Container Cloud Service).

   Step 3 of the Quick Start Wizard enables you to specify where and how to deploy the service or stack.



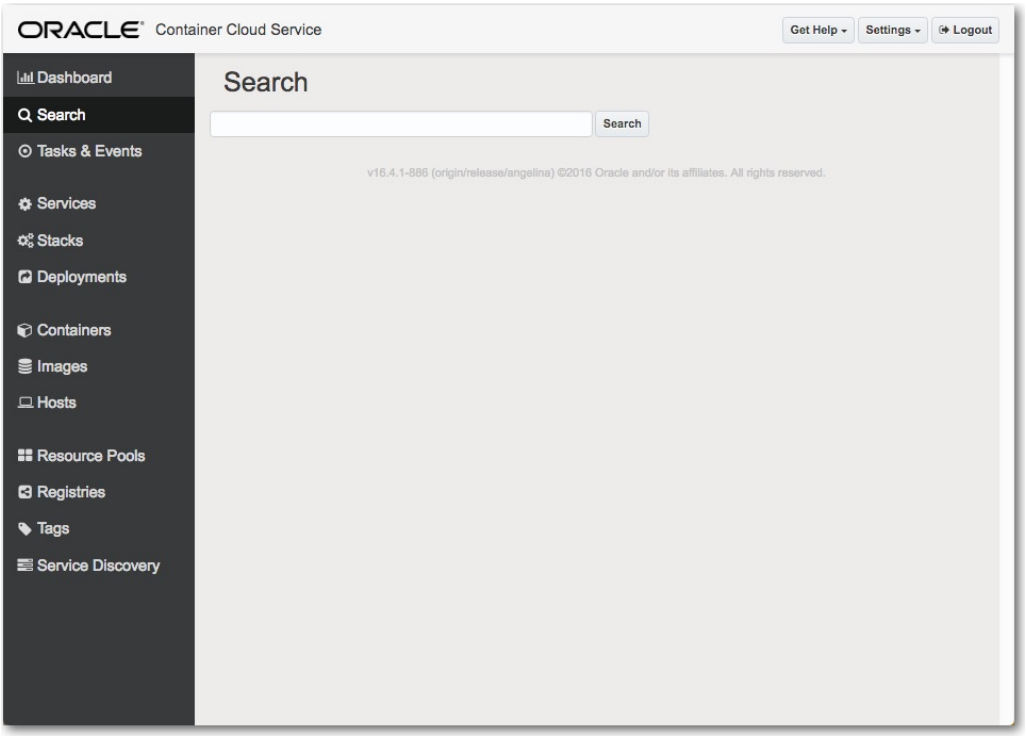
When you deploy a service or stack, Docker containers are launched on hosts in the resource pool you specify. See About Docker Containers and About Deployments.

4. Specify where and how to deploy the service or stack:
   a. Specify a name for the service or stack being deployed.
   b. Specify the resource pool containing the hosts that you want to run the containers for the service or stack.
   c. Specify the number of containers to start for each service in a stack.
   d. Click **Launch**.

The service, or the services in the stack, are deployed as Docker containers in the resource pool and on the number of hosts that you specified.

Searching Oracle Container Cloud Service

You can locate objects managed by Oracle Container Cloud Service using the Container Console **Search** page.



Use the **Search** page as a quick way to find and manage:

- services
- stacks
- deployments
- containers
- images
- hosts
- resource pools
- registries

Managing Resource Pools with Oracle Container Cloud Service

Learn about resource pools, how to create them, and how to manage them.

Topics

- About Resource Pools
- Managing Resource Pools
- Creating a Resource Pool

About Resource Pools

Resource pools are a way to organize hosts and combine them into isolated groups of compute resources. Resource pools enable you to manage your Docker environment more effectively by deploying services and stacks efficiently across multiple hosts.

Using resource pools, you can logically define locality as well as a grouping for hosts that share a usage or purpose.

It's important to note that a host can only exist in one resource pool at a time.

Oracle Container Cloud Service comes with three resource pools defined out of the box:

- default
- Development
- Production

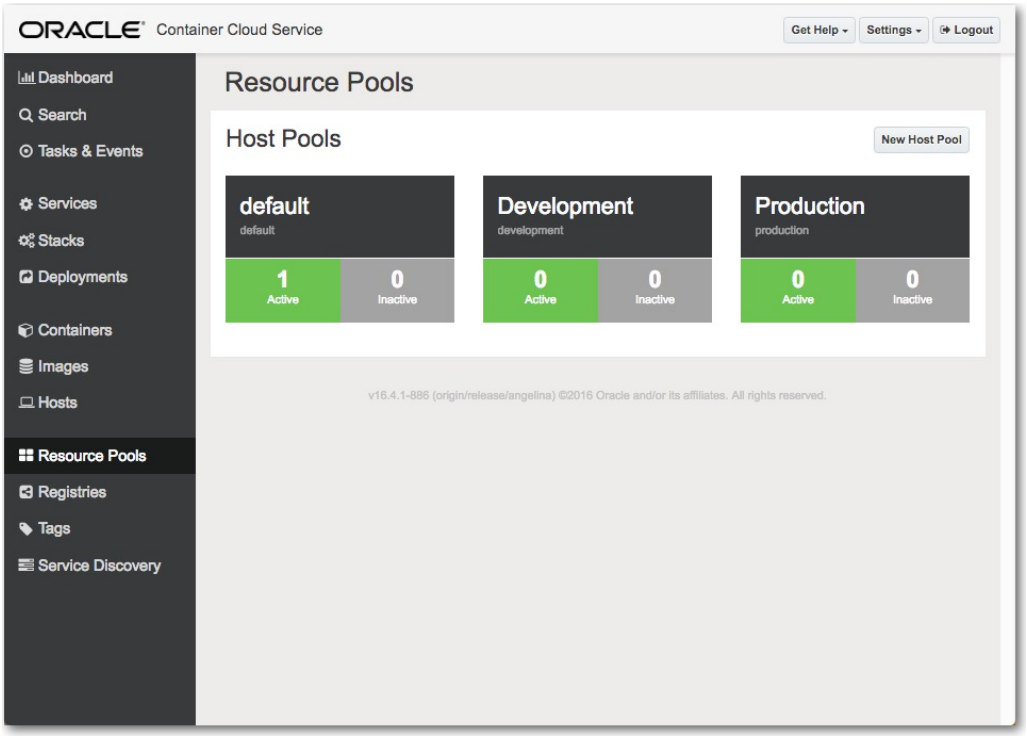
The naming of the default resource pools is arbitrary. You can rename these pools, or delete them and create new ones. The important thing is to use your resource pools to organize your hosts to accommodate your workflow.

Managing Resource Pools

Find out how to manage resource pools to deploy services and stacks efficiently across multiple hosts.

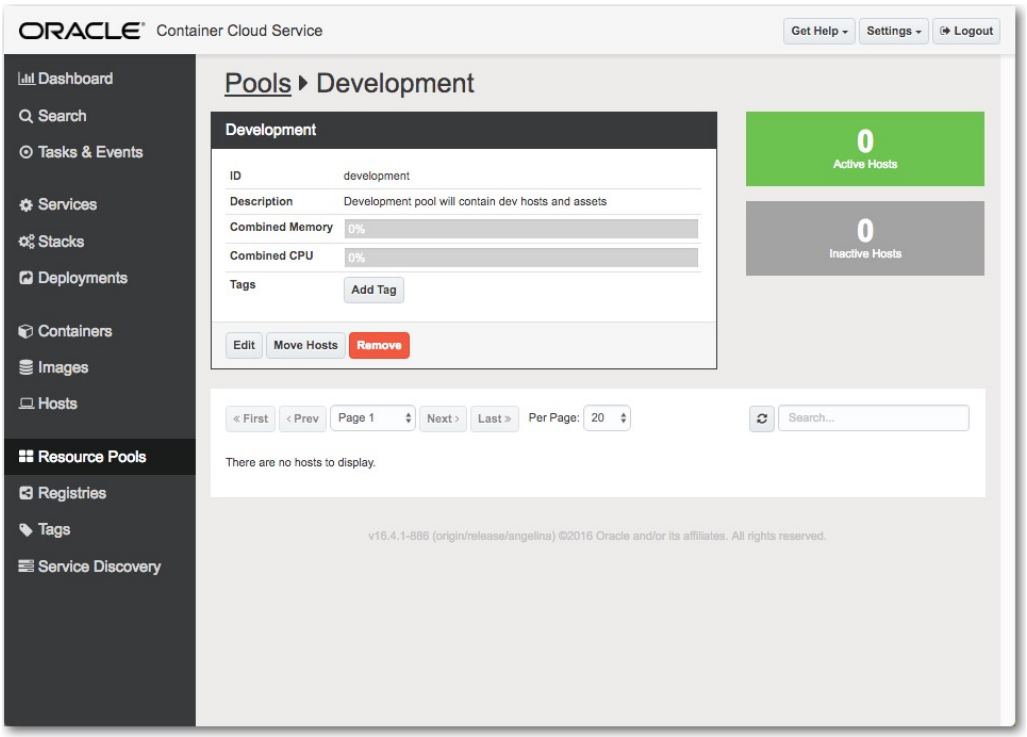
To manage resource pools using the Oracle Container Cloud Service Container Console:

1. On the **Resource Pools** page of the Container Console, review the resource pools currently being managed by Oracle Container Cloud Service.



The **Resource Pools** page gives a quick overview of the current pools as well as the number of active and inactive hosts assigned to each. A host is considered active if the Oracle Container Cloud Service agent running on it can successfully communicate with the manager node.

2. Click on the name of a resource pool to see and change details for that resource pool.



The **Resource Pools Details** page enables you to manage the resource pool, including the following options:

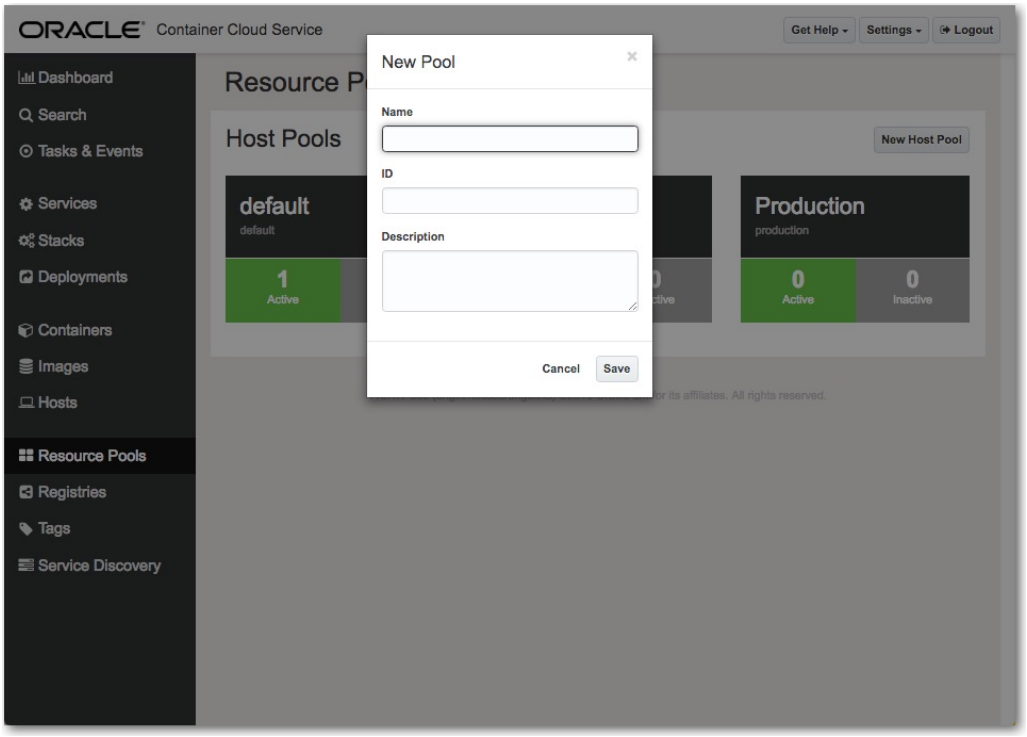
Option	Use to:
Tags	See the tags assigned to the resource pool. Click **Add Tag** to give the resource pool a tag that describes its purpose, or associates the resource pool with other resource pools with similar characteristics. A resource pool can be assigned multiple tags.
Edit	Change the resource pool's name or description.
Move Hosts	Assign one or more hosts to this resource pool (typically from the Default pool).
Remove	Remove the pool (only available if there are no hosts in the pool)
Move Selected to Pool	Move selected hosts to the pool you select. If you move a host from one pool to another, any running containers on the host you move are killed. Provided the original pool has resources available, Oracle Container Cloud Service will start new containers for the killed services on a different host in the original pool.
Add Tag to Selected	Add a tag to the selected hosts.

Creating a Resource Pool

Find out how to create a resource pool to deploy services and stacks efficiently across multiple hosts.

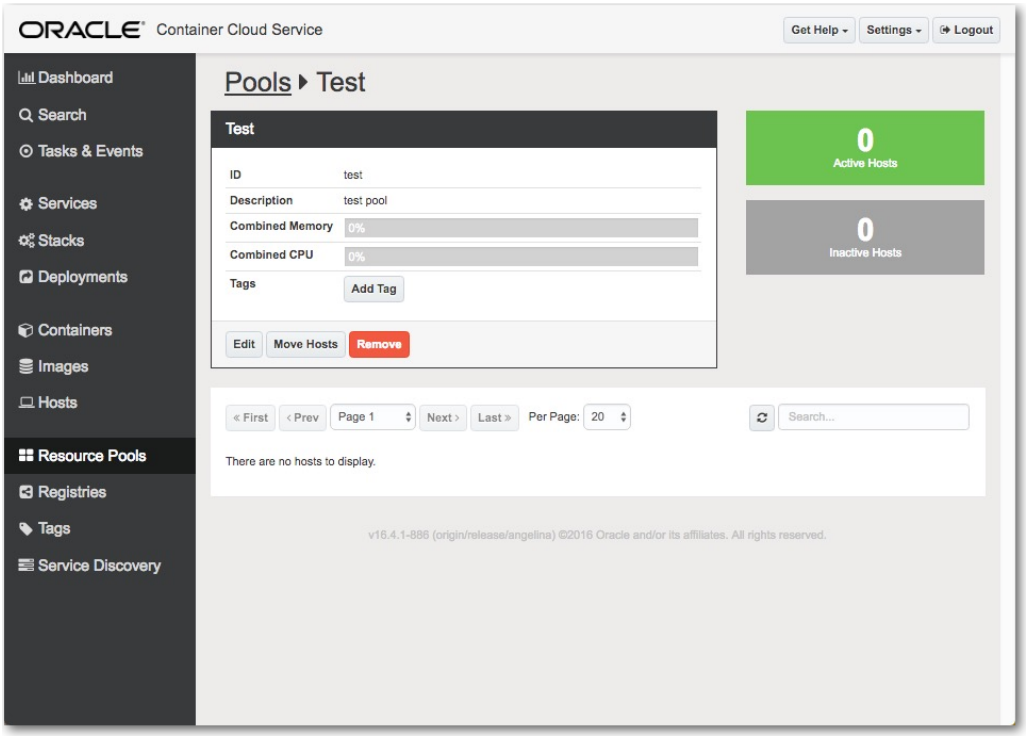
To create a new resource pool using the Oracle Container Cloud Service Container Console:

1. On the **Resource Pools** page of the Container Console, click **New Host Pool** to display the **New Pool** window.



2. In the **New Pool** window, enter:
   - a name for the pool (the name can contain spaces, and needn't be unique)
   - an id for the pool (the id mustn't contain spaces, must be unique, and you won't be able to change it later)
   - a description of the pool
3. Click **Save**.

   Oracle Container Cloud Service creates the new resource pool and shows you information about it.



As you can see, when it's first created, the new resource pool has no hosts assigned to it.

4. Click **Move Hosts** to assign one or more hosts to the new resource pool (typically from the Default pool).
5. Optional: Click **Add Tag** to give the new resource pool a tag that describes the resource pool's purpose, or that associates the resource pool with other resource pools with similar characteristics.

Managing Hosts with Oracle Container Cloud Service

Learn about hosts, how to manage them, and how to move them between resource pools.

Topics

- About Hosts
- Managing Hosts
- Moving Hosts Between Resource Pools

About Hosts

Hosts (or 'worker nodes') are the Oracle Compute virtual machines (VMs) managed by Oracle Container Cloud Service on which you deploy services and stacks to run Docker containers.

Every Oracle Container Cloud Service instance has one manager node, and a number of worker nodes (the 'hosts'). Oracle Container Cloud Service software running on the manager node orchestrates the deployment of Docker containers to the worker nodes

in the instance. Each worker node runs an Oracle Container Cloud Service agent to communicate Docker status to and from the manager node.

A worker node is considered active if the agent can successfully communicate with the manager node. If communication between the manager node and the agent on the worker node is lost for more than a minute or so (due to network, hardware, or software issues), the worker node is considered inactive.

You organize the hosts available to you into resource pools to accommodate your workflow, typically grouping together hosts that share a usage or purpose. It's important to note that a host can only exist in one resource pool at a time. See About Resource Pools.

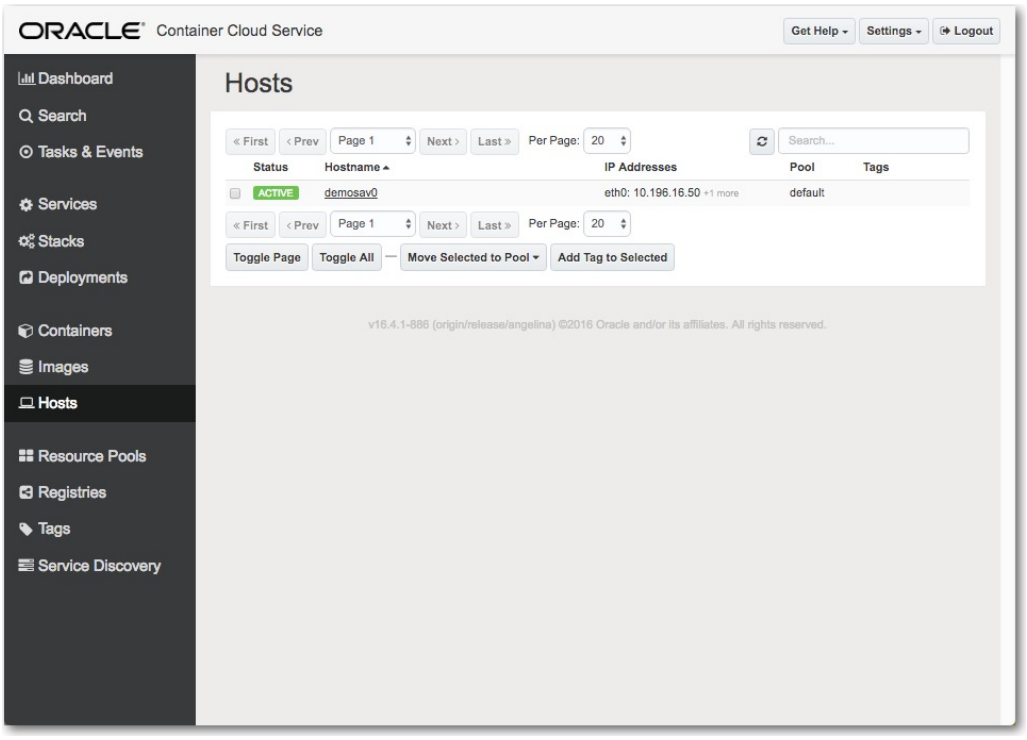
Using the Oracle Container Cloud Service Container Console, you can monitor the runtime status of a host (active or inactive). You also use the Container Console to manage the containers running on a host and the images downloaded to it, and to move hosts between resource pools.

Managing Hosts

Find out how to manage and monitor a host and the Docker containers and images on it.

To manage a host using the Oracle Container Cloud Service Container Console:

1. On the **Hosts** page of the Container Console, review the hosts for this Oracle Container Cloud Service instance.

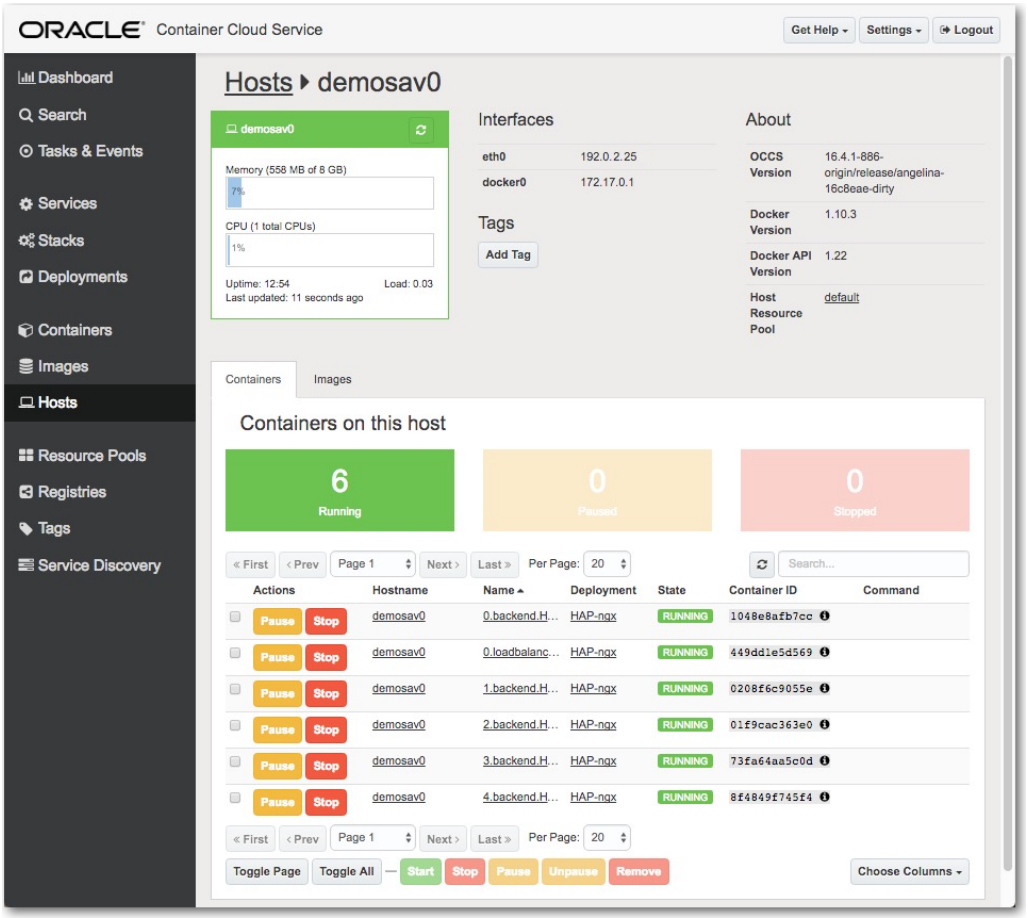


Each host is shown on a separate line, along with its IP addresses, the pool to which it belongs, its memory and CPU usage, and the number of containers running on it.

You can also:

- Move selected hosts to the pool you select. See Moving Hosts Between Resource Pools.
- Add a tag to selected hosts.

2. Perform management operations on the **Hosts** page by selecting hosts and:
   - clicking **Move Selected to Pool** to move all of the selected hosts to the same pool
   - clicking **Add Tag to Selected** to add the same tag to all of the selected hosts
3. To manage and monitor a host in more detail, use the **Hostname** field to locate the host that you're interested in and click the name of the host.



The **Host Details** page enables you to manage and monitor the host.

In the upper region of the page, you now see detailed information about the host, including:

Option	Use to see:
Memory	Absolute memory usage (as a number) and relative memory usage (as a percentage).
CPU	Number of CPUs, and relative CPU usage (as a percentage).
Load	Average system load over the last minute.

Option	Use to see:
Interfaces	The host's network interfaces and their addresses, including the host's public IP address (which is also included in the VM information shown in My Services for each Oracle Container Cloud Service instance). Use the public IP address to access any running containers that have exposed ports on this host.
Tags	A list of the tags currently assigned to the host. Click **Add Tag** to assign additional tags to the host. See About Tags.
OCCS Version	The version of the Oracle Container Cloud Service agent currently running on the host.
Docker Version	The version of the Docker daemon currently running on the host.
Docker API Version	The version of the Docker API that Oracle Container Cloud Service is using to interact with the Docker daemon on the host.
Host Resource Pool	The resource pool to which the host belongs.
Provider	OPC (Oracle Public Cloud)

4. Use the tabbed pages to access information about the host, including the containers running on it and the images downloaded to it:

Tab	Use to:
Containers	See the containers running on the host, access more detailed information about them, and perform management operations on them.
Images	See the images on the host, access more detailed information about them, and remove them from the host.

Moving Hosts Between Resource Pools

Find out how to move hosts between resource pools.

If you move a host from one pool to another, any running containers on the host you move are killed. Provided the original pool has resources available, Oracle Container Cloud Service will start new containers for the killed services on a different host in the original pool.

To move hosts between resource pools using the Oracle Container Cloud Service Container Console:

On the **Hosts** page of the Container Console, select the hosts that you want to move from one pool to another, click **Move Selected to Pool**, and specify the new pool to which you want to move the hosts.

Managing Services with Oracle Container Cloud Service

Learn about services, how to create them, and how to manage them.

Topics

- About Services
- Managing Services
- Creating a Service with Oracle Container Cloud Service

- Configuring the Default Orchestration Policy for a Service with Oracle Container Cloud Service
- Deploying a Service with Oracle Container Cloud Service
- Service Configuration Option Reference

About Services

An Oracle Container Cloud Service service comprises all of the necessary configuration for running a Docker image as a container on a host, plus default deployment directives.

Note that services themselves are neither containers nor images running in containers. Services are high-level configuration objects that you can create, deploy, and manage using Oracle Container Cloud Service. You could think of a service as a container 'template', or as a set of instructions to follow to deploy a running container.

Oracle Container Cloud Service comes with a number of pre-configured services. You can use these pre-configured services just as they are, or customize them and save them. You can also create a new service by specifying an image and then setting configuration options in the Oracle Container Cloud Service Container Console:

- by picking options from a list
- by copying and pasting a `docker run` command
- by copying and pasting YAML code

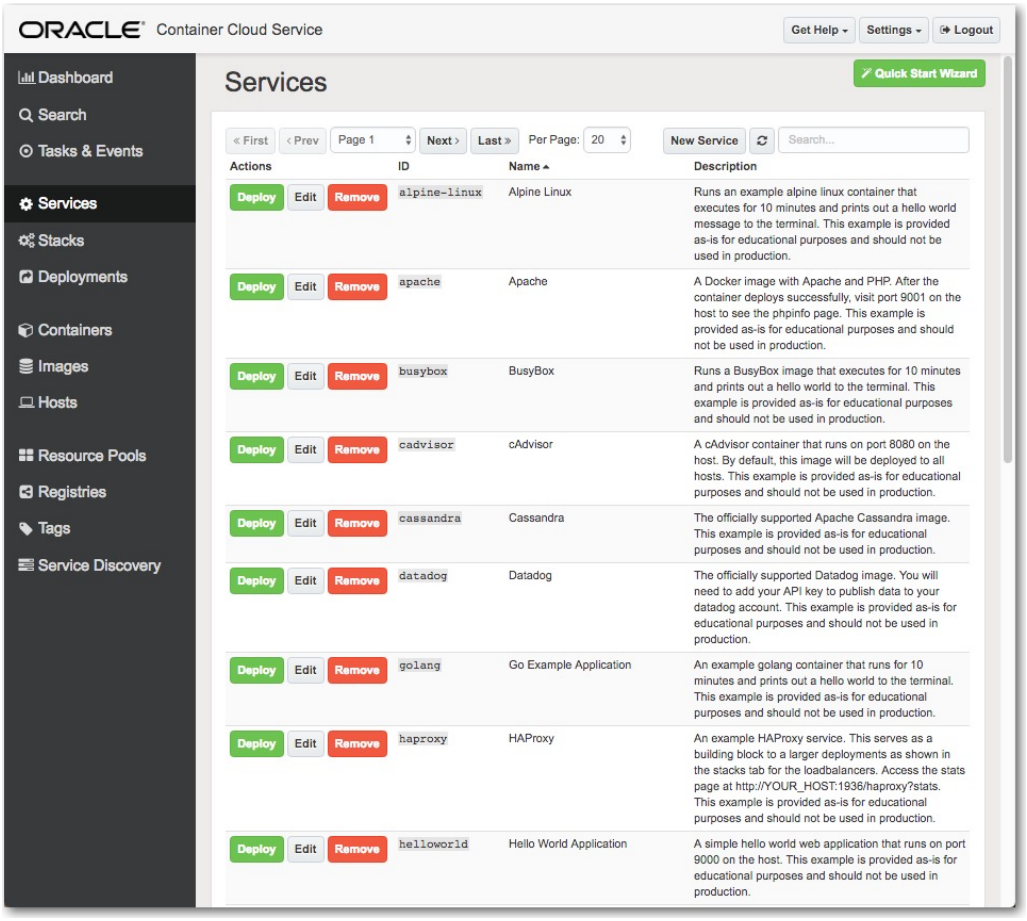
If you subsequently decide you no longer need the service, you can remove the service definition from the Oracle Container Cloud Service database. Note that if there's already a running deployment based on a service, removing the service definition doesn't affect any currently running containers.

Managing Services

You manage both services you've created and pre-configured services that come with Oracle Container Cloud Service using the Oracle Container Cloud Service Container Console.

To manage services using the Oracle Container Cloud Service Container Console:

1. On the **Services** page of the Container Console, review the list of services that you've created and the pre-configured services that come with Oracle Container Cloud Service.



Each service in the Oracle Container Cloud Service database is shown on a separate line, along with its description.

2. Perform management operations on the **Services** page by:
   - clicking **Deploy** beside a service name to deploy the service (see Deploying a Service with Oracle Container Cloud Service)
   - clicking **Remove** beside a service name to remove the service definition from the Oracle Container Cloud Service database (removing the service definition doesn't affect currently running deployments)
3. If you want to modify a service's configuration:
   a. Click **Edit** beside the service name to display the Service Editor.
   b. Modify the configuration options you want to change.

      See Creating a Service with Oracle Container Cloud Service for more information about the options you can set in the Service Editor.
   c. Click **Save**.

      Any changes you've made will take effect the next time you deploy the service, and won't affect currently running deployments.
4. If you want to create a new service, click **New Service** (see Creating a Service with Oracle Container Cloud Service).

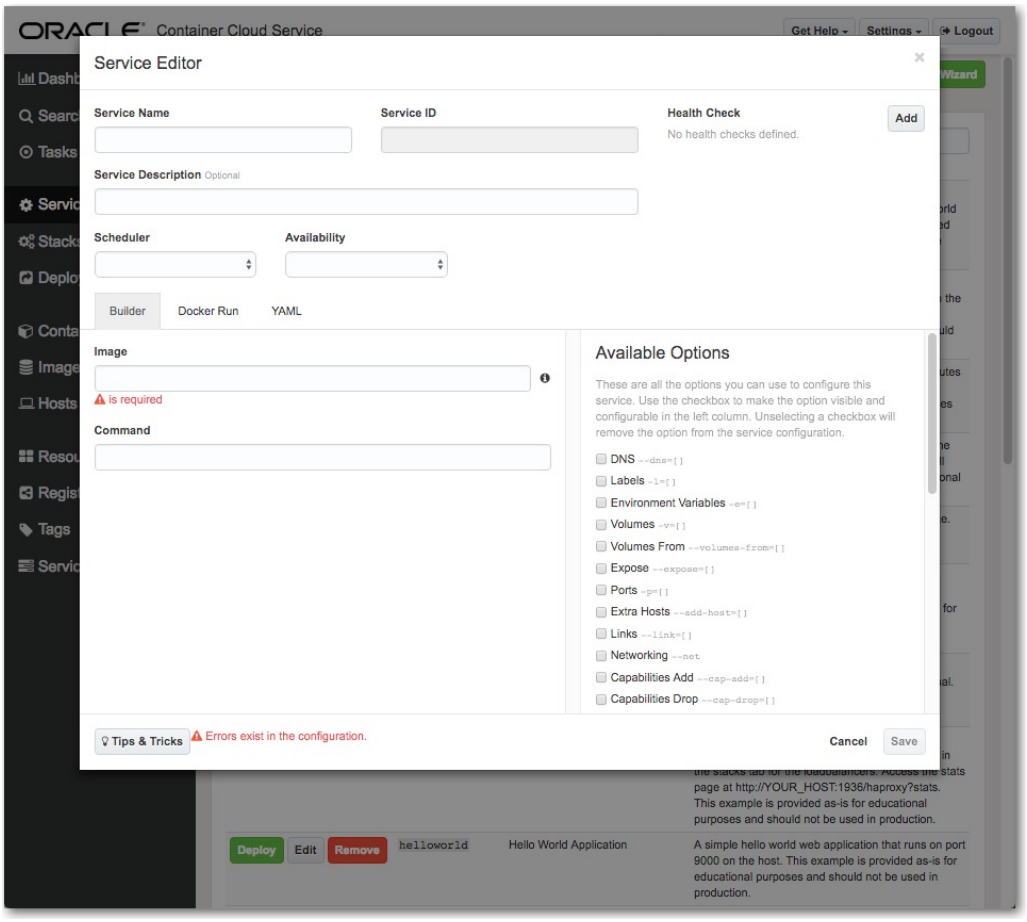
> **Tip:**
>
> If you want to create a new service that's similar to an existing service, click **Edit** beside the name of the existing service to display the Service Editor, and then click **Save As** to save a copy of the original service with a new name.

Creating a Service with Oracle Container Cloud Service

You can create a new service using the Oracle Container Cloud Service Container Console. The new service will comprise all of the necessary configuration for running a Docker container on a host, plus default deployment options.

To create a service using the Oracle Container Cloud Service Container Console:

1. On the **Services** page of the Container Console, click **New Service** to display the Service Editor.



2. Enter a unique name and a description for the new service. For example, Logstash.

   To make it easy to identify, you'll often give the service the same name as the Docker image on which it's based. But the service's name doesn't have to be the

same as the image's name. However, note that the service name you enter initially is used from now on as the Service ID. You can't change the Service ID once you've saved the service.

3. Enter default orchestration properties for the service to specify how and where to deploy it, using the **Scheduler**, **Availability**, and **Tag** fields as follows:

Option	Use to specify:
Scheduler	How Oracle Container Cloud Service determines the order to use when selecting the hosts on which to start containers when you deploy the service: • Random: Oracle Container Cloud Service uses a random method to start containers evenly across the available hosts, regardless of memory or CPU availability. • Memory: Oracle Container Cloud Service starts containers on hosts in order, beginning with the host that has the greatest amount of available memory. • CPU: Oracle Container Cloud Service starts containers on hosts in order, beginning with the host that has the greatest amount of available CPU. If you don't specify a value, Random is shown as the default when you deploy the service.
Availability	How Oracle Container Cloud Service determines where to start the number (N) of containers you specify when you deploy the service: • Per Pool: Oracle Container Cloud Service starts N containers across the resource pool as a whole, selecting hosts in order based on the setting of the **Scheduler** field. • Per Host: Oracle Container Cloud Service starts N containers on every host in the resource pool. • Per Tag: Oracle Container Cloud Service starts N containers on hosts assigned the tag in the **Tag** field, selecting hosts in order based on the setting of the **Scheduler** field. If you don't specify a value, Per Pool is shown as the default when you deploy the service.

If you expect to deploy a certain number of containers in the resource pool as a whole without worrying about which host they're on:

- Select **per-pool (across hosts in this pool)** from the **Availability** list.
- From the **Scheduler** list, specify how Oracle Container Cloud Service determines which hosts to start the containers on.

If you expect to deploy a certain number of containers on every host in the resource pool that has a particular tag associated with it:

- Select **per-tag (across hosts in this pool with a tag of...)** from the **Availability** list.
- Specify the tag to use to identify the hosts on which to deploy the service in the **Tag** field.
- From the **Scheduler** list, specify how Oracle Container Cloud Service determines which hosts to start the containers on.

If you expect to deploy a certain number of containers on every host in the resource pool, select **per-host (on each host in this pool)** from the **Availability** list.

The default orchestration properties you specify are used as default values when deploying the service, but you can override them when you come to deploy the service.

4. Optional: Define a health check to confirm that the service is running by clicking the Health Check **Add** button, entering a name for the health check, and a host port to poll. Now specify how Oracle Container Cloud Service is to check the service's health:

   - use the **Username** and **Password** fields if you want Oracle Container Cloud Service to simply establish a basic authenticated HTTP connection with the host
   - use the **Protocol**, **Interval**, and **Timeout** fields if you want Oracle Container Cloud Service to perform more sophisticated health checks

   If the service fails a health check you've defined, Oracle Container Cloud Service displays an error message.

   > **Note:**
   >
   > If you don't define a health check for the service, Oracle Container Cloud Service determines the service's status from the status of its container as reported by Docker Engine. If Docker Engine reports that the container is not running, Oracle Container Cloud Service will restart the container.

5. Specify the Docker image to download for the service, the command to run to deploy the service, and values for other Docker runtime configuration options required to deploy the service, in the following ways:

   - Use the **Builder** tab to:
     - Specify the Docker image to use.

       How to specify the image depends on its location:

       * If the image is in the public Docker Hub registry, simply specify the image name and version (for example, nginx:latest)
       * If the image is in a private registry, prefix the image name and version with the registry name and username (for example, my.internalregistry.com/myusername/myapp:latest)
       * If the image is in a private hosted registry, prefix the image name and version with the registry name and username (for example, quay.io/myusername/nginx:latest)
     - Specify an optional custom command that the container is to run.
     - Pick service configuration options from the **Available Options** list and specify values for them (see Service Configuration Option Reference)
   - Use the **Docker Run** tab to enter a `docker run` command to launch the service (including values for service configuration options). This tab is handy if you already have a `docker run` command that you can cut and paste into the **Docker Run** tab. For example:

     ```
     docker run \
       -e="occs:availability=per-pool" \
     ```

```
-e="occs:scheduler=cpu" \
"karthequian/helloworld:latest"
```

- Use the **YAML** tab to enter a YAML document to launch the service (including values for service configuration options). This tab is handy if you already have YAML code (for example, in a Docker Compose file) that you can cut and paste into the **YAML** tab. For example:

```
version: 2
services:
  my-helloworld-service:
    image: "karthequian/helloworld:latest"
    environment:
      - "occs:availability=per-pool"
      - "occs:scheduler=cpu"
```

If you don't include a registry when specifying the image to use, Oracle Container Cloud Service will attempt to pull the image from the Docker public registry.

> **Tip:**
>
> In most cases, it doesn't matter whether you use the **Builder** tab, the **Docker Run** tab, the **YAML** tab, or a combination of all three. Changes you make in one tab are reflected in the other tabs. See Service Configuration Option Reference.

6. Click **Save**.

The new service appears on the **Services** page. Having created the service, you're now ready to deploy it (see Deploying a Service with Oracle Container Cloud Service) or add it to an application stack (see Creating a Stack).

Configuring the Default Orchestration Policy for a Service with Oracle Container Cloud Service

A service's orchestration policy determines the number of Docker containers to run, and how and where to deploy them.

You can configure a service's orchestration policy:

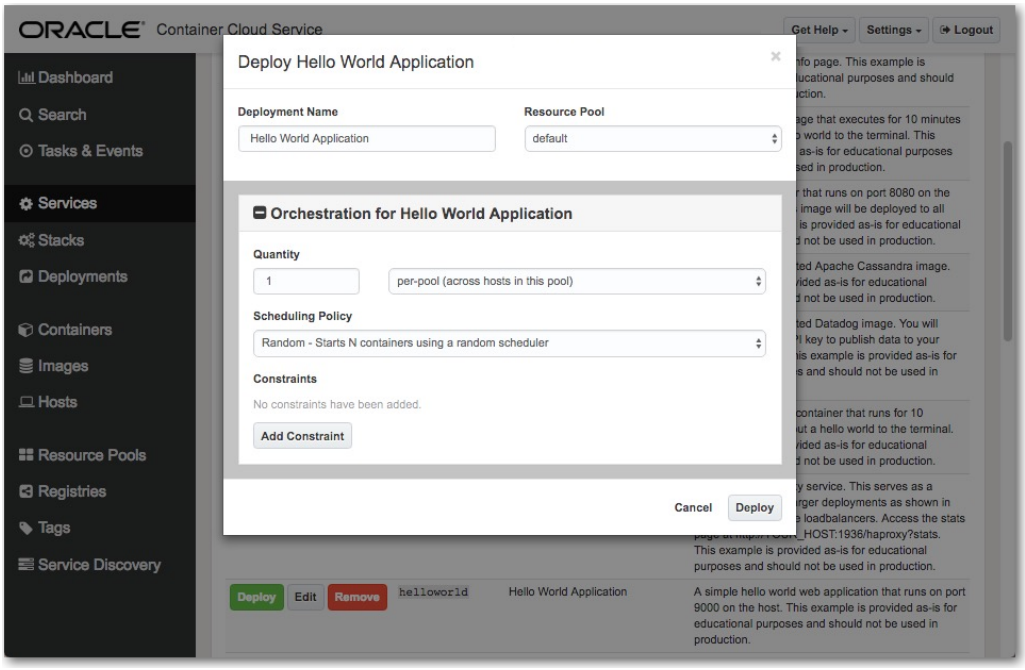
- By specifying default orchestration policy details when you create the service. The default details you specify are used as the default values for the service's orchestration policy when you deploy the service. See Creating a Service with Oracle Container Cloud Service.
- By overriding default orchestration policy details with deployment-specific details when you deploy the service. See Deploying a Service with Oracle Container Cloud Service.

Deploying a Service with Oracle Container Cloud Service

You can deploy a service individually rather than as part of a stack (for example, when an application comprises a single service, or a number of services running separately).

To deploy a service using the Oracle Container Cloud Service Container Console:

1. On the **Services** page of the Container Console, click the **Deploy** button beside the name of the service that you want to deploy.



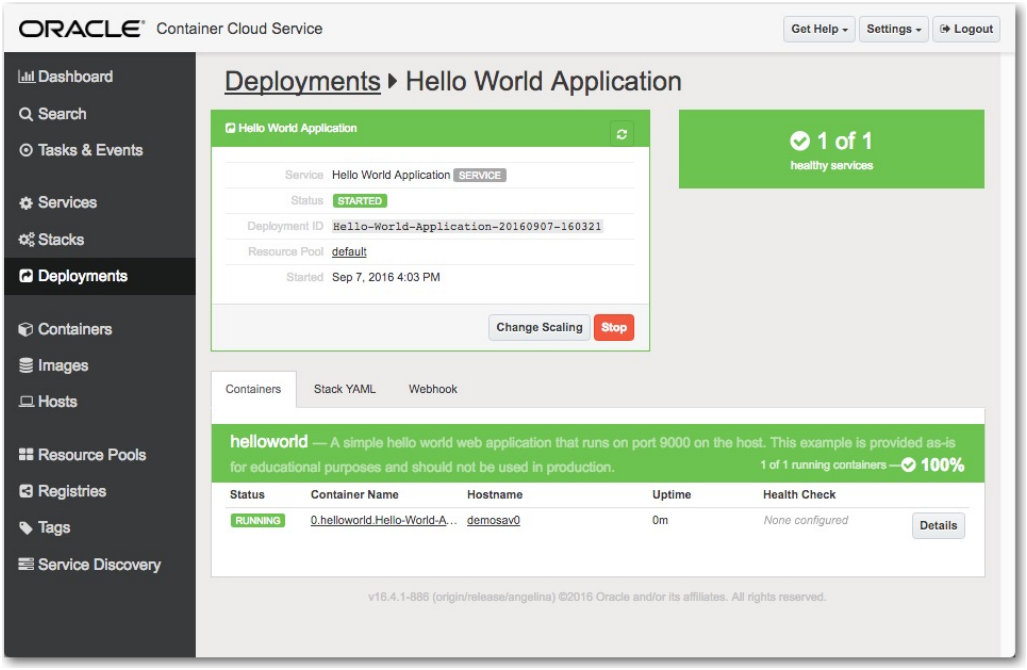
2. Specify a name for this specific deployment of the service (the name doesn't have to be unique, because Oracle Container Cloud Service identifies a deployment with an auto-generated unique deployment identifier).
3. Specify the resource pool on which to deploy the service.
4. Specify the number of containers to deploy in the **Quantity** field.
5. Review the default service orchestration information and, if necessary, update the details to specify how and where to deploy the number of containers you specified:
   - If you want to deploy a certain number of containers in the resource pool as a whole without worrying about which host they're on:
     - Select **per-pool (across hosts in this pool)** from the list.
     - From the **Scheduling Policy** list, specify how Oracle Container Cloud Service determines which hosts to start the containers on.
   - If you want to deploy a certain number of containers on every host in the resource pool that has a particular tag associated with it:
     - Select **per-tag (across hosts in this pool with a tag of...)** from the list.
     - Specify the tag to use to identify the hosts on which to deploy the service in the **Tag** field.

  - From the **Scheduling Policy** list, specify how Oracle Container Cloud Service determines which hosts to start the containers on.

- If you want to deploy a certain number of containers on every host in the resource pool, select **per-host (on each host in this pool)** from the list.

> **Tip:**
>
> If you want to have more control over the hosts a service can be deployed on, use the **Constraints** options to specify particular hosts based on their host name or associated tag.

6. Click **Deploy** to deploy and start the service in the resource pool and on the hosts that you specified.



The **Deployments** page appears and shows details of the deployment you just created by deploying the service, including:

- the names of the containers that have been started for the deployment
- the names of the hosts on which the deployment's containers are running
- the YAML that was executed for the deployment (note that if the original YAML is subsequently modified, the YAML shown for this deployment won't change)

Service Configuration Option Reference

Learn about the service configuration options available in the Oracle Container Cloud Service's Service Editor to manage your Docker environment.

You use the Service Editor to specify how to deploy the service:

- use the **Builder** tab to select configuration options using checkbox and dropdown lists
- use the **Docker Run** tab to enter Docker commands directly
- use the **YAML** tab to enter YAML commands directly

In most cases, it doesn't matter whether you use the **Builder** tab, the **Docker Run** tab, the **YAML** tab, or a combination of all three. Changes you make in one tab are reflected in the other tabs.

Builder tab Option	Docker Run equivalent (click to see Docker documentation)	YAML equivalent	Use to:
DNS	--dns=[ ]	dns	Specify a custom set of DNS servers for the container to use.
Labels	--label=[ ]	labels	Specify one or more labels for use by Docker Engine.
Environment Variables	-e=[ ]	environment	Specify the name and value of one or more environment variables.
Volumes	-v=[ ]	volumes	Specify the desired host, mount point and container target along with its read/write options.
Volumes From	--volumes-from=[ ]	volumes_from	Specify a comma-delimited list of container-IDs with an optional suffix for R/W mode. For example, 5d95413513ec:[ro|rw]. Mounts all the defined volumes from the referenced containers.
Expose	--expose=[ ]	expose	Expose ports without publishing them to the host machine. The ports will only be accessible to linked services.
Ports	-p=[ ]	ports	Publish a containers port or a range of ports to the host.
Extra Hosts	--add-host=[ ]	extra_hosts	Specify one or more additional hostname : IP mappings.
Links	--link=[ ]	links	Link to containers in another service. Either specify both the service name and the link alias (SERVICE:ALIAS), or just the service name (which will also be used for the alias).
Networking	--network_mode	network_mode	Specify networking mode. Choose from the same values as the docker client --network_mode parameter (none, bridge, host).
Capabilities Add	--cap-add=[ ]	cap_add	Specify Linux capabilities to add.
Capabilities Drop	--cap-drop=[ ]	cap_drop	Specify Linux capabilities to drop.
Container Name	--name	container_name	Specify a custom container name, rather than a generated default name.
CPU Shares	--cpu-shares	cpu_shares	Specify CPU shares using integers (relative weight).
CPU Set	--cpuset-cpus	cpuset	Specify CPUs in which to allow execution (0-3, 0,1).
Devices	--device=[ ]	devices	Specify a device mapping.
DNS Search	--dns-search=[ ]	dns_search	Specify the DNS search domain for the container to use.
Domain Name	No equivalent	domainname	Specify the domain name inside the container.

Builder tab Option	Docker Run equivalent (click to see Docker documentation)	YAML equivalent	Use to:
Hostname	--hostname	hostname	Specify the hostname to set for the container (not its host).
Entrypoint	--entrypoint	entrypoint	Specify a comma-delimited list of commands to execute in order. This overrides the Run Command.
Working Dir	--workdir	working_dir	Specify the base working directory from which any commands passed will be executed.
User	--user	user	Specify the username or UID to use when running.
Log Driver	--log-driver	log_driver	Specify a logging driver for the service's containers (None, JSON file, Syslog, Journald, GELF, Fluentd).
Log Options	--log-opt=[]	log_opt	Specify logging options for the logging driver. Logging options are key value pairs.
MAC Address	--mac-address	mac_address	Specify a MAC address. (format: 12:34:56:78:9a:bc).
Memory Limit	--memory	mem_limit	Specify the total virtual memory space for processes inside the container.
Memory+Swap Limit	--memory-swap	memswap_limit	Specify the total memory + swap space to provide to the container.
PID	--pid	pid	Set the PID mode to the host PID mode.
Privileged	--privileged	privileged	Start the container in privileged mode. Be sure to understand the security implications of this.
Read-Only	--read-only	read_only	Mount the container's root filesystem as read-only.
STDIN Open	-a stdin	stdin_open	Attach the container to STDIN (Standard Input).
TTY	-t	tty	Allocate a pseudo-tty.
Restart	--restart	restart	Specify the container restart policy on exit as one of: • **No** Do not automatically restart the container when it exits. This is the default. • **Always** Always restart the container regardless of the exit status. When you specify Always, the Docker daemon will try to restart the container indefinitely. • **On Failure** Restart only if the container exits with a non-zero exit status.
Security Options	--security-opt=[ ]	security_opt	Specify alternative security options.

Managing Application Stacks with Oracle Container Cloud Service

Learn about application stacks, how to create them, and how to manage them.

Topics

- About Stacks

- Managing Stacks
- Creating a Stack
- Deploying a Stack
- Defining Dependencies between Services in a Stack using Phases

About Stacks

An Oracle Container Cloud Service application stack (or simply ‘stack’) comprises all of the necessary configuration for running a set of services as Docker containers in a coordinated way and managed as a single entity, plus default deployment directives.

Note that stacks themselves are neither containers nor images running in containers, but rather are high-level configuration objects that you can create, deploy, and manage using Oracle Container Cloud Service.

For example, a stack might be one or more WordPress containers and a MariaDB container. Likewise a cluster of database or application nodes can be built as a stack.

Oracle Container Cloud Service comes with a number of pre-configured stacks. You can use these pre-configured stacks just as they are, or customize them and save them. You can also create new stacks, adding services to the stack as required.

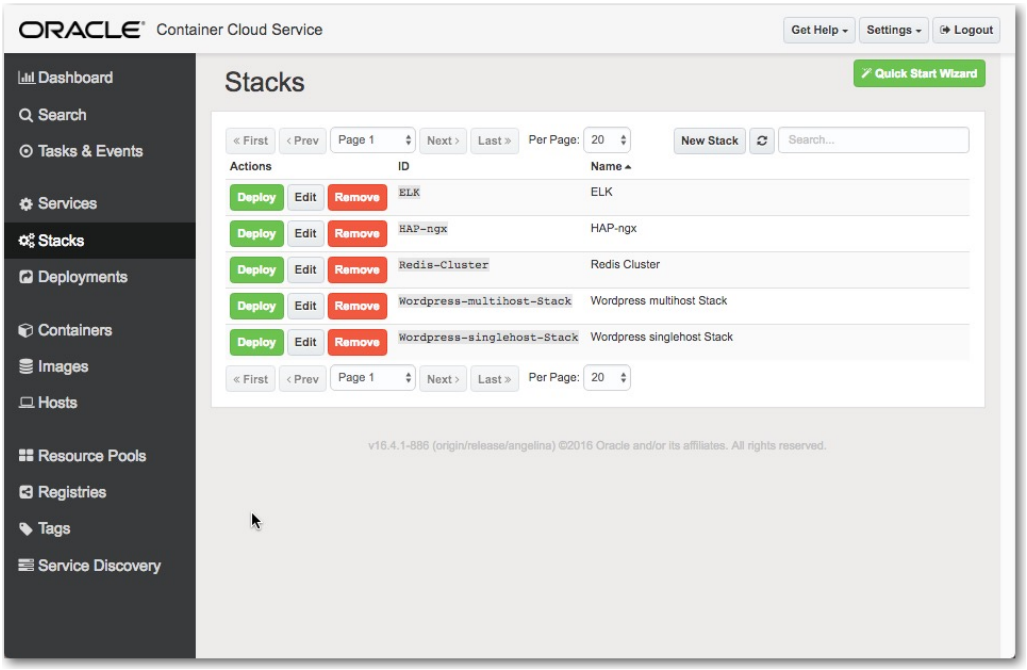
If you subsequently decide you no longer need a stack, you can remove the stack definition from the Oracle Container Cloud Service database. Note that if there’s already a running deployment based on a stack, removing the stack definition doesn’t affect any currently running containers.

Managing Stacks

You manage both stacks you’ve created and pre-configured stacks that come with Oracle Container Cloud Service using the Oracle Container Cloud Service Container Console.

To manage stacks using the Oracle Container Cloud Service Container Console:

1. On the **Stacks** page of the Container Console, review the list of stacks that you’ve created and the pre-configured stacks that come with Oracle Container Cloud Service.



Each stack in the Oracle Container Cloud Service database is shown on a separate line, along with its description.

2. Perform management operations on the **Stacks** page by:
   - clicking **Deploy** beside a stack name to deploy the stack (see Deploying a Stack)
   - clicking **Remove** beside a stack name to remove the stack definition from the Oracle Container Cloud Service database (removing the stack definition doesn't affect currently running deployments)
3. If you want to modify a stack:
   a. Click **Edit** beside the stack name to display the Stack Editor and:
      - View the properties of a service in this stack by hovering the mouse cursor over the service box on the Stack Editor canvas.
      - Modify the properties of a service in this stack by clicking the Gear icon in the center of the service box on the Stack Editor canvas to display the Service Editor. See Creating a Service with Oracle Container Cloud Service for more information about the options you can set in the Service Editor.

        Note that any changes to service properties that you make are specific to this occurrence of this service in this stack. The properties of the original service remain unchanged.

> **Tip:**
>
> If you modify the properties of a service in one stack, and later include the same service in a second stack, the changes you made to the occurrence of the service in the first stack are immaterial. However, if you do want to make those changes available for use in other stacks, save the service as a new service with a new name by clicking **Save As...** in the Service Editor. That way, the new service will appear in the **Available Services** list, in addition to the original service.

- Add a new service to the stack by dragging it from the Available Services list onto the Stack Editor canvas and setting its properties in the Service Editor.
- Remove a service from the stack by clicking the service's box on the Stack Editor canvas and pressing [Delete].

b. Click **Save**.

   Any changes you've made will take effect the next time you deploy the stack, and won't affect currently running deployments.

> **Tip:**
>
> You can also modify the properties of a stack by clicking **Advanced Editor** and updating the YAML that's executed when the stack is deployed.

4. If you want to create a new stack, click **New Stack** (see Creating a Stack).

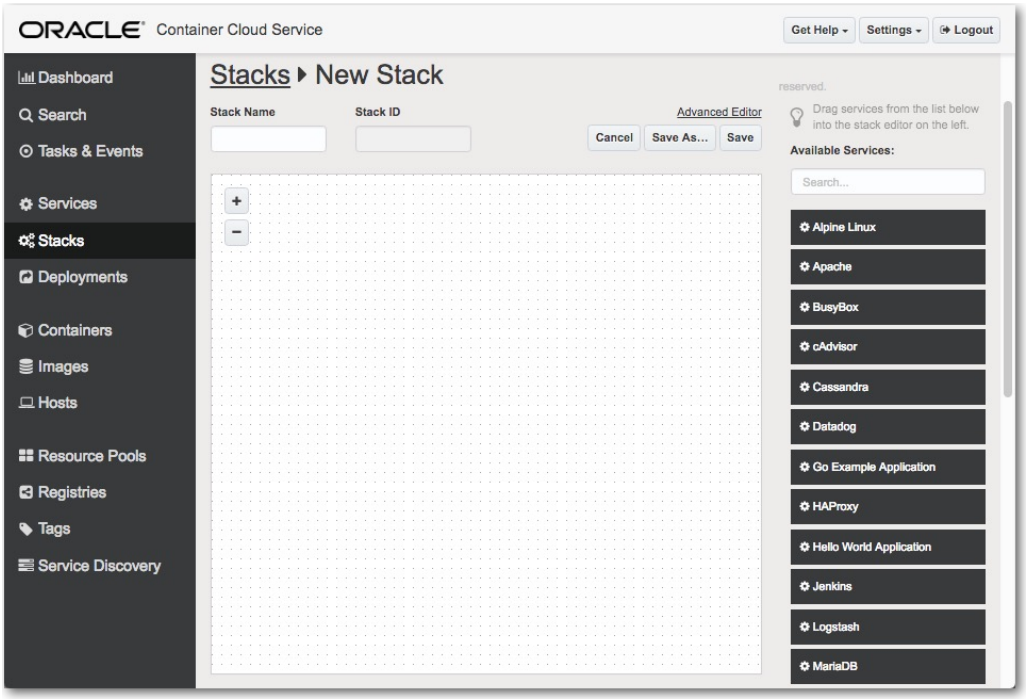
> **Tip:**
>
> If you want to create a new stack that's similar to an existing stack, click **Edit** beside the name of the existing stack to display the Stack Editor, and then click **Save As** to save a copy of the original stack with a new name.

Creating a Stack

You can create a new stack using the Oracle Container Cloud Service Container Console. The new stack will comprise all of the necessary configuration for running a set of services in a coordinated way, managed as a single entity, plus default deployment options.

To create a stack using the Oracle Container Cloud Service Container Console:

1. On the **Stacks** page of the Container Console, click **New Stack** to display the Stack Editor. The services you can include in the stack are shown in the **Available Services** list.

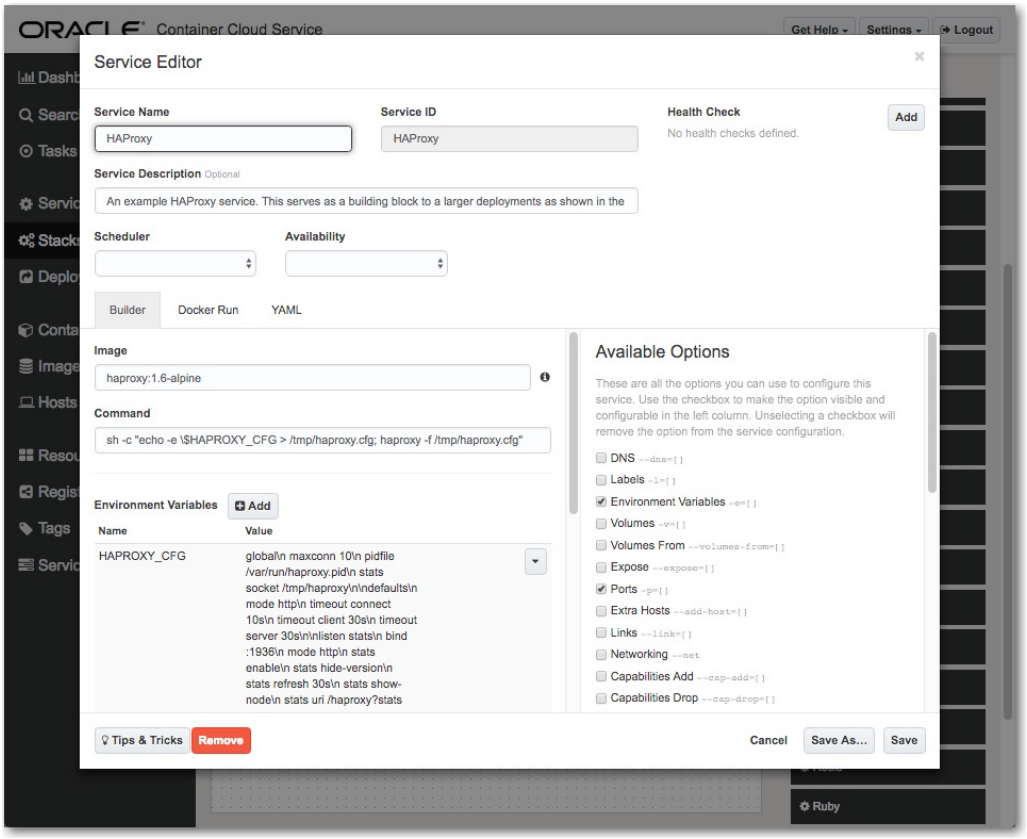


2. Enter a unique name for the new stack in the **Stack Name** field. For example, Load Balancer.

   The stack name you enter is used as the Stack ID. You can't change the Stack ID once you've saved the stack.

> **Tip:**
>
> You might already have a Docker Compose file or a YAML file that you want to use to define the stack. Rather than dragging and dropping services onto the Stack Editor canvas as described in the next few steps, simply click **Advanced Editor**, paste the contents of your existing file into the Advanced (YAML) Stack Editor, and save the stack. Changes you make in the Advanced (YAML) Stack Editor are reflected on the Stack Editor canvas, and vice versa. Note that if your file uses features in the very latest version of Docker Compose, Oracle Container Cloud Service might not yet support those features.

3. Drag the first service to include in the stack from the **Available Services** list onto the Stack Editor canvas. The Service Editor is displayed.



4. Specify properties for the service you want to include in the stack, including a name and description for this occurrence of the service, and the image from which to create the service.

   The properties you can set are identical to those you can set when you're defining a new service (see Creating a Service with Oracle Container Cloud Service).

   Note that any changes to service properties that you make are specific to this occurrence of this service in this stack. The properties of the original service remain unchanged.

5. Click **Save** to save the service for use in the current stack only. When you click **Save**, the service is added to the Stack Editor canvas.

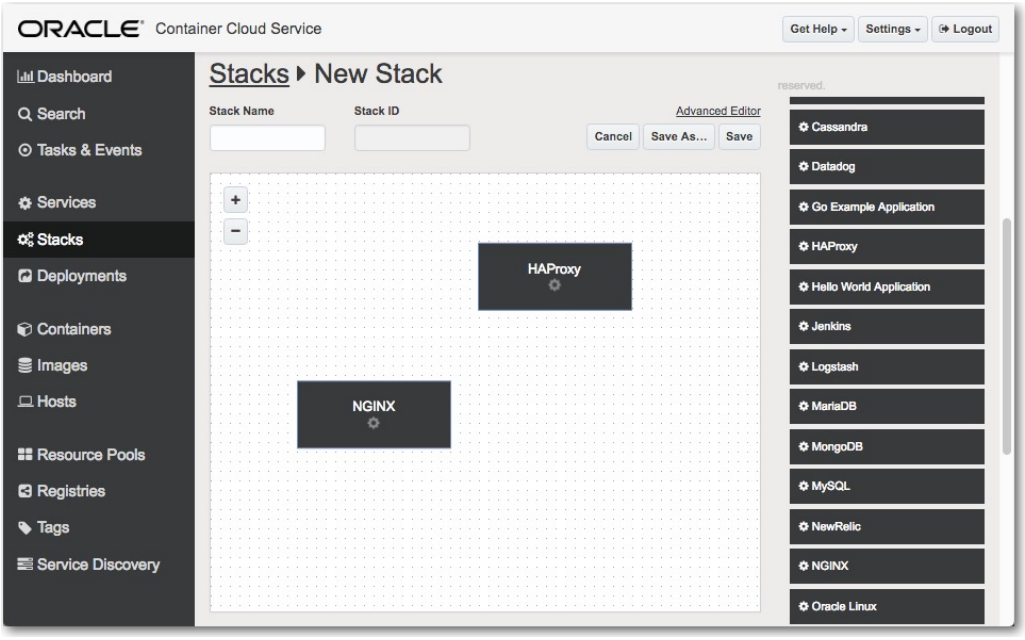
> **Tip:**
>
> If you modify the properties of a service in one stack, and later include the same service in a second stack, the changes you made to the occurrence of the service in the first stack are immaterial. However, if you do want to make those changes available for use in other stacks, save the service as a new service with a new name by clicking **Save As...** in the Service Editor. That way, the new service will appear in the **Available Services** list, in addition to the original service.

6. If you want to add another service to the stack, drag the service from the **Available Services** list onto the Stack Editor canvas and set properties for the service.

7. If you want to edit the properties of an existing service in a stack, display the Service Editor by clicking the Gear icon in the center of the service box on the Stack Editor canvas, or by double-clicking on the box itself.
8. Continue dragging services onto the Stack Editor canvas until you've added all the necessary services to the stack.

   Lines are automatically drawn between services that are related to each other, as determined from link statements in the service definitions.
9. Click **Save**.

The new stack appears on the **Stacks** page.



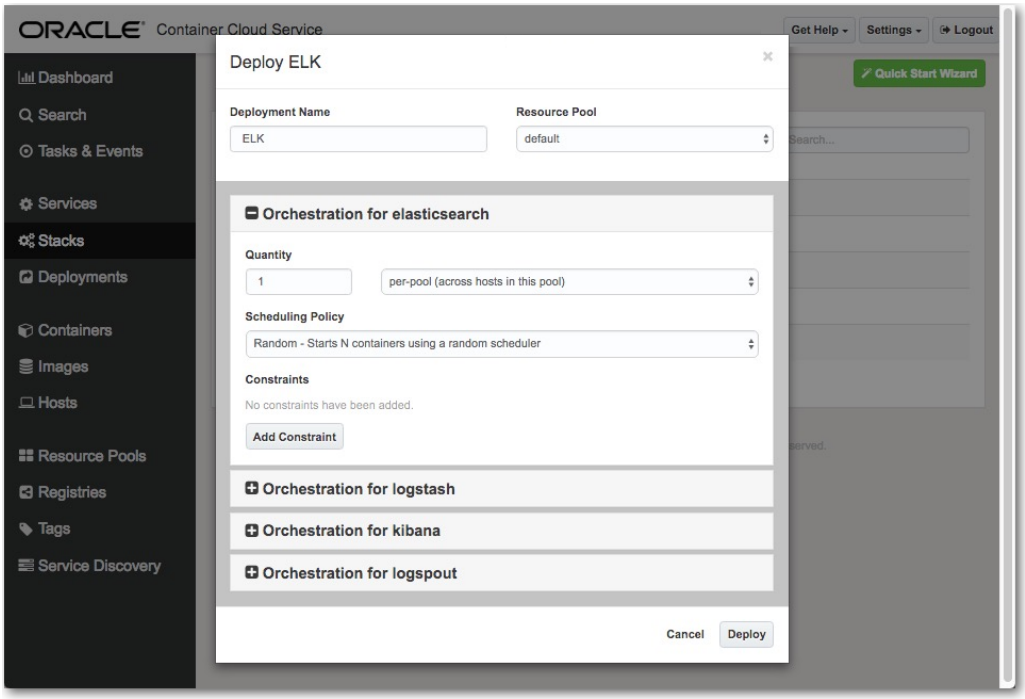
Having created the stack, you're now ready to deploy it (see Deploying a Stack).

Deploying a Stack

When you've defined an application stack, you're ready to deploy it on hosts in a resource pool that you specify.

To deploy a stack using the Oracle Container Cloud Service Container Console:

1. On the **Stacks** page of the Container Console, click the **Deploy** button beside the name of the stack that you want to deploy.

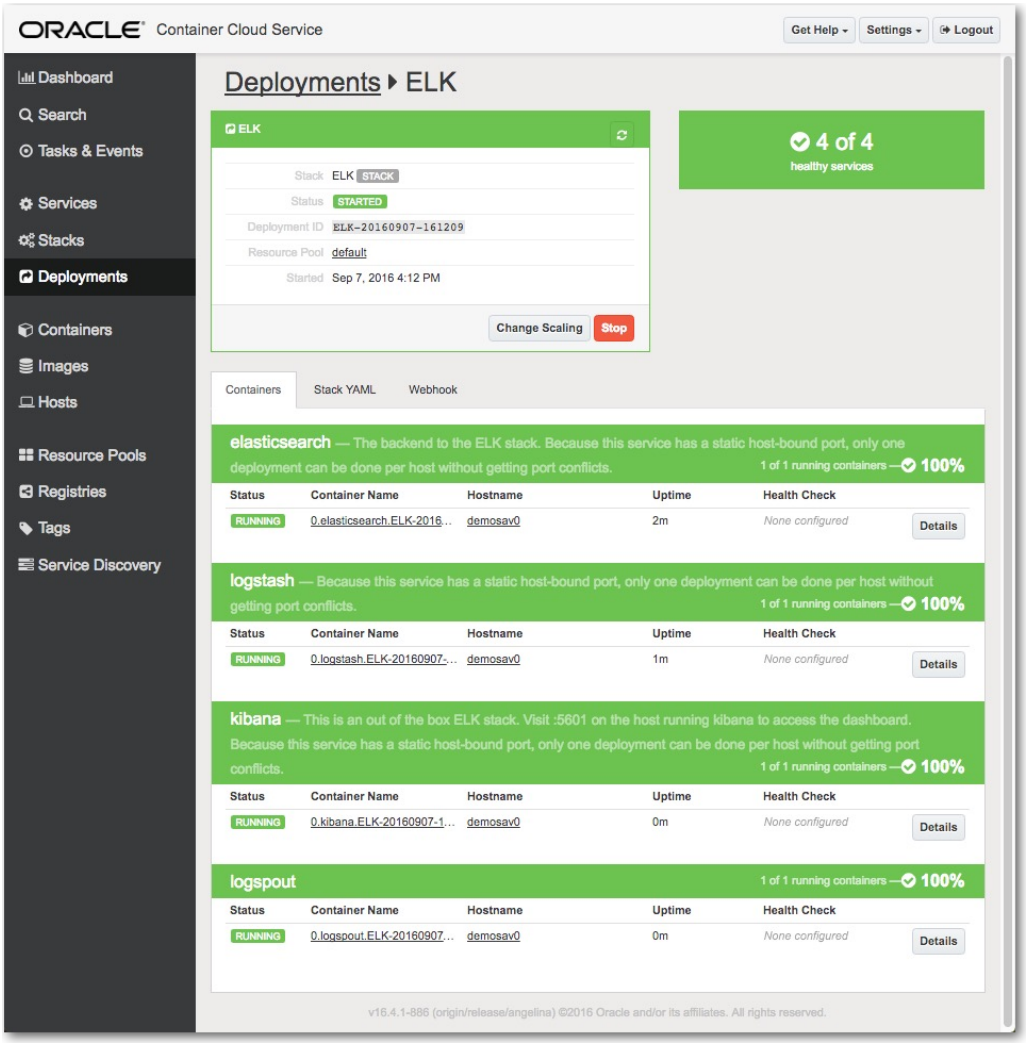


You can now enter deployment properties for the stack and its services.

2. Specify a name for this specific deployment of the stack (the name doesn't have to be unique, because Oracle Container Cloud Service identifies each deployment with an auto-generated unique deployment identifier).
3. Specify the resource pool on which to deploy the stack.
4. For each service in the stack, review the default service orchestration information and, if necessary, update the details by specifying:
   - the number of containers to deploy in the **Quantity** field
   - how and where to deploy the number of containers you specified:
     - If you want to deploy a certain number of containers in the resource pool as a whole without worrying about which host they're on, select **per-pool (across hosts in this pool)**. From the **Scheduling Policy** list, specify how Oracle Container Cloud Service determines which hosts to start the containers on.
     - If you want to deploy a certain number of containers on every host in the resource pool that has a particular tag associated with it, select **per-tag (across hosts in this pool with a tag of...)** and specify the tag to use to identify the hosts on which to deploy the service in the **Tag** field. From the **Scheduling Policy** list, specify how Oracle Container Cloud Service determines which hosts to start the containers on.
     - If you want to deploy a certain number of containers on every host in the resource pool, select **per-host (on each host in this pool)**.

> **Tip:**
>
> If you want to have more control over the hosts a service can be deployed on, use the **Constraints** options to specify particular hosts based on their host name or associated tag.

5. Click **Deploy** to deploy and start the stack and its services in the resource pool and on the hosts that you specified.



The **Deployments** page appears and shows details of the deployment you just created by deploying the stack, including:

- the names of the containers that have been started for the deployment
- the names of the hosts on which the deployment's containers are running
- the YAML that was executed for the deployment (note that if the original YAML is subsequently modified, the YAML shown for this deployment won't change)

Defining Dependencies between Services in a Stack using Phases

The services in a stack are often dependent on each other. In some cases, it's important that services start in a specific order. In other cases, you might not want one service to start until a number of other services have started successfully.

You can include *phases* in a stack definition to ensure that dependencies between services in the stack are preserved when you deploy and start the stack. Phases (each comprising one or more services) are identified by integers, starting at phase 0. If a phase comprises multiple services, all the services in that phase will start at the same time. Services in the next phase will not start until all the services in the previous phase have been started and are considered 'healthy', as follows:

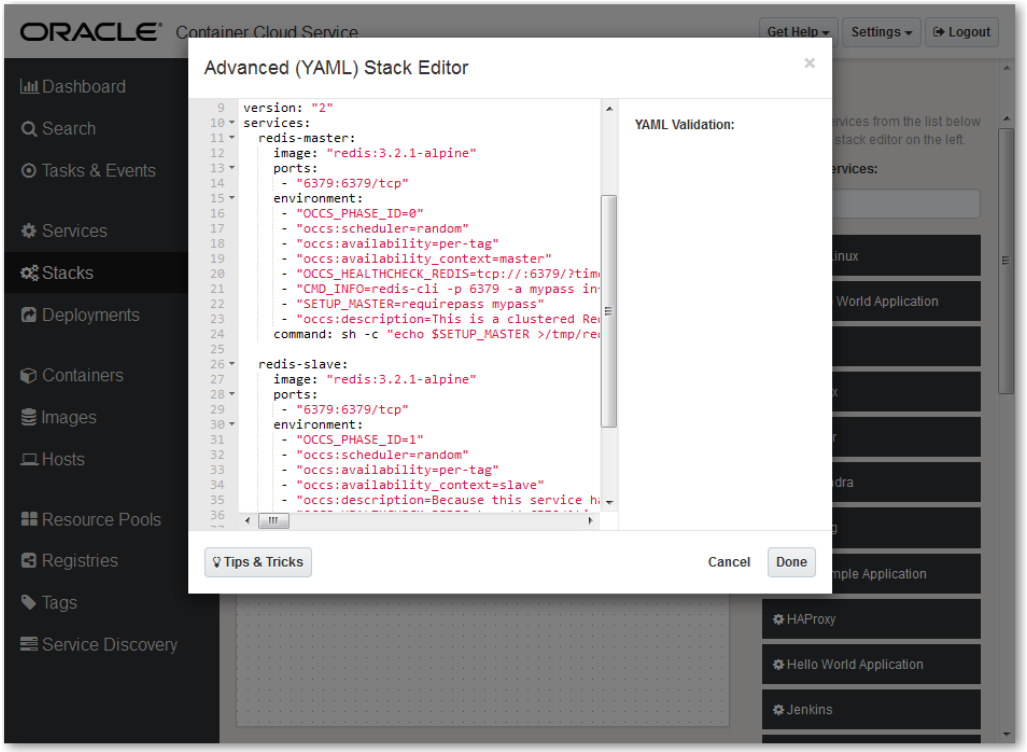
- in the case of a service for which health checks have been defined, the service is considered healthy if all the health checks are passing
- in the case of a service for which no health checks have been defined, the service is considered healthy if all containers for the service are running

Services in the lowest numbered phase (phase 0) are started first. When all the services in phase 0 have started successfully and are healthy, services in the next phase (phase 1) are started. When all the services in phase 1 have started successfully and are healthy, services in the next phase (phase 2) are started, and so on.

If you stop a deployed stack, running services are stopped in reverse order to the order in which they were started. So services in the phase with the highest identifier are stopped first, then services in the phase with the next highest identifier, and so on.

To define dependencies between services in a stack using phases:

1. On the **Stacks** page of the Container Console, click the **Edit** button beside the stack name to display the Stack Editor.
2. Click **Advanced Editor** to display the Advanced (YAML) Stack Editor, showing the YAML to deploy the stack.



3. For each service for which you want to define a phase:
   a. Under the `services:` top level node, locate the node for the service.
   b. Locate the `environment:` sub-node.
   c. Specify the phase in which the service is to start by setting a value for the `OCCS_PHASE_ID` key.

      For example:

      ```
      services:
        mysql:
          image: "mysql:latest"
          environment:
            - OCCS_PHASE_ID=0
      ```

   d. Add additional `OCCS_PHASE_ID` key value pairs for the other services in the stack. For example:
      - If you want three services to start one after the other, specify three different phase ids. For example:

        ```
        services:
          mysql:
            image: "mysql:latest"
            environment:
              - OCCS_PHASE_ID=0

          nginx:
            image: "nginx:latest"
            environment:
        ```

```
      - OCCS_PHASE_ID=2

  app:
    image: "app_image:latest"
    environment:
      - OCCS_PHASE_ID=1
```

- If you want two services to start at the same time, and a third service to start only after the other two have started successfully, give the same phase id to the two services you want to start at the same time, and specify a different phase id for the service you want to start later. For example:

```
services:
  mysql:
    image: "mysql:latest"
    environment:
      - OCCS_PHASE_ID=0

  nginx:
    image: "nginx:latest"
    environment:
      - OCCS_PHASE_ID=1

  app:
    image: "app_image:latest"
    environment:
      - OCCS_PHASE_ID=0
```

4. Click **Done** to save your changes and close the Advanced (YAML) Stack Editor.

When the stack is next deployed, services will start according to the phases that you specified.

Managing Docker Containers with Oracle Container Cloud Service

Learn about Docker containers and how to manage them.

Topics

- About Docker Containers
- Managing Docker Containers
- Accessing an Application Running Inside a Deployed Container
- Enabling Docker Containers to Communicate With Each Other
- Viewing the Log Files of a Running Container

About Docker Containers

A Docker container is a process that's created to run an application held in a Docker image.

The Docker container includes everything the application needs in order to run, including executable code, a runtime environment, system tools, and system libraries. This containerized approach ensures that the application will always run the same, regardless of the environment in which it's running.

When an operator executes a `docker run` command, the container process that runs is isolated in that it has its own file system, its own networking, and its own isolated process tree separate from the host.

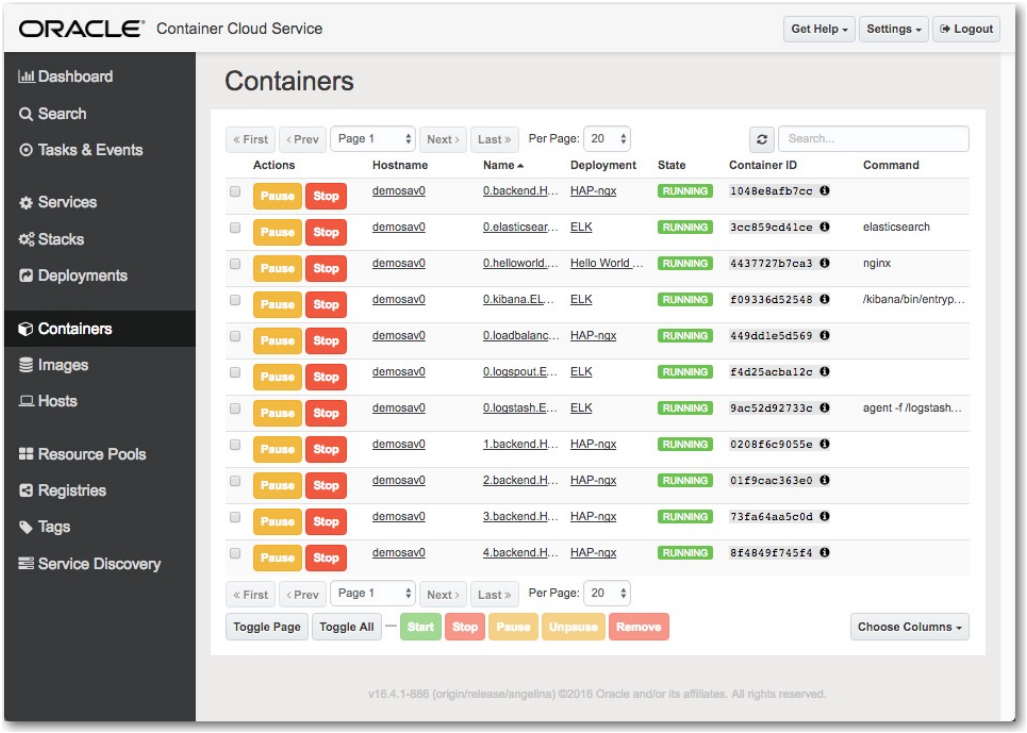
Managing Docker Containers

Having deployed a service or stack, you can use the Oracle Container Cloud Service Container Console to manage the Docker containers that are created for the deployment.

You can also manage Docker containers that are not associated with a deployment because they were started directly from a Docker image (see Starting a Docker Container Directly from a Docker Image).

To manage containers using the Oracle Container Cloud Service Container Console:

1. On the **Containers** page of the Container Console, review the Docker containers that have been created for deployed services.



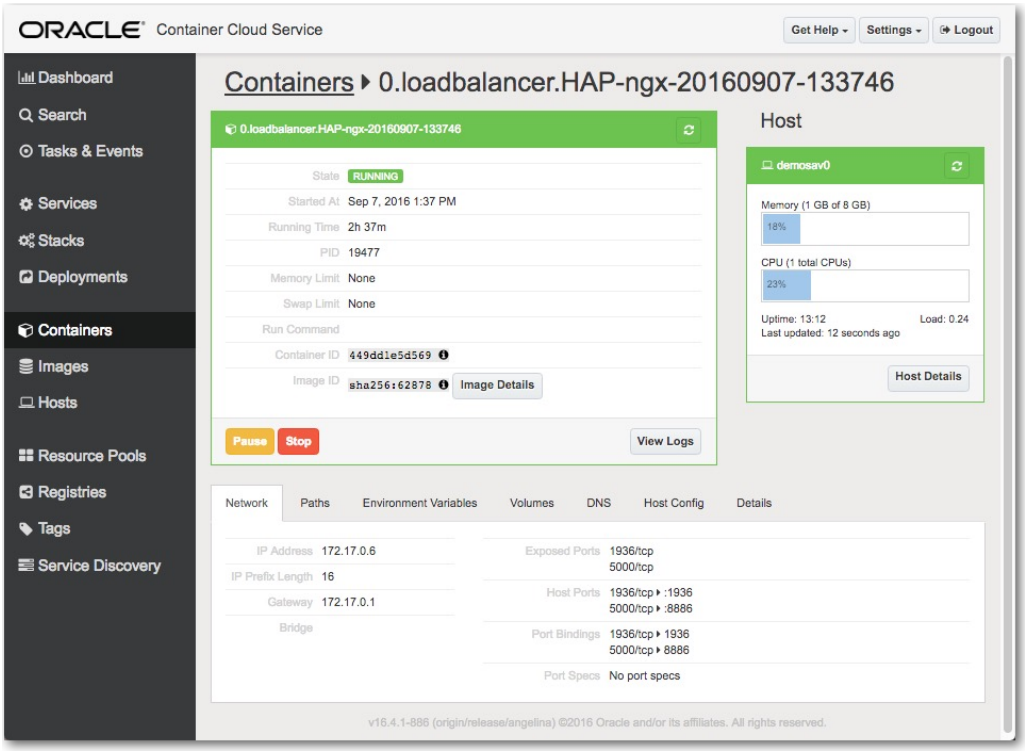
Each Docker container (identified by the first 12 characters of its unique container ID for easy comparison with the output of the `docker ps` command) is shown on a separate line, along with the name of the host on which it's running, the service it's running, and the command it was started with.

2. Perform management operations on the **Containers** page by selecting one or more containers and using the **Start**, **Stop**, **Pause**, **Unpause**, and **Remove** buttons.

> **Note:**
>
> Only use the **Start**, **Stop**, and **Remove** buttons to manage Docker containers that you've started directly from an image (see Starting a Docker Container Directly from a Docker Image).
>
> Don't use these buttons on individual Docker containers that are part of a running deployment. Instead, stop, start, and remove the containers by performing operations on the deployment itself (see Managing Deployments). If you stop a container that's part of a running deployment, the deployment will restart the container automatically.

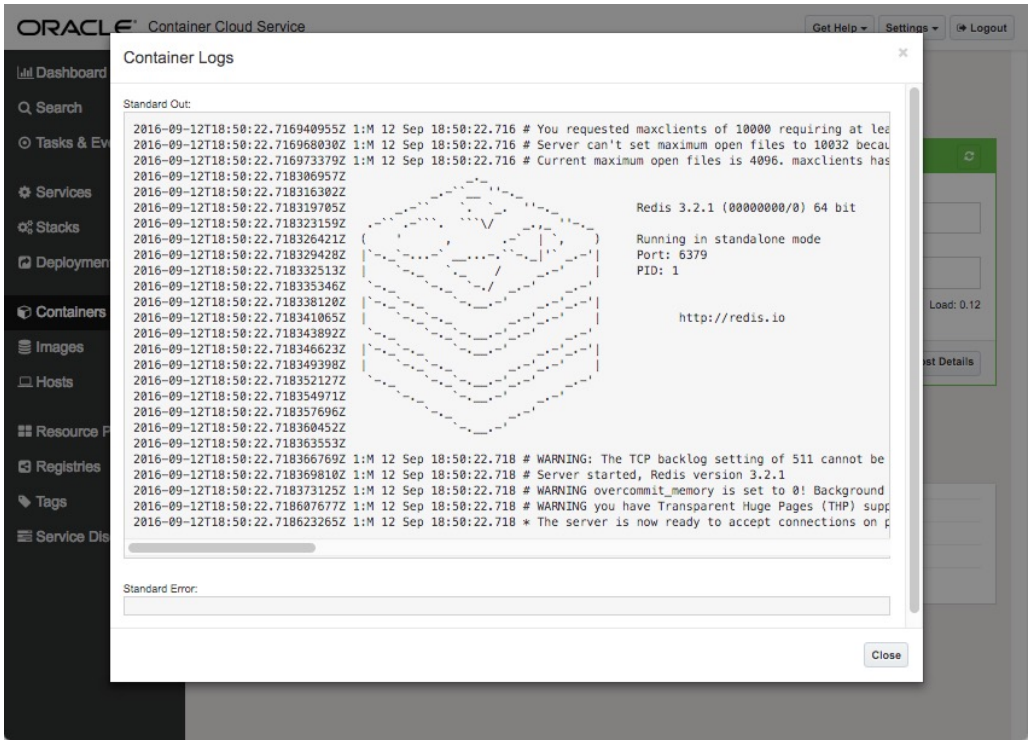
As well as performing management operations on Docker containers, typically you'll also want to view other information about the container such as configuration settings and runtime status.

3. Select the container you want to see more information about:
   a. Use the **Container ID** field, service **Name** field, and **Hostname** field to locate the Docker container.
   b. Click the name of the service that the container is running, shown in the **Name** field.



In the top left region, you now see detailed information about the Docker container, including runtime information and host usage. The color of the region's title bar shows the status of the container (green for running, orange for paused).

You can also view a snapshot of the container's Docker Standard Out and Standard Error log files by clicking **View Logs** to display the **Container Logs** window.



Note that output continues to be written to the log files. To see the latest output, close the **Container Logs** window and then click **View Logs** again.

4. Use the tabbed pages to access information about each Docker container.

Tab	Use to:
Network	See networking information about the container such as its IP address and port mappings.
Paths	See information about the container's physical location on the host.
Environment Variables	See the environment variables set by an operator at container run time.
Volumes	See the volume mounts both internal and external to the container.
DNS	See the DNS entries for the container.
Host Config	See the bind mount volumes and port mapping details.
Details	See miscellaneous information including the full run command, user, and create time.

Accessing an Application Running Inside a Deployed Container

After you've deployed a service or stack as one (or more) Docker containers, you'll want to test and use the applications running inside the deployed containers.

Before you can access the application inside a container, you'll need to find out the IP address and port number of the host on which the container has been deployed. You can then construct the URLto access the application.

To access an application running inside a deployed container using the Oracle Container Cloud Service Container Console:

1. On the **Deployments** page of the Container Console, click the name of the application that you want to access. The **Deployments Details** page is displayed.
2. Display the **Stack YAML** tab and note down the host port value specified for the `ports` option. If two values are shown, the host port is the value before the colon. For example, if you see `ports: -"9000:80/tcp"`, the host port is 9000.
3. Display the **Containers** tab, click the name of the host running the application you want to access, and note down the host's IP address shown in the **public_ip** field. For example, 198.51.100.254.
4. Assemble a URL from the host's IP address and the port specified for the deployment that you've just noted down. For example, http://198.51.100.254:9000.
5. In a browser, navigate to the URL you've assembled to access the application running inside the deployed container.

Enabling Docker Containers to Communicate With Each Other

Learn how to enable Docker containers to locate each other so they can communicate and operate together.

Topics

- About Container Communication
- Managing Entries in the Service Discovery Database to Enable Container Communication

About Container Communication

To communicate with each other, Docker containers require DNS information to find out the location of other containers. Oracle Container Cloud Service maintains DNS information for the running containers that it's managing in its Service Discovery database.

A service running in one Docker container might have a dependency on a second service running in another container on the same host or on a different host. Similarly, multiple containers might be running the same service on the same or different hosts for load balancing and high-availability reasons. In these situations, containers use DNS information to communicate with each other.

When a new container starts, Oracle Container Cloud Service automatically registers the new container's IP address and port number in the Service Discovery database. In some situations, you might want to manually enter or update DNS information in the Service Discovery database (for example, to initially bootstrap a container).

The record for each container in the Service Discovery database comprises:

- a key, made up of a unique id generated for the deployed container, concatenated with the id and name of the deployed service
- a value, made up of the IP address of the host running the container, concatenated with the port exposed by the service

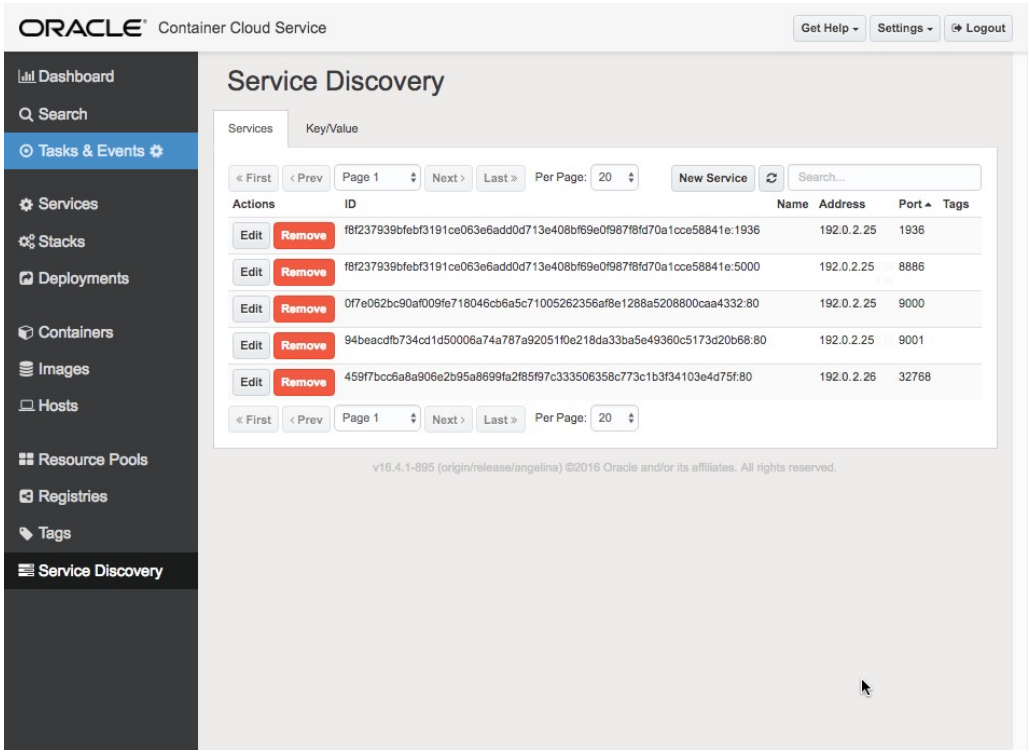
To enable a running container to become aware of a newly started container on which it depends, you'll need to include functionality in your stack to act as a listener to poll the Service Discovery database for container keys that contain the name of the dependency. That code can then write the DNS information for matching containers, which can in turn be regularly queried by other running containers to obtain the latest information about dependent containers. Some of the Oracle Container Cloud Service example stacks include this functionality. To find out more, see the information about building the example stacks on GitHub.

Managing Entries in the Service Discovery Database to Enable Container Communication

You can view the DNS information recorded for running Docker containers in the Oracle Container Cloud Service Service Discovery database. This information enables containers to communicate with each other.

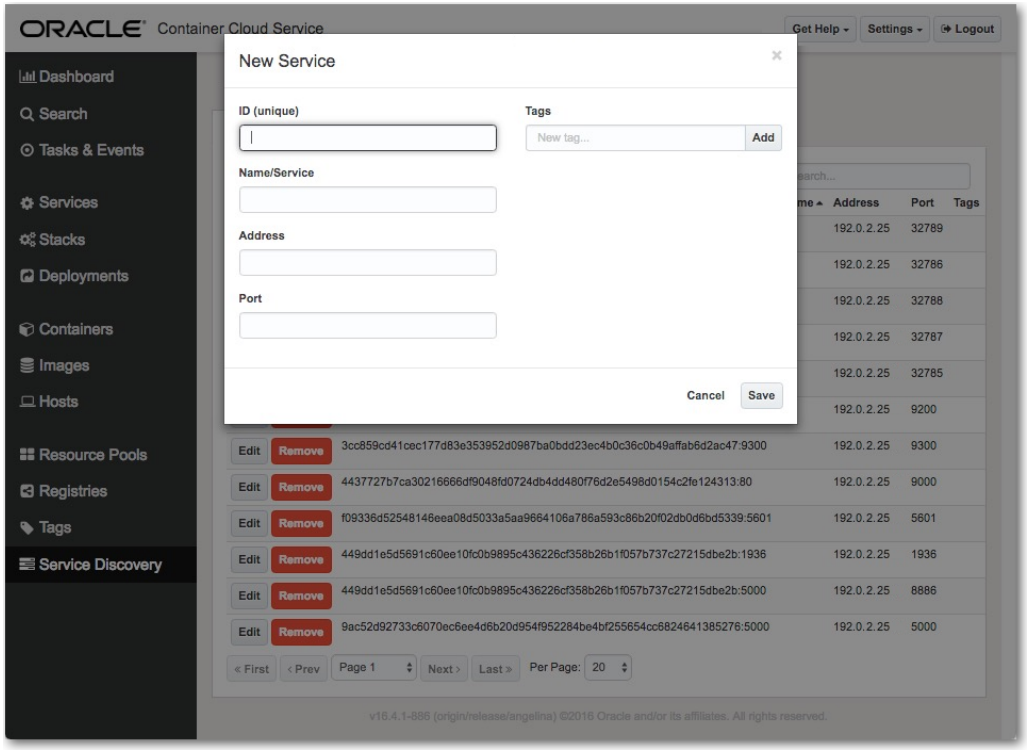
To view, update, and add DNS information for a service in the Service Discovery database using the Oracle Container Cloud Service Container Console:

1. On the **Service Discovery** page of the Container Console, review the services currently registered in the Service Discovery database.



2. Use the **Services** tab to:
   - View details for all current service entries in the Service Discovery database.
   - Manually add details for a new service to the Service Discovery database by clicking **New Service** and specifying options for the service entry:

Option	Use to specify:
ID	A unique identifier for this service entry in the Service Discovery database. For example, my-redis. Unique identifiers generated by Oracle Container Cloud Service are alphanumeric strings. For example, 85349d243fa7853fbe7516947916411bd3acec7380fb87049ebdbbde25d78a0d. The identifier maps to the DNS host name. You can't change the identifier once you've saved the service entry.
Name/Service	The service to which the entry applies. For example, redis. To group multiple entries for the same service into a Round Robin DNS pool, specify the same name in this field for all the entries. For example, if you were creating an entry for the redis service, you might specify redis in this field for the first service entry and for all subsequent entries for the redis service.
Address	The IP address of the host or service. For example, 192.0.2.175. IP addresses of service entries that have the same name in the **Name/Service** field are added into a Round Robin DNS pool.
Port	The port on which this service can be attached. For example, 6379. Don't append TCP or UDP to the port number because Oracle Container Cloud Service doesn't differentiate between TCP and UDP.
Tags	One or more tags to assign to this entry. For example, redis_tag. Tags you specify here are only relevant in the Service Discovery database, and are not available as tags elsewhere in Oracle Container Cloud Service.



- Update existing service details by clicking **Edit**.
- Remove details of a service from the Service Discovery database.

3. Use the **Key/Value** tab to:
   - View keys and values for all current service entries in the Service Discovery database.
   - Manually add details for a new key and associated value in the Service Discovery database by clicking **New Key/Value**.

Option	Use to specify:
Key	A unique key for this service entry in the Service Discovery database. Unique keys generated by Oracle Container Cloud Service include the name and id of the deployed service. For example, apps/ redis-001-redis-001-20160823-173743-80/containers/ 85349d243fa7853fbe7516947916411bd3acec7380fb87049ebdbbde25d78a0d.
Value	The IP address of the host running the container, concatenated with the port exposed by the service. For example, 192.0.2.175:9000.

   - Update an existing key or associated value by clicking **Edit** beside the entry you want to change.
   - Remove details for an existing key and associated value from the Service Discovery database.

Viewing the Log Files of a Running Container

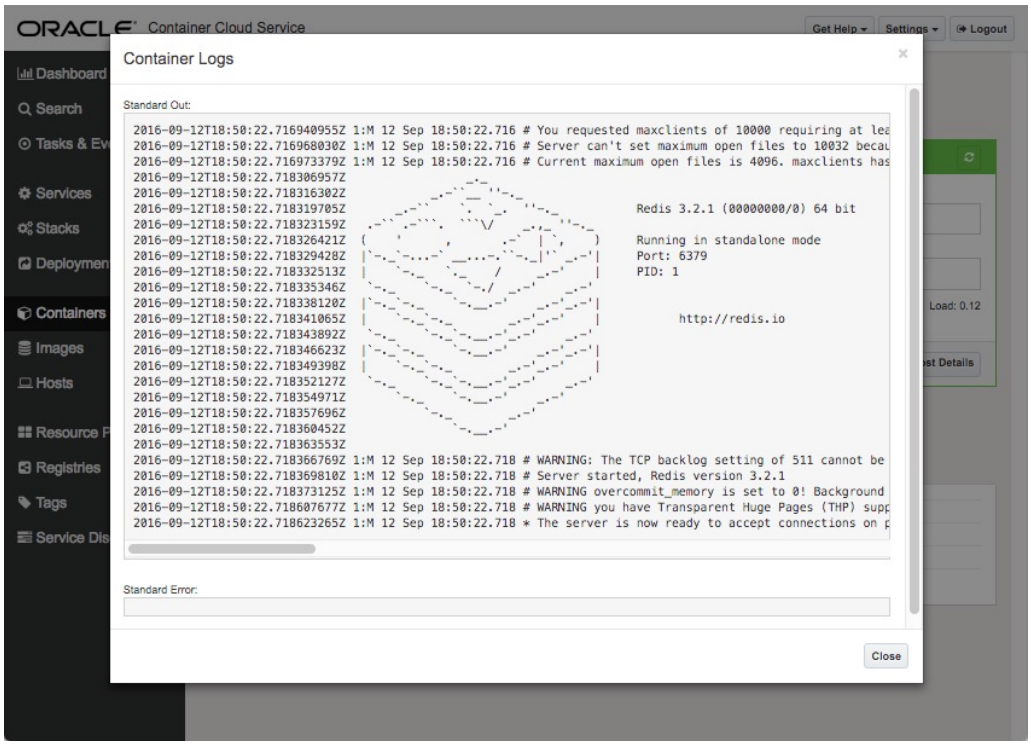
You can view the log files of a running container using the Oracle Container Cloud Service Container Console.

For example, you might want to view the log files of a particular container on a specific host to diagnose issues.

To view the log files of a running container using the Oracle Container Cloud Service Container Console:

1. On the **Containers** page of the Container Console, select the container you want to see more information about:
   a. Use the **Container ID** field, service **Name** field, and **Hostname** field to locate the Docker container.
   b. Click the name of the service that the container is running, shown in the **Name** field.

   Details about the container are displayed.
2. Click **View Logs** to display the **Container Logs** window.



You see a snapshot of the container's Docker Standard Out and Standard Error log files. Note that output continues to be written to the log files. To see the latest output, close the **Container Logs** window and then click **View Logs** again.

3. When you're done viewing the container logs, click **Close**.

Managing Docker Images with Oracle Container Cloud Service

Learn about Docker images and how to manage them.

Topics

- About Docker Images
- Managing Docker Images
- Starting a Docker Container Directly from a Docker Image
- Pushing a Docker Image to a Docker Repository

About Docker Images

A Docker image holds the application that you want Docker to run, along with any dependencies. A Docker image is stored in a Docker registry.

When a service is deployed (either singly, or as part of a stack) for the first time, the Docker image for the service is pulled from the specified Docker registry (either the public Docker registry or a private Docker registry) and added to the list of images managed by Oracle Container Cloud Service. The same Docker image can appear multiple times in the list of images managed by Oracle Container Cloud Service if:

- a service based on the image is deployed on multiple hosts
- the image has multiple tags associated with it

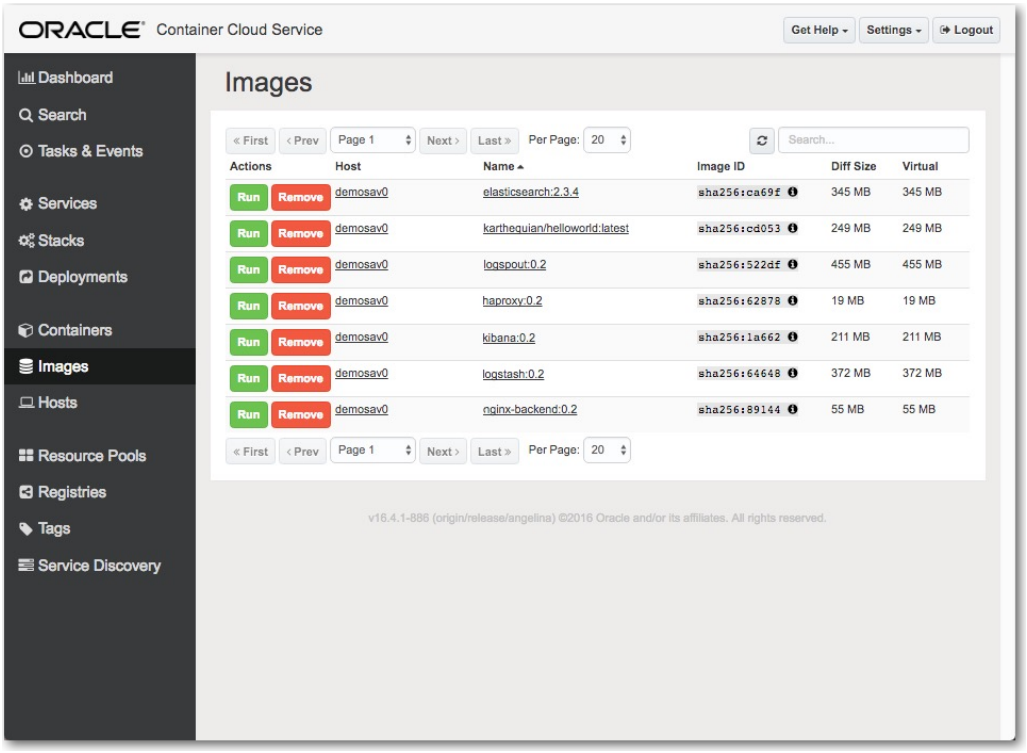
Once an image has been pulled from a Docker registry onto a host managed by Oracle Container Cloud Service, you can also start a container directly from the image (rather than starting a container by deploying a service). When you run the image in this way, you specify options for the container and manage the container directly using the Oracle Container Cloud Service Container Console. Note that in this case, Oracle Container Cloud Service doesn't create a deployment object for the running container. See Starting a Docker Container Directly from a Docker Image.

Managing Docker Images

Having deployed a service or stack, you can use the Oracle Container Cloud Service Container Console to manage the Docker images that have been pulled from public or private Docker registries.

To manage images using the Oracle Container Cloud Service Container Console:

1. On the **Images** page of the Container Console, review the images that have been pulled from Docker registries onto hosts managed by Oracle Container Cloud Service.



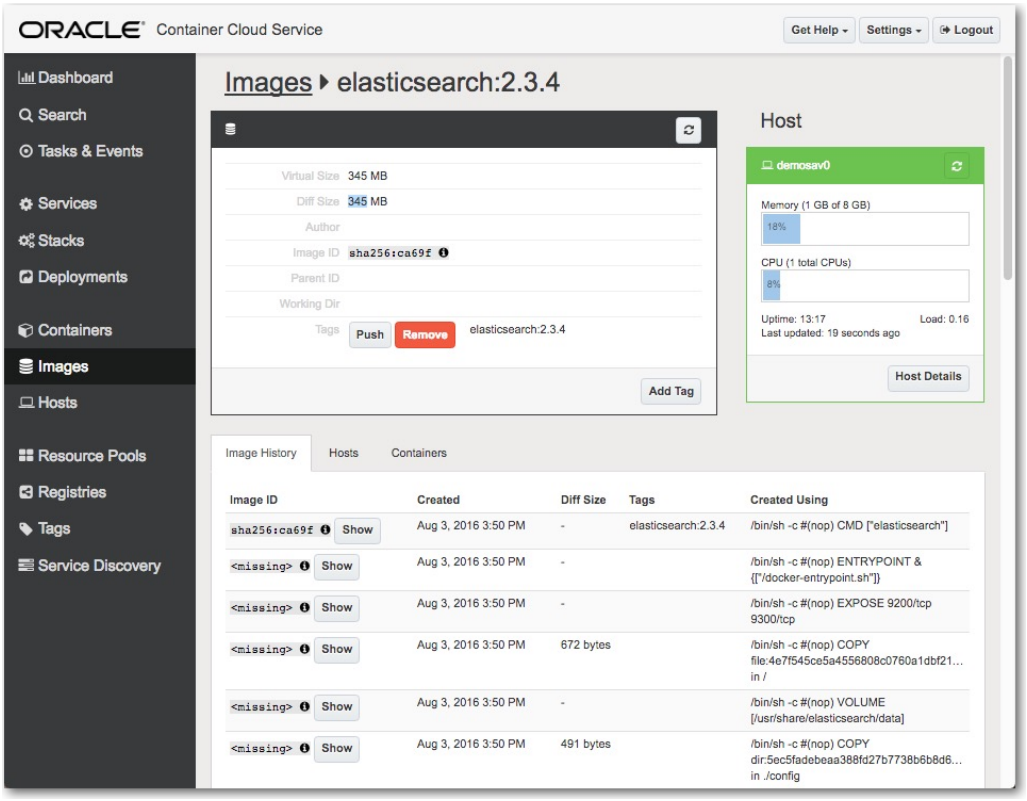
Each Docker image (identified by its unique Docker image ID) is shown on a separate line, along with the name of the image and the name of the host on which the image resides, and sizes for all layers in the image (Virtual Size) and the final layer in the image (Diff Size).

2. Perform management operations on the **Images** page by:

- clicking **Run** to start a Docker container from the Docker image on the associated host (see Starting a Docker Container Directly from a Docker Image)
- clicking **Remove** to delete a Docker image from the associated host, even if there are running Docker containers based on it (equivalent to the `docker rm -f` command)

As well as performing management operations on images, typically you'll also want to view other information about an image such as configuration settings and runtime status.

3. Use the **Host** field and **Name** field to locate the Docker image for which you want to view detailed information, and click the name of the image.



In the top left region, you now see detailed information about the Docker image, including:

Details	Use to:
Virtual Size	See the size of all layers of the container including ancestors.
Diff Size	See the size of only the final layer.
Author	See the author of the image as stated by the Dockerfile it was created from.
Image ID	See the full hash ID of the image. The first 12 characters are displayed for easy comparison with the `docker images` command.

Details	Use to:
Parent Id	See the ID of the image that the current image is built on top of (the parent image). Click on the parent image's ID to get more detail about it. Click the **Show** button to see the image that the parent image is built on top of.
Working Dir	The working directory specified for the image in its Dockerfile. If not specified, the default directory is used.
Tags	See the tag of the image. • Click **Push** to push the image (giving it this tag) to a public or private Docker registry that you've already defined in Oracle Container Cloud Service (see Creating a Docker Registry Definition). • Click **Remove** to remove a tag from the image. If you remove the last tag from an image, you're prompted to confirm that you want to remove the image from the host. • Click **Add Tag** to add a tag to the image. Having added a tag to an image, you can then push it to a public or private Docker registry.

4. Use the tabbed pages to access information about the Docker image including the hosts it's on, and the containers based on it:

Tab	Use to:
Image History	See details of the changes made to each layer of the image, from its base image upwards. Click **Show** for more details on each layer.
Hosts	A list of all hosts the image is currently on.
Containers	A list of currently available containers created from the image.

Starting a Docker Container Directly from a Docker Image

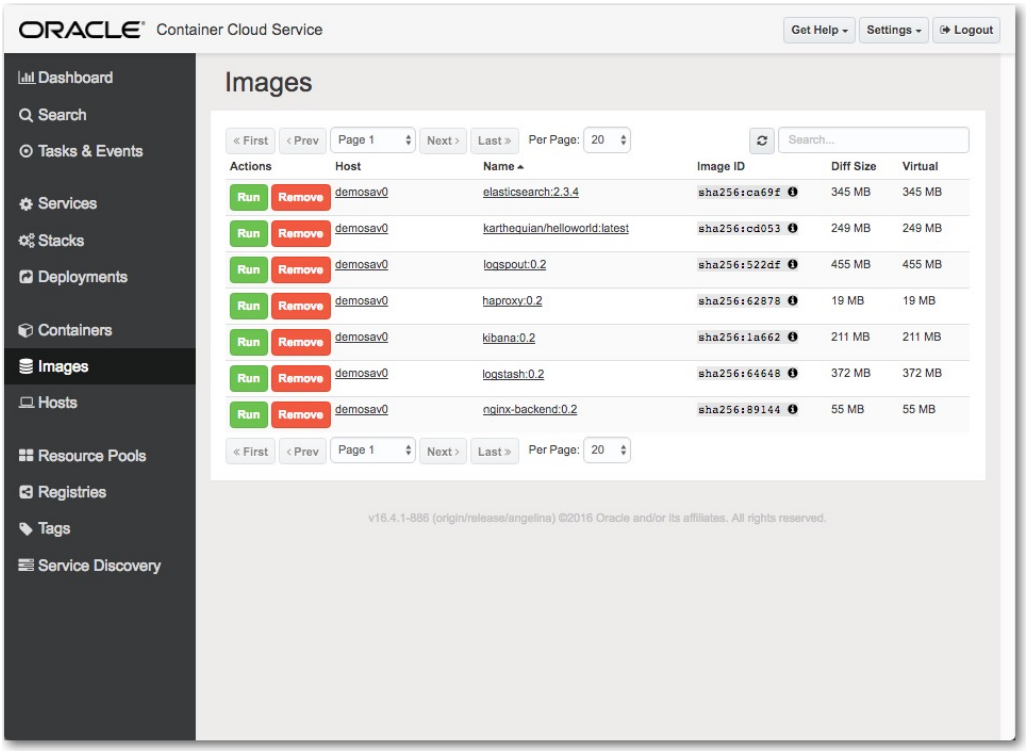
You don't have to deploy a service or stack to start a Docker container based on a Docker image. You can start a Docker container directly from an image that has already been pulled onto a host managed by Oracle Container Cloud Service.

For example, you might want to quickly start a simple container without specifying orchestration details.

Note that if you start a container directly from an image rather than by deploying a service or stack, Oracle Container Cloud Service enables you to manage the container directly, and doesn't create a deployment object.

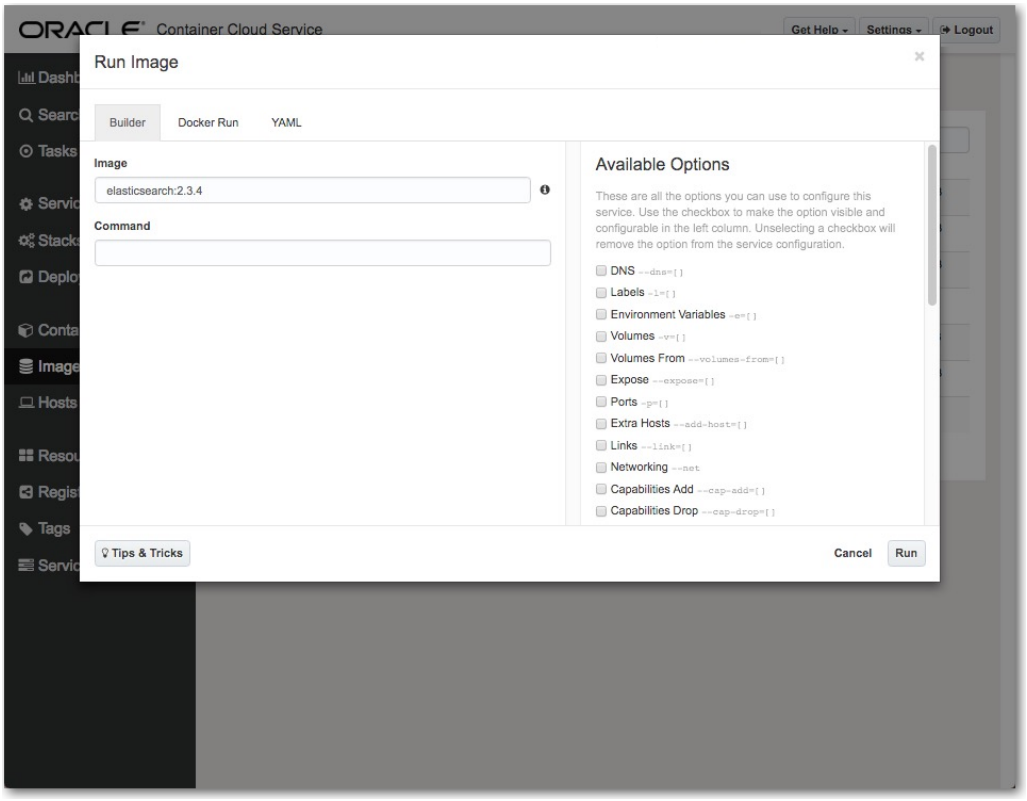
To start a Docker container directly from a Docker image using the Oracle Container Cloud Service Container Console:

1. On the **Images** page of the Container Console, review the images that have already been pulled from Docker registries onto hosts managed by Oracle Container Cloud Service.



Each Docker image (identified by its unique Docker image ID) is shown on a separate line, along with the name of the image and the name of the host on which the image resides.

2. Click the **Run** button beside the name of the host on which the Docker image resides to start a Docker container on that host.



3. In the **Run Image** window, specify the Docker image to download, the command to run to start the Docker container, and values for other Docker runtime configuration options required to start the container, in the following ways:

   - Use the **Builder** tab to:
     - specify the Docker image to use, in the format user/repository:tag
     - specify an optional custom command that the Docker container is to run
     - pick configuration options from the Available Options list and specify values for them (see Service Configuration Option Reference)
   - Use the **Docker Run** tab to enter a `docker run` command to start the Docker container (including values for configuration options). This tab is handy if you already have a `docker run` command that you can cut and paste into the **Docker Run** tab. For example:

     ```
     docker run \
       -e="occs:availability=per-pool" \
       -e="occs:scheduler=cpu" \
       "karthequian/helloworld:latest"
     ```

   - Use the **YAML** tab to enter a YAML document to start the Docker container (including values for configuration options). This tab is handy if you already have YAML code (for example, in a Docker Compose file) that you can cut and paste into the **YAML** tab. For example:

     ```
     version: 2
     services:
     ```

```
my-helloworld-service:
  image: "karthequian/helloworld:latest"
  environment:
    - "occs:availability=per-pool"
    - "occs:scheduler=cpu"
```

If you don't include a Docker registry when specifying the Docker image to use, Oracle Container Cloud Service will attempt to pull the image from the Docker public registry.

> **Tip:**
>
> In most cases, it doesn't matter whether you use the **Builder** tab, the **Docker Run** tab, the **YAML** tab, or a combination of all three. Changes you make in one tab are reflected in the other tabs. See Service Configuration Option Reference.

4. Click **Run**.

   A Docker container based on the Docker image is started on the host you specified. The container is shown on the **Containers** page of the Oracle Container Cloud Service Container Console with an auto-generated container name. However, no deployment is created for the container, so it's not shown on the **Deployments** page.

Pushing a Docker Image to a Docker Repository

When you want to store a new or updated image in a Docker repository, you can use the Oracle Container Cloud Service Container Console to add a tag to the image and then push it to the repository.

For example, you might have pulled the latest version of an application from Docker Hub, updated the local version and stored it in your local Docker repository as a new image, and then used the Container Console to deploy the image as a service to test it. If the image passes the tests, you'll want to push the new version to Docker Hub to replace the previous version by giving it the 'latest' tag.

You can use tags to push an image to any repository in any registry, not just to the registry it was originally pulled from. Note that a definition of the registry containing the repository to which you want to push the image must already exist in Oracle Container Cloud Service. Also note that the user specified in the registry definition must have sufficient privileges to write to the registry (see Creating a Docker Registry Definition).

You can use the same tag for multiple images. For example, you might want to push several different images to different repositories, but add the same 'latest' tag to each of them.

You can assign multiple tags to the same image. For example, you might add both a version number tag and the 'latest' tag to the same image.

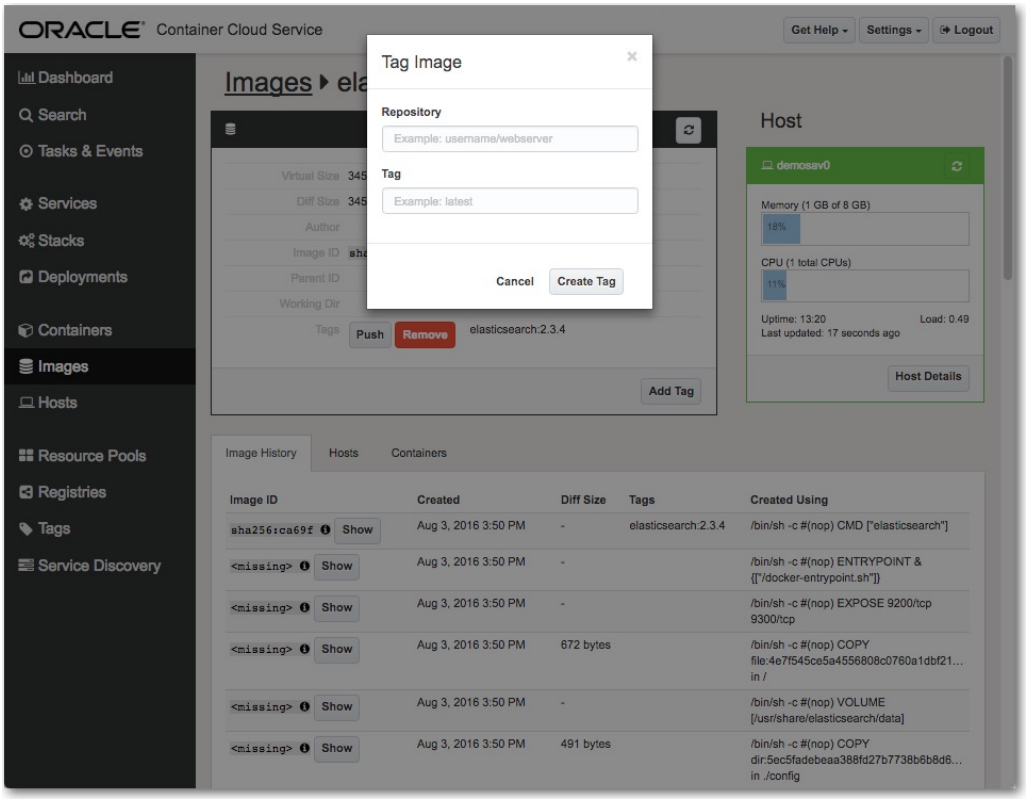
A typical scenario might be:

- The helloworld image is stored in the helloworld repository in the jsmith Docker registry. The most recent production version is version 5, and has the tags 'v5' and 'latest' in the repository.

- You've already created a service in the Container Console based on version 5 of the helloworld image, and deployed it. When you defined the service, you specified jsmith/helloworld:latest as the image to use.
- A developer pushes a new version of the helloworld image to the repository with the tag 'v6', which you want to test to assess whether it's production quality.
- You create a new service in the Container Console and specify jsmith/helloworld:v6 as the image to use. You deploy the service and test it.
- The image passes all the tests, so you decide to make version 6 the new production version.
- You add the 'latest' tag to the version 6 image in the Container Console and push the image to the jsmith/helloworld repository. The image that was previously tagged 'latest' is overwritten.

To push a Docker image to a Docker repository using the Oracle Container Cloud Service Container Console:

1. On the **Images** page of the Container Console, click the name of the Docker image that you want to push.
2. Click **Add Tag** to display the Tag Image window.



3. In the Tag Image window:
   a. Enter the name of the Docker repository to which you want to push the image in the **Repository** field, in the format <username>/<repository_name>. For example, jsmith/helloworld.

Note that the Docker repository you specify can be different to the repository that the image was pulled from originally. If the repository is in a different registry, a definition of that registry must already exist in Oracle Container Cloud Service. Also note that the user specified in the registry definition must have sufficient privileges to write to the registry (see Creating a Docker Registry Definition).

b. Enter a descriptive tag to add to the image when it's pushed to the Docker repository in the **Tag** field.

   The tag is up to you. For example, typical tags might be 'latest', '4.6.3 – 20160823', '7.5.2'.

c. Click **Create Tag**.

The new tag (which is the combination of the repository name and the description) is added to the image details.

4. Click the **Push** button beside the new tag to push the image to the Docker repository you specified with the tag you entered.

   If an image with the same tag already exists in the repository, it's overwritten. If the image didn't previously exist in the target registry, a new repository is created and the image is pushed to it.

Managing Deployments with Oracle Container Cloud Service

Learn about deployments and how to manage them.

Topics

- About Deployments
- Managing Deployments
- Scaling a Running Deployment with Oracle Container Cloud Service
- Monitoring the Health of a Deployment
- Stopping and Re-starting Service and Stack Deployments

About Deployments

An Oracle Container Cloud Service deployment comprises a service or stack in which Docker containers are managed, deployed, and scaled according to a set of orchestration rules that you've defined. Every time you deploy a service or a stack, Oracle Container Cloud Service creates a deployment. A single deployment can result in the creation of one or many Docker containers, across one or many hosts in a resource pool. And if you deploy the same service (or stack) multiple times, Oracle Container Cloud Service creates a corresponding number of deployments.

The orchestration functionality delivered by Oracle Container Cloud Service begins when you deploy a service and stack. The deployment of a service or stack generates a 'deployment object', or simply a 'deployment'. Docker images are pulled, Docker containers are created, and communication paths between services are set up. The orchestration rules specify how many containers to run per pool, per host, or per tag, and how to select the hosts on which to run the containers. You save default

orchestration policies with service and stack definitions, but you can override the default policies for specific deployments.

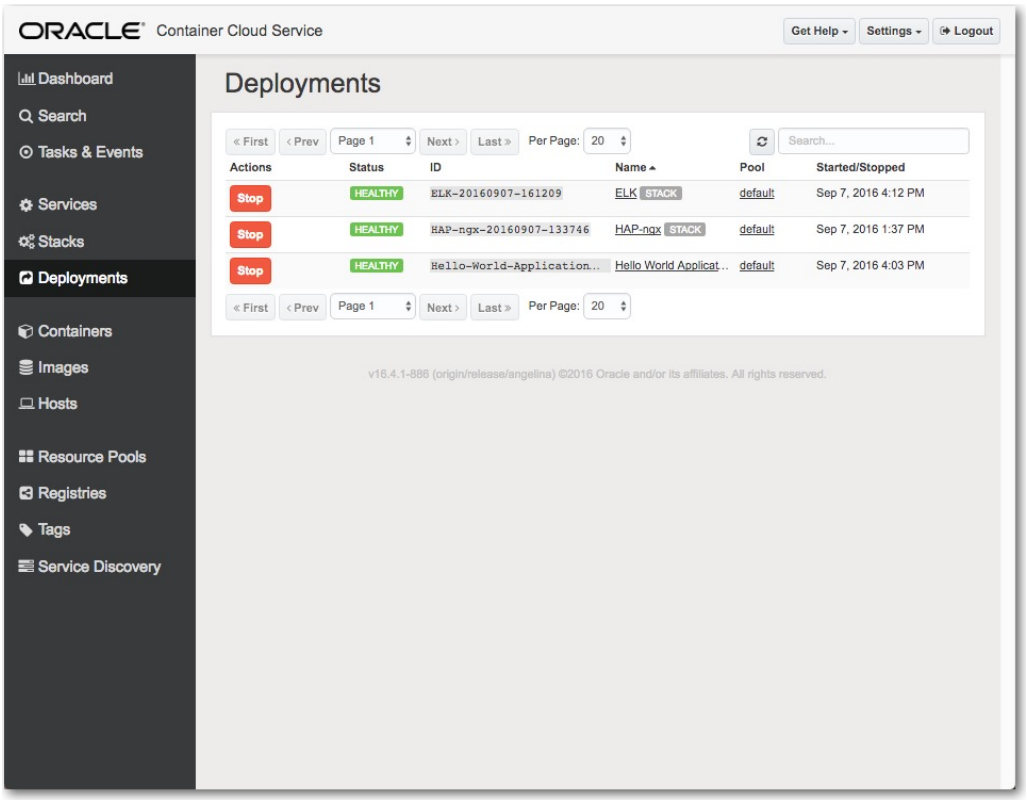
Each deployment is unique, and has its own lifecycle. A deployment is created when you click **Deploy**. If you subsequently stop a deployment, any containers created for the deployment are stopped and removed. If you restart a stopped deployment, it keeps the same deployment ID but new containers are created for it.

Managing Deployments

Having deployed a service or stack, you can use the Oracle Container Cloud Service Container Console to manage the deployment.

To manage a deployment using the Oracle Container Cloud Service Container Console:

1. On the **Deployments** page of the Container Console, review the deployments that have been created for deployed services and stacks.

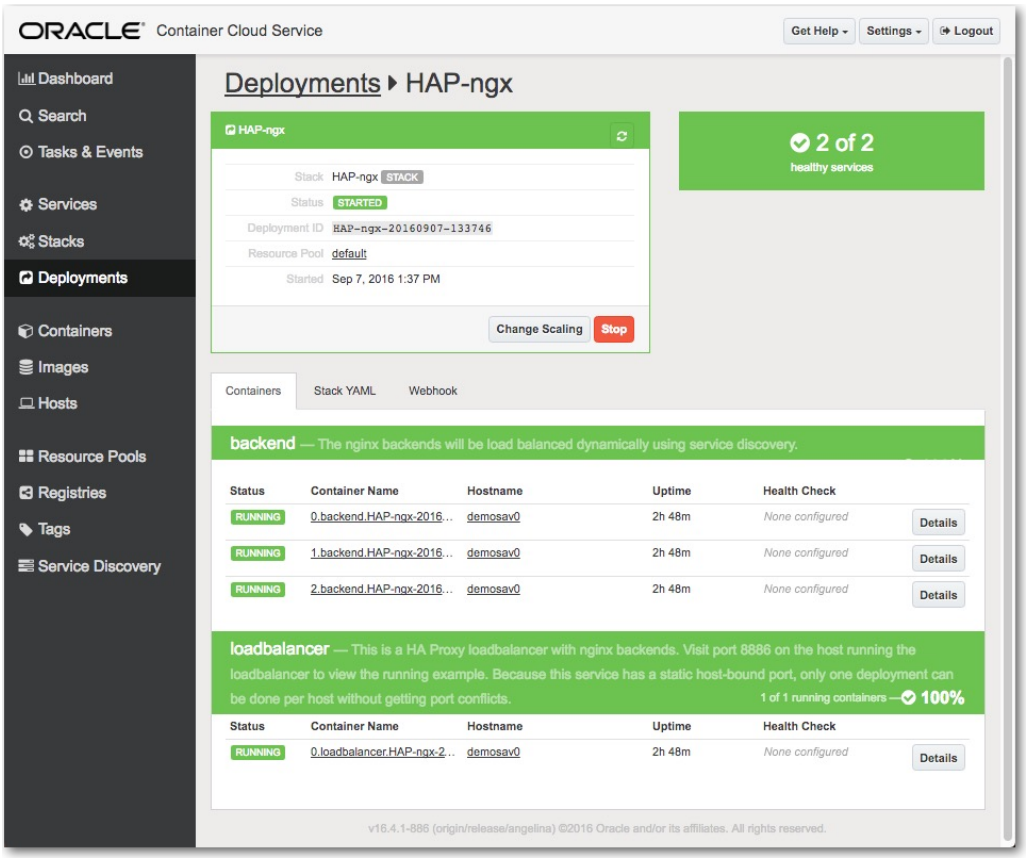


   Each deployment is shown on a separate line and uniquely identified by its deployment ID rather than by its name (which needn't be unique), along with the name of the pool to which the hosts running the deployment belong, and the name of the service or stack that's been deployed.

2. Perform management operations on the **Deployments** page using the **Start**, **Stop**, and **Remove** buttons. Note that you have to stop a deployment before you can remove it.

As well as performing management operations on deployments, typically you'll also want to view other information about the deployment such as configuration settings and runtime status.

3. Use the deployment **ID** field to locate the deployment running the service or stack for which you want to view detailed information, and click the name of the deployment.



In the top left region, you now see detailed information about the deployment, including runtime information and host usage. The color of the region's title bar shows the status of the deployment:

- green indicates the deployment is healthy and all Docker containers are running
- red indicates the deployment is unhealthy because one or more of its Docker containers isn't running
- grey indicates the deployment has stopped

4. Optional: Use the **Change Scaling** button and the **Stop** button to change the number of Docker containers for each service in the deployment and to stop all containers in the deployment respectively.
5. Use the tabbed pages to access information about each service in the deployment including:

Tab	Use to:
Containers	See the status of each of the Docker containers created for each service, the hosts they're running on, and the results of any health checks you've defined.
Stack YAML	See the YAML being executed by the deployment.
Webhook	See (and enable) a webhook url that will restart the deployment and pull the latest versions of any Docker images that have been updated. You'll find the webhook useful when integrating with continuous integration and continuous deployment (CI/CD) applications and pipelines. (A webhook is a simple event notification mechanism or callback that uses HTTP POST to send a message to a url when a particular event occurs.)

Scaling a Running Deployment with Oracle Container Cloud Service

When you deploy a service or stack, you specify how many Docker containers the deployment is to start. After it's been deployed, you can use the Oracle Container Cloud Service Container Console to increase (scale out) or decrease (scale in) the number of containers the running deployment uses.

To scale a running deployment in or out using the Oracle Container Cloud Service Container Console:

1. On the **Deployments** page of the Container Console, click the name of the deployment that you want to scale in or out.
2. On the **Deployments Details** page, click **Change Scaling**.
3. Increase or decrease the number of Docker containers the deployment uses.

   If the deployment is based on a stack, you can change the number of containers used by each service in the stack.
4. Click **Change**.

Monitoring the Health of a Deployment

Having deployed a service or stack, you can use the Oracle Container Cloud Service Container Console to monitor the status of that particular deployment so that you can take action to address any issues that arise.

To monitor the health of a deployment using the Oracle Container Cloud Service Container Console:

1. To see the status of deployed services and stacks:
   a. Go to the **Deployments** page of the Container Console to see a summary of the status of each deployed service or stack.

      The information includes the overall status of each deployment, and when each deployment was last started (if it's currently running) or when it was last stopped.
   b. Click the name of a deployed service or stack to see more detailed information about the status of that particular deployment.

The information includes status of individual services, when each service was last started (if it's currently running) or when it was last stopped, and results of service health checks.

2. To see the status of containers in a deployed service or stack:
   a. Go to the **Containers** page of the Container Console to see a summary of the status of each Docker container.
   b. Click the name of a Docker container to see detailed information about the container, including its current status, how long it has been running, and information it has output to log files.
3. To see even more detailed information about the status of each deployment:
   a. Go to the **Tasks & Events** page of the Container Console.
   b. Click one of **Event Log**, **Status Updates**, or **Tasks** according to the type of information you want to see.
   c. Select the information severity levels to display.
   d. Enter an appropriate search string so that you only see information about the services and stacks you're interested in.

Stopping and Re-starting Service and Stack Deployments

Periodically, you'll want to stop and restart service and stack deployments that you've previously started.

For example, you might want to deploy a more recent version of a service on the same host as an existing version. If you don't need access to deployed containers running the existing version, you can stop the existing deployment to release CPU and memory for containers running the new version. Having stopped the deployment, you can remove the deployment. Or you might decide to keep the deployment, just in case you want to restart and use it again later.

Topics

- Stopping a Service or Stack Deployment
- Re-starting a Service or Stack Deployment

Stopping a Service or Stack Deployment

Stop a deployment of a service or stack when you no longer require its containers to be running, perhaps because you want to temporarily (or permanently) free up resource on a host.

To stop a currently running service or stack deployment using the Oracle Container Cloud Service Container Console:

On the **Deployments** page of the Container Console, click the **Stop** button beside the name of the currently running deployment that you want to stop.
When you stop the deployment, the **Remove** and **Start** buttons appear. You can either permanently remove the deployment from the host by clicking **Remove**, or simply retain the deployment in case you want to restart it later.

Re-starting a Service or Stack Deployment

Re-start the deployment of a service or stack that's previously been stopped, perhaps to resume using an earlier version.

To restart a service or stack deployment using the Oracle Container Cloud Service Container Console:

On the **Deployments** page of the Container Console, click the **Start** button beside the name of the previously running deployment that you want to re-start.

Managing Docker Registries with Oracle Container Cloud Service

Learn about registries and how to manage them.

Topics

- About Docker Registries and Registry Definitions
- Managing Docker Registry Definitions
- Creating a Docker Registry Definition

About Docker Registries and Registry Definitions

A Docker registry is a system for storing and sharing tagged versions of Docker images. A Docker registry is organized into named repositories. Each Docker repository is associated with a particular version of one or more images.

Before Oracle Container Cloud Service can deploy a service based on a Docker image, it must already know the location of the Docker registry from which to pull the image. A registry definition containing location information must exist for each Docker registry from which Oracle Container Cloud Service will pull images. Similarly, if you're going to use Oracle Container Cloud Service to push images to a Docker registry, an appropriate registry definition must exist.

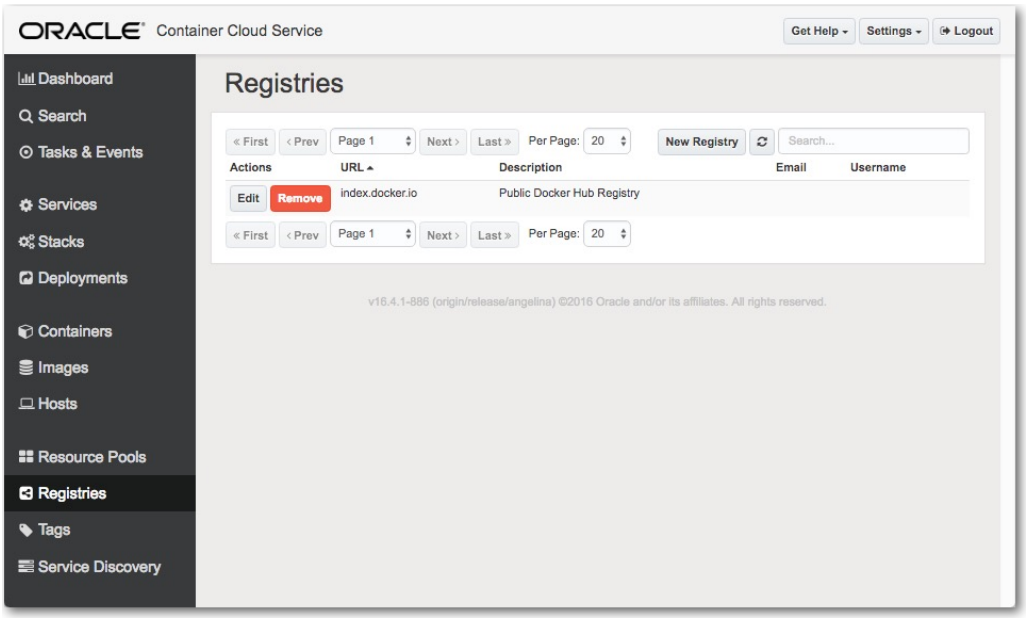
Oracle Container Cloud Service comes with a definition for the public Docker Hub registry out-of-the-box, so you can get started with public images right away. But you'll have to create a registry definition for any other public or private registry from which you want Oracle Container Cloud Service to pull images, or to which you want it to push images.

Managing Docker Registry Definitions

You manage definitions of public or private Docker registries that you want Oracle Container Cloud Service to pull images from or push images to, using the Oracle Container Cloud Service Container Console.

To manage registry definitions using the Oracle Container Cloud Service Container Console:

1. On the **Registries** page of the Container Console, review the list of registry definitions that you've created and the pre-defined registries that come with Oracle Container Cloud Service.



Each registry definition in the Oracle Container Cloud Service database is shown on a separate line, along with its description.

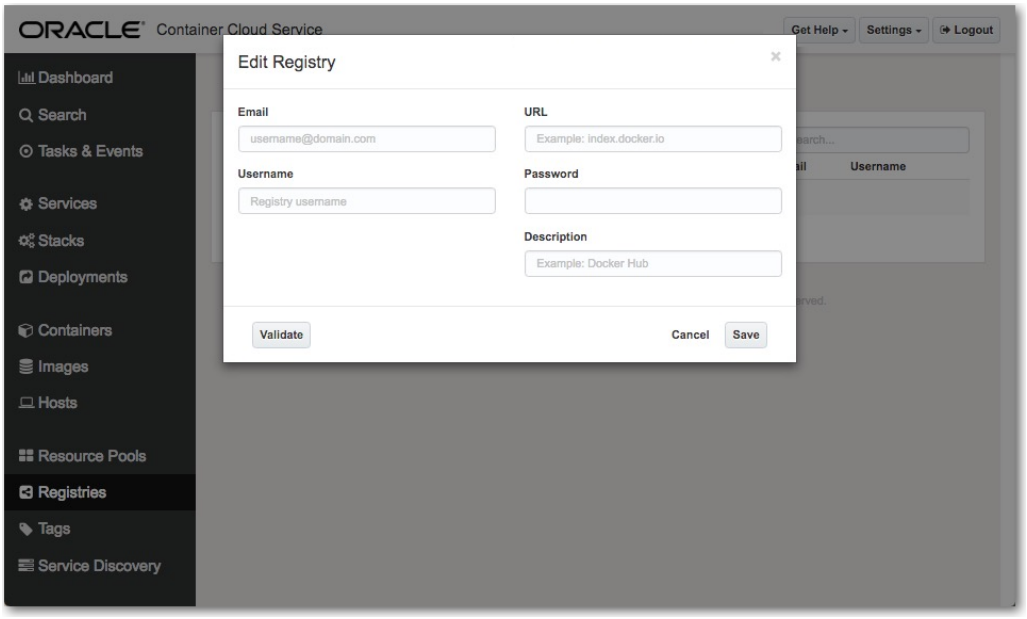
2. Perform management operations on the **Registries** page by:
   - clicking **Edit** beside the name of a registry definition to change any of the definition's properties (for example, password, description)
   - clicking **Remove** beside the name of a registry definition to delete the definition (Oracle Container Cloud Service will no longer pull images from or push images to repositories in the deleted registry)
3. If you want to create a new registry definition to pull images from or push images to, click **New Registry** (see Creating a Docker Registry Definition).

Creating a Docker Registry Definition

A registry definition must exist for each Docker registry that you want Oracle Container Cloud Service to pull images from or push images to.

To create a new Docker registry definition using the Oracle Container Cloud Service Container Console:

1. On the **Registries** page of the Container Console, click **New Registry**.



2. Enter the location of the public or private Docker registry in the **URL** field, as follows:
   - For Docker Hub specifically, append your account name to index.docker.io. For example, index.docker.io/jdoe
   - For other private registries, enter the appropriate URL. For example, registry.mydomain.com
3. If the registry is a private registry, enter the credentials to connect to the Docker registry in the **Username** and **Password** fields.
4. (Optional) Enter a valid email address (for example, to record who created the registry definition).
5. (Optional) Enter some meaningful text to describe the registry in the **Description** field.

   The description you enter appears as the registry name in the list of registries.
6. (Optional) Click **Validate** to confirm that Oracle Container Cloud Service can connect to your registry over SSL.

> **Note:**
>
> If SSL is not used, you might receive an error. However, the registry might still be reachable and usable.

7. Click **Save**.

   Oracle Container Cloud Service confirms that it can connect to the registry, and saves the registry definition.

You can now use the Docker registry you've just defined when specifying the image on which a service is based (see Creating a Service with Oracle Container Cloud Service) or when pushing an image (see Pushing a Docker Image to a Docker Repository).

Managing Tags with Oracle Container Cloud Service

Learn about tags, how to use them to manage your Docker environment, how to create them, and how to manage them.

Topics

- About Tags
- Creating and Assigning Tags
- Managing Tags Using the Tags Page

About Tags

Oracle Container Cloud Service tags are a way to categorize and organize resource pools and the hosts within them, enabling you to manage your Docker environment more effectively.

When defining orchestration policies, tags give you fine-grained control over which hosts in a resource pool to deploy a service or stack on.

Having created a new tag, you can use it:

- for resource pools
- for hosts, to collectively identify multiple hosts on which a service can be deployed
- for services in the Service Editor, when specifying the hosts on which a service is to run using the **per-tag (across hosts in this pool with tag of...** availability option.
- for deployments, to override a service's default orchestration details and specify the hosts on which the service is to run

You can also use tags as search criteria when looking for particular hosts across your entire managed Docker environment.

Note that the tags you use to organize resource pools and hosts are totally unrelated to the tags you use when pushing Docker images to a repository (see Pushing a Docker Image to a Docker Repository).

Creating and Assigning Tags

You can create and assign tags using the Oracle Container Cloud Service Container Console.

Tags are a useful way to identify Oracle Container Cloud Service objects with common characteristics, or that you want to use in a similar way.

Topics

- Creating New Tags Using the Tags Page
- Creating and Assigning New Tags When Defining Hosts and Resource Pools

Creating New Tags Using the Tags Page

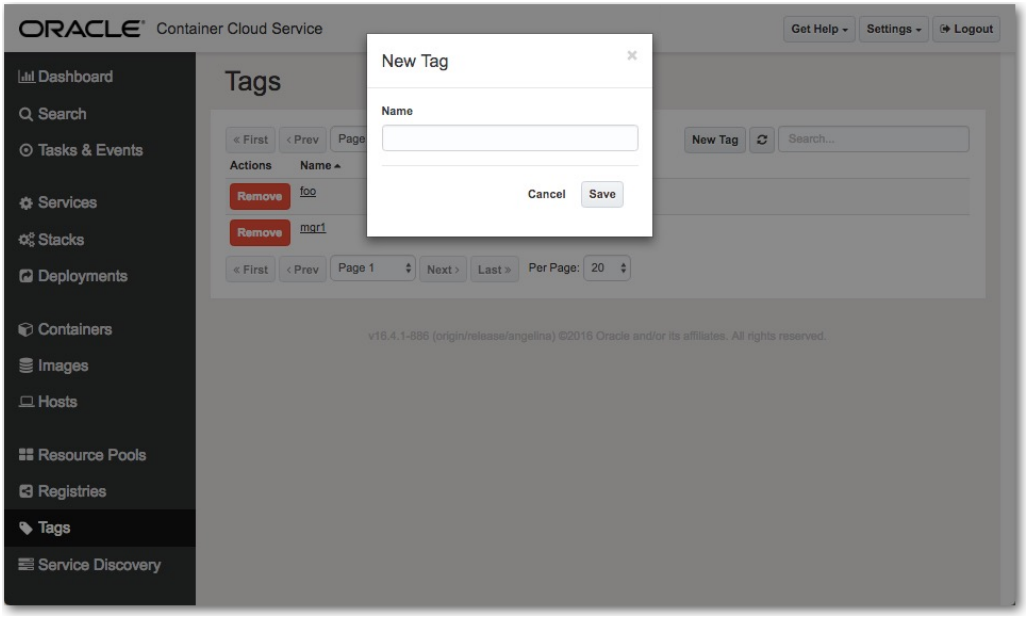
You can create new tags in advance of assigning them using the **Tags** page of the Oracle Container Cloud Service Container Console.

Using the **Tags** page to create new tags is handy when:

- you want to enter several tags in advance
- you want everyone on the team to use the same set of tags
- you want the tags to adhere to particular naming conventions

To create new tags using the **Tags** page of the Oracle Container Cloud Service Container Console:

1. On the **Tags** page of the Container Console, click **New Tag** to display the New Tag window.



2. Enter a unique name for the new tag.

   For example:

   - you might enter `LB` as the name of a tag you want to assign to hosts that will always run the load balancer container
   - you might enter `log` as the name of a tag you want to assign to hosts that will always run the kibana container for an ELK stack

   Tag names are case-sensitive. So although tag names must be unique, you could have two tags with identical names but different capitalization. For example, you could have two tags, one named ‘log’ and the other named ‘Log’.

3. Click **Save**.

Creating and Assigning New Tags When Defining Hosts and Resource Pools

You can create new tags when defining hosts and resource pools.

Creating new tags when defining hosts and resource pools is handy when:

- you're defining tags for your own use
- you're familiar with your team's tag naming standards

To create new tags when defining hosts and resource pools using the Oracle Container Cloud Service Container Console:

1. On the **Hosts** page of the Container Console, do one of the following to display the Add Tag window for the host to which you want to assign a new tag:
   - select the checkbox beside the name of the host and click **Add Tag to Selected**
   - select the name of the host to display the **Host Details** page, and click **Add Tag**
2. Click **New Tag** to display the New Tag window.
3. Enter a unique name for the new tag.

   For example:
   - you might enter `LB` as the name of a tag you want to assign to hosts that will always run the load balancer container
   - you might enter `log` as the name of a tag you want to assign to hosts that will always run the kibana container for an ELK stack

   Tag names are case-sensitive. So although tag names must be unique, you could have two tags with identical names but different capitalization. For example, you could have two tags, one named 'log' and the other named 'Log'.
4. Click **Save**.

Managing Tags Using the Tags Page

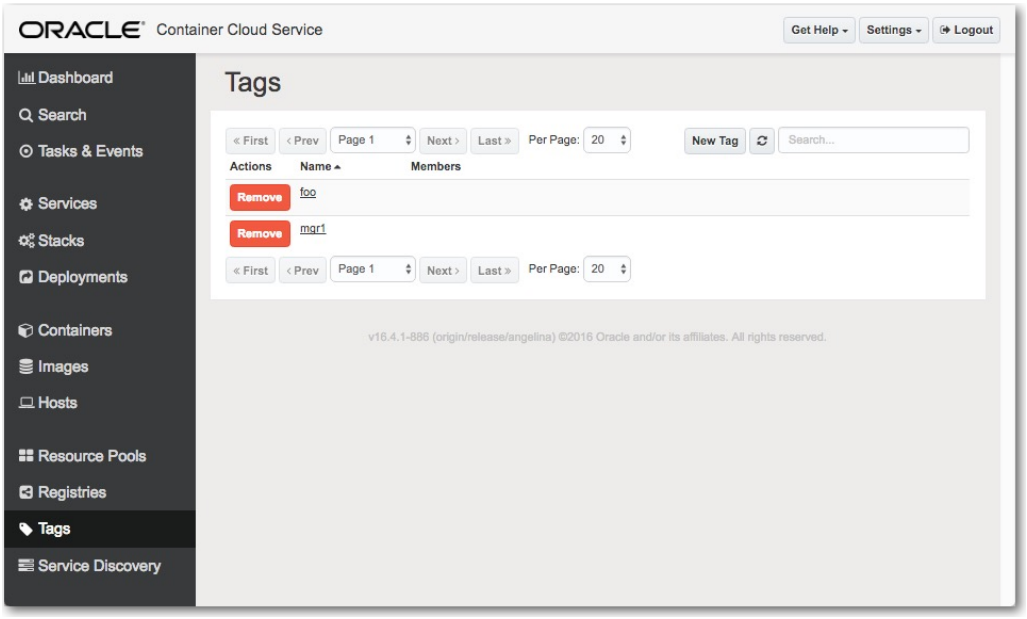
You can manage tags using the **Tags** page of the Oracle Container Cloud Service Container Console.

Using the **Tags** page to manage tags is handy when:

- you want to see the resource pools and hosts to which a specific tag is assigned
- you want to remove a tag from the system
- you want to assign a tag to (or remove a tag from) multiple hosts and resource pools

To manage tags using the **Tags** page using the Oracle Container Cloud Service Container Console:

1. On the **Tags** page of the Container Console, review the list of tags that have been defined and the number of resource pools and hosts to which they've been assigned.



2. Optional: If you want to delete a tag that hasn't been assigned to any resource pools or hosts, on the **Tags** page click the **Remove** button beside the name of the tag.
3. Optional: If you want to remove a tag that has been assigned to one or more resource pools or hosts:
   a. On the **Tags** page, click the name of the tag to see the resource pools and hosts that have been assigned that tag.
   b. Click the **Remove from Tag** button beside the name of every resource pool or host that's been assigned the tag.
   c. When you've removed all resource pools or hosts that have been assigned the tag, go back to the **Tags** page and click the **Remove** button beside the name of the tag to delete the tag.
4. Optional: If you want to assign an existing tag to one or more resource pools or hosts:
   a. On the **Tags** page, click the name of the tag to see the resource pools and hosts that have already been assigned that tag.
   b. Click **Add Pools** and **Add Hosts** to assign the tag to additional resource pools or hosts.

Monitoring Tasks, Events, and Status Updates with Oracle Container Cloud Service

Learn about tasks, events, status updates, and how to monitor them.

Topics

- About Tasks, Events, and Status Updates
- Monitoring Tasks, Events, and Status Updates

About Tasks, Events, and Status Updates

Tasks, events, and status updates enable you to monitor the Docker environment managed by Oracle Container Cloud Service.

Events are individual, discrete operations. For example, if you deploy a service using the Oracle Container Cloud Service Container Console, five separate events might be initiated:

- add a deployment
- pull a Docker image from a registry
- confirm that the docker image has been pulled
- create a docker image on host
- start the docker image

Status updates report the progress of particular recurring events and tasks (for example, starting a deployment, or running a health check). Status updates have a lifecycle, meaning their severity and description can change. For example, a container creation event might initially fail and generate a status update with a severity of Error. However, if the same container creation event is re-run successfully, the severity of the existing status update is downgraded to Cleared and its description changes. Note that not all events generate status updates.

Tasks are cancelable actions created by Oracle Container Cloud Service in response to your requests. Tasks are long-running, and can involve multiple events and objects. For example, if you stop several containers, Oracle Container Cloud Service creates a task to carry out the action on your behalf because it could take some time for multiple events on each of the containers to complete. All tasks generate status updates, and can be canceled while they are still running.

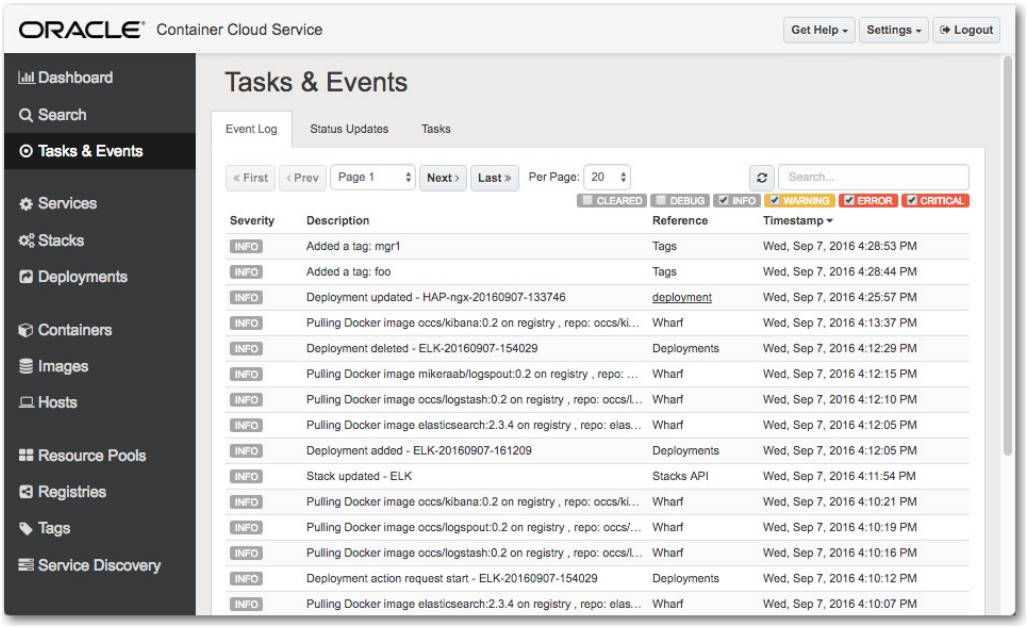
Oracle Container Cloud Service classifies events and status updates with a set of severity levels.

Monitoring Tasks, Events, and Status Updates

Find out how to monitor tasks, events, and status updates.

To monitor tasks, events, and status updates using the Oracle Container Cloud Service Container Console:

1. Go to the **Tasks & Events** page of the Container Console.



2. Use the tabs to monitor tasks, events, and status updates as follows:

Option	Use to:
Event Log	Monitor events of different severity levels. For example, events that occur when you deploy a service might include: • Docker pulling an image • Oracle Container Cloud Service creating a deployment • Docker creating a container • Docker starting a container
Status Updates	Monitor status updates of different severity levels, generated by events and tasks. For example, status updates when you deploy a service might include reporting successes for the following events: • parsing deployment YAML • getting a list of hosts • finding a host on which to deploy the service • starting the Docker container
Tasks	Monitor tasks created by Oracle Container Cloud Service in response to your requests. For example, if you stop several Docker containers, Oracle Container Cloud Service creates a task. If the task is still running, you can cancel it on this tab. Click **Details** to find out more about a particular task, including the user who requested it.

Managing Profile Settings with Oracle Container Cloud Service

Learn about when and how to change and view profile settings.

Topics

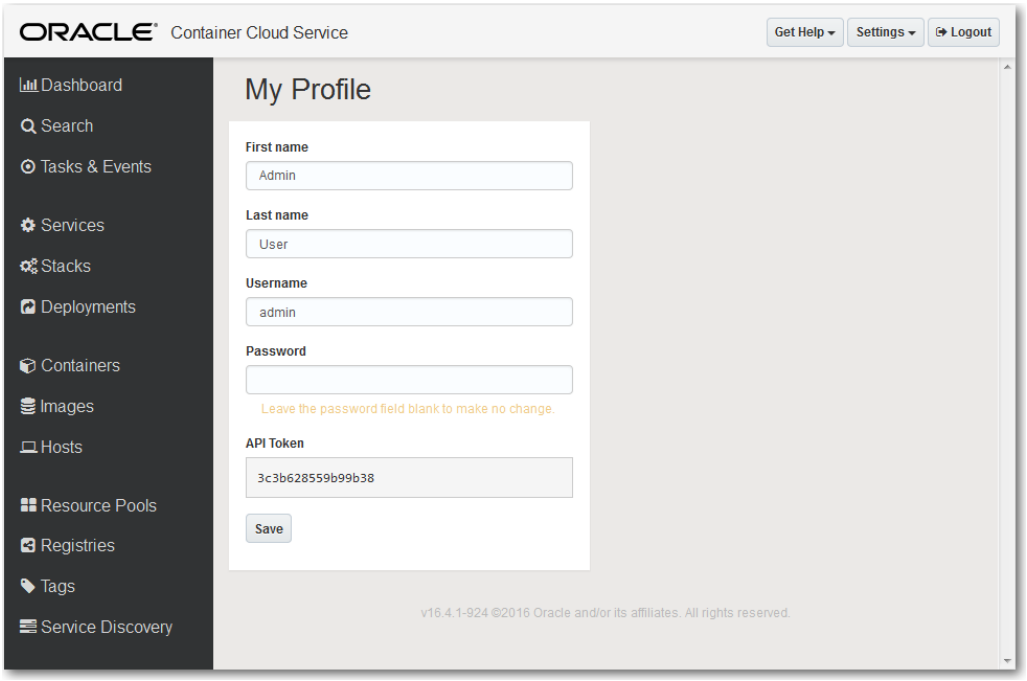
• Changing and Viewing Profile Settings

Changing and Viewing Profile Settings

Find out how to change the username and password with which you sign into Oracle Container Cloud Service Container Console, and how to see the value of the API token that uniquely identifies your username.

To change your username and password, and see the value of the API token using the Oracle Container Cloud Service Container Console:

1. From any Container Console page, select **My Profile** from the **Settings** menu.



2. Use the fields as follows:

Option	Use to:
First name	Associate a first name with the username, for information purposes only (optional).
Last name	Associate a last name with the username, for information purposes only (optional).
Username	Specify the name to use to sign into the Container Console. If you change this name, note the following: • Make sure you take a note of the new username because there's no way to recover or reset it later. • When you click **Save**, the Container Console session ends immediately and you're prompted to re-enter the username and password. • A new API token value is generated, so you'll have to update any scripts that included the old API token.

Option	Use to:
Password	Specify the password to use when signing into the Container Console with the name shown in the **Username** field. If you change other values on the **My Profile** page but want to keep the password unchanged, simply leave the **Password** field blank.
API Token	See the value to include as the Bearer token in the authorization headers of scripts that call Oracle Container Cloud Service REST API methods. Note that a new API token value is generated if you change the value in the **Username** field, so you'll have to update any scripts that included the old API token. See About the Oracle Container Cloud Service REST API.

3. Click **Save** to apply the changes. If you changed the value in the **Username** field, the Container Console session ends immediately and you're prompted to re-enter the username and password.

Using Template Functions and Arguments with Oracle Container Cloud Service

Learn about incorporating template functions and arguments in stack and service definitions.

Topics

- About Template Functions and Arguments
- Adding Template Functions and Arguments to Service Definitions
- Adding Template Functions and Arguments to Stack Definitions
- Template Function and Argument Reference

About Template Functions and Arguments

Template functions enable you to programmatically access and update details about service and stack deployments managed by Oracle Container Cloud Service. Template arguments expose a deployment's properties so that you can manipulate them using template functions.

You can incorporate Oracle Container Cloud Service template functions and arguments into service and stack definitions using the Container Console's Service Editor and Advanced (YAML) Stack Editor to programmatically get and set values for configuration options and environment variables.

Typically, template functions and arguments return values from:

- the Key/Value store in Oracle Container Cloud Service's Service Discovery database
- Oracle Container Cloud Service environment variables (with names starting 'OCCS_') related to the deployment

One way to use template functions and arguments is to call them to set environment variables in a running container's host. Application code in the container can then use those environment variables.

You'll also find template functions and arguments particularly useful when you want to create generic services and stacks that can be reused to meet different requirements. For example, you might want to define a stack to deploy a blogging application that comprises WordPress and a mariadb database. If you only have one such blogging application to deploy, you can create a stack definition that explicitly specifies the Docker images to run as Docker containers for the application, the host ports to use when the application is deployed, and the database root password.

Such a stack definition is shown below:

```
# https://hub.docker.com/_/wordpress/
# Example docker-compose.yml for wordpress:
# Run docker-compose up, wait for it to initialize completely,
# and visit http://localhost:8080 or http://host-ip:8080.
wordpress:
  image: wordpress:latest
  ports:
    - 8080:80
  links:
    - db:mysql

db:
  container_name: db
  image: mariadb:latest
  environment:
    - MYSQL_ROOT_PASSWORD=Eric-Password
```

So far, so good. But if you have ten similar blogging applications to deploy, creating and maintaining a separate stack definition for each application could quickly become tiresome.

By incorporating template functions in a stack definition, you can reuse the same stack definition to deploy multiple different applications. In the case of deploying the ten different blogging applications, you could incorporate template functions into a single stack definition to interrogate the Key/Value store in the Oracle Container Cloud Service Service Discovery database and return the port and the database root password to use for each blogging application, as well as the name and location of each application's data partition mount point.

Such a stack definition is shown below:

```
# https://hub.docker.com/_/worddivss/
# Example docker-compose.yml for wordpress:
# Run docker-compose up, wait for it to initialize completely,
# and visit http://localhost:8080 or http://host-ip:8080.
wordpress:
  image: wordpress:latest
  ports:
    - {{kv_get .EnvironmentID .StackID .ServiceID "port"}}:80
  links:
    - db:mysql

db: # ensure db sorts before wordpress
  container_name: db
  image: mariadb:latest
```

```
  environment:
    - MYSQL_ROOT_PASSWORD={{kv_get {{.ServiceID}} "pass" }}
  volumes:
    - /NFS/{{.EnvironmentID}}/{{.DeploymentID}}/{{.ServiceID}}/
{{.ServiceSlot}}:/mnt/data
```

Adding Template Functions and Arguments to Service Definitions

You can add template functions to a service definition, and pass in template arguments, enabling you to programmatically access and update details about the service when it's deployed.

To add a template function or argument to a service definition:

1. On the **Services** page of the Container Console, click the **Edit** button beside the service name to display the Service Editor.
2. Add the required template function or template argument on the **Builder** tab, the **Docker Run** tab, or the **YAML** tab.

   You'll usually find it easiest to add and edit template functions and arguments using the **YAML** tab.

   Note that template function calls must always be enclosed within double curly braces as `{{`*`function-name`*`}}`, argument names must always be preceded by a period, and that function and argument names must always be capitalized exactly as shown in Template Function and Argument Reference. Beyond that, the steps to add template functions or arguments vary from tab to tab as follows:

   - On the **Builder** tab, click **Add** or **Edit** beside a configuration option, and enter the template function or argument in the appropriate text field. For example:
     - to call the `api_token` function to return the token of the deployment creator and set an environment variable named MY_DEPLOY_CREATOR_TOKEN_ENV_VAR to that value, you'd select Environment Variables from the **Available Options** list, click **Add**, and then type `MY_DEPLOY_CREATOR_TOKEN_ENV_VAR` in the **Name** field and `{{api_token}}` in the **Value** field
     - to use the `.ServiceID` template argument to set the value of an environment variable named MY_SERVICE_ID_ENV_VAR to the service ID, you'd select Environment Variables from the **Available Options** list, click **Add**, and then type `MY_SERVICE_ID_ENV_VAR` in the **Name** field and `{{.ServiceID}}` in the **Value** field
   - On the **Docker Run** tab, enter the template function or argument in the appropriate location. For example:
     - to call the `api_token` function to return the token of the deployment creator and set an environment variable named MY_DEPLOY_CREATOR_TOKEN_ENV_VAR to that value, you'd enter:

       ```
       -e="MY_DEPLOY_CREATOR_TOKEN_ENV_VAR={{api_token}}"
       ```

  - to use the `.ServiceID` template argument to set the value of an environment variable named MY_SERVICE_ID_ENV_VAR to the service ID, you'd enter:

    ```
    -e="MY_SERVICE_ID_ENV_VAR={{.ServiceID}}"
    ```

    Note that on the **Docker Run** tab, any double quotation marks surrounding arguments in function calls must be preceded by a backslash (\) escape character.

- On the **YAML** tab, enter the template function or argument in the appropriate location. For example:
  - to call the `api_token` function to return the token of the deployment creator and set an environment variable named MY_DEPLOY_CREATOR_TOKEN_ENV_VAR to that value, you'd enter:

    ```
    environment:
      - 'MY_DEPLOY_CREATOR_TOKEN_ENV_VAR={{api_token}}'
    ```

  - to use the `.ServiceID` template argument to set the value of an environment variable named MY_SERVICE_ID_ENV_VAR to the service ID, you'd enter:

    ```
    environment:
      - 'MY_SERVICE_ID_ENV_VAR={{.ServiceID}}'
    ```

  To find out the template functions and arguments that are available, along with the correct syntax to use, see Template Function and Argument Reference.

3. Click **Save** to save your changes and close the Service Editor.

When the service is next deployed, the template functions and arguments you've specified will be used.

Adding Template Functions and Arguments to Stack Definitions

You can add template functions to a stack definition, and pass in template arguments, enabling you to programmatically access and update details about the stack when it's deployed.

To add a template function or argument to a stack definition:

1. On the **Stacks** page of the Container Console, click the **Edit** button beside the stack name to display the Stack Editor.
2. Click **Advanced Editor** to display the Advanced (YAML) Stack Editor, showing the YAML to deploy the stack.
3. Edit the YAML and add the required template function or template argument. Note that template function calls must always be enclosed within double curly braces as {{*function-name*}}, argument names must always be preceded by a period, and that function and argument names must always be capitalized exactly as shown in Template Function and Argument Reference.

   For example, say you're defining a stack to deploy a number of different applications, each requiring a mariadb service. You've added a number of key value pairs to the Key/Value store in the Oracle Container Cloud Service's Service

Discovery database to specify a different root password for each application. In the Key/Value store, each application's password is the value, and the corresponding key is the service ID and the string "pass", separated by a "/". To extract the password for a particular application when deploying the stack and assign it to the `MYSQL_ROOT_PASSWORD` environment variable, you can use the `kv_get` template function and the `.ServiceID` template argument. You'd add `MYSQL_ROOT_PASSWORD={{kv_get {{.ServiceID}} "pass" }}` to the YAML. The db section of the stack definition would now look like:

```
db: # ensure db sorts before wordpress
  container_name: db
  image: mariadb:latest
  environment:
    - MYSQL_ROOT_PASSWORD={{kv_get {{.ServiceID}} "pass" }}
```

To find out the template functions and arguments that are available, along with the correct syntax to use, see Template Function and Argument Reference.

4. Click **Done** to save your changes and close the Advanced (YAML) Stack Editor.

When the stack is next deployed, the template functions and arguments you've specified will be used.

Template Function and Argument Reference

When including template functions and arguments you can include in service and stack definitions using the Oracle Container Cloud Service's Service Editor and Advanced (YAML) Stack Editor, you must use the correct syntax.

About Template Functions and Arguments

For template functions and arguments to be evaluated by Docker Compose, you must insert them into the YAML beneath the correct service level keys:

- If you're editing the YAML indirectly using the Service Editor's **Builder** tab or **Docker Run** tab, provided you've used the template functions and arguments appropriately to set configuration options, they are automatically included in the YAML in the correct location.
- If you're editing the YAML directly using the Service Editor's **YAML** tab or the Advanced (YAML) Stack Editor, you manually insert the template functions and arguments beneath the following service level keys:
  - command
  - environment
  - hostname
  - labels
  - network_mode
  - ports
  - volumes
  - volumes_from

Template Functions

Use the following template functions when defining services in the Oracle Container Cloud Service's Service Editor and Advanced (YAML) Stack Editor. Note that template function calls must always be enclosed within double curly braces as {{*function-name*}}, and that function names must always be capitalized exactly as shown below.

Template Function and *parameters*	**Description**	**Example Value of Environment Variable in Deployment**	**Example Function Call with Arguments**	**Example Evaluation**
api_token	Extracts the token of the deployment creator.	(not applicable)	USER_API_TOKEN={{api_token}}	USER_API_TOKEN=36bf51e24218d492
expand "*separator*" "*string*" *min qty*	Ranges over an index from a given minimum value up to the number specified by qty minus 1, and appends that value to a string separated by a separator.	(not applicable)	MY_ZKS={{expand "," "zk-" 0 3}}	MY_ZKS=zk-0,zk-1,zk-2
hostip_for_interface *host-interfaces-and-ip-addresses* "*interface-to-match*"	Extracts the IP address for an interface of the host running the container.	OCCS_HOSTIPS=eth0: 172.16.44.183, eth1: 10.9.1.2, docker0: 172.17.42.1	MY_ETH1_IP2={{hostip_for_interface .HostIPs "eth1"}}	MY_ETH1_IP2=10.9.1.2
hostip_with_prefix *host-interfaces-and-ip-addresses* "*prefix-to-match*"	Extracts the IP address that matches the prefix of the host running the container.	OCCS_HOSTIPS=eth0: 172.16.44.183, eth1: 10.9.1.2, docker0: 172.17.42.1	MY_172_16_IP={{hostip_with_prefix .HostIPs "172.16."}}	MY_172_16_IP=172.16.44.183
hostips_keys *host-interfaces-and-ip-addresses*	Lists the interfaces of the host running the container.	OCCS_HOSTIPS=eth0: 172.16.44.183, eth1: 10.9.1.2, docker0: 172.17.42.1	MY_INTERFACES={{hostips_keys .HostIPs}}	MY_INTERFACES=eth0 eth1 docker0
hostips_values *host-interfaces-and-ip-addresses*	Lists the IP addresses of the host running the container.	OCCS_HOSTIPS=eth0: 172.16.44.183, eth1: 10.9.1.2, docker0: 172.17.42.1	MY_IPS={{hostips_values .HostIPs}}	MY_IPS=172.16.44.183 10.9.1.2 172.17.42.1
kv_get "*arg-1*" "*arg-2*" "*arg-n*"	Accepts any number of arguments, joins them together using a "/" separator, then uses the generated key to look up a value in the Key/Value store. If a value is not found, the lookup results in an error and templating fails.	(not applicable)	MY_TEST={{kv_get "arbitrarily" "defined" "key" "path"}}	MY_TEST=172.16.44.183
kv_path "*arg-1*" "*arg-2*" "*arg-n*"	Accepts any number of arguments and joins them together using a "/" separator as the evaluated result. kv_get calls kv_path to generate the path key.	(not applicable)	MY_TEST_PATH={{kv_path "arbitrarily" "defined" "key" "path"}}	MY_TEST_PATH=arbitrarily/defined/key/path

Template Function and *parameters*	**Description**	**Example Value of Environment Variable in Deployment**	**Example Function Call with Arguments**	**Example Evaluation**
map_get "*key-value-pairs-string*" "*record-separator*" "*key-value-separator*" "*key*"	Lists the value of a particular key in a specified string of key/value pairs, given a list of key/value pairs, a record separator, and a key/value separator.	OCCS_HOSTIPS=eth0: 172.16.44.183, eth1: 10.9.1.2, docker0: 172.17.42.1	MY_ETH1_IP2={{map_get .HostIPs "," ":" "eth1"}}	MY_ETH1_IP2=10.9.1.2
map_keys "*key-value-pairs-string*" "*record-separator*" "*key-value-separator*"	Lists all the keys in a specified string of key/value pairs, given a record separator, and key/value separator.	OCCS_HOSTIPS=eth0: 172.16.44.183, eth1: 10.9.1.2, docker0: 172.17.42.1	MY_ETH1_IP2={{map_keys .HostIPs "," ":"}}	MY_ETH1_IP2=eth0 eth1 docker0
map_values "*key-value-pairs-string*" "*record-separator*" "*key-value-separator*"	Lists all the values in a specified string of key/value pairs, given a record separator, and key/value separator.	OCCS_HOSTIPS=eth0: 172.16.44.183, eth1: 10.9.1.2, docker0: 172.17.42.1	MY_IPS={{map_values .HostIPs "," ":"}}	MY_IPS=172.16.44.183 10.9.1.2 172.17.42.1
plus *number-1 number-2*	Adds a value with another value.	(not applicable)	MY_1542={{plus 1500 42}}	MY_1542=1542
plus_ip *ip-address number*	Adds a number to an IP address.	OCCS_SERVICE_SLOT=2	MY_OTHERIP={{plus_ip "10.1.0.100" .ServiceSlot}}	MY_OTHERIP=10.1.0.102
plus_suffix "*separator*" "*string*" *number*	Extracts the suffix of a given string (up to an occurrence of the specified separator character), adds the specified number to the extracted suffix, and returns the string with the updated suffix.	OCCS_SERVICE_SLOT=2	MY_OTHERIP={{plus_suffix "." "10.1.0.100" .ServiceSlot}}	MY_OTHERIP=10.1.0.102
proxy "*service-id:port*"	A multi-host directive that routes the endpoint of given SERVICE_ID:PORT The proxy directive creates a TCP proxy that the container uses to reach a remote service endpoint. When the container makes a connection to the endpoint, the proxy looks up the given service in the Service Discovery database. If the proxy finds matching services, it chooses one at random and proxies the incoming connection to it. If the proxy finds no matching services, it closes the incoming connection and an error is logged.	WORDPRESS_DB_HOST={{ proxy "SERVICE_ID:PORT" }}	WORDPRESS_DB_HOST={{ proxy "db:3306" }}	WORDPRESS_DB_HOST=172.17.42.1:20009

Template Function and *parameters*	Description	Example Value of Environment Variable in Deployment	Example Function Call with Arguments	Example Evaluation
sd_deployment_containers_path "*sibling-service*" *port*	Generates the Service Discovery container path given a sibling service and port.	OCCS_DEPLOYMENT_ID=NGINX-load-balanced-with-HAProxy	MY_BACKEND_KEY={{sd_deployment_containers_path "nginx-backend-eg" 80}}	MY_BACKEND_KEY=apps/nginx-backend-eg-NGINX-load-balanced-with-HAProxy-80/containers

Template Arguments

Use the following arguments when defining services in the Oracle Container Cloud Service's Service Editor and Advanced (YAML) Stack Editor.

The examples below assume you've defined a stack with the ID "user-blogs" to deploy a blogging application named "Eric's Blog". The "user-blogs" stack comprises a WordPress service and a mariadb service.

Note that argument names must always be preceded by a period and capitalized exactly as shown.

Template Argument	Description	Example Use to set MY_ENV_VAR	Example Evaluation
.ContainerHostname	The hostname used within the container.	MY_ENV_VAR={{.ContainerHostname}}	MY_ENV_VAR=2.wordpress.erics-blog
.ContainerName	The name of the container.	MY_ENV_VAR={{.ContainerName}}	MY_ENV_VAR=2.wordpress.erics-blog
.CreatedBy	The username who created the deployment.	MY_ENV_VAR={{.CreatedBy}}	MY_ENV_VAR="admin"
.DeploymentID	The deployment ID for a deployment.	MY_ENV_VAR={{.DeploymentID}}	MY_ENV_VAR="erics-blog"
.DeploymentName	The deployment name for a deployment.	MY_ENV_VAR={{.DeploymentName}}	MY_ENV_VAR="Eric's Blog"
.EnvironmentID	The optional environment for a deployment. If not specified for a deployment, "default" is used.	MY_ENV_VAR={{.EnvironmentID}}	MY_ENV_VAR="default"
.HostID	The GUID of the host running the container.	MY_ENV_VAR={{.HostID}}	MY_ENV_VAR=172.16.44.183
.HostIPs	A map of interface to IP addresses for the host.	MY_ENV_VAR={{.HostIPs}}	MY_ENV_VAR=eth0: 172.16.44.183, eth1: 10.9.1.2, docker0: 172.17.42.1
.Hostname	The hostname of the host running the container.	MY_ENV_VAR={{.Hostname}}	MY_ENV_VAR=somehost.example.com
.KVDeploymentPath	A shortcut for joining the environment and the deployment ID.	MY_ENV_VAR={{.KVDeploymentPath}}	MY_ENV_VAR=production/erics-blog
.KVServicePath	A shortcut for joining the environment, deployment ID, and service ID.	MY_ENV_VAR={{.KVServicePath}}	MY_ENV_VAR=production/erics-blog/wordpress
.KVServiceSlotPath	A shortcut for joining the environment, deployment ID, service ID, and service slot.	MY_ENV_VAR={{.KVServiceSlotPath}}	MY_ENV_VAR=production/erics-blog/wordpress/2

Template Argument	Description	Example Use to set MY_ENV_VAR	Example Evaluation
.ServiceID	The ID of the service used to create the container.	MY_ENV_VAR={{.ServiceID}}	MY_ENV_VAR=wordpress
.ServiceIDs	The service IDs of the deployment used to create the containers.	MY_ENV_VAR={{.ServiceIDs}}	MY_ENV_VAR=db wordpress
.ServiceSlot	The integer slot of the container from 0 to quantity-1 of the service. If three containers were specified for the WordPress deployment for roberts-blog, .ServiceSlot would be 0 for the first container, 1 for the second, and 2 for the third.	MY_ENV_VAR={{.ServiceSlot}}	MY_ENV_VAR=0
.ServiceSlotStr	The slot in string format.	MY_ENV_VAR={{.ServiceSlotStr}}	MY_ENV_VAR=0
.StackID	The ID of the stack used to define the deployment.	MY_ENV_VAR={{.StackID}}	MY_ENV_VAR="user-blogs"

Using the Oracle Container Cloud Service REST API

Learn about the Oracle Container Cloud Service REST API and how to use it to manage your Docker environment.

Topics

- About the Oracle Container Cloud Service REST API
- Calling the Oracle Container Cloud Service REST API
- Adding Keys and Values to the Key/Value Store Database
- Example REST API Calls to Access the Key/Value Store Database

About the Oracle Container Cloud Service REST API

You can use the Oracle Container Cloud Service REST API to incorporate Oracle Container Cloud Service functionality into your web applications.

The Oracle Container Cloud Service REST API exposes Oracle Container Cloud Service functionality as HTTP methods to enable you to incorporate the functionality using Representational State Transfer (REST) requests and responses. REST offers a simple, light-weight API for applications that don't require the server to maintain state information or message exchange. See Calling the Oracle Container Cloud Service REST API.

If you want to hold information in persistent storage, you can save keys and values in the Oracle Container Cloud Service Key/Value Store database for retrieval by calls to the Oracle Container Cloud Service REST API. See Adding Keys and Values to the Key/Value Store Database and Example REST API Calls to Access the Key/Value Store Database.

Some Oracle Container Cloud Service REST API methods (usually those performing PUT, POST, and DELETE operations) require authentication details in order to run, in the form of an API token. The API token uniquely identifies the SSO user associated with it. You can see and copy the API token associated with your SSO username on

the **My Profile** page (see Changing and Viewing Profile Settings). Typically, you'll include the API token as the value of the Bearer token in the authorization header of a script calling a method that requires authentication. Note that a new API token value is generated if you change the value in the **Username** field on the **My Profile** page, so you'll have to update any scripts that included the old API token.

Calling the Oracle Container Cloud Service REST API

As a developer, you can use the functionality provided by the Oracle Container Cloud Service REST API to incorporate Oracle Container Cloud Service functionality into your web applications using Representational State Transfer (REST) requests and responses. REST offers a simple, light-weight API for applications that don't require the server to maintain state information or message exchange.

For more information, see REST API for Oracle Container Cloud Service.

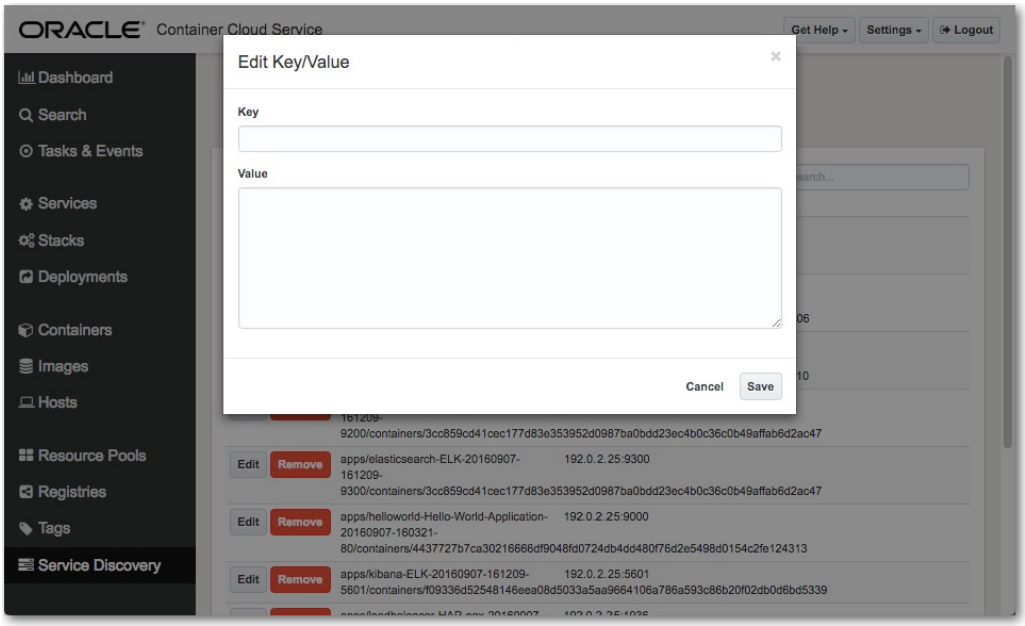
Adding Keys and Values to the Key/Value Store Database

You can store values for keys in the Key/Value Store database using the Oracle Container Cloud Service Container Console for subsequent retrieval by calls to the Oracle Container Cloud Service REST API.

As well as using the Container Console to add keys and values as described here, you can also use the REST API. For more information about accessing the Key/Value Store database using the REST API, see Example REST API Calls to Access the Key/Value Store Database.

To add keys and values to the Key/Value Store database using the Oracle Container Cloud Service Container Console:

1. On the **Service Discovery** page of the Container Console, display the **Key/Value** tab, and click **New Key/Value**.



2. Enter a unique identifier for the key. You can't change the identifier once you've saved the key.
3. Enter a value for the key.

   Note that key values are always shown in the **Edit Key/Value** window in the human-readable form in which you enter them. However, the key values are actually stored in the Key/Value Store database as base64 encoded data, and API calls to retrieve the values will return the base64 encoded data.
4. Click **Save**.

Example REST API Calls to Access the Key/Value Store Database

You can access the Key/Value Store database using the Oracle Container Cloud Service REST API.

To retrieve values from the Key/Value Store database, you'll typically make calls to the Oracle Container Cloud Service REST API (which you can also use to add and remove keys and values). For example:

- to insert a value in the Key/Value Store database using the REST API:

  ```
  curl -si -X "PUT" -H "Authorization: Bearer 394ed660d52ac8d8" "http://
  apitest.occs.example.oraclecloud.com:80/api/kv/dummykey" -d 'dummy
  value'
  ```

- to get a value from the Key/Value Store database using the REST API:

  ```
  curl -si -X "GET" -H "Authorization: Bearer 394ed660d52ac8d8" "http://
  apitest.occs.example.oraclecloud.com:80/api/kv/dummykey"
  ```

  which might generate the response:

  ```
  [{"Key":"dummykey","Value":"ZHVtbXkgdmFsdWU="}]
  ```

- to delete a value from the Key/Value Store database using the REST API:

  ```
  curl -si -X "DELETE" -H "Authorization: Bearer 394ed660d52ac8d8"
  "http://apitest.occs.example.oraclecloud.com:80/api/kv/dummykey"
  ```

As well as using the Oracle Container Cloud Service REST API, you can also add, edit, and remove values in the Key/Value Store database using the Oracle Container Cloud Service Container Console (see Adding Keys and Values to the Key/Value Store Database).

A

Using Oracle Container Cloud Service Example Stacks

Learn how to use the Oracle Container Cloud Service example stacks.

Topics:

- About the Example Stacks
- Exploring an Example Stack

About the Example Stacks

If you want to try out Oracle Container Cloud Service but you don't have any containers of your own yet, you can explore the example stacks available on GitHub.

The example stacks in the Oracle Container Cloud Service GitHub repository include:

- web servers such as Apache HTTP Server and NGINX
- databases such as MySQL and Mongo
- WordPress for blogging

Explore the examples to see how to:

- create one-click deployments
- enable built-in service discovery
- import existing Docker Run and Docker Compose files
- copy and paste YAML, to encourage code sharing and cross-team collaboration
- scale your containerized applications as needed

Use these examples as learning tools, or edit and extend them to incorporate them into your own containerized applications.

Exploring an Example Stack

You can explore one of the example stacks available on GitHub by building the stack, then using Oracle Container Cloud Service to create and deploy it as a Docker container, and then accessing the application running inside the deployed container.

To explore a example stack:

1. Go to the GitHub repository at https://github.com/oracle/docker-images/tree/master/ContainerCloud/.
2. Decide which of the example stacks to explore and follow the instructions in the README file to build it.
3. In Oracle Container Cloud Service Container Console, create a new stack based on the stack you've just built (see Creating a Stack).

4. In Oracle Container Cloud Service Container Console, deploy the new stack (see Deploying a Stack).
5. Assemble the URL to access the running stack and enter the URL in a browser (see Accessing an Application Running Inside a Deployed Container).