Document title	CloudStack Container Service Installation and Administration Guide
Date	13 July 2016
Version	1.0.0

1 Introduction

The CloudStack Container Service (CCS) orchestrates provisioning of Kubernetes container clusters in a virtual machines and networks managed by Apache CloudStack. By default, container clusters are built on CoreOS (Alpha Channel) and Kubernetes 1.2.4. Once provisioned, users are able to configure the cluster and deploy containers using standard Kubernetes tools such as kubectl. Finally, CCS securely embeds the Kubernetes Dashboard into the CloudStack Administrative Console – providing a web UI to deploy containerized applications.

CCS is implemented as a CloudStack management server plugin. Therefore, a functioning CloudStack management server is required for installation. CCS is distributed as a RPM package installed from a private Yum repository. Access to this repository requires an installation token issued by ShapeBlue. To obtain an installation token, please contact CCS support. The RPM includes an script to download and install the default CoreOS template into each zone. Following installation of the default CoreOS template, CCS will be ready for use.

This guide details installation and administration of container clusters using CCS. For information on installing CCS, please see the CloudStack Container Service Installation Guide.

1.1 Prerequistes and Supported Environments

The CCS plugin requires that the management server has access to the public network in the zonein order to communicate with the container clusters it is managing.

The following table lists the matrix of Management Server OS, CloudStack, and Hypervisor versions that have been tested with CCS:

Management Server OS	CloudStack Version	Hypervisor Version
RHEL/CentOS 6.8	4.5.2.1	XenServer 6.2 SP1
RHEL/CentOS 6.8	4.5.2.1	XenServer 6.5 SP1
RHEL/CentOS 6.8	4.5.2.1	VMware 5.5
RHEL/CentOS 6.8	4.5.2.1	CentOS 6.8 KVM
RHEL/CentOS 7.2	4.5.2.1	CentOS 7.2 KVM
RHEL/CentOS 6.8	4.6.2.1	VMware 5.5
RHEL/CentOS 7.2	4.6.2.1	XenServer 6.5 SP1

If your environment configuration is not listed, please contact CCS support for assistance determining whether or not CCS will function in your environment.

1.2 Security

CCS generates a root CA certificate for a management server cluster. When container clusters are provisioned, a new SSL certificate is generated and signed by the generated root CA. This certificate is used to secure all Kubernetes endpoints. Additionally, CCS generates a unique administrative password for each cluster provisioned.

1.3 Getting Support

If you have any questions about using CCS or encounter any issues, please contact CCS support via email or create a support ticket.

1 Installation

To install CCS, execute the following steps on each management server:

1. Stop the CloudStack management and usage services, as applicable:
    a. `service cloudstack-management stop`
    b. `service cloudstack-usage stop`
2. Download the Yum repository installation script from https://downloads.shapeblue.com/ccs/1.0.0/install_yum_repo.sh
3. Make the script executable and run it:
    a. `chmod +x install_yum_repo.sh`
    b. `./install_yum_repo.sh` <installation token>
4. Install the CCS package `yum install shapeblue-ccs`
5. Start the CloudStack management and usage services, as applicable:
    a. `service cloudstack-management start`
    b. `service cloudstack-usage start`

On one management server, execute the template installation script, `ccs-template-install -m` <hypervisor (XenServer|VMware|KVM)>. This script installs the default CoreOS template into each zone. CCS uses this template to deploy Kubernetes masters and nodes. Once the template installation is complete, CCS is ready for use.

2 Uninstallation

To uninstall CCS, execute the following steps:

1. Stop the CloudStack management and usage services, as applicable on each CloudStack management server:
    a. `service cloudstack-management stop`
    b. `service cloudstack-usage stop`
2. Remove the CCS from each CloudStack management server: `yum erase cloudstack-ccs`
3. Start the CloudStack management and usage services, as applicable, on each CloudStack management server:
    a. `service cloudstack-management start`
    b. `service cloudstack-usage start`

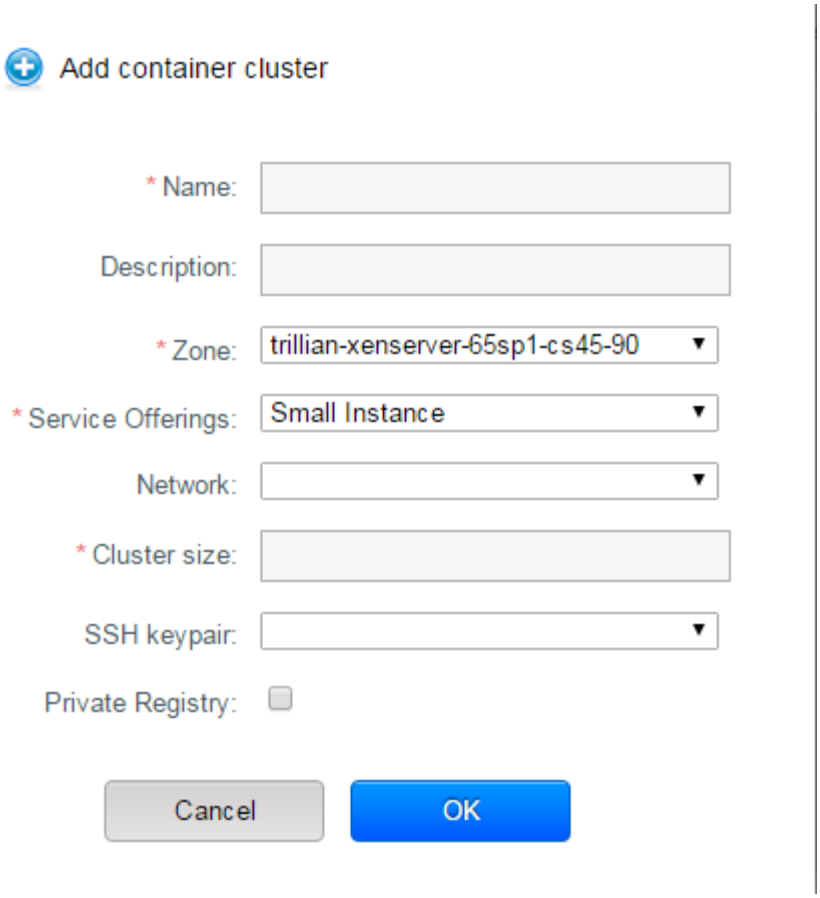
On one management server, execute database cleanup script, `ccs-cleanup-database`. This script will remove the CCS database tables from the management server database.

2 User Interface Usage

With the CCS plugin installed, a new "Container Service" tab will be added in the UI. In this tab, container clusters can be added and removed.

2.1 Adding a Container Cluster

To add a new container cluster, selecting the "Container Service" tab and click the "Add container cluster" button in the upper right-hand corner of the navigation ribbon. A dialog such as the following will displayed where a new container cluster is defined:



- Name – The name for the cluster.
- Description – The description for the cluster.
- Zone – The zone where the cluster should be deployed.
- Service Offerings – The Compute Offering to use when creating the cluster VMs.
- Network – The network to use for the cluster, or blank to make CloudStack create a new network for the cluster.
- Cluster Size – The number of compute VMs to deploy in the cluster (i.e. excluding the management VM).
- SSH keypair – The SSH key pair to use to log in to the cluster VMs.
- Note: While this is not a required field, it is necessary to log in via SSH to the VMs. Kubernetes can still be managed without this, but it may be useful in some situations, such as troubleshooting.
- Private Registry – whether or not to use a private container registry. By default, the Docker public registry will be used.

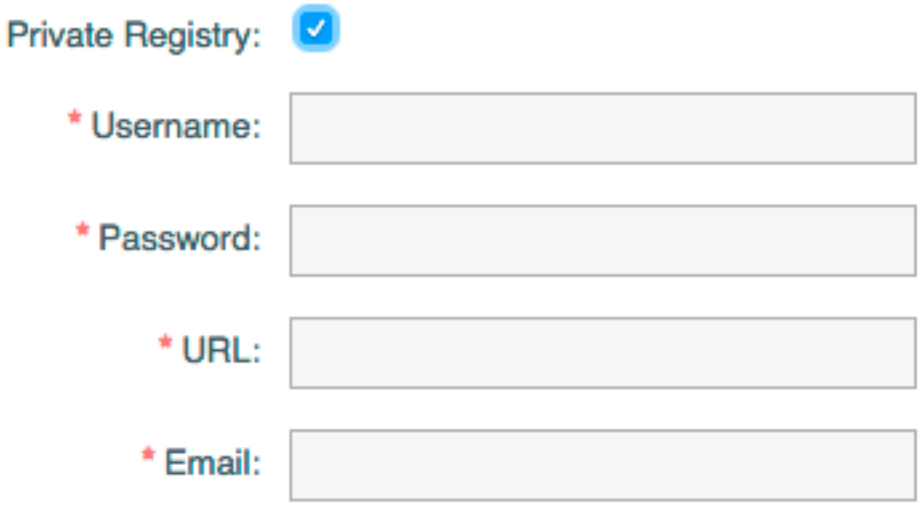
The following limitations apply to container cluster configurations:

- The size of a container cluster cannot exceed the account's virtual machine resource limit or one-hundred (100) nodes – whichever is smaller. CCS places an upper limit of a one hundred nodes in a container cluster to avoid overwhelming the cluster's master.
- A network may only be assigned to one container cluster at any given time.

Clicking "Ok" will trigger creation of the container cluster including provisioning of the underlying virtual machines, configuration of Kubernetes, network creation, and cluster startup.

2.1.1 Using a Private Registry

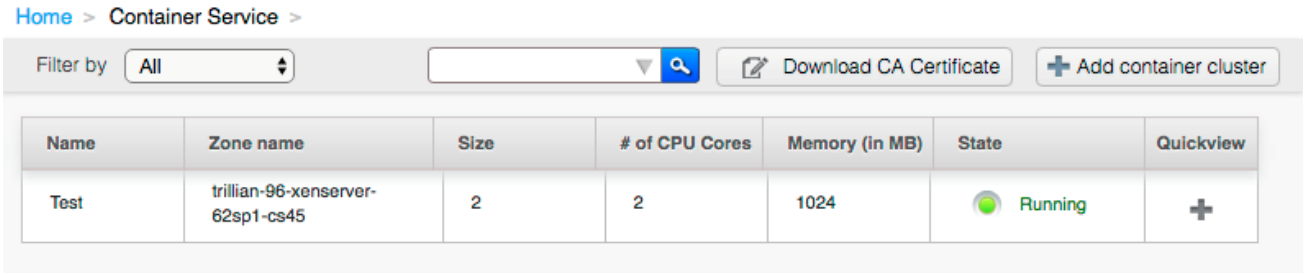
To use a external, private container registry, select the "Private Registry" option in the container cluster create dialog which will expand the fields in the following manner:



- **Username**: A username with access to the repository
- **Password**: The password for the username to access the repository
- **URL**: The location of the repository. It must be reachable from the container cluster's network.
- **Email**: The email address of the user to access the repository

2.2 Browsing Available Container Clusters

Selecting the "Container Service" tab will list all available container clusters for the account as depicted in the following screenshot:



These columns provide the following information:

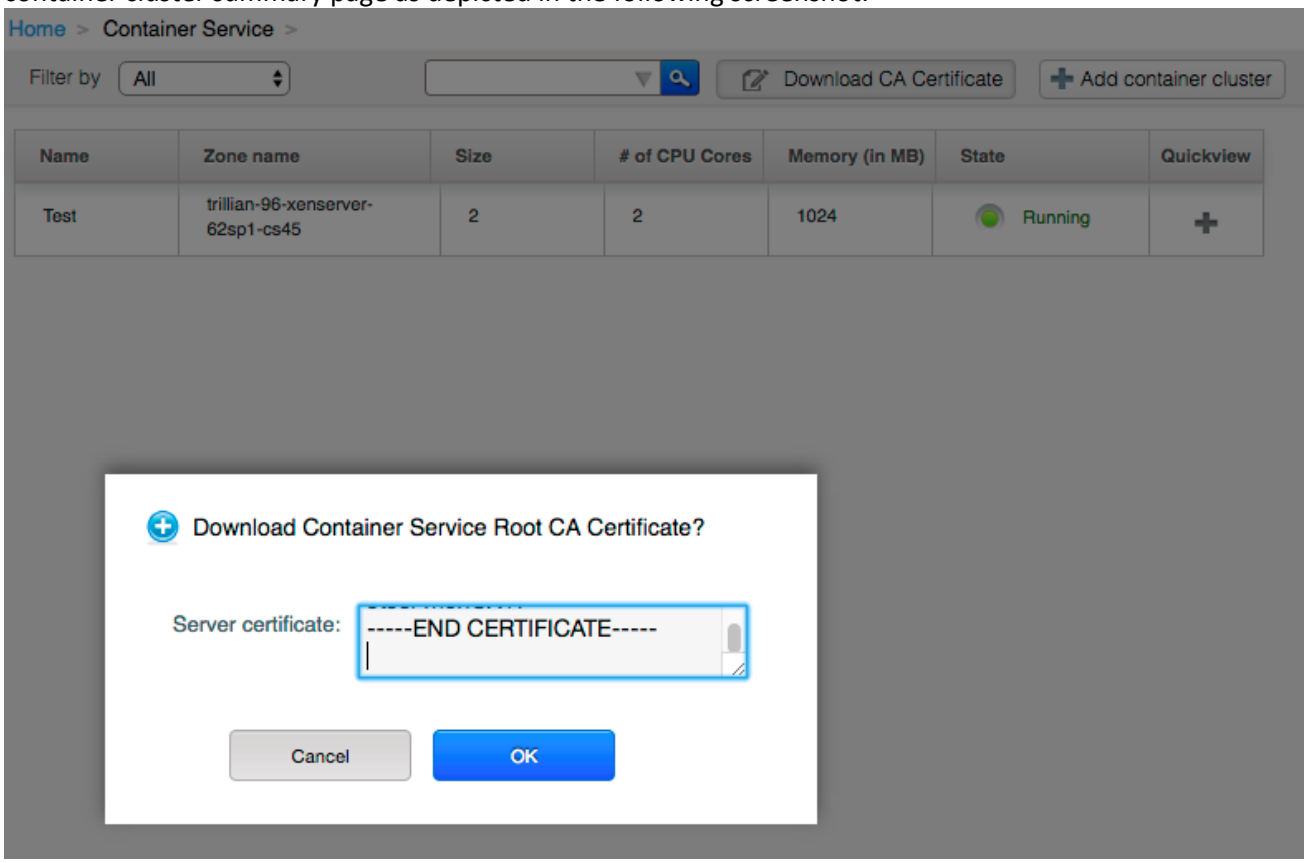
- **Name**: The name of the cluster
- **Zone**: The zone in which the cluster is deployed
- **Size**: The number of VMs on which the cluster is deployed not including the VM for the Kubernetes master
- **# of CPU Cores**: Total CPU codes available for container deployment
- **Memory (in MB)**: Total memory available for container deployment
- **State**: The current state of the cluster. The following are the possible states:
    - **Created**: Initial State of a container cluster when has been defined but no resources consumed
    - **Starting**: Resources needed for container cluster are being provisioned and the container cluster is being configured and started.
    - **Running**: Resources have been provisioned and container cluster is in operational ready state to launch containers
    - **Stopping**: Resources for the container cluster are being destroyed
    - **Stopped**: All resources for the container cluster are destroyed
    - **Alert**: The container cluster is in unexpected operational state (operationally in-active control place, stopped cluster VM's etc).
    - **Recovering**: The container cluster is recovering from alert state to a healthy state
    - **Destroyed**: All resources for the container cluster are destroyed. The cluster is no longer useable.
    - **Destroying**: Resources for the container cluster are being cleaned up or are awaiting garbage collection
    - **Error**: Creation of the container cluster failed

2.3 Download and Install Root CA Certificate

As discussed in "Section 1.2: Security", CCS secures the Kubernetes Dashboard with an SSL certificate signed by a certificate authority (CA) created for the management server. Since the certificate authority is unknown to user's web browsers, they must perform a one-time operation to add this CA to their browser. The

following steps will download the CA certificate from the the management server to be installed in a browser:

1. Select the “Container Service” tab and click the “Download Root CA Certificate” button on the container cluster summary page as depicted in the following screenshot:

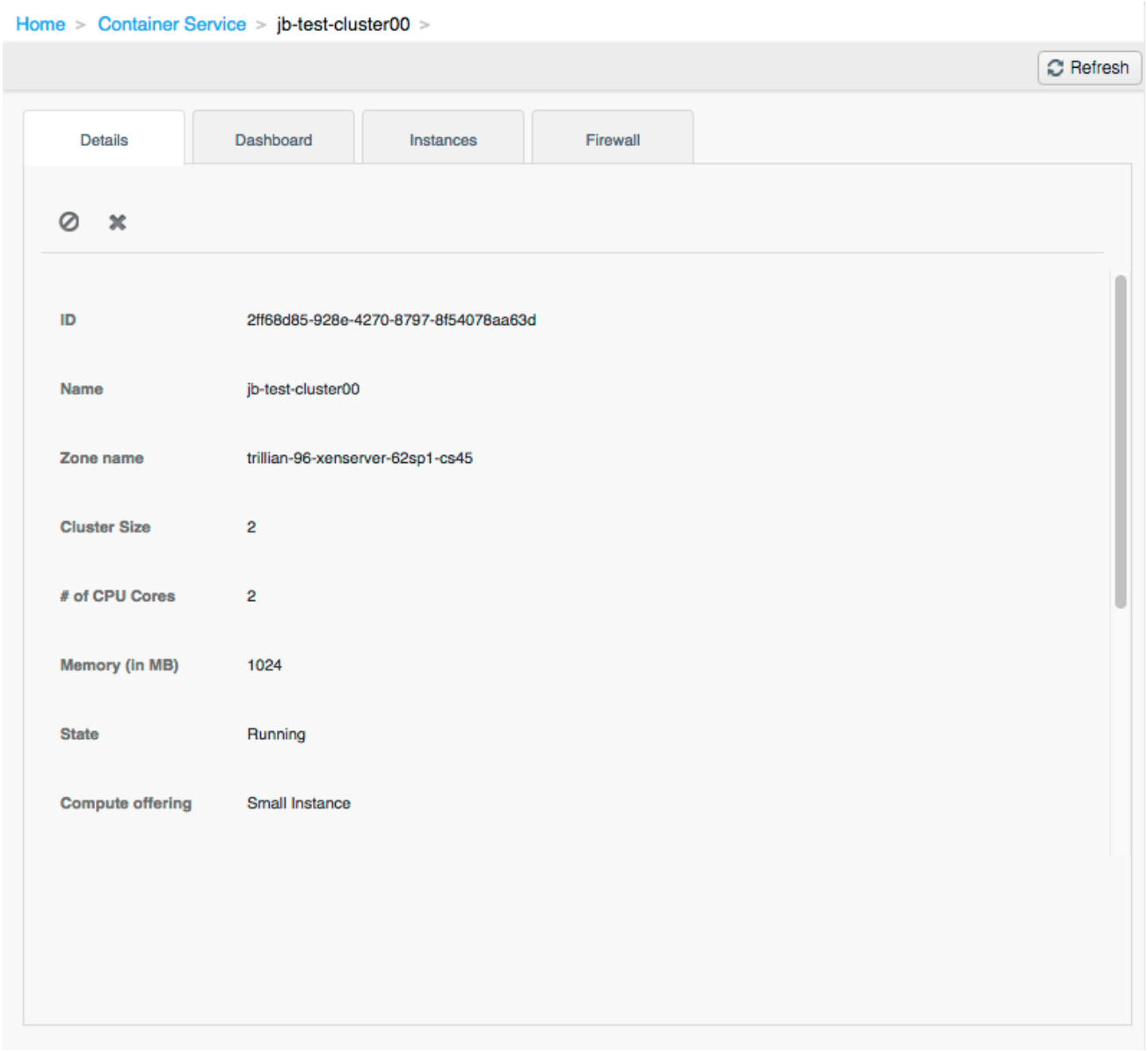


2. Clicking “Ok” will prompt the user to save the certificate file to local storage as “cloudstack-containerservice.pem”

Users import the “cloudstack-containerservice.pem” in their browser list of CAs per the browser’s documentation. Once the CA certificate has been successfully imported, the user will be able access the embedded Kubernetes Dashboard securely.

2.4 Inspecting Container Cluster Details

Click the name of a container cluster on the container cluster summary page will display the details about a container cluster as depicted in the following screenshot:



The tabs provide access to the following functions and/or information:

- **Details**: Configuration information about the container cluster. In addition to the fields defined above in "Section 2.2: Browsing Avaiable Container Clusters", the page provides the following addition fields about the container cluster:
    - **ID**: The unique identifier of the container cluster. This value is used when interacting with the container cluster API.
    - **Compute Offering**: The compute offering used to provision the VMs on which the cluster is running.
    - **Ssh Key Pair**: The SSH key pair used to authenticate to the container cluster

  - **API Endpoint**: The API endpoint for the container cluster. This endpoint is used to connect `kubectl` to the container cluster.
  - **Dashboard Endpoint**: The URL for the Kubernetes Dashboard for the container cluster
  - **Username**: The username with which to authenticate to the container cluster. This username is used to authenticate when connecting to the cluster with `kubectl`.
  - **Password**: The password with which to authenticate to the container cluster. This password is used to authenticate when connection to the cluster with `kubectl`.

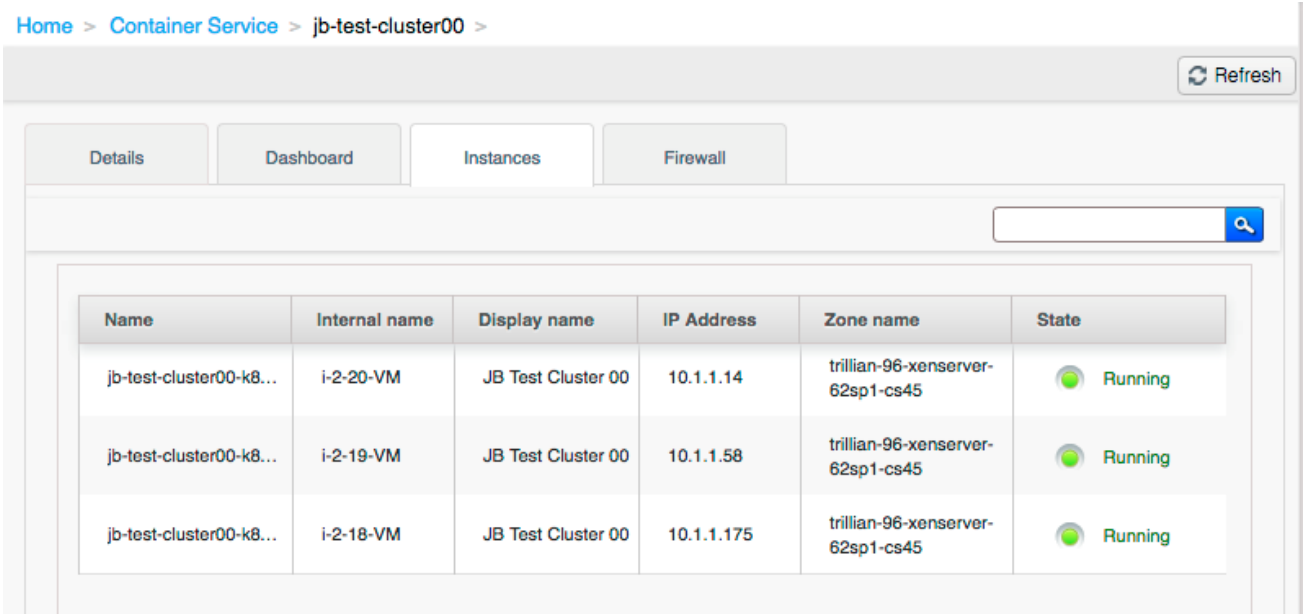
- **Dashboard**: Access to the embedded Kubernetes Dashboard. Please see the Kubernetes Dashboard User Guide for more information about using deploy containerized applications. **N.B.** The dashboard is not accessible until the root CA certificate has been installed in each user's browser. This installation process is described in "Section 2.3: Download and Install Root CA Certificate".
- **Instances**: Information about the virtual machines on which the container cluster runs. Please see "Section 2.4.1: Container Cluster Instance Details" below for more details about the information provided.
- **Firewall**: Shortcut to the Firewall configuration page for the network attached to the container cluster

The buttons below the tabs provide access to the following functions (left to right):

- **Start/Stop Container Cluster**: Dependent on the current state of the container cluster, starts or stops it.
- **Destroy Container Cluster**: Marks the container cluster for removal. Removal of the underlying resources will occur on the next garbage collection run.

2.4.1 Container Cluster Instance Details

When viewing details about a container cluster, the "Instances" tab provides information about the VMs on which the container cluster is running. The list includes the VMs on which the Kubernetes nodes run, as well as, the master. Therefore, the number of VM instances will be cluster size plus one (1). For example, in the following screenshot, the "jb-test-cluster-00" container cluster is defined with a size of two (2) requiring three (3) virtual machines to one which to deploy:



It is important to note that container cluster VMs are standard user VMs. Therefore, they can be controlled (e.g. started, stopped, deleted, etc) outside of CCS' control. Extreme care should be taken when manually controlling these VMs as it is very likely to negatively impact the stability of the container cluster.

3 Accessing a Container Cluster using kubectl

`kubectl` is a powerful command line tool for accessing and controlling Kubernetes clusters. While not required for using CCS, it is a popular means to control and monitor Kubernetes clusters.

Using CloudMonkey, execute a `listContainerCluster` command to acquire the `username`, `password`, and `endpoint` of the cluster. With this information, `kubectl` without certificate validation can be used as follows:

```
kubectl <COMMAND> -s <endpoint> --username=<username> --
password=<password> --insecure-skip-tls-verify=true
```

To use `kubectl` with certificate validation, download the root CA certificate via the UI (as described in "Section 2.3: Download and Install Root CA Certificate") or executing a `listContainerClusterCACert` command using CloudMonkey. The following command demonstrates using `kubectl` with certification validation:

```
kubectl <COMMAND> -s <endpoint> --username=<username> --
password=<password> --certificate-authority=<path to downloaded
certificate>
```

To reduce the complexity of the these commands, the username, password, endpoint, and certificate authority information can be saved to ~/.kube/config using the following command:

```
kubectl config -s <endpoint> --username=<username> --
password=<password> --certificate-authority=<path to downloaded
certificate> --cluster=<name of cluster>
```

For more information about `kubectl` usage, please see the kubectl User Guide.

4 Billing

By default, CCS is built with the assumption that providers will bill container service users based on the resources consumed by the VMs underlying the container clusters. Therefore, all CloudStack resources associated with a container cluster continue to generate the same usage events as they would if they being provisioned for standalone usage. CCS generates two additional usage events – `CONTAINER.CLUSTER.CREATE` and `CONTAINER.CLUSTER.DELETE`.

It is also important to note that, by default, a network with a public IP assigned for NAT is created for a container cluster. Dependent on the provider's billing scheme, this behaviour could impact user costs.

5 API Reference

With the CCS plugin installed, a number of CloudStack API commands will be available.

5.1 createContainerCluster

Creates a Container Cluster.

The following are the request parameters for the createContainerCluster API command:

Name	Description	Required
name	name for the container cluster	Yes
description	description of container cluster	No
zoneid	zone in which container cluster to be launched	Yes
serviceofferingid	service offering from which container cluster VMs are expected to be created	Yes
accountname	account for which container cluster to be created	No
domainid	domain in which account belongs	No
networkid	network that will be used for launching container cluster VMs, if specified. If left unspecified, an isolated network will be provisioned for the container cluster automatically	No
sshKeyPairName	SSH key pair with user can log into the cluster VMs	No
clustersize	desired size of the container cluster	Yes
dockerRegistryUserName	Docker private registry user name	No
dockerRegistryPassword	Docker private registry password for the specified user	No
dockerRegistryUrl	URL for the docker private registry	No
dockerRegistryEmail	email of the user	No

The following are the response tags for the createContainerCluster API Command:

Name	Description
id	UUID of the container cluster
name	Name of the container cluster
description	Description of the container cluster
zoneid	UUID of the zone in which container cluster is created
zonename	Name of the zone in which container cluster is create
serviceofferingid	UUID of the service offering with which VM's in the container cluster were created
serviceofferingname	Name of the service offering with which VM's in the container cluster were created
templateid	UUID of the template used to create VM's in the container cluster
networkids	Network in which container cluster VM's were created
associatednetworkname	Name of the network in which container cluster VM's were created
keypair	SSH keypair that is used to provision VM's
size	Size of the cluster as specified by the user when creating the cluster
state	Current state of the container cluster
cores	Total number of CPU cores used by the container cluster
memory	Total memory used by the container cluster
endpoint	URL for the cluster orchestrator.
consoleendpoint	URL for the UI/dashboard of the container cluster
virtualmachineids	List of UUID's of the virtual machines that are part of the container cluster
username	CCS generated username for the container cluster
Password	CCS generated password for the container cluster

5.2 deleteContainerCluster

Deletes a provisioned Container Cluster. All the VM's associated with the container cluster will be deleted.

The following are the request parameters for the deleteContainerCluster API command:

Name	Description	Required
id	UUID of the container cluster which is requested to be deleted	Yes

The following are the response tags for the deleteContainerCluster API Command:

Name	Description
success	True if operation executed successfully

5.3 listContainerCluster

List the Container Clusters. Domain admins can list all the clusters in the domain. Root admins can list all clusters. Regular users can only list clusters owned.

The following are the request parameters for the listContainerCluster API command:

Name	Description	Required
id	UUID of the container cluster to be listed	No
state	State by which the container clusters to be searched and listed	No
name	Name by which the container clusters to be searched and listed	No

The following are the response tags for the listContainerCluster API Command:

Name	Description
id	UUID of the container cluster
name	Name of the container cluster
description	Description of the container cluster
zoneid	UUID of the zone in which container cluster is created
zonename	Name of the zone in which container cluster is create
serviceofferingid	UUID of the service offering with which VM's in the container cluster were created
serviceofferingname	Name of the service offering with which VM's in the container cluster were created
templateid	UUID of the template used to create VM's in the container cluster
networkids	Network in which container cluster VM's were created
associatednetworkname	Name of the network in which container cluster VM's were created
keypair	SSH keypair that is used to provision VM's
size	Size of the cluster as specified by the user when creating the cluster
state	Current state of the container cluster
cores	Total number of CPU cores used by the container cluster
memory	Total memory used by the container cluster

endpoint	URL for the cluster orchestrator.
consoleendpoint	URL for the UI/dashboard of the container cluster
virtualmachineids	List of UUID's of the virtual machines that are part of the container cluster
username	CCS generated username for the container cluster
Password	CCS generated password for the container cluster

5.4 stopContainerCluster

Stops container cluster. All the VM's in the cluster will be stopped.

The following are the request parameters for the stopContainerCluster API command:

Name	Description	Required
Id	UUID of the container cluster which should be stopped	Yes

The following are the response tags for the stopContainerCluster API Command:

Name	Description
Success	True if operation executed successfully

5.5 startContainerCluster

Stops a Container Cluster. All the VM's in the cluster will be started.

The following are the request parameters for the startContainerCluster API command:

Name	Description	Required
id	UUID of the container cluster which should be started	

The following are the response tags for the startContainerCluster API Command:

Name	Description
success	True if operation executed successfully

5.6 listContainerClusterCACert

A root certificate will be generated by CCS for provisioning the clusters nodes. Users have to use this API to download the certificate and trust it in order to use the cluster dashboard from the CloudStack UI.

The listContainerClusterCACert API command has no parameters. The following are the response tags for the listContainerClusterCACert API Command:

Name	Description

certificate	Root CA certificate in PEM format.

6 Plugin Configuration

6.1 Global Settings

With the CCS plugin installed, a number of settings will be added in Global Settings.

6.1.1 cloud.container.cluster.template.name

Name of the installed CloudStack template that will be used for provisioning the container cluster VM's. By default, it is set to "ShapeBlue-CCS-Template", and "ccd-template-install" script will register template with same name as well. However, admin can register a CCS template certified by ShapeBlue for CCS, manually and give template name of his choice. In which case this setting need to be updated accordingly with registered template name.

6.1.2 cloud.container.cluster.network.offering

Name of the network offering that CloudStack container service will use to provision the network for container cluster. By default, installing CCS package, a network offering by name 'DefaultNetworkOfferingforContainerService' is created. And this global setting is set to 'DefaultNetworkOfferingforContainerService'. However, an administrator can create a network offering (with source nat, firewall,port-forwarding and user data services, with egress default policy set to allow) and specify in this global settings.

6.1.3 cloud.container.cluster.master.cloudconfig, cloud.container.cluster.node.cloudconfig

These two global settings point to the user data templates that will be rendered and passed to container cluster VM's on start. These two global settings will be set by default on installing CCS packages and are expected not to be modified. Defaulted to "/etc/cloudstack/management/k8s-master.yml" and "/etc/cloudstack/management/k8s-node.yml" respectively for "cloud.container.cluster.master.cloudconfig" and "cloud.container.cluster.node.cloudconfig".

6.2 Tab Name Customization

To modify the tab name, edit the ccs.js file located at /usr/share/cloudstack-management/webapps/client/plugins/ccs with desired name. Default value is "Container Service"

To modify the logo, replace the /usr/share/cloudstack-management/webapps/client/plugins/ccs/icon.png file with a custom image.