Red Hat Container Development Kit 3.16

Getting Started Guide

Quick-start guide to using and developing with Red Hat Container Development Kit

Last Updated: 2021-05-11

Red Hat Container Development Kit 3.16 Getting Started Guide

Quick-start guide to using and developing with Red Hat Container Development Kit

Yana Hontyk
yhontyk@redhat.com

Kevin Owen
kowen@redhat.com

Chris Negus

Robert Krátký

Legal Notice

Abstract

This guide shows how to get up to speed using Red Hat Container Development Kit. Included instructions and examples guide through first steps developing containerized applications using Docker, Kubernetes, and OpenShift Container Platform, both from your host workstation (Microsoft Windows, macOS, or Red Hat Enterprise Linux) and from within the Container Development Environment provided by Red Hat Container Development Kit.

Table of Contents

CHAPTER 1. GETTING STARTED WITH CONTAINER DEVELOPMENT KIT

This section contains information about setting up, installing, and uninstalling Container Development Kit.

1.1. INTRODUCING RED HAT CONTAINER DEVELOPMENT KIT

Red Hat Container Development Kit provides a platform for developing containerized applications. It is a set of tools that enables developers to quickly and easily set up an environment for developing and testing containerized applications on the Red Hat Enterprise Linux platform.

- Container Development Kit provides a personal Container Development Environment you can install on your own laptop, desktop, or server system. The Container Development Environment is provided in the form of a Red Hat Enterprise Linux virtual machine.
- Container Development Kit is available for the Microsoft Windows, macOS, and Linux operating systems, thus allowing developers to use their preferred platform while producing applications ready to be deployed in the Red Hat Enterprise Linux ecosystem.

Container Development Kit is a part of the Red Hat Developers program, which provides tools, resources, and support for developers who wish to utilize Red Hat solutions and products to create applications, both locally and in the cloud. For additional information and to register to become a part of the program, visit developers.redhat.com.

1.1.1. Understanding Container Development Kit documentation

- The Red Hat Container Development Kit 3.16 Release Notes and Known Issues contains information about the current release of the product as well as a list of known problems that users may encounter when using it.
- The Container Development Kit Getting Started Guide contains instructions on how to install and start using the Container Development Environment to develop Red Hat Enterprise Linux-based containers using tools and services such as **OpenShift Container Platform**, **Docker**, **Eclipse**, and various command line tools.
- Report issues with Red Hat Container Development Kit or request new features using the **CDK** project at https://issues.jboss.org/projects/CDK.
- Report issues with the Red Hat Container Development Kit 3.16 Release Notes and Known Issues and Container Development Kit Getting Started Guide using the **RHDEVDOCS** project at https://issues.jboss.org/projects/RHDEVDOCS.

1.2. PREPARING TO INSTALL CDK

1.2.1. Overview

The following section describes how to install CDK and the required dependencies.

These are the basic steps for setting up CDK on your personal system:

1. Set up your virtualization environment

2. Download CDK software for your operating system from the Red Hat Container Development Kit Download page
3. Install CDK
4. Set up and start CDK
5. Configure CDK so you can use it efficiently

The setup procedure should be run as a regular user with permission to launch virtual machines. In the procedure, you will see how to assign that permission, along with ways to configure your hypervisor and command shell to start and effectively interact with CDK.

1.2.2. Prerequisites

CDK requires a hypervisor to start the virtual machine on which the OpenShift cluster is provisioned. Verify that the hypervisor of your choice is installed and enabled on your system before you set up CDK. Once the hypervisor is up and running, additional setup is required for CDK to work with that hypervisor.

Depending on your host operating system, you have the choice of the following recommended native hypervisors:

macOS

- hyperkit

Linux

- KVM

Windows

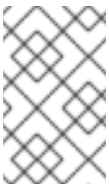
- Hyper-V

NOTE

To use CDK with Hyper-V ensure that, after you install Hyper-V, you also add a Virtual Switch using the Hyper-V Manager and set the configuration option **hyperv-virtual-switch** to this virtual switch. For specific configuration steps see the Setting Up the Hyper-V Hypervisor section.

All Platforms

- VirtualBox



NOTE

VirtualBox 5.1.12 or later is recommended on Windows to avoid the issue Error: machine does not exist. If you encounter issues related to the hypervisor, see the Troubleshooting Driver Plug-ins section.

Refer to the documentation for each hypervisor to determine the hardware and operating system versions needed to run that hypervisor.

1.3. SETTING UP THE VIRTUALIZATION ENVIRONMENT

1.3.1. Overview

Follow the appropriate procedure to set up the hypervisor for your particular operating system. CDK uses libmachine and its driver plug-in architecture to provide a consistent way to manage the CDK VM.

Some hypervisors require manual installation of the driver plug-in. CDK embeds the VirtualBox driver plug-in, so no additional steps are required to configure it. However, VirtualBox will need to be identified to CDK via the **--vm-driver virtualbox** flag or persistant configuration settings. See Setting Up CDK to Use VirtualBox for more information.

See the appropriate section for your hypervisor and operating system:

- For Red Hat Enterprise Linux, set up the KVM driver
- For macOS, set up the hyperkit driver
- For Windows, set up the Hyper-V hypervisor
- For VirtualBox, set up CDK to use VirtualBox

1.3.2. Red Hat Enterprise Linux

1.3.2.1. Setting Up the KVM Driver

CDK is currently tested against **docker-machine-driver-kvm** version 0.10.0.

1. As root, install the KVM binary and make it executable as follows:

```
# curl -L https://github.com/dhiltgen/docker-machine-kvm/releases/download/v0.10.0/docker-machine-driver-kvm-centos7 -o /usr/local/bin/docker-machine-driver-kvm
# chmod +x /usr/local/bin/docker-machine-driver-kvm
```

   For more information, see the GitHub documentation of the Docker Machine KVM driver.

2. As root, install **libvirt** and **qemu-kvm** on your system:

```
# yum install libvirt qemu-kvm
```

3. As root, add yourself to the **libvirt** group:

```
# usermod -a -G libvirt username
```

4. Update your current session to apply the group change:

```
$ newgrp libvirt
```

5. Start the **libvirtd** service as root:

```
# systemctl start libvirtd
# systemctl enable libvirtd
```

1.3.2.2. Start libvirtd service

1. Check the status of **libvirtd**:

```
$ systemctl is-active libvirtd
```

1. If **libvirtd** is not active, start the **libvirtd** service as root:

```
# systemctl start libvirtd
```

1.3.2.2.1. Next Steps

Proceed to Installing CDK once your hypervisor has been installed and configured.

1.3.3. macOS

1.3.3.1. Setting Up the hyperkit Driver

CDK is currently tested against **docker-machine-driver-hyperkit** version 1.0.0.

Using hyperkit requires having both **hyperkit** and **docker-machine-driver-hyperkit** installed.

1.3.3.1.1. Installing hyperkit

- If you have Docker Desktop for macOS installed, **hyperkit** is already installed.
- If you use Homebrew you can install the latest version of **hyperkit**:

```
$ brew install hyperkit
```

1.3.3.1.2. Installing docker-machine-driver-hyperkit

- If you use Homebrew you can install the latest version of **docker-machine-driver-hyperkit**:

```
$ brew install docker-machine-driver-hyperkit
```

- Alternatively, you can download and install the **docker-machine-driver-hyperkit** binary and place it in a directory which is on your **PATH** environment variable. The directory */usr/local/bin* is a good choice, since it is the default installation directory for Docker Machine binaries.

The following steps explain the installation of the **docker-machine-driver-hyperkit** binary to the */usr/local/bin/* directory:

1. Download the **docker-machine-driver-hyperkit** binary using:

```
$ sudo curl -L  https://github.com/machine-drivers/docker-machine-driver-hyperkit/releases/download/v1.0.0/docker-machine-driver-hyperkit -o /usr/local/bin/docker-machine-driver-hyperkit
```

2. Enable root access for the **docker-machine-driver-hyperkit** binary and add it to the default **wheel** group:

```
$ sudo chown root:wheel /usr/local/bin/docker-machine-driver-hyperkit
```

3. Set owner User ID (SUID) for the binary as follows:

```
$ sudo chmod u+s,+x /usr/local/bin/docker-machine-driver-hyperkit
```

NOTE

The downloaded **docker-machine-driver-hyperkit** binary is compiled against a specific version of macOS. It is possible that the driver will fail to work after a macOS version upgrade. In this case you can try to compile the driver from source:

```
$ go get -u -d github.com/machine-drivers/docker-machine-driver-hyperkit
$ cd $GOPATH/src/github.com/machine-drivers/docker-machine-driver-hyperkit

# Install docker-machine-driver-hyperkit binary into /usr/local/bin
$ make build
```

For more information, see the hyperkit driver documentation on GitHub.

1.3.3.1.3. Next Steps

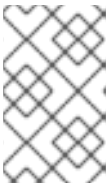
Proceed to Installing CDK once your hypervisor has been installed and configured.

1.3.4. Windows

1.3.4.1. Setting Up the Hyper-V Hypervisor

To use CDK with Hyper-V:

1. Install Hyper-V.
2. Add the user to the local Hyper-V Administrators group.



NOTE

This is required to allow the user to create and delete virtual machines with the Hyper-V Management API. For more information, see Hyper-V commands must be run as an Administrator.

3. Add an External Virtual Switch.
4. Verify that you pair the virtual switch with a network card (wired or wireless) that is connected to the network.
5. Use the configuration option **hyperv-virtual-switch** or startup flag **--hyperv-virtual-switch** to set the name of the external virtual switch you want to use for CDK.
   For example, on PowerShell use

```
PS> minishift config set hyperv-virtual-switch "External (Wireless)"
```

   or

```
PS> minishift start --hyperv-virtual-switch "External (Wireless)"
```

NOTE

- The name of the virtual switch is case sensitive.
- The use of the environment variable **HYPERV_VIRTUAL_SWITCH** has been deprecated. Instead **MINISHIFT_HYPERV_VIRTUAL_SWITCH** can be used as a configuration option, although this is not recommended as environment variables on Windows do not support non-ASCII characters.

1.3.4.1.1. Next Steps

Proceed to Installing CDK once your hypervisor has been installed and configured.

1.3.5. Setting Up CDK to Use VirtualBox

VirtualBox must be manually installed in order to use the embedded VirtualBox drivers. VirtualBox version 5.1.12 or later is required. Ensure that you download and install VirtualBox before using the embedded drivers.

VirtualBox must be identified to CDK through either the **--vm-driver virtualbox** flag or persistant configuration options.

1.3.5.1. Use VirtualBox Temporarily

The **--vm-driver virtualbox** flag will need to be given on the command line each time the **minishift start** command is run. For example:

```
$ minishift start --vm-driver virtualbox
```

1.3.5.2. Use VirtualBox Permanently

Setting the **vm-driver** option as a persistent configuration option allows you to run **minishift start** without explicitly passing the **--vm-driver virtualbox** flag each time. You may set the **vm-driver** persistent configuration option as follows:

```
$ minishift config set vm-driver virtualbox
```

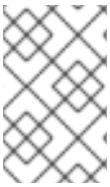
NOTE

The **vm-driver** persistent configuration option must be supplied before **minishift start** has been run. If you have already run **minishift start**, ensure that you run **minishift delete**, set the configuration with **minishift config set vm-driver virtualbox**, then run **minishift start** in order to make use of the VirtualBox driver.

1.3.5.3. Next Steps

Proceed to Installing CDK once your hypervisor has been installed and configured.

1.4. INSTALLING CDK



NOTE

Before you can download CDK software, you need to either register with the Red Hat Developer Program site or login to the Red Hat customer portal with Red Hat subscription credentials.

1. Download CDK for your operating system from the Red Hat Container Development Kit Download page.

2. Copy the downloaded **minishift** file to a directory in your **PATH** and make it executable. The downloaded executable is named **cdk-3.16.0-1-minishift-darwin-amd64** (for macOS), **cdk-3.16.0-1-minishift-linux-amd64** (for Red Hat Enterprise Linux) or **cdk-3.16.0-1-minishift-windows-amd64.exe** (for Windows). Assuming the executable is in the ***~/Downloads*** directory, follow the procedure for your operating system:
For Red Hat Enterprise Linux:

```
$ mkdir -p ~/bin
$ cp ~/Downloads/cdk-3.16.0-1-minishift* ~/bin/minishift
$ chmod +x ~/bin/minishift
$ export PATH=$PATH:$HOME/bin
$ echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
```

For macOS:

```
$ mkdir -p ~/bin
$ cp ~/Downloads/cdk-3.16.0-2-minishift* ~/bin/minishift
$ chmod +x ~/bin/minishift
$ export PATH=$PATH:$HOME/bin
$ echo export PATH=$PATH:$HOME/bin >> ~/.bash_profile
```

For Windows:

Create the desired directory and copy the downloaded CDK binary to the directory, renaming the binary to **minishift.exe**. Add the directory path to the Windows **PATH** variable.

If it's difficult to get **minishift.exe** in your **PATH**, you can simply run it from the current directory as **./minishift.exe** (or **.\minishift.exe** in Command Prompt).

NOTE

- On the Windows operating system, due to issue #236, you need to execute the CDK binary from your local ***C:*** drive. You cannot run CDK from a network drive.

1.5. CDK QUICKSTART

1.5.1. Overview

This section contains a brief demo of CDK and of the provisioned OpenShift cluster. For details on the usage of CDK, see the Basic Usage section.

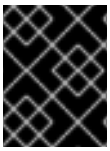
The interaction with OpenShift is with the command line tool **oc** which is copied to your host. For more information on how CDK can assist you in interacting with and configuring your local OpenShift instance, see the OpenShift Client Binary section.

For more information about the OpenShift cluster architecture, see Architecture Overview in the OpenShift documentation.

The following steps describe how to get started with CDK on a Linux operating system with the KVM hypervisor driver.

1.5.2. Setting up CDK

The **minishift setup-cdk** command gets and configures the components needed to run CDK on your system. By default, **minishift setup-cdk** places CDK content in your *~/.minishift* directory (*%USERPROFILE%/.minishift* on Windows).



IMPORTANT

To use a directory other than *~/.minishift*, you must set the **--minishift-home** flag or **MINISHIFT_HOME** environment variable, as described in Environment Variables.

Run the following command to set up CDK for Red Hat Enterprise Linux:

```
$ minishift setup-cdk
Setting up CDK 3 on host using '/home/user/.minishift' as Minishift's home directory
Copying minishift-rhel7.iso to '/home/user/.minishift/cache/iso/minishift-rhel7.iso'
Copying oc to '/home/user/.minishift/cache/oc/v3.10.45/linux/oc'
Creating configuration file '/home/user/.minishift/config/config.json'
Creating marker file '/home/user/.minishift/cdk'
Default add-ons anyuid, admin-user, xpaas, registry-route, che, eap-cd installed
Default add-ons anyuid, admin-user, xpaas enabled
CDK 3 setup complete.
```

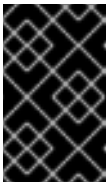
For Windows or macOS: Running the **minishift setup-cdk** command on Windows and macOS results in slightly different output, based on some different components and pathnames.

1.5.3. Starting CDK

1. By default, **minishift start** prompts you for your Red Hat Subscription Manager account username and password. You can enter that information or choose instead to:

   - Skip registration: Add the **--skip-registration** option to **minishift start** to not register the CDK VM.

   - Register permanently: You can export registration information to environment variables so that **minishift** picks it up automatically each time it starts.

     IMPORTANT

     Storing unencrypted registration information in environment variables is not secure. Entering your credentials through the **minishift start** prompt is recommended for security.

     Export your registration information as follows:

     For Red Hat Enterprise Linux:

     ```
     $ export MINISHIFT_USERNAME='<RED_HAT_USERNAME>'
     ```

```
$ export MINISHIFT_PASSWORD='<RED_HAT_PASSWORD>'
$ echo export MINISHIFT_USERNAME=$MINISHIFT_USERNAME >> ~/.bashrc
$ echo export MINISHIFT_PASSWORD=$MINISHIFT_PASSWORD >> ~/.bashrc
```

For macOS:

```
$ export MINISHIFT_USERNAME='<RED_HAT_USERNAME>'
$ export MINISHIFT_PASSWORD='<RED_HAT_PASSWORD>'
$ echo export MINISHIFT_USERNAME=$MINISHIFT_USERNAME >> ~/.bash_profile
$ echo export MINISHIFT_PASSWORD=$MINISHIFT_PASSWORD >> ~/.bash_profile
```

For Windows:

Using Command Prompt:

```
C:\> set MINISHIFT_USERNAME='<RED_HAT_USERNAME>'
C:\> set MINISHIFT_PASSWORD='<RED_HAT_PASSWORD>'
C:\> setx MINISHIFT_USERNAME %MINISHIFT_USERNAME%
C:\> setx MINISHIFT_PASSWORD %MINISHIFT_PASSWORD%
```

Using PowerShell:

```
PS> $env:MINISHIFT_USERNAME = '<RED_HAT_USERNAME>'
PS> $env:MINISHIFT_PASSWORD = '<RED_HAT_PASSWORD>'
PS> setx MINISHIFT_USERNAME $env:MINISHIFT_USERNAME
PS> setx MINISHIFT_PASSWORD $env:MINISHIFT_PASSWORD
```

2. Run the **minishift start** command:

```
$ minishift start
-- Starting profile 'minishift'
...
-- Minishift VM will be configured with ...
   Memory:    4 GB
   vCPUs :    2
   Disk size: 20 GB
-- Starting Minishift VM ..... OK
-- Registering machine using subscription-manager
   Registration in progress ..... OK [42s]
...
   OpenShift server started.

   The server is accessible via web console at:
      https://192.168.42.60:8443/console

   You are logged in as:
      User:     developer
      Password: <any value>

   To login as administrator:
      oc login -u system:admin
...
```

NOTE

- The IP is dynamically generated for each OpenShift cluster. To check the IP, run the **minishift ip** command.
- By default, CDK uses the driver most relevant to the host OS. To use a different driver, set the **--vm-driver** flag in **minishift start**. For example, to use VirtualBox instead of KVM on Linux operating systems, run **minishift start --vm-driver=virtualbox**.
- While CDK starts it runs several checks to make sure that the CDK VM and the OpenShift cluster are able to start correctly. If any startup checks fail, see the Troubleshooting Getting Started topic for information about possible causes and solutions.

For more information about **minishift start** options, see the **minishift start** command reference.

3. Use **minishift oc-env** to display the command you need to type into your shell in order to add the **oc** binary to your **PATH** environment variable. The output of **oc-env** will differ depending on OS and shell type.

```
$ minishift oc-env
export PATH="/home/user/.minishift/cache/oc/v3.11.374/linux:$PATH"
# Run this command to configure your shell:
# eval $(minishift oc-env)
```

For more information about interacting with OpenShift with the command line interface and the Web console, see the OpenShift Client Binary section.

1.5.4. Deploying a Sample Application

OpenShift provides various sample applications, such as templates, builder applications, and quickstarts. The following steps describe how to deploy a sample Node.js application from the command line.

1. Create a Node.js example application:

```
$ oc new-app https://github.com/openshift/nodejs-ex -l name=myapp
```

2. Track the build log until the application is built and deployed:

```
$ oc logs -f bc/nodejs-ex
```

3. Expose a route to the service:

```
$ oc expose svc/nodejs-ex
```

4. Access the application:

```
$ minishift openshift service nodejs-ex --in-browser
```

5. To stop CDK, use the following command:

```
$ minishift stop
Stopping local OpenShift cluster...
Unregistering machine
Cluster stopped.
```

For more information about creating applications in OpenShift, see Creating New Applications in the OpenShift documentation.

1.6. UNINSTALLING CDK

1.6.1. Overview

This section describes how you can uninstall the **minishift** binary and delete associated files.

1.6.2. Uninstalling CDK

1. Delete the CDK VM and any VM-specific files:

   ```
   $ minishift delete
   ```

   This command deletes everything in the ***MINISHIFT_HOME/.minishift/machines/minishift*** directory. Other cached data and the persistent configuration are not removed.

2. To completely uninstall CDK, delete everything in the **MINISHIFT_HOME** directory (default ***~/.minishift***):
   For Red Hat Enterprise Linux and macOS:

   ```
   $ rm -rf ~/.minishift
   ```

   For Windows PowerShell:

   Replace **<MINISHIFT_HOME>** with the location of your home directory.

   ```
   PS C:\> Remove-Item -Recurse -Force C:\<MINISHIFT_HOME>\.minishift\
   ```

   For Windows command prompt:

   Replace **<MINISHIFT_HOME>** with the location of your home directory. (You may need to use the **del /s** command instead.)

   ```
   c:\> rm -r c:\<MINISHIFT_HOME>\.minishift
   ```

3. With your hypervisor management tool, confirm that there are no remaining artifacts related to the CDK VM. For example, if you use KVM, you need to run the **virsh** command.

CHAPTER 2. USING CDK

This section describes how to use the **minishift** command-line interface (CLI). It covers the basic commands as well as advanced features like profiles, image caching, add-ons and host folders.

For details about using the **minishift** and **oc** commands to manage your local OpenShift cluster, see Chapter 3, *Interacting with OpenShift using Container Development Kit*.

2.1. BASIC USAGE

2.1.1. Overview

When you use CDK, you interact with the following components:

- the CDK virtual machine (VM)
- the Docker daemon running on the VM
- the OpenShift cluster running on the Docker daemon

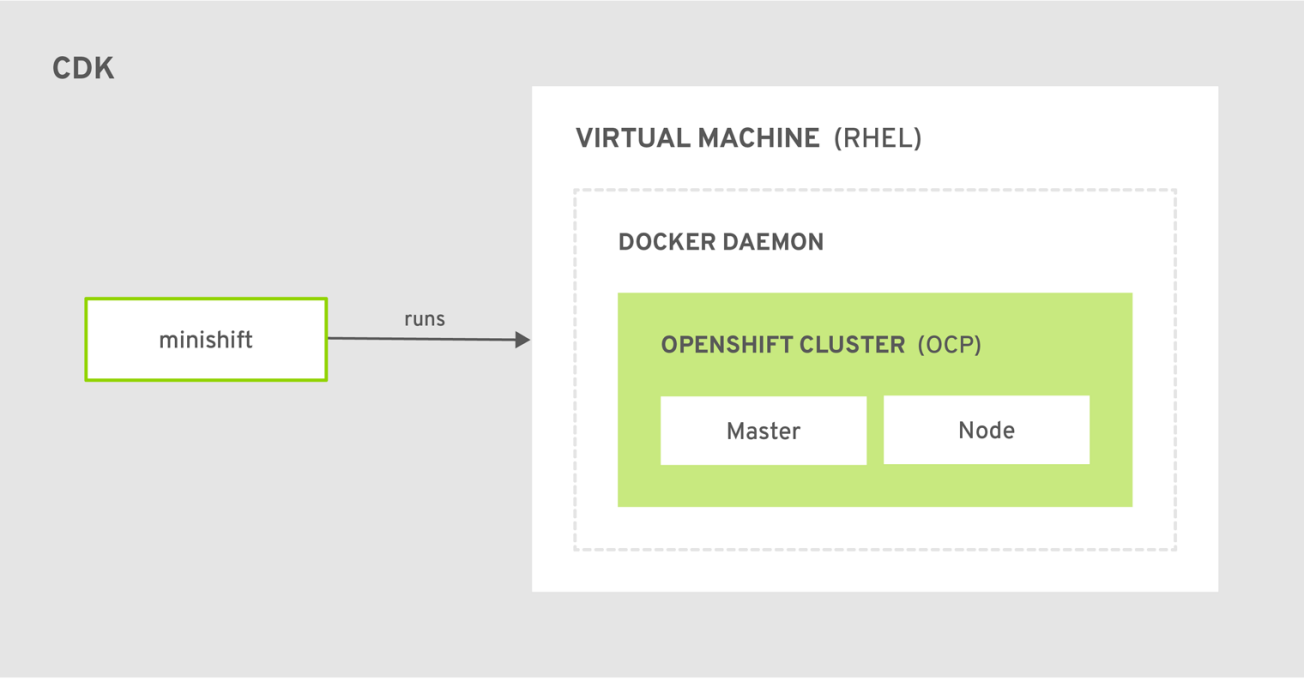
The CDK architecture diagram outlines these components. The **minishift** binary, placed on the **PATH** for easy execution, is used to **start**, **stop**, and **delete** the CDK VM. The VM itself is bootstrapped off of a Red Hat Enterprise Linux Live ISO.

Some CDK commands, for example **docker-env**, interact with the Docker daemon, whilst others communicate with the OpenShift cluster, for example the **openshift** command.

Once the OpenShift cluster is up and running, you interact with it using the **oc** binary. CDK caches this binary under **$MINISHIFT_HOME** (per default *~/.minishift*). **minishift oc-env** is an easy way to add the **oc** binary to your **PATH**.

For more details about using CDK to manage your local OpenShift cluster, see Chapter 3, *Interacting with OpenShift using Container Development Kit*.

Figure 2.1. : CDK architecture



2.1.2. CDK Life-cycle

2.1.2.1. The minishift setup-cdk Command

The **minishift setup-cdk** command copies a Red Hat Enterprise Linux ISO image to the CDK cache, creates a default configuration file, installs and enables default add-ons, and creates a marker file to confirm that the command has been run.

The command also copies the **oc** binary to your host so that you can interact with OpenShift through the **oc** command line tool or through the Web console, which can be accessed through the URL provided in the output of the **minishift start** command.

2.1.2.2. The minishift start Command

The **minishift start** command creates and configures the CDK VM and provisions a local, single-node OpenShift cluster within the CDK VM.

2.1.2.3. The minishift stop Command

The **minishift stop** command stops your OpenShift cluster and shuts down the CDK VM, but preserves the OpenShift cluster state.

Starting CDK again will restore the OpenShift cluster, allowing you to continue working from the last session. However, you must enter the same parameters that you used in the original start command.

See the Persistent Configuration section for more information on how to persist CDK settings.

2.1.2.4. The minishift delete Command

The **minishift delete** command deletes the OpenShift cluster, and also shuts down and deletes the CDK VM. No data or state are preserved.

2.1.3. Runtime Options

The runtime behavior of CDK can be controlled through flags, environment variables, and persistent configuration options.

The following precedence order is applied to control the behavior of CDK. Each action in the following list takes precedence over the action below it:

1. Use command line flags as specified in the Flags section.
2. Set environment variables as described in the Environment Variables section.
3. Use persistent configuration options as described in the Persistent Configuration section.
4. Accept the default value as defined by CDK.

2.1.3.1. Flags

You can use command line flags with CDK to specify options and direct its behavior. This has the highest precedence. Almost all commands have flags, although different commands might have different flags. Some of the commonly-used command line flags of the **minishift start** command are **cpus**, **memory** or **vm-driver**.

2.1.3.2. Environment Variables

CDK allows you to specify command line flags you commonly use through environment variables. To do so, apply the following rules to the flag you want to set as an environment variable.

1. Apply **MINISHIFT_** as a prefix to the flag you want to set as an environment variable. For example, the **vm-driver** flag of the **minishift start** command becomes **MINISHIFT_vm-driver**.
2. Use uppercase characters for the flag, so **MINISHIFT_vm-driver** in the above example becomes **MINISHIFT_VM-DRIVER**.
3. Replace **-** with _, so **MINISHIFT_VM-DRIVER** becomes **MINISHIFT_VM_DRIVER**.

Environment variables can be used to replace any option of any CDK command. A common example is the URL of the ISO to be used. Usually, you specify it with the **iso-url** flag of the **minishift start** command. Applying the above rules, you can also specify this URL by setting the environment variable as **MINISHIFT_ISO_URL**.

NOTE

You can also use the **MINISHIFT_HOME** environment variable, to choose a different home directory for CDK. By default, the **minishift** command places all runtime state into *~/.minishift*. This environment variable is currently experimental and semantics might change in future releases.

To use **MINISHIFT_HOME**, you should set the new home directory when you first set up CDK. For example, this sets the **minishift** home directory to *~/.mynewdir* on a Linux system:

```
$ minishift setup-cdk --minishift-home ~/.mynewdir
$ export MINISHIFT_HOME=~/.mynewdir
$ echo 'export MINISHIFT_HOME=~/.mynewdir' >> ~/.bashrc
```

2.1.3.3. Persistent Configuration

Using persistent configuration allows you to control CDK behavior without specifying actual command line flags, similar to the way you use environment variables. You can also define global configuration using the **--global** flag. Global configuration is applicable to all profiles.

CDK maintains a configuration file in *$MINISHIFT_HOME/config/config.json*. This file can be used to set commonly-used command line flags persistently for individual profiles. The global configuration file is maintained at *MINISHIFT_HOME/config/global.json*.

NOTE

- Persistent configuration can only be applied to the set of supported configuration options that are listed in the synopsis of the **minishift config** sub-command. This is different from environment variables, which can be used to replace any option of any command.
- By default, **minishift** persists start flags to persistent configuration. Use **minishift config set save-start-flags false** to disable this behavior.

2.1.3.3.1. Setting Persistent Configuration Values

The easiest way to change a persistent configuration option is with the **minishift config set** sub-command. For example:

```
# Set default memory 4096 MB
$ minishift config set memory 4096
```

The easiest way to set a persistent configuration option across all profiles is with the **minishift config set --global** sub-command.

For example, you can set the default memory to 8192 MB for every profile as follows:

```
$ minishift config set --global memory 8192
```

Flags which can be used multiple times per command invocation, like **docker-env** or **insecure-registry**, need to be comma-separated when used with the **config set** command. For example, from the CLI, you can use **insecure-registry** like this:

```
$ minishift start --insecure-registry hub.foo.com --insecure-registry hub.bar.com
```

To configure the same registries in the persistent configuration, run the following command:

```
$ minishift config set insecure-registry hub.foo.com,hub.bar.com
```

To view all persistent configuration values, you can use the **minishift config view** sub-command:

```
$ minishift config view
- memory: 4096
```

To view all persistent configuration values in the global configuration file, you can use the **minishift config view --global** sub-command:

```
$ minishift config view --global
- memory: 8192
```

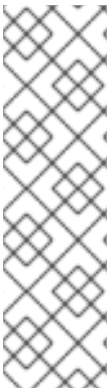
Alternatively, you can display a single value with the **minishift config get** sub-command:

```
$ minishift config get memory
4096
```

2.1.3.3.2. Unsetting Persistent Configuration Values

To remove a persistent configuration option, you can use the **minishift config unset** sub-command. For example:

```
$ minishift config unset memory
```



NOTE

The precedence for user-defined values is as follows:

1. Command line flags.
2. Environment variables.
3. Instance-specific configuration.
4. Global configuration.

2.1.4. Persistent Volumes

As part of the OpenShift cluster provisioning, 100 persistent volumes are created for your OpenShift cluster. This allows applications to make persistent volumes claims. The location of the persistent data is determined in the **host-pv-dir** flag of the **minishift start** command and defaults to */var/lib/minishift/openshift.local.pv* on the CDK VM.

2.1.5. HTTP/HTTPS Proxies

If you are behind an HTTP/HTTPS proxy, you need to supply proxy options to allow Docker and OpenShift to work properly. To do this, pass the required flags during **minishift start**.

For example:

```
$ minishift start --http-proxy http://YOURPROXY:PORT --https-proxy https://YOURPROXY:PORT
```

In an authenticated proxy environment, the **proxy_user** and **proxy_password** must be a part of proxy URI.

```
$ minishift start --http-proxy http://<proxy_username>:<proxy_password>@YOURPROXY:PORT \
                  --https-proxy https://<proxy_username>:<proxy_password>@YOURPROXY:PORT
```

You can also use the **--no-proxy** flag to specify a comma-separated list of hosts that should not be proxied.

Using the proxy options will transparently configure the Docker daemon as well as OpenShift to use the specified proxies.

NOTE

- **minishift start** honors the environment variables **HTTP_PROXY**, **HTTPS_PROXY** and **NO_PROXY**. The environment variables can be specified in lower case or upper case. The lower case variables take higher precedence over upper case variables. If these variables are set, they are implicitly used during **minishift start** unless explicitly overridden by the corresponding command line flags.
- CDK does not escape the special characters in environment variables for proxies. Special characters must be manually escaped.
- Use the **minishift start --openshift-version** flag to request a specific version of OpenShift Container Platform. You can list all CDK-compatible OpenShift Container Platform versions with the **minishift openshift version list** command.

2.1.6. Networking

The CDK VM is exposed to the host system with a host-only IP address that can be obtained with the **minishift ip** command.

2.1.7. Connecting to the CDK VM with SSH

You can use the **minishift ssh** command to interact with the CDK VM.

You can run **minishift ssh** without a sub-command to open an interactive shell and run commands on the CDK VM in the same way that you run commands interactively on any remote machine using SSH.

You can also run **minishift ssh** with a sub-command to send the sub-command directly to the CDK VM and return the result to your local shell. For example:

```
$ minishift ssh -- docker ps
CONTAINER    IMAGE                  COMMAND                  CREATED       STATUS       NAMES
71fe8ff16548 openshift3/ose:v3.11.374 "/usr/bin/openshift s" 4 minutes ago  Up 4 minutes  origin
```

2.2. CDK PROFILES

2.2.1. Overview

IMPORTANT

You must run **minishift setup-cdk** before using profiles.

A profile is an instance of the Minishift VM along with all of its configuration and state. The profile feature allows you to create and manage these isolated instances of Minishift.

Each CDK profile is created with its own configuration (memory, CPU, disk size, add-ons, and so on) and is independent of other profiles. Refer to the use of environment variables if you want to make sure that certain configuration settings, for example **cpus** or **memory**, get applied to all profiles.

The *active* profile is the profile against which all commands are executed, unless the global **--profile** flag is used. You can determine the active profile by using the **minishift profile list** command. You can execute single commands against a non-active profile by using the **--profile** flag, for example **minishift --profile profile-demo console** to open the OpenShift console for the specified **profile-demo** profile.

On top of the **--profile** flag, there are commands for listing, deleting and setting the active profile. These commands are described in the following sections.

WARNING

Even though profiles are independent of each other, they share the same cache for ISOs, **oc** binaries and container images. **minishift delete --clear-cache** will affect all profiles for this reason. We recommend using **--clear-cache** with caution.

2.2.2. Creating Profiles

There are two ways to create a new profile.

NOTE

Profile names can only consist of alphanumeric characters. Underscores (_) and hyphens (-) are allowed as separators.

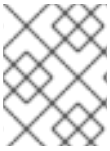
2.2.2.1. Using the --profile Flag

When you run the **minishift start** command with the **--profile** flag the profile is created if it does not already exist. For example:

```
$ minishift --profile profile-demo start
-- Starting profile 'profile-demo'
-- Check if deprecated options are used ... OK
-- Checking if https://github.com is reachable ... OK
-- Checking if requested OpenShift version 'v3.11.0' is valid ... OK
-- Checking if requested OpenShift version 'v3.11.0' is supported ... OK
-- Checking if requested hypervisor 'hyperkit' is supported on this platform ... OK
-- Checking if hyperkit driver is installed ...
```

```
   Driver is available at /usr/local/bin/docker-machine-driver-hyperkit
   Checking for setuid bit ... OK
-- Checking the ISO URL ... OK
-- Checking if provided oc flags are supported ... OK
-- Starting the OpenShift cluster using 'hyperkit' hypervisor ...
-- Minishift VM will be configured with ...
   Memory:    4 GB
   vCPUs :    2
   Disk size: 20 GB
-- Starting Minishift VM .....
```

See also Example Workflow for Profile Configuration.



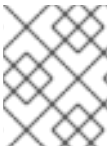
NOTE

A profile automatically becomes the active profile when a CDK instance is started successfully via **minishift start**.

2.2.2.2. Using the profile set Command

The other option to create a profile is to use the **profile set** command. If the specified profile does not exist, it is implicitly created:

```
$ minishift profile set demo
Profile 'demo' set as active profile
```



NOTE

The default profile is **minishift**. It will be present by default and it does not need to be created.

2.2.3. Listing Profiles

You can list all existing profiles with the **minishift profile list** command. You can also see the active profile highlighted in the output.

```
$ minishift profile list
- minishift     Running      (Active)
- profile-demo  Does Not Exist
```

2.2.4. Switching Profiles

To switch between profiles use the **minishift profile set** command:

```
$ minishift profile set profile-demo
Profile 'profile-demo' set as active profile
```

NOTE

Only one profile can be active at any time.

2.2.5. Deleting Profiles

To delete a profile, run:

```
$ minishift profile delete profile-demo
You are deleting the active profile. It will remove the VM and all related artifacts. Do you want to
continue [y/N]?: y
Deleted:  /Users/user/.minishift/profiles/profile-demo
Profile 'profile-demo' deleted successfully
Switching to default profile 'minishift' as the active profile.
```

NOTE

The default profile **minishift** cannot be deleted.

2.2.6. Example Workflow for Profile Configuration

You have two options to create a new profile and configure its persistent configuration. The first option is to implicitly create the new profile by making it the active profile using the **profile set** command. Once the profile is active you can run any **minishift config** command. Lastly, start the instance:

```
$ minishift profile set profile-demo
$ minishift config set memory 8GB
$ minishift config set cpus 4
$ minishift addon enable anyuid
$ minishift start
```

The alternative is to execute a series of commands each specifying the targeted profile explicitly using the **--profile** flag:

```
$ minishift --profile profile-demo config set memory 8GB
$ minishift --profile profile-demo config set cpus 4
$ minishift --profile profile-demo addon enable anyuid
$ minishift --profile profile-demo start
```

2.3. IMAGE CACHING

2.3.1. Overview

To speed up the provisioning of the OpenShift cluster and to minimize network traffic, container images can be cached on the host. These images can then be imported into the running Docker daemon, either explicitly on request or implicitly during **minishift start**. The following sections describe image caching and its configuration in more detail.

NOTE

The format in which images are cached has changed with CDK version 3.3.0. Prior to 3.3.0, the images were stored as tar files. As of 3.3.0, images are stored in the OCI image format.

If you used image caching prior to CDK 3.3.0, your cache will need to be recreated. If you want to remove the obsolete pre-3.3.0 images, you can clear your cache via:

```
$ minishift delete --clear-cache
```

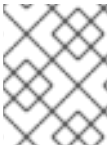
2.3.2. Explicit Image Caching

CDK provides the **image** command together with its sub-commands to control the behavior of image caching. To export and import images from the Docker daemon of the CDK VM, use **minishift image export** and **minishift image import**.

2.3.2.1. Importing and Exporting Single Images

Once the CDK VM is running, images can be explicitly exported from the Docker daemon:

```
$ minishift image export <image-name-0> <image-name-1> ...
Pulling image <image-name-0> .. OK
Exporting <image-name-0>. OK
Pulling image <image-name-1> .. OK
Exporting <image-name-2>. OK
```



NOTE

Images which are not available in the Docker daemon will be pulled prior to being exported to the host.

To import previously cached images, use the **minishift image import** command:

```
$ minishift image import <image-name-0> <image-name-1> ...
Importing <image-name-0> . OK
```

2.3.2.2. Listing Cached Images

The **minishift image list** command lists either the currently cached images or the images available in the CDK Docker daemon.

To view currently cached images on the host:

```
$ minishift image list
registry.access.redhat.com/openshift3/ose-docker-registry:v3.11.374
registry.access.redhat.com/openshift3/ose-haproxy-router:v3.11.374
registry.access.redhat.com/openshift3/ose:v3.11.374
```

To view images available in the Docker daemon:

```
$ minishift image list --vm
```

```
registry.access.redhat.com/openshift3/ose-deployer:v3.11.374
registry.access.redhat.com/openshift3/ose-docker-registry:v3.11.374
registry.access.redhat.com/openshift3/ose-haproxy-router:v3.11.374
registry.access.redhat.com/openshift3/ose-pod:v3.11.374
registry.access.redhat.com/openshift3/ose-web-console:v3.11.374
registry.access.redhat.com/openshift3/ose:v3.11.374
```

2.3.2.3. Persisting Cached Image Names

In order to avoid having to type the image names explicitly as part of the **image export** or **image import** command, you can store a list of image names for import and export in the persistent configuration.

Use **minishift image cache-config view** to view the list of currently configured images and **minishift image cache-config add** to add images to the list:

```
$ minishift image cache-config view
$ minishift image cache-config add alpine:latest busybox:latest
$ minishift image cache-config view
alpine:latest
busybox:latest
```

To remove images from the list use **minishift image cache-config remove**:

```
$ minishift image cache-config remove alpine:latest
$ minishift image cache-config view
busybox:latest
```

Once the image names are stored in the persistent configuration, you can run **minishift image export** and **minishift image import** without any arguments.

2.3.2.4. Exporting and Importing All Images

You can export and import all images using the **--all** flag. For the export command, this means that all images currently available on the Docker daemon will be exported to the host. For the import command, it means that all images available in the local CDK cache will be imported into the Docker daemon of the CDK VM.

WARNING

Exporting and importing all images can take a long time and locally cached images can take up a considerable amount of disk space. We recommend using this feature with caution.

2.3.3. Implicit Image Caching

Image caching is enabled by default for CDK. It occurs in a background process after the **minishift start** command is completed for the first time. Once the images are cached under ***$MINISHIFT_HOME/cache/image***, successive CDK VM creations will use these cached images.

To disable this feature you need to disable the **image-caching** property in the persistent configuration using the **minishift config set** command:

```
$ minishift config set image-caching false
```

NOTE

Implicit image caching will transparently add the required OpenShift images to the list of cached images as specified per **cache-images** configuration option. See Persisting Cached Image Names.

NOTE

Each time an image exporting background process runs, a log file is generated under ***$MINISHIFT_HOME/logs*** which can be used to verify the progress of the export.

You can re-enable the caching of the OpenShift images by setting **image-caching** to **true** or removing the setting altogether using **minishift config unset**:

```
$ minishift config unset image-caching
```

2.3.4. Delete Image from Local Cache

Images are cached under ***$MINISHIFT_HOME/cache/images*** and stored as an SHA256 blob along with an index which contains details about matching the SHA with an image name.

To delete an image from the local cache, use the **minishift image delete** command:

```
$ minishift image delete <image-name-0> <image-name-1> ...
Deleting <image-name-0> from the local cache OK
Deleting <image-name-1> from the local cache OK
```

2.4. ADD-ONS

2.4.1. Overview

CDK allows you to extend the vanilla OpenShift setup provided by **cluster up** with an add-on mechanism.

Add-ons are directories that contain a text file with the ***.addon*** extension. The directory can also contain other resource files such as JSON template files. However, only one ***.addon*** file is allowed per add-on.

The following example shows the contents of an add-on file, including the name and description of the add-on, additional metadata, and the actual add-on commands to apply.

Example: anyuid add-on definition file

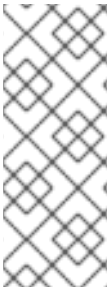
```
# Name: anyuid                                                                  1
# Description: Allows authenticated users to run images under a non pre-allocated UID  2
# Required-Vars: ACME_TOKEN                                                     3
# OpenShift-Version: >3.6.0                                                     4
```

```
# Minishift-Version: >1.22.0 5

mytoken := oc sa get-token oauth-client 6

oc new-app -p OPENSHIFT_OAUTH_CLIENT_SECRET=#{mytoken} 7
oc adm policy add-scc-to-group anyuid system:authenticated 8
```

1. (Required) Name of the add-on.
2. (Required) Description of the add-on.
3. (Optional) Comma separated list of required interpolation variables. See Variable Interpolation.
4. (Optional) OpenShift version required to run a specific add-on. See OpenShift Version Semantics.
5. (Optional) CDK version required to run a specific add-on. See Minishift Version Semantics.
6. Run an actual add-on command and store its output in a variable. See Internal Variables.
7. Actual add-on command. In this case, the command executes the **oc** binary using the **mytoken** variable value.
8. Actual add-on command. In this case, the command executes the **oc** binary.



NOTE

- Comment lines, starting with the '#' character, can be inserted anywhere in the file.
- Commands starting with the **!** character ignore execution failure. With the help of this, add-on will be idempotent i.e. a command can be executed multiple times without changing the final behavior of the add-on.

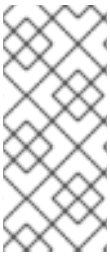
Enabled add-ons are applied during **minishift start**, immediately after the initial cluster provisioning is successfully completed.

2.4.2. OpenShift-Version Semantics

As part of the add-on metadata you can specify the OpenShift version which needs to be running in order to apply the add-on. To do so, you can specify the optional **OpenShift-Version** metadata field. The semantics are as follows:

>3.6.0	OpenShift version higher than 3.6.0 needs to be running in order to apply the add-on.
>=3.6.0	OpenShift version 3.6.0 or higher needs to be running in order to apply the add-on.
3.6.0	OpenShift version 3.6.0 needs to be running in order to apply the add-on.
>=3.5.0, <3.8.0	OpenShift version should higher or equal to 3.5.0 but lower than 3.8.0.

>=3.5.0, <=3.8.0	OpenShift version should higher or equal to 3.5.0 but lower or equal to 3.8.0.



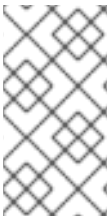
NOTE

- If the metadata field **OpenShift-Version** is not specified in the add-on header, the add-on can be applied against any version of OpenShift.
- **OpenShift-Version** only supports versions in the form of <major>.<minor>.<patch>.

2.4.3. Minishift-Version Semantics

As part of the add-on metadata you can specify the Minishift version which needs to be running in order to apply the add-on. To do so, you can specify the optional **Minishift-Version** metadata field. The semantics are as follows:

>1.22.0	Minishift version greater than 1.22.0 needs to be running in order to apply the add-on.
>=1.22.0	Minishift version greater than or equal to 1.22.0 needs to be running in order to apply the add-on.
1.22.0	Minishift version 1.22.0 needs to be running in order to apply the add-on.
>=1.21.0, <1.25.0	Minishift version must be greater than or equal to 1.21.0, but less than 1.25.0.
>=1.22.0, <=1.25.0	Minishift version must be greater than or equal to 1.22.0, but less than or equal to 1.25.0.



NOTE

- If the metadata field **Minishift-Version** is not specified in the add-on header, the add-on can be applied against any version of CDK.
- **Minishift-Version** only supports versions in the form of <major>.<minor>.<patch>.

2.4.4. Defining Add-on Dependencies

Add-on dependencies can be defined using the *Depends-On* metadata field. Multiple dependencies may be defined using a comma-separated list of add-on names.

Example: Defining add-on dependency in add-on definition file

```
# Name: example
# Description: Shows the use of the Depends-On metadata field
# Depends-On: anyuid, admin-user

echo Depends on the anyuid and admin-user add-ons, requiring them to be installed.
```

2.4.5. Add-on Commands

This section describes the commands that an add-on file can contain. They form a small domain-specific language for CDK add-ons:

ssh

If the add-on command starts with **ssh**, you can run any command within the CDK-managed VM. This is similar to running **minishift ssh** and then executing any command on the VM. For more information about **minishift ssh** command usage, see Connecting to the CDK VM with SSH .

oc

If the add-on command starts with **oc**, it uses the **oc** binary that is cached on your host to execute the specified **oc** command. This is similar to running **oc --as system:admin ...** from the command line.

NOTE

The **oc** command is executed as **system:admin**.

openshift

If the add-on command starts with **openshift**, it uses the **oc** binary present in the OpenShift container to execute the command. This means that any file parameters or other system-specific parameters must match the environment of the container instead of your host.

docker

If the add-on command starts with **docker**, it executes a **docker** command against the Docker daemon within the CDK VM. This is the same daemon on which the single-node OpenShift cluster is running as well. This is similar to running **eval $(minishift docker-env)** on your host and then executing any **docker** command. See also **minishift docker-env**.

echo

If the add-on command starts with **echo**, the arguments following the **echo** command are printed to the console. This can be used to provide additional feedback during add-on execution.

sleep

If the add-on command starts with **sleep**, it waits for the specified number of seconds. This can be useful in cases where you know that a command such as **oc** might take a few seconds before a certain resource can be queried.

cat

If the add-on command starts with **cat**, the arguments following the **cat** command must be a valid file path. This will print the contents of a valid file to the console or display an error message if the file path is invalid. This can be useful to use file contents with another command.

NOTE

Trying to use an undefined command will cause an error when the add-on gets parsed.

2.4.6. Variable Interpolation

CDK allows the use of variables within the add-on commands. Variables have the format **#{<variable-name>}**. The following example shows how the OpenShift routing suffix can be interpolated into an **openshift** command to create a new certificate as part of securing the OpenShift registry. The used variable **#{routing-suffix}** is part of the built-in add-on variables.

Example: Usage of the routing-suffix variable

```
$ openshift admin ca create-server-cert \
  --signer-cert=/var/lib/origin/openshift.local.config/master/ca.crt \
  --signer-key=/var/lib/origin/openshift.local.config/master/ca.key \
  --signer-serial=/var/lib/origin/openshift.local.config/master/ca.serial.txt \
  --hostnames='docker-registry-default.#{routing-suffix},docker-
registry.default.svc.cluster.local,172.30.1.1' \
  --cert=/etc/secrets/registry.crt \
  --key=/etc/secrets/registry.key
```

2.4.6.1. Built-in Variables

There exist several built-in variables which are available for interpolation at all times. The following table shows these variables.

Table 2.1. Supported built-in add-on variables

Variable	Description
ip	IP of the CDK VM.
routing-suffix	OpenShift routing suffix for the application.
addon-name	Name of the current add-on.
user	User used by CDK for execution of commands through SSH.

2.4.6.2. Dynamic Variables

The commands **minishift start** as well as **minishift addons apply** also provide an **--addon-env** flag which allows to dynamically pass variables for interpolation, for example:

```
$ minishift addons apply --addon-env PROJECT_USER=user acme
```

The **--addon-env** flag can be specified multiple times to define more than one variables for interpolation.

Specifying dynamic variables also works in conjunction with setting persistent configuration values.

```
$ minishift config set addon-env PROJECT_USER=user
$ minishift addons apply acme
```

NOTE

Multiple variables must be comma separated when the **minishift config set** command is used.

There is also the possibility to dynamically interpolate a variable with the value of an environment variable at the time the add-on gets applied. For this the variable value needs to be prefixed with *env.*

```
$ minishift config set addon-env PROJECT_USER=env.USER      (1)
$ minishift addons apply acme                               (2)
```

(1) Using the *env* prefix ensures that instead of literally replacing '#{PROJECT_USER}' with 'env.USER', the value of the environment variable **USER** is used. If the environment variable is not set, interpolation does not occur.

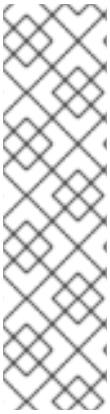
(2) When the add-on is applied, each occurrence of **#{PROJECT_USER}** within an add-on command gets replaced with the value of the environment variable **USER**.

As an add-on developer, you can enforce that a variable value is provided when the add-on gets applied by adding the variable name to the *Required-Vars* metadata header. Multiple variables need to be comma separated.

```
# Name: acme
# Description: ACME add-on
# Required-Vars: PROJECT_USER
```

You can also provide default values for variables using the *Var-Defaults* metadata header. *Var-Defaults* needs to be specified in the format of **<key>=<value>**. Multiple default key/value pairs need to be comma separated.

```
# Name: acme
# Description: ACME add-on
# Required-Vars: PROJECT_USER
# Var-Defaults: PROJECT_USER=user
```



NOTE

- **=** and **,** are metacharacters and cannot be used as part of keys or values.
- An empty value will be set if "NULL", "Null", or "null" is specified as the value for a *Var-Defaults* key. For example:

  ```
  # Var-Defaults: PROJECT_USER=null
  ```

- Variable values specified via the command line using the **--addon-env** or set via **minishift config set addon-env** have precedence over *Var-Defaults*.

2.4.6.3. Internal Variables

Add-ons allow the definition of variables. These variables can be used to store the output of an add-on command.

For example, you can save **oc** command output in a **mytoken** variable and use it in subsequent add-on commands like so:

```
mytoken := oc sa get-token oauth-client
oc new-app -p OPENSHIFT_OAUTH_CLIENT_SECRET=#{mytoken}
```

2.4.7. Default Add-ons

CDK provides a set of built-in add-ons that offer some common OpenShift customization to assist with development. During **minishift setup-cdk**, Minishift automatically installs and enables the **xpaas**, **anyuid**, and **admin-user** add-ons. To install the default add-ons, run:

```
$ minishift addons install --defaults
```

This command extracts the default add-ons to the add-on installation directory ***$MINISHIFT_HOME/addons***. To view the list of installed add-ons, you can then run:

```
$ minishift addons list --verbose=true
```

This command prints a list of installed add-ons. You should at least see the **anyuid** add-on listed. This is an important add-on that allows you to run images that do not use a pre-allocated UID. By default, this is not allowed in OpenShift.

Table 2.2. Default add-ons

Add-on Name	Description
anyuid	Changes the default security context constraints to allow pods to run with arbitrary UID.
admin-user	Creates a user named 'admin' and assigns the **cluster-admin** role to it.
che	Runs an instance of the Eclipse Che integrated development environment.
eap-cd	Imports JBoss EAP CD templates.
htpasswd-identity-provider	User can change and add default login username and password for the OpenShift instance.
registry-route	Creates an edge terminated route for the OpenShift registry.
xpaas	Imports xPaaS templates.

NOTE

The **eap-cd** add-on provides templates for JBoss EAP CD 19. To use a more recent version of JBoss EAP CD, use the upstream **eap-cd** add-on. See Add-ons by the Community for more information on how to install the upstream add-on.

2.4.7.1. Add-ons by the Community

Apart from the several default add-ons, there are a number of community developed add-ons for CDK. Community add-ons can be found in the minishift-addons repository. You can get all the information about these add-ons in the repository. The instructions for installing them can be found in the README.

2.4.8. Installing Add-ons

Add-ons are installed with the **minishift addons install** command.

The following example shows how to install an add-on.

Example: Installing an add-on

```
$ minishift addons install <path_to_addon_directory>
```

2.4.9. Enabling and Disabling Add-ons

Add-ons are enabled with the **minishift addons enable** command and disabled with the **minishift addons disable** command. Enabled add-ons automatically get executed during **minishift start**.

The following examples show how to enable and disable the **anyuid** add-on.

Example: Enabling the anyuid add-on

```
$ minishift addons enable anyuid
```

Example: Disabling the anyuid add-on

```
$ minishift addons disable anyuid
```

2.4.9.1. Add-on Priorities

When you enable an add-on, you can also specify a priority, which determines the order that the add-ons are applied.

The following example shows how to enable the **registry** add-on with a higher priority value.

Example: Enabling the registry add-on with priority

```
$ minishift addons enable registry --priority=5
```

The add-on priority attribute determines the order in which add-ons are applied. By default, an add-on has the priority 0. Add-ons with a lower priority value are applied first.

In the following example, the **anyuid**, **registry**, and **eap** add-ons are enabled with the respective priorities of 0, 5 and 10. This means that **anyuid** is applied first, followed by **registry**, and lastly the **eap** add-on.

Example: List command output with explicit priorities

```
$ minishift addons list
- anyuid        : enabled    P(0)
- registry      : enabled    P(5)
- eap           : enabled    P(10)
```

NOTE

If two add-ons have the same priority the order in which they are getting applied is not determined.

2.4.10. Applying Add-ons

Add-ons can be explicitly executed with the **minishift addons apply** command. You can use the **apply** command for both enabled and disabled add-ons. To apply multiple add-ons with a single command, specify add-on names separated by space.

The following example shows how to explicitly apply the **anyuid** and the **admin-user** add-ons.

Example: Applying anyuid and admin-user add-ons

```
$ minishift addons apply anyuid admin-user
```

2.4.11. Removing Add-ons

Add-ons can be removed with the **minishift addons remove** command. It is the mirror command to **minishift addons apply** and similarly can be used regardless of whether the add-on is enabled or not. Provided the specified add-on is installed and has a **<addon_name>.addon.remove** file, **minishift addons remove** will execute the commands specified in this file.

To remove multiple add-ons with a single command, specify the add-on names separated by a space. The following example shows how to explicitly remove the **admin-user** add-on.

Example: Removing admin-user add-on

```
$ minishift addons remove admin-user
-- Removing addon 'admin-user':.
admin user deleted
```

2.4.12. Uninstalling Add-ons

Add-ons can be uninstalled with the **minishift addons uninstall** command. It is the mirror command to **minishift addons install** and can be used regardless of whether the add-on is enabled or not. Provided the specified add-on is installed, **minishift addons uninstall** will delete the corresponding add-on directory from ***$MINISHIFT_HOME/addons***.

The following example shows how to explicitly uninstall the **admin-user** add-on:

Example: Uninstalling admin-user add-on

```
$ minishift addons uninstall admin-user
Add-on 'admin-user' uninstalled
```

2.4.13. Writing Custom Add-ons

To write a custom add-on, you should create a directory and in it create at least one text file with the extension ***.addon***, for example ***admin-role.addon***.

This file needs to contain the **Name** and **Description** metadata fields, as well as the commands that you want to execute as a part of the add-on.

The following example shows the definition of an add-on that gives the developer user cluster-admin privileges.

Example: Add-on definition for admin-role

```
# Name: admin-role
# Description: Gives the developer user cluster-admin privileges

oc adm policy add-role-to-user cluster-admin developer
```

After you define the add-on, you can install it by running:

```
$ minishift addons install <ADDON_DIR_PATH>
```

NOTE

You can also write metadata with multiple lines.

Example: Add-on definition which contain multiline description

```
# Name: prometheus
# Description: This template creates a Prometheus instance preconfigured to gather
OpenShift and
# Kubernetes platform and node metrics and report them to admins. It is protected by
an
# OAuth proxy that only allows access for users who have view access to the
prometheus
# namespace. You may customize where the images (built from openshift/prometheus
# and openshift/oauth-proxy) are pulled from via template parameters.
# Url: https://prometheus.io/
```

You can also edit your add-on directly in the CDK add-on install directory ***$MINISHIFT_HOME/addons***. Be aware that if there is an error in the add-on, it will not show when you run any **addons** commands, and it will not be applied during the **minishift start** process.

To provide add-on removal instructions, you can create text file with the extension ***.addon.remove***, for example ***admin-user.addon.remove***. Similar to the ***.addon*** file, it needs the **Name** and **Description** metadata fields. If a ***.addon.remove*** file exists, it can be applied via the **remove** command.

2.5. HOST FOLDERS

2.5.1. Overview

Host folders are directories on the host which are shared between the host and the CDK VM. This allows for two way file synchronization between the host and the VM. The following sections discuss usage of the **minishift hostfolder** command.

2.5.2. The minishift hostfolder Command

CDK provides the **minishift hostfolder** command to list, add, mount, unmount and remove host folders. You can use the **hostfolder** command to mount multiple shared folders onto custom specified mount points.

NOTE

Currently CIFS and SSHFS based host folders are supported.

2.5.2.1. Prerequisites

2.5.2.1.1. SSHFS

SSHFS is the default technology for sharing host folders. It works without prerequisites.

2.5.2.1.2. CIFS

Host folders can be shared using CIFS. It is possible to use CIFS on Windows, macOS, and Linux.

On macOS, you can enable CIFS-based shares under **System Preferences > Sharing**. See How to connect with File Sharing on your Mac for detailed setup instructions.

On Linux, follow your distribution-specific instructions to install Samba. Refer to Samba File and Print Server in RHEL to learn how to configure the Samba implementation of CIFS in Red Hat Enterprise Linux.

2.5.2.2. Displaying Host Folders

The **minishift hostfolder list** command gives you an overview of the defined host folders, their names, mount points, remote paths and whether they are currently mounted.

An example output could look like:

```
$ minishift hostfolder list
Name    Type    Source              Mountpoint      Mounted
test    sshfs   /Users/user/test    /mnt/sda1/test  N
```

In this example, there is an SSHFS based host folder with the name **test** which mounts ***/Users/user/test*** onto ***/mnt/sda1/test*** in the CDK VM. The share is currently not mounted.

2.5.2.3. Adding Host Folders

The **minishift hostfolder add** command allows you to define a new host folder.

The exact syntax to use depends on the host folder type. Independent of the type you can choose between non-interactive and interactive configuration. The default is non-interactive. By specifying the **--interactive** flag you can select the interactive configuration mode.

The following sections give examples for configuring CIFS and SSHFS host folders.

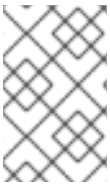
2.5.2.3.1. CIFS

Adding a CIFS based hostfolder

```
$ minishift hostfolder add -t cifs --source //192.168.99.1/MYSHARE --target /mnt/sda1/myshare --
options username=user,password=mysecret,domain=MYDOMAIN myshare
```

The above command will create a new host folder with the name of **myshare**. The source of the host folder is the UNC path ***//192.168.99.1/MYSHARE*** which is required. The target or mount point is ***/mnt/sda1/myshare*** within the CDK VM. Using the **--options** flag host folder type specific options can be specified using a list of comma separated key/value pairs. In the case of a CIFS host folder the required options are the user name for the CIFS share as well as the password. Via the **--options** flag the

domain of the share can be specified as well. Often this can be left out, but for example on Windows, when your account is linked to a Microsoft account, you must use the Microsoft account email address as user name as well as your machine name as displayed by **$env:COMPUTERNAME** as a domain.



NOTE

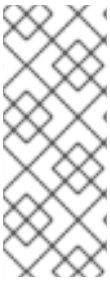
On Windows hosts, the **minishift hostfolder add** command also provides a **users-share** option. When this option is specified, no UNC path needs to be specified and ***C:\Users*** is assumed.

2.5.2.3.2. SSHFS

Adding an SSHFS based hostfolder

```
$ minishift hostfolder add -t sshfs --source /Users/user/myshare --target /mnt/sda1/myshare myshare
```

For the case of an SSHFS based host folder only the source and target of the host folder need to be specified.



NOTE

When adding host folders, one can give type-specific options to be used while mounting by using the **--options** flag. For example:

```
$ minishift hostfolder add -t sshfs --source ~/myshare/ --target /mnt/sda1/myshare --
options "-o compression=no" myshare
```

2.5.2.3.3. Instance-Specific Host Folders

By default, host folder definitions are persistent, similar to other persistent configuration options. This means that these host folder definitions will survive the deletion and subsequent re-creation of a CDK VM.

In some cases you might want to define a host folder just for a specific CDK instance. To do so, you can use the **--instance-only** flag of the **minishift hostfolder add** command. Host folder definitions that are created with the **--instance-only** flag will be removed together with any other instance-specific state during **minishift delete**.

2.5.2.4. Mounting Host Folders

After you have added host folders, you use the **minishift hostfolder mount** command to mount a host folder by its name:

```
$ minishift hostfolder mount myshare
```

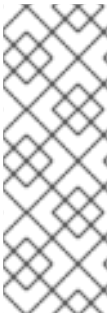
You can verify that the host folder is mounted by running:

```
$ minishift hostfolder list
Name      Mountpoint          Remote path              Mounted
myshare   /mnt/sda1/myshare   //192.168.99.1/MYSHARE   Y
```

Alternatively, you can list the actual content of the mounted host folder:

```
-
```

```
$ minishift ssh "ls -al /mnt/sda1/myshare"
```



NOTE

When mounting SSHFS based host folders an SFTP server process is started on port 2022 of the host. Make sure that your network and firewall settings allow this port to be opened. If you need to configure this port you can make use of CDK's persistent configuration using the key **hostfolders-sftp-port**, for example:

```
$ minishift config set hostfolders-sftp-port 2222
```

2.5.2.4.1. Auto-Mounting Host Folders

Host folders can also be mounted automatically each time you run **minishift start**. To set auto-mounting, you need to set the **hostfolders-automount** option in the CDK configuration file.

```
$ minishift config set hostfolders-automount true
```

After the **hostfolders-automount** option is set, CDK will attempt to mount all defined host folders during **minishift start**.

2.5.2.5. Unmounting Host Folders

You use the **minishift hostfolder umount** command to unmount a host folder.

```
$ minishift hostfolder umount myshare

$ minishift hostfolder list
Name       Mountpoint          Remote path              Mounted
myshare    /mnt/sda1/myshare   //192.168.99.1/MYSHARE   N
```

2.5.2.6. Deleting Host Folders

You use the **minishift hostfolder remove** command to remove a host folder definition.

```
$ minishift hostfolder list
Name       Mountpoint          Remote path              Mounted
myshare    /mnt/sda1/myshare   //192.168.1.82/MYSHARE   N

$ minishift hostfolder remove myshare

$ minishift hostfolder list
No host folders defined
```

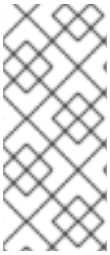
2.6. ASSIGN STATIC IP ADDRESS

2.6.1. Overview

Most hypervisors do not support extending the lease time when the IP is assigned using DHCP. This might lead to a new IP being assigned to the VM after a restart as it will conflict with the security certificates generated for the old IP. This will make CDK completely unusable until a new instance is set

up by running **minishift delete** followed by **minishift start**.

To prevent this, CDK includes the functionality to set a static IP address to the VM. This will prevent the IP address from changing between restarts. However, it will not work on all of the driver plug-ins at the moment due to the way the IP address is resolved.



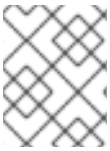
NOTE

- Assigning a static IP address to the CDK VM is only officially supported for Hyper-V.
- The CDK VM cannot be assigned a static IP address when using the KVM driver plug-in.

2.6.2. Assign IP Address to Hyper-V

Since the Internal Virtual Switch for Hyper-V does not provide a DHCP offer option, an IP address needs to be provided in a different way. Functionality is provided to assign an IP address on startup using the Data Exchange Service for Hyper-V.

To make this work, you need to create a Virtual Switch using NAT .



NOTE

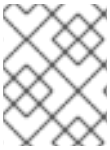
WinNAT is limited to one NAT network per host. For more details about capabilities and limitations, please see the WinNAT capabilities and limitations blog .

The following command will attempt to assign an IP address for use on the Internal Virtual Switch 'MyInternal':

```
PS> minishift.exe config set hyperv-virtual-switch "MyInternal"
PS> minishift.exe start `
  --network-ipaddress 192.168.1.10 `
  --network-gateway 192.168.1.1 `
  --network-nameserver 8.8.8.8
```

If you want to use the 'DockerNAT' network, the following commands are needed to setup the correct NAT networking and assign an IP in the range expected:

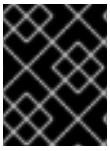
```
PS> New-NetNat -Name SharedNAT -InternalIPInterfaceAddressPrefix 10.0.75.1/24
PS> minishift.exe config set hyperv-virtual-switch "DockerNAT"
PS> minishift.exe start `
  --network-ipaddress 10.0.75.128 `
  --network-gateway 10.0.75.1 `
  --network-nameserver 8.8.8.8
```



NOTE

Be sure to specify a valid gateway and nameserver. Failing to do so will result in connectivity issues.

2.6.3. Set Fixed IP Address



IMPORTANT

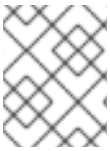
This feature is not supported on KVM as the driver plug-in relies on the DHCP offer to determine the IP address.

The following command will set the IP address that was assigned as fixed:

```
$ minishift ip --set-static
```

If you prefer to use dynamic assignment, you can use:

```
$ minishift ip --set-dhcp
```



NOTE

Use **minishift config set static-ip false** to stop automatically assigning a static IP address for supported hypervisors.

2.7. CDK DOCKER DAEMON

2.7.1. Overview

When running OpenShift in a single VM, you can reuse the Docker daemon managed by CDK for other Docker use-cases as well. By using the same Docker daemon as CDK, you can speed up your local development.

2.7.2. Console Configuration

In order to configure your console to reuse the CDK Docker daemon, follow these steps:

1. Make sure that you have the Docker client binary installed on your machine. For information about specific binary installations for your operating system, see the Docker installation site.
2. Start CDK with the **minishift start** command.
3. Run the **minishift docker-env** command to display the command you need to type into your shell in order to configure your Docker client. The command output will differ depending on OS and shell type.

   ```
   $ minishift docker-env
   export DOCKER_TLS_VERIFY="1"
   export DOCKER_HOST="tcp://192.168.99.101:2376"
   export DOCKER_CERT_PATH="/Users/user/.minishift/certs"
   export DOCKER_API_VERSION="1.24"
   # Run this command to configure your shell:
   # eval $(minishift docker-env)
   ```

4. Test the connection by running the following command:

   ```
   $ docker ps
   ```

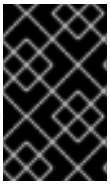
   If successful, the shell will print a list of running containers.

2.8. EXPERIMENTAL FEATURES

2.8.1. Overview

If you want to get early access to some upcoming features and experiment, you can set the environment variable **MINISHIFT_ENABLE_EXPERIMENTAL**, which makes additional feature flags available:

```
$ export MINISHIFT_ENABLE_EXPERIMENTAL=y
```



IMPORTANT

Experimental features are not officially supported, and might break or result in unexpected behavior. To share your feedback on these features, you are welcome to contact the Minishift community.

2.8.2. Enabling Experimental oc cluster up Flags

By default, CDK does not expose all **oc cluster up** flags in the CDK CLI.

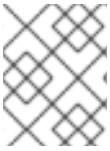
You can set the **MINISHIFT_ENABLE_EXPERIMENTAL** environment variable to enable the following options for the **minishift start** command:

extra-clusterup-flags

Enables passing flags directly to **oc cluster up** that are not directly exposed in the CDK CLI.

For example, the following command will provision the OpenShift **service catalog**:

```
$ MINISHIFT_ENABLE_EXPERIMENTAL=y minishift start --extra-clusterup-flags "--
enable='*,service-catalog'"
```



NOTE

The recommended way to enable service catalog is by using the **minishift openshift component add** command.

2.8.3. Local Proxy Server

To help in the situation when security certificates are used in an organization, but cannot be easily shared with the instance, CDK can run a local proxy server on the host, which the CDK instance can use to access external resources.

Enabling the proxy server is done using the following command:

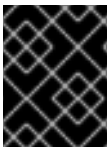
```
$ minishift config set local-proxy true
```

This will start a proxy server on the host which will be automatically assigned to the CDK instance.

When the corporate or upstream proxy is known, you can specify this with the following configuration option:

```
$ minishift config set local-proxy-upstream http(s)://[username:password@]host:port
```

When this option is used, all traffic will be re-encrypted with a CDK-specific certificate. For this reason, this proxy should only be used for development and use with CDK.

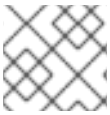


IMPORTANT

To allow external traffic to your local host you might have to enable port **3128/tcp** in your host firewall.

2.8.4. Local DNS Server

CDK provides a DNS server for offline usage or the possibility of overriding DNS records while testing. This will allow you to access the OpenShift routes without Internet.



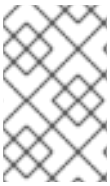
NOTE

The DNS server is specific to a profile.

Starting the DNS server can be done as follows:

```
$ minishift dns start
```

After starting the DNS server you need to configure your device settings to use this nameserver. The start command will show you a temporary option that can be used when you entered offline usage.



NOTE

In the current implementation you need to start the server and do required changes in the host settings manually. The DNS configuration is not permanent and might reset when the network state of the device changes.

Stopping the DNS server can be done as follows:

```
$ minishift dns stop
```

To get the status of the DNS server:

```
$ minishift dns status
```

2.8.4.1. Local DNS Setup for macOS

Recent versions of macOS do not send out DNS queries in offline mode, and the process for using a local DNS server from CDK is more involved than other operating systems.

2.8.4.1.1. Enable tap devices

Check for the presence of **tap** devices in */dev*:

```
$ ls /dev | grep tap
```

If no **tap** devices are present, install the **tuntap** package:

```
$ brew install tuntap
```

2.8.4.1.2. Use a tap device to create a network service

As root, open the */Library/Preferences/SystemConfiguration/preferences.plist* file and add the following XML under the **<key>NetworkServices</key>** element:

```
<key>D16F22CE-6DDE-4E63-837C-E16538EA5CCB</key> (1)
<dict>
  <key>DNS</key>
  <dict />
  <key>IPv4</key>
  <dict>
    <key>Addresses</key>
    <array>
      <string>10.10.90.1</string> (2)
    </array>
    <key>ConfigMethod</key>
    <string>Manual</string>
    <key>SubnetMasks</key>
    <array>
      <string>255.255.0.0</string>
    </array>
  </dict>
  <key>IPv6</key>
  <dict>
    <key>ConfigMethod</key>
    <string>Automatic</string>
  </dict>
  <key>Interface</key>
  <dict>
    <key>DeviceName</key>
    <string>tap0</string> (3)
    <key>Hardware</key>
    <string>Ethernet</string>
    <key>Type</key>
    <string>Ethernet</string>
    <key>UserDefinedName</key>
    <string>MiniTap</string> (4)
  </dict>
  <key>Proxies</key>
  <dict>
    <key>ExceptionsList</key>
    <array>
      <string>*.local</string>
      <string>169.254/16</string>
    </array>
    <key>FTPPassive</key>
    <integer>1</integer>
  </dict>
  <key>SMB</key>
  <dict />
```

```
    <key>UserDefinedName</key>
    <string>MiniTap</string>  5
</dict>
```

1 This is the UUID for the network service. Replace this value with the output of **uuidgen**.

2 The IP address for the network service.

3 The **/dev/tap** device to use.

4 5 Name for the network service (This will appear in the Network Preferences GUI).

2.8.4.1.3. Adding the Network Service to*ServiceOrder* array

In the */Library/Preferences/SystemConfiguration/preferences.plist* file, look for the **<key>ServiceOrder</key>** element. As root, append the UUID for our **MiniTap** network service to this array.

```
<key>ServiceOrder</key>
    <array>
        <string>06BFF3C7-13DA-420F-AE9C-B036401184D7</string>
 <string>58231F56-CA25-4D41-930F-46D83CA07BFE</string>
 <string>304203B0-AC87-459F-9761-C2799EEBB2E3</string>
 <string>8655D244-C6E7-4CC0-BF06-BB18F9C3BB85</string>
 <string>3C26FB9D-D918-4B79-9C7B-ADECD8EFE00F</string>
 <string>D16F22CE-6DDE-4E63-837C-E16538EA5CCB</string>  1
    </array>
```

1 The UUID for **MiniTap** network service.

2.8.4.1.4. Adding the Network Service to*Service* dictionary

In the */Library/Preferences/SystemConfiguration/preferences.plist* file, look for the **<key>Service</key>** element. As root, append the following XML to its dictionary:

```
<key>Service</key>
    <dict>
        <key>06BFF3C7-13DA-420F-AE9C-B036401184D7</key>
        <dict>
            <key>__LINK__</key>
            <string>/NetworkServices/06BFF3C7-13DA-420F-AE9C-B036401184D7</string>
        </dict>
        <key>304203B0-AC87-459F-9761-C2799EEBB2E3</key>
        <dict>
            <key>__LINK__</key>
            <string>/NetworkServices/304203B0-AC87-459F-9761-C2799EEBB2E3</string>
        </dict>
        <key>3C26FB9D-D918-4B79-9C7B-ADECD8EFE00F</key>
        <dict>
            <key>__LINK__</key>
            <string>/NetworkServices/3C26FB9D-D918-4B79-9C7B-ADECD8EFE00F</string>
        </dict>
        <key>58231F56-CA25-4D41-930F-46D83CA07BFE</key>
```

```
        <dict>
          <key>__LINK__</key>
          <string>/NetworkServices/58231F56-CA25-4D41-930F-46D83CA07BFE</string>
        </dict>
        <key>8655D244-C6E7-4CC0-BF06-BB18F9C3BB85</key>
        <dict>
          <key>__LINK__</key>
          <string>/NetworkServices/8655D244-C6E7-4CC0-BF06-BB18F9C3BB85</string>
        </dict>
        <key>D16F22CE-6DDE-4E63-837C-E16538EA5CCB</key>  (1)
        <dict>
          <key>__LINK__</key>
          <string>/NetworkServices/D16F22CE-6DDE-4E63-837C-E16538EA5CCB</string> (2)
        </dict>
      </dict>
```

1. The UUID of the **MiniTap** service.
2. Replace this UUID with the UUID of your **MiniTap** service.

Reboot macOS and you should see a **MiniTap** service in the Network Preferences GUI. This service will be disconnected. To turn it on, issue the following commands:

```
$ exec 4<>/dev/tap0  (1)
$ ifconfig tap0 10.10.90.1 255.255.0.0  (2) (3)
$ ifconfig tap0 up  (4)
```

1 2 4 Replace it with the **/dev/tap** device used by **MiniTap** Service.

3. IP address should be same as the one in the **MiniTap** Service definition.

2.8.4.1.5. Adding resolver config

Create the file **/etc/resolver/nip.io** with the following content:

```
nameserver <ip_address_of_the_minishfit_vm>
search_order 1
```

2.8.5. CDK System Tray

To help the users of CDK on macOS and Windows perform simple tasks like starting and stopping a profile from the GUI, the binaries for these platforms have been compiled to include an experimental system tray.

By default, the system tray is automatically started on running **minishift start**. To disable this behavior, use the following command:

```
$ minishift config set auto-start-tray false
```

To start the system tray:

```
$ minishift service start systemtray
```

-

2.8.6. Timezone Setup

If you want to set a different timezone from the default, use the following command:

```
$ minishift timezone --set <Valid_Timezone>
```

To see the available timezones, use the following command:

```
$ minishift timezone --list
```

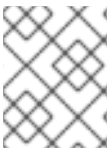
To check the current timezone of the CDK instance, use the following command:

```
$ minishift timezone
```

2.9. RUN AGAINST AN EXISTING MACHINE

2.9.1. Overview

CDK can be run against an existing remote machine using **vm-driver** as **generic**.



NOTE

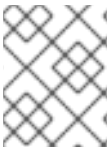
CentOS 7, Red Hat Enterprise Linux 7, and Fedora > 26 are the suggested Linux distributions for this feature.

2.9.2. Configuring an Existing Remote Machine

To use an existing machine with CDK, it needs to be configured as follows:

1. Establish password-less SSH from the host to the existing remote machine:

```
Host$ ssh-copy-id <user>@<remote_IP_address> # Ensure that the user has sudo access without a password or use root
Host$ ssh <user>@<remote_IP_address>         # Ensure that this login works without a password
```



NOTE

Skip the following steps if you are using CentOS 7, Red Hat Enterprise Linux 7 or Fedora (version > 26).

1. Configure the existing remote machine:

```
Remote_Machine$ yum install -y docker net-tools firewalld
Remote_Machine$ systemctl start docker
Remote_Machine$ systemctl enable docker
Remote_Machine$ systemctl start firewalld
Remote_Machine$ systemctl enable firewalld
```

2. Allow the **2376**, **8443**, and **80** TCP ports through the firewall on the remote machine to have communication from the host:

```
Remote_Machine$ firewall-cmd --permanent --add-port 2376/tcp --add-port 8443/tcp --add-port 80/tcp
```

3. Determine the Docker bridge network container subnet:

```
Remote_Machine$ docker network inspect -f "{{range .IPAM.Config }}{{ .Subnet }}{{end}}" bridge
```

This command displays a subnet, such as **172.17.0.0/16**.

4. Using the Docker bridge network container subnet, create a **minishift** zone for the firewall with the subnet as its source:

```
Remote_Machine$ firewall-cmd --permanent --new-zone minishift
Remote_Machine$ firewall-cmd --permanent --zone minishift --add-source <subnet>
```

5. Allow the **53** and **8053** UDP ports through the firewall on the remote machine to allow containers to access the OpenShift master API and DNS endpoints:

```
Remote_Machine$ firewall-cmd --permanent --zone minishift --add-port 53/udp --add-port 8053/udp
```

6. Reload the firewall on the remote machine:

```
Remote_Machine$ firewall-cmd --reload
```

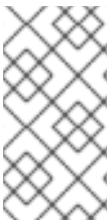
2.9.3. Running Against an Existing Remote Machine

Use the following command on the host to run CDK against a remote machine:

```
$ minishift start --vm-driver generic --remote-ipaddress <remote_IP_address> --remote-ssh-user <username> --remote-ssh-key <private_ssh_key>
$ minishift addon apply htpasswd-identity-provider --addon-env USER_PASSWORD=<NEW_PASSWORD>
```

NOTE

The value of the **--remote-ssh-key** flag must be the location of a private SSH key on the host machine.



NOTE

To change the default username and password, you must apply the **htpasswd-identity-provider** add-on with the desired username and password after running the **minishift start** command. By default, CDK uses the **Allow All** identity provider. The **Allow All** identity provider allows any non-empty username and password to log in.

2.10. CONVERTING AN EXISTING DOCKER COMPOSE PROJECT

2.10.1. Installing Kompose

Kompose is a tool for converting Docker Compose to container orchestrators such as OpenShift and Kubernetes. Kompose may be installed manually or via a package manager (Chocolatey on Microsoft Windows, Homebrew on macOS, or Yum on Red Hat Enterprise Linux).

2.10.1.1. Manually

1. Download the binary for your operating system from the Kompose GitHub Releases page.
2. Add the binary to a location in your **PATH**.

2.10.1.2. With Chocolatey

On Microsoft Windows, you can install Kompose via Chocolatey:

```
PS> choco.exe install kubernetes-kompose
```

2.10.1.3. With Homebrew

On macOS, you can install Kompose via Homebrew:

```
$ brew install kompose
```

2.10.1.4. On Red Hat Enterprise Linux

Kompose can be installed from the command line by enabling the Red Hat Developer Tools and Red Hat Software Collections repositories:

```
$ subscription-manager repos --enable rhel-7-server-devtools-rpms
$ subscription-manager repos --enable rhel-server-rhscl-7-rpms
$ yum install kompose -y
```

2.10.2. Using Kompose

To convert your Docker Compose project using Kompose, follow these steps:

1. Start CDK so you have an OpenShift cluster to communicate with.

```
$ minishift start
Starting local OpenShift cluster using 'kvm' hypervisor...
-- Checking OpenShift client ... OK
-- Checking Docker client ... OK
-- Checking Docker version ... OK
-- Checking for existing OpenShift container ... OK
...
```

2. Download an example Docker Compose file , or use your own.

```
wget https://raw.githubusercontent.com/kubernetes/kompose/master/examples/docker-compose.yaml
```

3. Convert your Docker Compose file to OpenShift. Run **kompose convert --provider=openshift** in the same directory as your ***docker-compose.yaml*** file.

```
$ kompose convert --provider=openshift
INFO OpenShift file "frontend-service.yaml" created
INFO OpenShift file "redis-master-service.yaml" created
INFO OpenShift file "redis-slave-service.yaml" created
INFO OpenShift file "frontend-deploymentconfig.yaml" created
INFO OpenShift file "frontend-imagestream.yaml" created
INFO OpenShift file "redis-master-deploymentconfig.yaml" created
INFO OpenShift file "redis-master-imagestream.yaml" created
INFO OpenShift file "redis-slave-deploymentconfig.yaml" created
INFO OpenShift file "redis-slave-imagestream.yaml" created
```

Alternatively, you can convert and deploy directly to OpenShift with **kompose up --provider=openshift**.

```
$ kompose up --provider=openshift
INFO We are going to create OpenShift DeploymentConfigs, Services and
PersistentVolumeClaims for your Dockerized application.
If you need different kind of resources, use the 'kompose convert' and 'oc create -f'
commands instead.

INFO Deploying application in "myproject" namespace
INFO Successfully created Service: frontend
INFO Successfully created Service: redis-master
INFO Successfully created Service: redis-slave
INFO Successfully created DeploymentConfig: frontend
INFO Successfully created ImageStream: frontend
INFO Successfully created DeploymentConfig: redis-master
INFO Successfully created ImageStream: redis-master
INFO Successfully created DeploymentConfig: redis-slave
INFO Successfully created ImageStream: redis-slave

Your application has been deployed to OpenShift. You can run 'oc get dc,svc,is,pvc' for
details.
```

4. Access the newly deployed application with CDK.
After deployment, you must create an OpenShift route in order to access the service.

Create a route for the **frontend** service using **oc**.

```
$ oc expose service/frontend
route "frontend" exposed
```

Access the **frontend** service with **minishift**.

```
$ minishift openshift service frontend --namespace=myproject --in-browser
Opening the service myproject/frontend in the default browser...
```

You can also access the GUI of OpenShift for an overview of the deployed containers.

```
$ minishift console
Opening the OpenShift Web console in the default browser...
```

CHAPTER 3. INTERACTING WITH OPENSHIFT USING CONTAINER DEVELOPMENT KIT

CDK creates a virtual machine and provisions a local, single-node OpenShift cluster in this VM. The following sections describe how CDK can assist you in interacting and configuring your local OpenShift cluster.

For details about managing the CDK VM, see the Basic Usage section.

3.1. USING THE OPENSHIFT CLIENT BINARY (OC)

3.1.1. Overview

The **minishift start** command creates an OpenShift cluster using the cluster up approach. For this purpose it copies the **oc** binary onto your host.

The **oc** binary is located in the *$MINISHIFT_HOME/cache/oc/v3.11.374* directory, assuming that you use the default OpenShift version for CDK. You can add this binary to your **PATH** using **minishift oc-env**, which displays the command you need to type into your shell.

The output of **minishift oc-env** differs depending on the operating system and the shell type.

```
$ minishift oc-env
export PATH="/home/user/.minishift/cache/oc/v3.11.374:$PATH"
# Run this command to configure your shell:
# eval $(minishift oc-env)
```

3.1.2. CDK CLI Profile

As part of the **minishift start** command, a CLI profile named **minishift** is also created. This profile, also known as a *context*, contains the configuration to communicate with your OpenShift cluster.

CDK activates this context automatically, but if you need to switch back to it after, for example, logging into another OpenShift instance, you can run:

```
$ oc config use-context minishift
```

For an introduction to **oc** usage, see the Get Started with the CLI topic in the OpenShift documentation.

3.1.3. Logging Into the Cluster

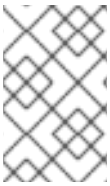
By default, **cluster up** uses AllowAllPasswordIdentityProvider to authenticate against the local cluster. This means any non-empty user name and password can be used to login to the local cluster.

The recommended user name and password is **developer** and **developer**, respectively. This is because they are already assigned to the default project **myproject** and also can impersonate the administrator. This allows you to run administrator commands using the **--as system:admin** parameter.

To login as administrator, use the system account:

```
$ oc login -u system:admin
```

In this case, client certificates are used. The certificates are stored in *~/.kube/config*. The **cluster up** command installs the appropriate certificates as a part of the bootstrap.



NOTE

If you run the command **oc login -u system -p admin**, you will log in but not as an administrator. Instead, you will be logged in as an unprivileged user with no particular rights.

To view the available login contexts, run:

```
$ oc config view
```

3.1.4. Accessing the Web Console

To access the OpenShift Web console, you can run this command in a shell after starting CDK to get the URL of the Web console:

```
$ minishift console --url
```

Alternatively, after starting CDK, you can use the following command to directly open the console in a browser:

```
$ minishift console
```

3.1.5. Accessing OpenShift Services

To access a service that is exposed with a route, run this command in a shell:

```
$ minishift openshift service [-n NAMESPACE] [--url] NAME
```

For more information refer also to Exposing Services.

3.1.6. Viewing OpenShift Logs

To access OpenShift logs, run the following command after starting CDK:

```
$ minishift logs
```

3.1.7. Updating OpenShift Configuration

While OpenShift is running, you can view and change the master or the node configuration of your cluster.

To view the OpenShift master configuration file ***master-config.yaml***, run the following command:

```
$ minishift openshift config view
```

To show the node or kubeapi-server configuration instead of the master configuration, specify the **target** flag.

For details about the **view** sub-command, see the **minishift openshift config view** synopsis.

NOTE

After you update the OpenShift configuration, OpenShift will transparently restart.

3.1.7.1. Example: Configuring cross-origin resource sharing

In this example, you configure cross-origin resource sharing (CORS) by updating the OpenShift master configuration to allow additional IP addresses to request resources.

By default, OpenShift only allows cross-origin resource requests from the IP address of the cluster or from localhost. This setting is stored in the **corsAllowedOrigins** property of the master configuration file ***master-config.yaml***.

To change the property value and allow cross-origin requests from all domains, run the following command:

```
$ minishift openshift config set --patch '{"corsAllowedOrigins": [".*"]}'
```

NOTE

If you get the error *The specified patch need to be a valid JSON.* when you run the above command, you need to modify the above command depending on your operating system, your shell environment and its interpolation behavior.

For example, if you use PowerShell on Windows 7 or 10, modify the above command to:

```
PS> minishift.exe openshift config set --patch '{\"corsAllowedOrigins\": [\".*\"]}'
```

If you use Command Prompt, use the following:

```
C:\> minishift.exe openshift config set --patch "{\"corsAllowedOrigins\": [\".*\"]}"
```

3.1.7.2. Example: Changing the OpenShift Routing Suffix

In this example, you change the OpenShift routing suffix in the master configuration.

If you use a static routing suffix, you can set the **routing-suffix** flag as part of the **minishift start** command. By default, CDK uses a dynamic routing prefix based on nip.io, in which the IP address of the VM is a part of the routing suffix, for example **192.168.99.103.nip.io**.

If you experience issues with **nip.io**, you can use xip.io, which is based on the same principles.

To set the routing suffix to **xip.io**, run the following command:

```
$ minishift openshift config set --patch '{"routingConfig": {"subdomain": "<IP-ADDRESS>.xip.io"}}'
```

Make sure to replace **IP-ADDRESS** in the above example with the IP address of your CDK VM. You can retrieve the IP address by running the **minishift ip** command.

3.1.8. Add Component to OpenShift Cluster

To add a component to a running OpenShift cluster, use the following:

```
$ minishift openshift component add <component-name>
```

3.1.8.1. Example: Add service-catalog component

In this example, the **service-catalog** component can be added as follows:

```
$ minishift openshift component add service-catalog
```

3.1.9. List Valid Components to Add to OpenShift Cluster

To list valid components which can be added to the running OpenShift cluster, use the following:

```
$ minishift openshift component list
```

3.2. EXPOSING SERVICES

3.2.1. Overview

There are several ways you can expose your service after you deploy it on OpenShift. The following sections describe the various methods and when to use them.

3.2.2. Routes

If you are deploying a Web application, the most common way to expose it is by a route. A route exposes the service as a host name. You can create a route using the Web console or the CLI:

```
$ oc expose svc/frontend --hostname=www.example.com
```

To see a full example of creating an application and exposing it with a route, see the Deploying a Sample Application section.

3.2.3. NodePort Services

In case the service you want to expose is not HTTP based, you can create a **NodePort** service. In this case, each OpenShift node will proxy that port into your service. To access this port on your CDK VM, you need to configure an Ingress IP using **oc expose** with the parameter **type=LoadBalancer**.

A common use-case for Ingress IP Self-Service is the ability to expose a database service. The following example shows the complete workflow to create and expose a MariaDB instance using CDK:

```
$ minishift start
$ eval $(minishift oc-env)
$ oc new-app -e MYSQL_ROOT_PASSWORD=admin
https://raw.githubusercontent.com/openshift/origin/master/examples/db-templates/mariadb-persistent-
template.json
$ oc rollout status -w dc/mariadb
$ oc expose dc mariadb --type=LoadBalancer --name=mariadb-ingress
$ oc export svc mariadb-ingress
 ....
```

```
ports:
   - nodePort: 30907
 ....
```

After the service is exposed, you can access MariaDB with the **mysql** CLI using the CDK VM IP and the exposed NodePort service.

```
$ mysql --user=root --password=admin --host=$(minishift ip) --port=30907
```

3.2.4. Port Forwarding

3.2.4.1. Using oc port-forward

If you want to quickly access a port of a specific pod of your cluster, you can also use the **oc port-forward** command of the OpenShift CLI.

```
$ oc port-forward POD [LOCAL_PORT:]REMOTE_PORT
```

3.2.4.2. Using VirtualBox tools

In case you're using the VirtualBox driver plug-in there is another method you can use for port forwarding. This method will allow for permanent port forwarding as well as for forwarding multiple ports at the same time. This method requires you to set up a **nodePort** as outlined above.

If we go by the example above, we can do the following:

```
$ VBoxManage controlvm minishift natpf1 "mariadb,tcp,,3306,,30907"
```

This will allow us to communicate with the mariadb service on **localhost:3306** which might be convenient if you don't want to change default ports. Additionally, an important advantage of this way of forwarding ports is that we can talk to a service as opposed to just a single pod.

3.3. ACCESSING THE OPENSHIFT DOCKER REGISTRY

3.3.1. Overview

OpenShift provides an integrated Docker registry which can be used for development as well. Images present in the registry can directly be used for applications, speeding up the local development workflow.

3.3.2. Logging Into the Registry

1. Start CDK and add the **oc** binary to the **PATH**. For a detailed example, see the CDK Quickstart section.
2. Make sure your shell is configured to reuse the CDK Docker daemon.
3. Log into the OpenShift Docker registry.

   ```
   $ docker login -u developer -p $(oc whoami -t) $(minishift openshift registry)
   ```

3.3.3. Deploying Applications

The following example shows how to deploy an OpenShift application directly from a locally-built Docker image. This example uses the OpenShift project **myproject**. This project is automatically created by **minishift start**.

1. Make sure your shell is configured to reuse the CDK Docker daemon.
2. Build the Docker image as usual.
3. Tag the image against the OpenShift registry:

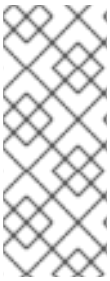
```
$ docker tag my-app $(minishift openshift registry)/myproject/my-app
```

4. Push the image to the registry to create an image stream with the same name as the application:

```
$ docker push $(minishift openshift registry)/myproject/my-app
```

5. Create an application from the image stream and expose the service:

```
$ oc new-app --image-stream=my-app --name=my-app
$ oc expose service my-app
```



NOTE

If you want to deploy an application using **oc run --image [...]** then exposed internal registry route doesn't work. You should use internal registry IP along with your project and app to deploy, as following:

```
$ oc run myapp --image 172.30.1.1:5000/myproject/myapp
```

CHAPTER 4. TROUBLESHOOTING CDK

This section contains solutions to common problems that you might encounter while setting up and using CDK.

4.1. TROUBLESHOOTING GETTING STARTED

4.1.1. Overview

This section contains solutions to common problems that you might encounter while installing and configuring CDK.

4.1.2. CDK startup check failed

While CDK starts, it runs several startup checks to make sure that the CDK VM and the OpenShift Cluster are able to start without any issues. If any configuration is incorrect or missing, the startup checks fail and CDK does not start.

- You can skip the startup checks by executing the following command:

```
$ minishift config set skip-startup-checks true
```

The following sections describe the different startup checks.

4.1.2.1. Driver plug-in configuration

One of the startup checks verifies that the relevant driver plug-in is configured correctly. If this startup check fails, review the Setting Up the Virtualization Environment topic and configure the appropriate driver.

If you want to force CDK to start despite a failing driver plug-in check, you can instruct CDK to treat these errors as warnings:

- For KVM/libvirt on Linux, run the following command:

```
$ minishift config set warn-check-kvm-driver true
```

- For hyperkit on macOS, run the following command:

```
$ minishift config set warn-check-hyperkit-driver true
```

- For Hyper-V on Windows, run the following command:

```
C:\> minishift.exe config set warn-check-hyperv-driver true
```

4.1.2.2. Persistent storage volume configuration and usage

CDK checks whether the persistent storage volume is mounted and that enough disk space is available. If the persistent storage volume, for example, uses more than 95% of the available disk space, CDK will not start.

If you want to recover the data, you can skip this test and start CDK to access the persistent volume:

```
$ minishift config set skip-check-storage-usage true
```

4.1.2.3. External network connectivity

After the CDK VM starts, it runs several network checks to verify whether external connectivity is possible from within the CDK VM.

By default, network checks are configured to treat any errors as warnings, because of the diversity of the development environments. You can configure the network checks to optimize them for your environment.

For example, one of the network checks pings an external host. You can change the host by running the following command:

```
$ minishift config set check-network-ping-host <host-IP-address>
```

Replace **<host-IP-address>** with the address of your internal DNS server, proxy host, or an external host that you can reach from your machine.

Because proxy connectivity might be problematic, you can run a check that tries to retrieve an external URL. You can configure the URL by running:

```
$ minishift config set check-network-http-host <URL>
```

4.1.3. OpenShift web console does not work with older versions of Safari

minishift console does not work on older versions of Safari web browser such as version 10.1.2 (12603.3.8). Attempting to access the web console results in the following error:

```
Error unable to load details about the server
```

Retry after updating it to the latest version or use Firefox or Chrome browser for this. Version 11.0.3 (13604.5.6) has been tested and works with OpenShift web console. You can use **minishift console --url** to get the web console URL.

4.2. TROUBLESHOOTING DRIVER PLUG-INS

4.2.1. Overview

This section contains solutions to common problems that you might encounter while configuring the driver plug-ins for CDK.

4.2.2. KVM/libvirt

4.2.2.1. Undefining virsh snapshots fail

If you use **virsh** on KVM/libvirt to create snapshots in your development workflow then use **minishift delete** to delete the snapshots along with the VM, you might encounter the following error:

```
$ minishift delete
Deleting the Minishift VM...
```

```
Error deleting the VM:  [Code-55] [Domain-10] Requested operation is not valid: cannot delete
inactive domain with 4 snapshots
```

Cause: The snapshots are stored in ***~/.minishift/machines***, but the definitions are stored in ***/var/lib/libvirt/qemu/snapshot/minishift***.

Workaround: To delete the snapshots, you need to perform the following steps as root.

1. Delete the definitions:

```
# virsh snapshot-delete --metadata minishift <snapshot-name>
```

2. Undefine the CDK domain:

```
# virsh undefine minishift
```

You can now run **minishift delete** to delete the VM and restart CDK.

NOTE

If these steps do not resolve the issue, you can also use the following command to delete the snapshots:

```
$ rm -rf ~/.minishift/machines
```

It is recommended to avoid using metadata when you create snapshots. To ensure this, you can specify the **--no-metadata** flag. For example:

```
# virsh snapshot-create-as --domain vm1 overlay1 --diskspec vda,file=/export/overlay1.qcow2 --disk-
only --atomic --no-metadata
```

4.2.2.2. Error creating new host: dial tcp: missing address

The problem is likely that the **libvirtd** service is not running. You can check this with the following command:

```
$ systemctl status libvirtd
```

If **libvirtd** is not running, start it and enable it to start on boot:

```
$ systemctl start libvirtd
$ systemctl enable libvirtd
```

4.2.2.3. Failed to connect socket to '/var/run/libvirt/virtlogd-sock'

The problem is likely that the **virtlogd** service is not running. You can check this with the following command:

```
$ systemctl status virtlogd
```

If **virtlogd** is not running, start it and enable it to start on boot:

```
$ systemctl start virtlogd
$ systemctl enable virtlogd
```

4.2.2.4. Domain 'minishift' already exists...

If you try **minishift start** and this error appears, ensure that you use **minishift delete** to delete the VMs that you created earlier. However, if this fails and you want to completely clean up CDK and start fresh, do the following:

1. As root, check if any existing CDK VMs are running:

```
# virsh list --all
```

2. If any CDK VM is running, stop it:

```
# virsh destroy minishift
```

3. Delete the VM:

```
# virsh undefine minishift
```

4. As your regular user, delete the ***~/.minishift/machines*** directory:

```
$ rm -rf ~/.minishift/machines
```

In case all of this fails, you might want to uninstall CDK and do a fresh install of CDK.

4.2.3. VirtualBox

4.2.3.1. Error machine does not exist

If you use Windows, ensure that you set the **--vm-driver virtualbox** flag in the **minishift start** command. Alternatively, the problem might be an outdated version of VirtualBox.

To avoid this issue, it is recommended to use VirtualBox 5.1.12 or later.

4.2.4. Hyper-V

4.2.4.1. Hyper-V commands must be run as an Administrator

If you run CDK with Hyper-V on Windows as a normal user or as a user with Administrator privileges, you might encounter the following error:

```
Error starting the VM: Error creating the VM. Error with pre-create check: "Hyper-V commands must be run as an Administrator".
```

Workaround: You can either add yourself to the Hyper-V Administrators group, which is recommended, or run the shell in an elevated mode.

If you are using PowerShell, you can add yourself to the Hyper-V Administrators group as follows:

1. As an administrator, run the following command:

```
PS> ([adsi]"WinNT://./Hyper-V
Administrators,group").Add("WinNT://$env:UserDomain/$env:Username,user")
```

2. Log out and log back in for the change to take effect.

You can also use the GUI to add yourself to the Hyper-V Administrators group as follows:

1. Click the **Start** button and select **Computer Management**.
2. In the **Computer Management** window, select **Local Users And Groups**, then double-click **Groups**.
3. Double-click the **Hyper-V Administrators** group, the **Hyper-V Administrators Properties** dialog box is displayed.
4. Add your account to the Hyper-V Administrators group, log off, then log in for the change to take effect.

Now you can run the Hyper-V commands as a normal user.

For more options for Hyper-V see creating Hyper-V administrators local group .

4.2.4.2. CDK running with Hyper-V fails when connected to OpenVPN

If you try to use CDK with Hyper-V using an external virtual switch while you are connected to a VPN such as OpenVPN, CDK might fail to provision the VM.

Cause: Hyper-V networking might not route the network traffic in both directions properly when connected to a VPN.

Workaround: Disconnect from the VPN and try again after stopping the VM from the Hyper-V manager.

4.3. TROUBLESHOOTING MISCELLANEOUS

4.3.1. Overview

This section contains solutions to common problems that you might encounter while using various components of CDK.

4.3.2. The root filesystem of the CDK VM exceeds overlay size

Installing additional packages or copying large files to the root filesystem of the CDK VM might exceed the allocated overlay size and lock the CDK VM.

Cause: The CDK VM root filesystem contains core packages that are configured to optimize running the CDK VM and containers. The available storage on the root filesystem is determined by the overlay size, which is smaller than the total available storage.

Workaround: Avoid installing packages or storing large files in the root filesystem of the CDK VM. Instead, you can create a sub-directory in the ***/mnt/sda1/*** persistent storage volume or define and mount host folders that can share storage space between the host and the CDK VM.

If you want to perform development tasks inside the CDK VM, it is recommended that you use containers, which are stored in persistent storage volumes, and reuse the CDK Docker daemon.

4.3.3. Special characters cause passwords to fail

Depending on your operating system and shell environment, certain special characters can trigger variable interpolation and therefore cause passwords to fail.

Workaround: When creating and entering passwords, wrap the string with single quotes in the following format: **'<password>'**

4.3.4. Cannot access web console with Microsoft Edge

Attempting to access the OpenShift web console using Microsoft Edge can result in the following error:

```
Site not reachable.
```

Alternatively, the OpenShift web console may be rendered as a blank page.

Cause: This is an inconsistent error caused by Microsoft Edge.

Workaround: Use an alternative web browser to access the OpenShift web console. Mozilla Firefox and Google Chrome both work as expected.

4.3.5. X.509 certificate is valid for 10.0.2.15, 127.0.0.1, 172.17.0.1, 172.30.0.1, 192.168.99.100, not 192.168.99.101

Starting a stopped CDK VM can produce the following X.509 certificate error:

```
$ minishift start
-- Checking if requested hypervisor 'kvm' is supported on this platform ... OK
-- Checking the ISO URL ... OK
-- Starting local OpenShift cluster using 'kvm' hypervisor ...
-- Starting Minishift VM ..... OK
[...]
FAIL
  Error: cannot access master readiness URL https://192.168.99.101:8443/healthz/ready
  Details:
    No log available from "origin" container

  Caused By:
    Error: Get https://192.168.99.101:8443/healthz/ready: x509: certificate is valid for 10.0.2.15,
127.0.0.1, 172.17.0.1, 172.30.0.1, 192.168.99.100, not 192.168.99.101
Error during 'cluster up' execution: Error starting the cluster.
```

The reason for the above error is that OpenShift cluster certificates contain the IP of the CDK VM. The certificates are generated only when the CDK VM is freshly started. After restart, the CDK VM might be assigned a new IP address. If this happens, the certificate becomes invalid.

Workaround: Delete the existing CDK VM and start again.

```
$ minishift delete --force
$ minishift start
```

4.3.6. Removing the subscription password from an OS-native keychain

4.3.6.1. Windows

On Windows, the password is stored using **Credential Manager**. Run the following command in Command Prompt to delete the stored password:

```
C:\> cmdkey /delete:minishift:<username>
```

4.3.6.2. Red Hat Enterprise Linux

On Red Hat Enterprise Linux, the password is stored using the **D-Bus Secret Service API**provided by **libsecret**. Run the following command to delete the stored password:

```
$ secret-tool clear service minishift username <username>
```

4.3.6.3. macOS

On macOS, the password is stored using **Keychain Access**. Run the following command to delete the stored password:

```
$ security delete-generic-password -s minishift
```