miniDC/OS

Release 2020.08.14.0

Adam Dangoor

Aug 14, 2020

CONTENTS

miniDC/OS can be used for spinning up and managing DC/OS clusters in test environments.

It is built on top of the DC/OS E2E Python library.

CHAPTER

ONE

DOCKER

The *minidcos docker* CLI allows you to create, manage and destroy open source DC/OS and DC/OS Enterprise clusters on Docker nodes.

A typical CLI workflow for open source DC/OS may look like the following. Install the CLI (see *Installation*), then create and manage a cluster:

```
# Fix issues shown by minidcos docker doctor
$ minidcos docker doctor
$ minidcos docker download-installer
$ minidcos docker create ./dcos_generate_config.sh --agents 0
default
$ minidcos docker wait
$ minidcos docker run --test-env --sync-dir /path/to/dcos/checkout pytest -k test_tls
...
# Get onto a node
$ minidcos docker run bash
[master-0]# exit
$ minidcos docker destroy
```

Each of these and more are described in detail below.

* *wait*
* *web*

1.1 Requirements

1.1.1 Docker 17.06+

Docker version 17.06 or later must be installed.

Plenty of memory must be given to Docker. On Docker for Mac, this can be done from Docker > Preferences > Advanced. This backend has been tested with a four node cluster with 9 GB memory given to Docker.

1.1.2 IP Routing Set Up for Docker

On macOS, hosts cannot connect to containers IP addresses by default. This is required, for example, to access the web UI, to SSH to nodes and to use the DC/OS CLI.

Once the CLI is installed, run *setup-mac-network* to set up IP routing.

Without this, it is still possible to use some features. Specify the `--transport docker-exec` and `--skip-http-checks` options where available.

1.1.3 `ssh`

The `ssh` command must be available to use the `ssh` transport options.

1.1.4 Operating System

This tool has been tested on macOS with Docker for Mac and on Linux.

It has also been tested on Windows on Vagrant.

Windows

The only supported way to use the Docker backend on Windows is using Vagrant and VirtualBox.

- Ensure Virtualization and VT-X support is enabled in your PC's BIOS. Disable Hyper-V virtualization. See https://www.howtogeek.com/213795/how-to-enable-intel-vt-x-in-your-computers-bios-or-uefi-firmware/.
- Install VirtualBox and VirtualBox Extension Pack.
- Install Vagrant.
- Install the Vagrant plugin for persistent disks:

```
vagrant plugin install vagrant-persistent-storage
```

- Optionally install the Vagrant plugins to cache package downloads and keep guest additions updates:

```
vagrant plugin install vagrant-cachier
vagrant plugin install vagrant-vbguest
```

- Start Powershell and download the miniDC/OS `Vagrantfile` to a directory containing a DC/OS installer file:

```
((New-Object System.Net.WebClient).DownloadString('https://raw.githubusercontent.com/
↪|github-owner|/|github-repository|/master/vagrant/Vagrantfile')) | Set-Content -
↪LiteralPath Vagrantfile
```

- By default, the `Vagrantfile` installs miniDC/OS from the most recent release at the time it is downloaded. To use a different release, or any Git reference, set the environment variable `DCOS_E2E_REF`:

```
$env:DCOS_E2E_REF = "master"
```

- Start the virtual machine and login:

```
vagrant up
vagrant ssh
```

You can now run *Docker* commands.

To connect to the cluster nodes from the Windows host (e.g. to use the DC/OS web interface), in PowerShell Run as Administrator, and add the Virtual Machine as a gateway:

```
route add 172.17.0.0 MASK 255.255.0.0 192.168.18.2
```

To shutdown, logout of the virtual machine shell, and destroy the virtual machine and disk:

```
vagrant destroy
```

The route will be removed on reboot. You can manually remove the route in PowerShell Run as Administrator using:

```
route delete 172.17.0.0
```

1.1.5 `doctor` command

minidcos docker comes with the *doctor* command. Run this command to check your system for common causes of problems.

1.2 Creating a Cluster

To create a cluster you first need to download a DC/OS installer.

This can be done via the releases page or with the *download-installer* command.

DC/OS Enterprise is also supported. Ask your sales representative for installers.

Creating a cluster is possible with the *create* command. This command allows you to customize the cluster in many ways.

The command returns when the DC/OS installation process has started. To wait until DC/OS has finished installing, use the *wait* command.

To use this cluster, it is useful to find details using the *inspect* command.

1.2.1 Using a custom Docker network

By default DC/OS clusters are launched on the `docker0` network.

To launch a DC/OS cluster on a custom Docker network the network must first be created using the standard Docker CLI. During *create* the command line option `--network` then takes the name of the Docker network as a parameter.

DC/OS nodes utilize an environment-specific `ip-detect` script to detect their current private IP address. The default `ip-detect` script used by *minidcos docker* does only account for the `docker0` network case. Therefore, in order for DC/OS to operate on a custom network a custom `ip-detect` script needs to be provided and put into the `genconf` directory before installing DC/OS.

The following IP detect script works for any custom Docker network:

```
#!/bin/bash -e
if [ -f /sbin/ip ]; then
   IP_CMD=/sbin/ip
else
   IP_CMD=/bin/ip
fi
$IP_CMD -4 -o addr show dev eth1 | awk 'split($4,a,"/");print a[1]'
```

The *create* command supports overwriting the default `genconf` directory with the contents of the directory supplied through the command line option `--genconf-dir`.

```
# Create ip-detect as mentioned above
$ docker network create custom-bridge
$ mkdir custom-genconf
$ mv ip-detect custom-genconf/ip-detect
$ minidcos docker create /path/to/dcos_generate_config.sh      --network custom-bridge      --genconf
```

The custom Docker network is not cleaned up by the *minidcos docker* CLI.

1.2.2 DC/OS Enterprise

There are multiple DC/OS Enterprise-only features available in *create*.

The only extra requirement is to give a valid license key, for DC/OS 1.11+. See *create* for details on how to provide a license key.

Ask your sales representative for DC/OS Enterprise installers.

For, example, run the following to create a DC/OS Enterprise cluster in strict mode:

```
$ minidcos docker create /path/to/dcos_generate_config.ee.sh       --license-key /path/to/license.txt
```

The command returns when the DC/OS installation process has started. To wait until DC/OS has finished installing, use the *wait* command.

See *create* for details on this command and its options.

1.3 Cluster IDs

Clusters have unique IDs. Multiple commands take `--cluster-id` options. Specify a cluster ID in *create*, and then use it in other commands. Any command which takes a `--cluster-id` option defaults to using "default" if no cluster ID is given.

1.4 Running commands on Cluster Nodes

It is possible to run commands on a cluster node in multiple ways. These include using *run*, `docker exec` and `ssh`.

1.4.1 Running commands on a cluster node using run

It is possible to run the following to run a command on an arbitrary master node.

```
$ minidcos docker run systemctl list-units
```

See *run* for more information on this command. In particular see the `--node` option to choose a particular node to run the command on.

1.4.2 Running commands on a cluster node using `docker exec`

Each cluster node is a Docker container. This means that you can use tools such as `docker exec` to run commands on nodes. To do this, first choose the container ID of a node. Use *inspect* to see all node container IDs.

1.4.3 Running commands on a cluster node using `ssh`

One SSH key allows access to all nodes in the cluster. See this SSH key's path and the IP addresses of nodes using *inspect*. The available SSH user is `root`.

1.5 Getting on to a Cluster Node

Sometimes it is useful to get onto a cluster node. To do this, you can use any of the ways of *Running commands on Cluster Nodes*.

For example, to use *run* to run `bash` to get on to an arbitrary master node:

```
$ minidcos docker run example bash
```

or, similarly, to use `docker exec` to get on to a specific node:

```
$ eval $(minidcos docker inspect --env)
$ docker exec -it $MASTER_0 bash
```

See *Running commands on Cluster Nodes* for details on how to choose particular nodes.

1.6 Destroying Clusters

There are two commands which can be used to destroy clusters. These are *destroy* and *destroy-list*.

Either destroy a cluster with *destroy*:

```
$ minidcos docker destroy
default
$ minidcos docker destroy --cluster-id pr_4033_strict
pr_4033_strict
```

or use *destroy-list* to destroy multiple clusters:

```
$ minidcos docker destroy-list pr_4033_strict pr_4019_permissive
pr_4033_strict
pr_4019_permissive
```

To destroy all clusters, run the following command:

```
$ minidcos docker destroy-list $(minidcos docker list)
pr_4033_strict
pr_4019_permissive
```

1.7 Running Integration Tests

The *run* command is useful for running integration tests.

To run integration tests which are developed in the a DC/OS checkout at `/path/to/dcos`, you can use the following workflow:

```
$ minidcos docker create ./dcos_generate_config.sh
$ minidcos docker wait
$ minidcos docker run --test-env --sync-dir /path/to/dcos/checkout pytest -k test_tls.py
```

There are multiple options and shortcuts for using these commands. See *run* for more information on this command.

1.8 Viewing the Web UI

To view the web UI of your cluster, use the *web* command. To see the web UI URL of your cluster, use the *inspect* command.

Before viewing the UI, you may first need to configure your browser to trust your DC/OS CA, or choose to override the browser protection.

1.8.1 macOS

On macOS, by default, viewing the web UI requires IP routing to be set up. Use *setup-mac-network* to set up IP routing.

The web UI is served by master nodes on port `80`. To view the web UI on macOS without setting up IP routing, use the `--one-master-host-port-map` option on the *create* command to forward port `80` to your host. For example:

```
$ minidcos docker create ./dcos_generate_config.sh --one-master-host-port-map 70:80
$ minidcos docker wait
$ open localhost:70
```

1.9 Using a Custom CA Certificate

On DC/OS Enterprise clusters, it is possible to use a custom CA certificate. See the Custom CA certificate documentation for details. It is possible to use *create* to create a cluster with a custom CA certificate.

1. Create or obtain the necessary files:

   `dcos-ca-certificate.crt`, `dcos-ca-certificate-key.key`, and `dcos-ca-certificate-chain.crt`.

2. Put the above-mentioned files into a directory, e.g. `/path/to/genconf/`.

3. Create a file containing the "extra" configuration.

   create takes an `--extra-config` option. This adds the contents of the specified YAML file to a minimal DC/OS configuration.

   Create a file with the following contents:

```
ca_certificate_path: genconf/dcos-ca-certificate.crt
ca_certificate_key_path: genconf/dcos-ca-certificate-key.key
ca_certificate_chain_path: genconf/dcos-ca-certificate-chain.crt
```

4. Create a cluster.

```
$ minidcos docker create      /path/to/dcos_generate_config.ee.sh      --variant enterprise
```

5. Verify that everything has worked.

   See Verify installation for steps to verify that the DC/OS Enterprise cluster was installed properly with the custom CA certificate.

1.10 Using a Loopback Sidecar

The *create-loopback-sidecar* command can be used to create a loopback sidecar. This will provide all containers with a unformatted block device, mounted as a loopback device. All containers have access to this loopback device. Therefore, care must be taken that only a single container has write-access to it.

```
$ minidcos docker create-loopback-sidecar sidecar1
/dev/loop0
$ minidcos docker create /tmp/dcos_generate_config.sh
$ minidcos docker wait
$ minidcos docker destroy-loopback-sidecar sidecar1
```

Loopback sidecars can be listed with *list-loopback-sidecars*. Loopback sidecars can be destroyed with *destroy-loopback-sidecar*.

1.11 Limitations

Docker does not represent a real DC/OS environment with complete accuracy. This section describes the currently known differences between clusters created with *minidcos docker* and a real DC/OS environment.

1.11.1 SELinux

Tests inherit the host's environment. Any tests that rely on SELinux being available require it be available on the host.

1.11.2 Storage

Docker does not support storage features expected in a real DC/OS environment.

1.11.3 Reboot

DC/OS nodes cannot be rebooted. The cluster cannot survive a system reboot.

1.11.4 Resources UI

The DC/OS web UI Resources tab shows resources available to each DC/OS node. On a *minidcos docker* cluster, resources are not specific to one node. That means, if the cluster has 16 GB memory available in total, the web UI will show that each node has 16 GB memory available.

1.11.5 Marathon-LB

Network configuration options vary by kernel version. If you see the following when installing Marathon-LB, change the Marathon-LB `sysctl-params` configuration value:

```
sysctl: cannot stat /proc/sys/net/ipv4/tcp_tw_reuse: No such file or directory
sysctl: cannot stat /proc/sys/net/ipv4/tcp_max_syn_backlog: No such file or directory
sysctl: cannot stat /proc/sys/net/ipv4/tcp_max_tw_buckets: No such file or directory
sysctl: cannot stat /proc/sys/net/ipv4/tcp_max_orphans: No such file or directory
```

Remove the relevant `sysctl-params` values. This may leave:

```
net.ipv4.tcp_fin_timeout=30 net.core.somaxconn=10000
```

1.12 CLI Reference

1.12.1 minidcos docker

Manage DC/OS clusters on Docker.

```
minidcos docker [OPTIONS] COMMAND [ARGS]...
```

clean

Remove containers, volumes and networks created by this tool.

```
minidcos docker clean [OPTIONS]
```

Options

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

create

Create a DC/OS cluster.

DC/OS Enterprise

DC/OS Enterprise clusters require different configuration variables to DC/OS OSS. For example, enterprise clusters require the following configuration parameters:

`superuser_username`, `superuser_password_hash`, `fault_domain_enabled`, `license_key_contents`

These can all be set in `--extra-config`. However, some defaults are provided for all but the license key.

The default superuser username is `bootstrapuser`. The default superuser password is `deleteme`. The default `fault_domain_enabled` is `false`.

`license_key_contents` must be set for DC/OS Enterprise 1.11 and above. This is set to one of the following, in order:

* The `license_key_contents` set in `--extra-config`. * The contents of the path given with `--license-key`. * The contents of the path set in the `DCOS_LICENSE_KEY_PATH` environment variable.

If none of these are set, `license_key_contents` is not given.

```
minidcos docker create [OPTIONS] INSTALLER
```

Options

--docker-version <docker_version>
The Docker version to install on the nodes. This can be provided by setting the *MINIDCOS_NODE_DOCKER_VERSION* environment variable. [default: 18.06.3-ce]

Options 1.11.2|1.13.1|17.12.1-ce|18.06.3-ce

--linux-distribution <linux_distribution>
The Linux distribution to use on the nodes. [default: centos-7]

Options centos-7|centos-8|coreos|flatcar|ubuntu-16.04

--docker-storage-driver <docker_storage_driver>
The storage driver to use for Docker in Docker. By default this uses the host's driver. [default: auto]

Options aufs|auto|overlay|overlay2

--mount-sys-fs-cgroup, --no-mount-sys-fs-cgroup
Mounting `/sys/fs/cgroup` from the host is required to run applications which require `cgroup` isolation. Choose to not mount `/sys/fs/cgroup` if it is not available on the host. [default: True]

--masters <masters>
The number of master nodes. [default: 1]

--agents <agents>
The number of agent nodes. [default: 1]

--public-agents <public_agents>
The number of public agent nodes. [default: 1]

--extra-config <extra_config>
The path to a file including DC/OS configuration YAML. The contents of this file will be added to add to a default configuration.

--security-mode <security_mode>
The security mode to use for a DC/OS Enterprise cluster. This overrides any security mode set in `--extra-config`.

Options disabled|permissive|strict

-c, --cluster-id <cluster_id>
A unique identifier for the cluster. Use the value "default" to use this cluster for other commands without specifying –cluster-id.

--license-key <license_key>
This is ignored if using open source DC/OS. If using DC/OS Enterprise, this defaults to the value of the *DCOS_LICENSE_KEY_PATH* environment variable.

--genconf-dir <files_to_copy_to_genconf_dir>
Path to a directory that contains additional files for the DC/OS installer. All files from this directory will be copied to the "genconf" directory before running the DC/OS installer.

--copy-to-master <copy_to_master>
Files to copy to master nodes before installing DC/OS. This option can be given multiple times. Each option should be in the format /absolute/local/path:/remote/path.

--custom-volume <custom_volume>
Bind mount a volume on all cluster node containers. See https://docs.docker.com/engine/reference/run/#volume-shared-filesystems for the syntax to use.

--custom-master-volume <custom_master_volume>
Bind mount a volume on all cluster master node containers. See https://docs.docker.com/engine/reference/run/#volume-shared-filesystems for the syntax to use.

--custom-agent-volume <custom_agent_volume>
Bind mount a volume on all cluster agent node containers. See https://docs.docker.com/engine/reference/run/#volume-shared-filesystems for the syntax to use.

--custom-public-agent-volume <custom_public_agent_volume>
Bind mount a volume on all cluster public agent node containers. See https://docs.docker.com/engine/reference/run/#volume-shared-filesystems for the syntax to use.

--workspace-dir <workspace_dir>
Creating a cluster can use approximately 2 GB of temporary storage. Set this option to use a custom "workspace" for this temporary storage. See https://docs.python.org/3/library/tempfile.html#tempfile.gettempdir for details on the temporary directory location if this option is not set.

--variant <variant>
Choose the DC/OS variant. If the variant does not match the variant of the given installer, an error will occur. Using "auto" finds the variant from the installer. Finding the variant from the installer takes some time and so using another option is a performance optimization.

Options auto|oss|enterprise

--wait-for-dcos
Wait for DC/OS after creating the cluster. This is equivalent to using "minidcos docker wait" after this command. "minidcos docker wait" has various options available and so may be more appropriate for your use case. If the chosen transport is "docker-exec", this will skip HTTP checks and so the cluster may not be fully ready.

--network <network>
The Docker network containers will be connected to.It may not be possible to SSH to containers on a custom network on macOS.

--transport <transport>
The communication transport to use. On macOS the SSH transport requires IP routing to be set up. See "minidcos docker setup-mac-network". It also requires the "ssh" command to be available. This can be provided by setting the *MINIDCOS_DOCKER_TRANSPORT* environment variable. When using a TTY, different transports may use different line endings. [default: docker-exec]

Options docker-exec|ssh

--one-master-host-port-map <one_master_host_port_map>
Publish a container port of one master node to the host. Only Transmission Control Protocol is supported currently. The syntax is <HOST_PORT>:<CONTAINER_PORT>

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

Arguments

INSTALLER
Required argument

Environment variables

MINIDCOS_NODE_DOCKER_VERSION

Provide a default for *--docker-version*

DCOS_LICENSE_KEY_PATH

Provide a default for *--license-key*

MINIDCOS_DOCKER_TRANSPORT

Provide a default for *--transport*

create-loopback-sidecar

Create a loopback sidecar.

A loopback sidecar provides a loopback device that points to a (unformatted) block device.

```
minidcos docker create-loopback-sidecar [OPTIONS] NAME
```

Options

--size <size>
Size (in Megabytes) of the block device.

Arguments

NAME
Required argument

destroy

Destroy a cluster.

```
minidcos docker destroy [OPTIONS]
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--transport <transport>
The communication transport to use. On macOS the SSH transport requires IP routing to be set up. See "minidcos docker setup-mac-network". It also requires the "ssh" command to be available. This can be provided by setting the *MINIDCOS_DOCKER_TRANSPORT* environment variable. When using a TTY, different transports may use different line endings. [default: docker-exec]

Options docker-exec|ssh

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

Environment variables

MINIDCOS_DOCKER_TRANSPORT

Provide a default for *--transport*

destroy-list

Destroy clusters.

To destroy all clusters, run `minidcos docker destroy $(minidcos docker list)`.

```
minidcos docker destroy-list [OPTIONS] [CLUSTER_IDS]...
```

Options

--transport <transport>
The communication transport to use. On macOS the SSH transport requires IP routing to be set up. See "minidcos docker setup-mac-network". It also requires the "ssh" command to be available. This can be provided by setting the *MINIDCOS_DOCKER_TRANSPORT* environment variable. When using a TTY, different transports may use different line endings. [default: docker-exec]

Options docker-exec|ssh

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

Arguments

CLUSTER_IDS
Optional argument(s)

Environment variables

MINIDCOS_DOCKER_TRANSPORT

Provide a default for *--transport*

destroy-loopback-sidecar

Destroy a loopback sidecar.

```
minidcos docker destroy-loopback-sidecar [OPTIONS] NAME
```

Options

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

Arguments

NAME
Required argument

destroy-mac-network

Destroy containers created by "minidcos docker setup-mac-network".

```
minidcos docker destroy-mac-network [OPTIONS]
```

Options

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

doctor

Diagnose common issues which stop this CLI from working correctly.

```
minidcos docker doctor [OPTIONS]
```

Options

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

download-installer

Download a DC/OS Open Source installer.

For DC/OS Enterprise installers, contact your sales representative.

```
minidcos docker download-installer [OPTIONS]
```

Options

--dcos-version <dcos_version>
The DC/OS Open Source installer version to download. This can be in one of the following formats: `stable`, `testing/master`, `testing/<DC/OS MAJOR RELEASE>`, `stable/<DC/OS MINOR RELEASE>`, `testing/pull/<GITHUB-PR-NUMBER>`. See https://dcos.io/releases/ for available releases. If an HTTP or HTTPS URL is given, that is downloaded. [default: stable]

--download-path <download_path>
The path to download an installer to. [default: ./dcos_generate_config.sh]

inspect

Show cluster details.

```
minidcos docker inspect [OPTIONS]
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--transport <transport>
The communication transport to use. On macOS the SSH transport requires IP routing to be set up. See "minidcos docker setup-mac-network". It also requires the "ssh" command to be available. This can be provided by setting the *MINIDCOS_DOCKER_TRANSPORT* environment variable. When using a TTY, different transports may use different line endings. [default: docker-exec]

Options docker-exec|ssh

Environment variables

MINIDCOS_DOCKER_TRANSPORT

Provide a default for *--transport*

install

Install DC/OS on the given Docker cluster.

```
minidcos docker install [OPTIONS] INSTALLER
```

Options

--genconf-dir <files_to_copy_to_genconf_dir>
Path to a directory that contains additional files for the DC/OS installer. All files from this directory will be copied to the "genconf" directory before running the DC/OS installer.

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--extra-config <extra_config>
The path to a file including DC/OS configuration YAML. The contents of this file will be added to add to a default configuration.

--security-mode <security_mode>
The security mode to use for a DC/OS Enterprise cluster. This overrides any security mode set in `--extra-config`.

Options disabled|permissive|strict

--license-key <license_key>
This is ignored if using open source DC/OS. If using DC/OS Enterprise, this defaults to the value of the *DCOS_LICENSE_KEY_PATH* environment variable.

--transport <transport>
The communication transport to use. On macOS the SSH transport requires IP routing to be set up. See "minidcos docker setup-mac-network". It also requires the "ssh" command to be available. This can be provided by setting the *MINIDCOS_DOCKER_TRANSPORT* environment variable. When using a TTY, different transports may use different line endings. [default: docker-exec]

Options docker-exec|ssh

--variant <variant>
Choose the DC/OS variant. If the variant does not match the variant of the given installer, an error will occur. Using "auto" finds the variant from the installer. Finding the variant from the installer takes some time and so using another option is a performance optimization.

Options auto|oss|enterprise

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--wait-for-dcos
Wait for DC/OS after creating the cluster. This is equivalent to using "minidcos docker wait" after this command. "minidcos docker wait" has various options available and so may be more appropriate for your use case. If the chosen transport is "docker-exec", this will skip HTTP checks and so the cluster may not be fully ready.

--workspace-dir <workspace_dir>
Creating a cluster can use approximately 2 GB of temporary storage. Set this option to use a custom "workspace" for this temporary storage. See https://docs.python.org/3/library/tempfile.html#tempfile.gettempdir for details on the temporary directory location if this option is not set.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

Arguments

INSTALLER
Required argument

Environment variables

DCOS_LICENSE_KEY_PATH

Provide a default for *`--license-key`*

MINIDCOS_DOCKER_TRANSPORT

Provide a default for *`--transport`*

list

List all clusters.

```
minidcos docker list [OPTIONS]
```

list-loopback-sidecars

List loopback sidecars.

```
minidcos docker list-loopback-sidecars [OPTIONS]
```

provision

Provision Docker containers to install a DC/OS cluster.

```
minidcos docker provision [OPTIONS]
```

Options

--docker-version <docker_version>
The Docker version to install on the nodes. This can be provided by setting the *MINIDCOS_NODE_DOCKER_VERSION* environment variable. [default: 18.06.3-ce]

Options 1.11.2|1.13.1|17.12.1-ce|18.06.3-ce

--linux-distribution <linux_distribution>
The Linux distribution to use on the nodes. [default: centos-7]

Options centos-7|centos-8|coreos|flatcar|ubuntu-16.04

--docker-storage-driver <docker_storage_driver>
The storage driver to use for Docker in Docker. By default this uses the host's driver. [default: auto]

Options aufs|auto|overlay|overlay2

--mount-sys-fs-cgroup, --no-mount-sys-fs-cgroup
Mounting `/sys/fs/cgroup` from the host is required to run applications which require `cgroup` isolation. Choose to not mount `/sys/fs/cgroup` if it is not available on the host. [default: True]

--masters <masters>
The number of master nodes. [default: 1]

--agents <agents>
The number of agent nodes. [default: 1]

--public-agents <public_agents>
The number of public agent nodes. [default: 1]

-c, --cluster-id <cluster_id>
A unique identifier for the cluster. Use the value "default" to use this cluster for other commands without specifying –cluster-id.

--custom-volume <custom_volume>
Bind mount a volume on all cluster node containers. See https://docs.docker.com/engine/reference/run/#volume-shared-filesystems for the syntax to use.

--custom-master-volume <custom_master_volume>
Bind mount a volume on all cluster master node containers. See https://docs.docker.com/engine/reference/run/#volume-shared-filesystems for the syntax to use.

--custom-agent-volume <custom_agent_volume>
Bind mount a volume on all cluster agent node containers. See https://docs.docker.com/engine/reference/run/#volume-shared-filesystems for the syntax to use.

--custom-public-agent-volume <custom_public_agent_volume>
Bind mount a volume on all cluster public agent node containers. See https://docs.docker.com/engine/reference/run/#volume-shared-filesystems for the syntax to use.

--workspace-dir <workspace_dir>
Creating a cluster can use approximately 2 GB of temporary storage. Set this option to use a custom "workspace" for this temporary storage. See https://docs.python.org/3/library/tempfile.html#tempfile.gettempdir for details on the temporary directory location if this option is not set.

--network <network>
The Docker network containers will be connected to.It may not be possible to SSH to containers on a custom network on macOS.

--transport <transport>
The communication transport to use. On macOS the SSH transport requires IP routing to be set up. See "minidcos docker setup-mac-network". It also requires the "ssh" command to be available. This can be provided by setting the *MINIDCOS_DOCKER_TRANSPORT* environment variable. When using a TTY, different transports may use different line endings. [default: docker-exec]

Options docker-exec|ssh

--one-master-host-port-map <one_master_host_port_map>
Publish a container port of one master node to the host. Only Transmission Control Protocol is supported currently. The syntax is <HOST_PORT>:<CONTAINER_PORT>

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

Environment variables

MINIDCOS_NODE_DOCKER_VERSION

Provide a default for *--docker-version*

MINIDCOS_DOCKER_TRANSPORT

Provide a default for *--transport*

run

Run an arbitrary command on a node or multiple nodes.

To use special characters such as single quotes in your command, wrap the whole command in double quotes.

```
minidcos docker run [OPTIONS] NODE_ARGS...
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--dcos-login-uname <dcos_login_uname>
The username to set the `DCOS_LOGIN_UNAME` environment variable to. [default: bootstrapuser]

--dcos-login-pw <dcos_login_pw>
The password to set the `DCOS_LOGIN_PW` environment variable to. [default: deleteme]

--sync-dir <sync_dir>
The path to a DC/OS checkout. Part of this checkout will be synced to all master nodes before the command is run. The bootstrap directory is synced if the checkout directory variant matches the cluster variant.Integration tests are also synced.Use this option multiple times on a DC/OS Enterprise cluster to sync both DC/OS Enterprise and DC/OS Open Source tests.

-te, --test-env
With this flag set, environment variables are set and the command is run in the integration test directory. This means that "pytest" will run the integration tests.

--node <node>
A reference to a particular node to run the command on. This can be one of: The node's IP address, the node's Docker container name, the node's Docker container ID, a reference in the format "<role>_<number>". These details be seen with `minidcos docker inspect`. [default: master_0]

--env <env>
Set environment variables in the format "<KEY>=<VALUE>"

--transport <transport>
The communication transport to use. On macOS the SSH transport requires IP routing to be set up. See "minidcos docker setup-mac-network". It also requires the "ssh" command to be available. This can be provided by setting the *MINIDCOS_DOCKER_TRANSPORT* environment variable. When using a TTY, different transports may use different line endings. [default: docker-exec]

Options docker-exec|ssh

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

Arguments

NODE_ARGS
Required argument(s)

Environment variables

MINIDCOS_DOCKER_TRANSPORT

Provide a default for *--transport*

send-file

Send a file to a node or multiple nodes.

```
minidcos docker send-file [OPTIONS] SOURCE DESTINATION
```

Options

-c, --cluster-id `<cluster_id>`
The ID of the cluster to use. [default: default]

--transport `<transport>`
The communication transport to use. On macOS the SSH transport requires IP routing to be set up. See "minidcos docker setup-mac-network". It also requires the "ssh" command to be available. This can be provided by setting the *MINIDCOS_DOCKER_TRANSPORT* environment variable. When using a TTY, different transports may use different line endings. [default: docker-exec]

Options docker-exec|ssh

--node `<node>`
A reference to a particular node to run the command on. This can be one of: The node's IP address, the node's Docker container name, the node's Docker container ID, a reference in the format "<role>_<number>". These details be seen with `minidcos docker inspect`. [default: master_0]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

Arguments

SOURCE
Required argument

DESTINATION
Required argument

Environment variables

MINIDCOS_DOCKER_TRANSPORT

Provide a default for *`--transport`*

setup-mac-network

Set up a network to connect to nodes on macOS.

This creates an OpenVPN configuration file and describes how to use it.

```
minidcos docker setup-mac-network [OPTIONS]
```

Options

--configuration-dst `<configuration_dst>`
The location to create an OpenVPN configuration file. [default: ~/Documents/docker-for-mac.ovpn]

--force
Overwrite any files and destroy conflicting containers from previous uses of this command.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

sync

Sync files from a DC/OS checkout to master nodes.

This syncs integration test files and bootstrap files.

`DCOS_CHECKOUT_DIR` should be set to the path of clone of an open source DC/OS or DC/OS Enterprise repository.

By default the `DCOS_CHECKOUT_DIR` argument is set to the value of the `DCOS_CHECKOUT_DIR` environment variable.

If no `DCOS_CHECKOUT_DIR` is given, the current working directory is used.

This makes an assumption that all DC/OS Enterprise and DC/OS OSS integration tests are in the top level `packages/dcos-integration-test` directory.

```
minidcos docker sync [OPTIONS] [DCOS_CHECKOUT_DIR]
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--transport <transport>
The communication transport to use. On macOS the SSH transport requires IP routing to be set up. See "minidcos docker setup-mac-network". It also requires the "ssh" command to be available. This can be provided by setting the *MINIDCOS_DOCKER_TRANSPORT* environment variable. When using a TTY, different transports may use different line endings. [default: docker-exec]

Options docker-exec|ssh

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

Arguments

DCOS_CHECKOUT_DIR
Optional argument

Environment variables

DCOS_CHECKOUT_DIR

Provide a default for *DCOS_CHECKOUT_DIR*

MINIDCOS_DOCKER_TRANSPORT

Provide a default for *--transport*

upgrade

Upgrade a cluster to a given version of DC/OS.

```
minidcos docker upgrade [OPTIONS] INSTALLER
```

Options

-c, --cluster-id `<cluster_id>`
The ID of the cluster to use. [default: default]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--extra-config `<extra_config>`
The path to a file including DC/OS configuration YAML. The contents of this file will be added to add to a default configuration.

--variant `<variant>`
Choose the DC/OS variant. If the variant does not match the variant of the given installer, an error will occur. Using "auto" finds the variant from the installer. Finding the variant from the installer takes some time and so using another option is a performance optimization.

Options auto|oss|enterprise

--transport `<transport>`
The communication transport to use. On macOS the SSH transport requires IP routing to be set up. See "minidcos docker setup-mac-network". It also requires the "ssh" command to be available. This can be provided by setting the *MINIDCOS_DOCKER_TRANSPORT* environment variable. When using a TTY, different transports may use different line endings. [default: docker-exec]

Options docker-exec|ssh

--workspace-dir `<workspace_dir>`
Creating a cluster can use approximately 2 GB of temporary storage. Set this option to use a custom "workspace" for this temporary storage. See https://docs.python.org/3/library/tempfile.html#tempfile.gettempdir for details on the temporary directory location if this option is not set.

--security-mode `<security_mode>`
The security mode to use for a DC/OS Enterprise cluster. This overrides any security mode set in `--extra-config`.

Options disabled|permissive|strict

--wait-for-dcos
Wait for DC/OS after creating the cluster. This is equivalent to using "minidcos docker wait" after this command. "minidcos docker wait" has various options available and so may be more appropriate for your use case. If the chosen transport is "docker-exec", this will skip HTTP checks and so the cluster may not be fully ready.

--license-key `<license_key>`
This is ignored if using open source DC/OS. If using DC/OS Enterprise, this defaults to the value of the *DCOS_LICENSE_KEY_PATH* environment variable.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

--genconf-dir `<files_to_copy_to_genconf_dir>`
Path to a directory that contains additional files for the DC/OS installer. All files from this directory will be copied to the "genconf" directory before running the DC/OS installer.

Arguments

INSTALLER
Required argument

Environment variables

MINIDCOS_DOCKER_TRANSPORT

Provide a default for *--transport*

DCOS_LICENSE_KEY_PATH

Provide a default for *--license-key*

wait

Wait for DC/OS to start.

```
minidcos docker wait [OPTIONS]
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--superuser-username <superuser_username>
The superuser username is needed only on DC/OS Enterprise clusters. [default: bootstrapuser]

--superuser-password <superuser_password>
The superuser password is needed only on DC/OS Enterprise clusters. [default: deleteme]

--skip-http-checks
Do not wait for checks which require an HTTP connection to the cluster. If this flag is used, this command may return before DC/OS is fully ready. Use this flag in cases where an HTTP connection cannot be made to the cluster. For example this is useful on macOS without a VPN set up. [default: False]

--transport <transport>
The communication transport to use. On macOS the SSH transport requires IP routing to be set up. See "minidcos docker setup-mac-network". It also requires the "ssh" command to be available. This can be provided by setting the *MINIDCOS_DOCKER_TRANSPORT* environment variable. When using a TTY, different transports may use different line endings. [default: docker-exec]

Options docker-exec|ssh

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

Environment variables

MINIDCOS_DOCKER_TRANSPORT

Provide a default for *--transport*

web

Open the browser at the web UI.

Note that the web UI may not be available at first. Consider using `minidcos docker wait` before running this command.

```
minidcos docker web [OPTIONS]
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--transport <transport>
The communication transport to use. On macOS the SSH transport requires IP routing to be set up. See "minidcos docker setup-mac-network". It also requires the "ssh" command to be available. This can be provided by setting the *MINIDCOS_DOCKER_TRANSPORT* environment variable. When using a TTY, different transports may use different line endings. [default: docker-exec]

Options docker-exec|ssh

Environment variables

MINIDCOS_DOCKER_TRANSPORT

Provide a default for *--transport*

CHAPTER

TWO

VAGRANT

The *minidcos vagrant* CLI allows you to create, manage and destroy open source DC/OS and DC/OS Enterprise clusters on Vagrant VMs.

A typical CLI workflow for open source DC/OS may look like the following. Install the CLI (see *Installation*), then create and manage a cluster:

```
# Fix issues shown by minidcos vagrant doctor
$ minidcos vagrant doctor
$ minidcos vagrant create ./dcos_generate_config.sh --agents 0
default
$ minidcos vagrant wait
$ minidcos vagrant run --test-env --sync-dir /path/to/dcos/checkout pytest -k test_tls
...
# Get onto a node
$ minidcos vagrant run bash
[master-0]# exit
$ minidcos vagrant destroy
```

Each of these and more are described in detail below.

2.1 Requirements

2.1.1 Hardware

A minimum of 2 GB of free memory is required per DC/OS node.

2.1.2 `ssh`

The `ssh` command must be available.

2.1.3 Vagrant by HashiCorp

Vagrant must be installed. This has been tested with:

- Vagrant 2.1.1
- Vagrant 2.1.2

2.1.4 Oracle VirtualBox

VirtualBox must be installed. This has been tested with VirtualBox 5.1.18.

2.1.5 `vagrant-vbguest` plugin

vagrant-vbguest must be installed.

2.1.6 `doctor` command

minidcos vagrant comes with the *doctor* command. Run this command to check your system for common causes of problems.

2.2 Creating a Cluster

To create a cluster you first need to download a DC/OS installer.

This can be done via the releases page or with the *download-installer* command.

DC/OS Enterprise is also supported. Ask your sales representative for installers.

Creating a cluster is possible with the *create* command. This command allows you to customize the cluster in many ways.

The command returns when the DC/OS installation process has started. To wait until DC/OS has finished installing, use the *wait* command.

To use this cluster, it is useful to find details using the *inspect* command.

2.2.1 DC/OS Enterprise

There are multiple DC/OS Enterprise-only features available in *create*.

The only extra requirement is to give a valid license key, for DC/OS 1.11+. See *create* for details on how to provide a license key.

Ask your sales representative for DC/OS Enterprise installers.

For, example, run the following to create a DC/OS Enterprise cluster in strict mode:

```
$ minidcos vagrant create /path/to/dcos_generate_config.ee.sh       --license-key /path/to/license.tx
```

The command returns when the DC/OS installation process has started. To wait until DC/OS has finished installing, use the *wait* command.

See *create* for details on this command and its options.

2.3 Cluster IDs

Clusters have unique IDs. Multiple commands take `--cluster-id` options. Specify a cluster ID in *create*, and then use it in other commands. Any command which takes a `--cluster-id` option defaults to using "default" if no cluster ID is given.

2.4 Running commands on Cluster Nodes

It is possible to run commands on a cluster node in multiple ways. These include using *run* and `ssh`.

2.4.1 Running commands on a cluster node using run

It is possible to run the following to run a command on an arbitrary master node.

```
$ minidcos vagrant run systemctl list-units
```

See *run* for more information on this command.

2.4.2 Running commands on a cluster node using `ssh`

One SSH key allows access to all nodes in the cluster. See this SSH key's path and the IP addresses of nodes using *inspect*.

2.5 Getting on to a Cluster Node

Sometimes it is useful to get onto a cluster node. To do this, you can use any of the ways of *Running commands on Cluster Nodes*.

For example, to use *run* to run `bash` to get on to an arbitrary master node:

```
$ minidcos vagrant run bash
```

2.6 Destroying Clusters

There are two commands which can be used to destroy clusters. These are *destroy* and *destroy-list*.

Either destroy a cluster with *destroy*:

```
$ minidcos vagrant destroy
default
$ minidcos vagrant destroy --cluster-id pr_4033_strict
pr_4033_strict
```

or use *destroy-list* to destroy multiple clusters:

```
$ minidcos vagrant destroy-list pr_4033_strict pr_4019_permissive
pr_4033_strict
pr_4019_permissive
```

To destroy all clusters, run the following command:

```
$ minidcos vagrant destroy-list $(dcos-vagrant list)
pr_4033_strict
pr_4019_permissive
```

2.7 Running Integration Tests

The *run* command is useful for running integration tests.

To run integration tests which are developed in the a DC/OS checkout at `/path/to/dcos`, you can use the following workflow:

```
$ minidcos vagrant create ./dcos_generate_config.sh
$ minidcos vagrant wait
$ minidcos vagrant run --test-env --sync-dir /path/to/dcos/checkout pytest -k test_tls.py
```

There are multiple options and shortcuts for using these commands. See *run* for more information on this command.

2.8 Viewing the Web UI

To view the web UI of your cluster, use the *web* command. To see the web UI URL of your cluster, use the *inspect* command.

Before viewing the UI, you may first need to configure your browser to trust your DC/OS CA, or choose to override the browser protection.

2.9 Using a Custom CA Certificate

On DC/OS Enterprise clusters, it is possible to use a custom CA certificate. See the Custom CA certificate documentation for details. It is possible to use *create* to create a cluster with a custom CA certificate.

1. Create or obtain the necessary files:

   `dcos-ca-certificate.crt`, `dcos-ca-certificate-key.key`, and `dcos-ca-certificate-chain.crt`.

2. Put the above-mentioned files into a directory, e.g. `/path/to/genconf/`.

3. Create a file containing the "extra" configuration.

   create takes an `--extra-config` option. This adds the contents of the specified YAML file to a minimal DC/OS configuration.

   Create a file with the following contents:

   ```
   ca_certificate_path: genconf/dcos-ca-certificate.crt
   ca_certificate_key_path: genconf/dcos-ca-certificate-key.key
   ca_certificate_chain_path: genconf/dcos-ca-certificate-chain.crt
   ```

4. Create a cluster.

   ```
   $ minidcos vagrant create /path/to/dcos_generate_config.ee.sh --variant enterprise
   ```

5. Verify that everything has worked.

   See Verify installation for steps to verify that the DC/OS Enterprise cluster was installed properly with the custom CA certificate.

2.10 CLI Reference

2.10.1 minidcos vagrant

Manage DC/OS clusters on Vagrant.

```
minidcos vagrant [OPTIONS] COMMAND [ARGS]...
```

clean

Remove VMs created by this tool.

This is useful in removing paused and aborted VMs. VMs are aborted when the host is shut down.

```
minidcos vagrant clean [OPTIONS]
```

Options

--destroy-running-clusters
Destroy running clusters. [default: False]

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

create

Create a DC/OS cluster.

DC/OS Enterprise

DC/OS Enterprise clusters require different configuration variables to DC/OS OSS. For example, enterprise clusters require the following configuration parameters:

`superuser_username`, `superuser_password_hash`, `fault_domain_enabled`, `license_key_contents`

These can all be set in `--extra-config`. However, some defaults are provided for all but the license key.

The default superuser username is `bootstrapuser`. The default superuser password is `deleteme`. The default `fault_domain_enabled` is `false`.

`license_key_contents` must be set for DC/OS Enterprise 1.11 and above. This is set to one of the following, in order:

* The `license_key_contents` set in `--extra-config`. * The contents of the path given with `--license-key`. * The contents of the path set in the `DCOS_LICENSE_KEY_PATH` environment variable.

If none of these are set, `license_key_contents` is not given.

```
minidcos vagrant create [OPTIONS] INSTALLER
```

Options

--masters <masters>
The number of master nodes. [default: 1]

--agents <agents>
The number of agent nodes. [default: 1]

--extra-config <extra_config>
The path to a file including DC/OS configuration YAML. The contents of this file will be added to add to a default configuration.

--public-agents <public_agents>
The number of public agent nodes. [default: 1]

--workspace-dir <workspace_dir>
Creating a cluster can use approximately 2 GB of temporary storage. Set this option to use a custom "workspace" for this temporary storage. See https://docs.python.org/3/library/tempfile.html#tempfile.gettempdir for details on the temporary directory location if this option is not set.

--variant <variant>
Choose the DC/OS variant. If the variant does not match the variant of the given installer, an error will occur. Using "auto" finds the variant from the installer. Finding the variant from the installer takes some time and so using another option is a performance optimization.

Options auto|oss|enterprise

--license-key <license_key>
This is ignored if using open source DC/OS. If using DC/OS Enterprise, this defaults to the value of the *DCOS_LICENSE_KEY_PATH* environment variable.

--genconf-dir <files_to_copy_to_genconf_dir>
Path to a directory that contains additional files for the DC/OS installer. All files from this directory will be copied to the "genconf" directory before running the DC/OS installer.

--security-mode <security_mode>
The security mode to use for a DC/OS Enterprise cluster. This overrides any security mode set in `--extra-config`.

Options disabled|permissive|strict

--copy-to-master <copy_to_master>
Files to copy to master nodes before installing DC/OS. This option can be given multiple times. Each option should be in the format /absolute/local/path:/remote/path.

-c, --cluster-id <cluster_id>
A unique identifier for the cluster. Use the value "default" to use this cluster for other commands without specifying –cluster-id.

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--vm-memory-mb <vm_memory_mb>
The amount of memory to give each VM. [default: 2048]

--enable-selinux-enforcing
With this flag set, SELinux is set to enforcing before DC/OS is installed on the cluster.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

--vagrant-box-url <vagrant_box_url>
The URL of the Vagrant box to use. [default: https://downloads.dcos.io/dcos-vagrant/metadata.json]

--vagrant-box-version <vagrant_box_version>
The version of the Vagrant box to use. See https://www.vagrantup.com/docs/boxes/versioning.html#version-constraints for details. [default: ~> 0.10]

--wait-for-dcos
Wait for DC/OS after creating the cluster. This is equivalent to using "minidcos vagrant wait" after this command.

Arguments

INSTALLER
Required argument

Environment variables

DCOS_LICENSE_KEY_PATH

Provide a default for *--license-key*

destroy

Destroy a cluster.

```
minidcos vagrant destroy [OPTIONS]
```

Options

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

destroy-list

Destroy clusters.

To destroy all clusters, run `minidcos vagrant destroy $(minidcos vagrant list)`.

```
minidcos vagrant destroy-list [OPTIONS] [CLUSTER_IDS]...
```

Options

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

Arguments

CLUSTER_IDS
Optional argument(s)

doctor

Diagnose common issues which stop this CLI from working correctly.

```
minidcos vagrant doctor [OPTIONS]
```

Options

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

download-installer

Download a DC/OS Open Source installer.

For DC/OS Enterprise installers, contact your sales representative.

```
minidcos vagrant download-installer [OPTIONS]
```

Options

--dcos-version <dcos_version>
The DC/OS Open Source installer version to download. This can be in one of the following formats: `stable`, `testing/master`, `testing/<DC/OS MAJOR RELEASE>`, `stable/<DC/OS MINOR RELEASE>`, `testing/pull/<GITHUB-PR-NUMBER>`. See https://dcos.io/releases/ for available releases. If an HTTP or HTTPS URL is given, that is downloaded. [default: stable]

--download-path <download_path>
The path to download an installer to. [default: ./dcos_generate_config.sh]

inspect

Show cluster details.

```
minidcos vagrant inspect [OPTIONS]
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

install

Install DC/OS on a provisioned Vagrant cluster.

```
minidcos vagrant install [OPTIONS] INSTALLER
```

Options

--extra-config <extra_config>
The path to a file including DC/OS configuration YAML. The contents of this file will be added to add to a default configuration.

--workspace-dir <workspace_dir>
Creating a cluster can use approximately 2 GB of temporary storage. Set this option to use a custom "workspace" for this temporary storage. See https://docs.python.org/3/library/tempfile.html#tempfile.gettempdir for details on the temporary directory location if this option is not set.

--variant <variant>
Choose the DC/OS variant. If the variant does not match the variant of the given installer, an error will occur. Using "auto" finds the variant from the installer. Finding the variant from the installer takes some time and so using another option is a performance optimization.

Options auto|oss|enterprise

--license-key <license_key>
This is ignored if using open source DC/OS. If using DC/OS Enterprise, this defaults to the value of the *DCOS_LICENSE_KEY_PATH* environment variable.

--genconf-dir <files_to_copy_to_genconf_dir>
Path to a directory that contains additional files for the DC/OS installer. All files from this directory will be copied to the "genconf" directory before running the DC/OS installer.

--security-mode <security_mode>
The security mode to use for a DC/OS Enterprise cluster. This overrides any security mode set in `--extra-config`.

Options disabled|permissive|strict

-c, --cluster-id <cluster_id>
A unique identifier for the cluster. Use the value "default" to use this cluster for other commands without specifying –cluster-id.

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

--wait-for-dcos
Wait for DC/OS after creating the cluster. This is equivalent to using "minidcos vagrant wait" after this command.

Arguments

INSTALLER
Required argument

Environment variables

DCOS_LICENSE_KEY_PATH

Provide a default for *--license-key*

list

List all clusters.

```
minidcos vagrant list [OPTIONS]
```

provision

Provision a Vagrant cluster for installing DC/OS.

```
minidcos vagrant provision [OPTIONS]
```

Options

--masters <masters>
The number of master nodes. [default: 1]

--agents <agents>
The number of agent nodes. [default: 1]

--public-agents <public_agents>
The number of public agent nodes. [default: 1]

--workspace-dir <workspace_dir>
Creating a cluster can use approximately 2 GB of temporary storage. Set this option to use a custom "workspace" for this temporary storage. See https://docs.python.org/3/library/tempfile.html#tempfile.gettempdir for details on the temporary directory location if this option is not set.

-c, --cluster-id <cluster_id>
A unique identifier for the cluster. Use the value "default" to use this cluster for other commands without specifying –cluster-id.

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--enable-selinux-enforcing
With this flag set, SELinux is set to enforcing before DC/OS is installed on the cluster.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

--vagrant-box-url <vagrant_box_url>
The URL of the Vagrant box to use. [default: https://downloads.dcos.io/dcos-vagrant/metadata.json]

--vagrant-box-version <vagrant_box_version>
The version of the Vagrant box to use. See https://www.vagrantup.com/docs/boxes/versioning.html#version-constraints for details. [default: ~> 0.10]

--vm-memory-mb <vm_memory_mb>
The amount of memory to give each VM. [default: 2048]

run

Run an arbitrary command on a node or multiple nodes.

To use special characters such as single quotes in your command, wrap the whole command in double quotes.

```
minidcos vagrant run [OPTIONS] NODE_ARGS...
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--dcos-login-uname <dcos_login_uname>
The username to set the `DCOS_LOGIN_UNAME` environment variable to. [default: bootstrapuser]

--dcos-login-pw <dcos_login_pw>
The password to set the `DCOS_LOGIN_PW` environment variable to. [default: deleteme]

--sync-dir <sync_dir>
The path to a DC/OS checkout. Part of this checkout will be synced to all master nodes before the command is run. The bootstrap directory is synced if the checkout directory variant matches the cluster variant.Integration tests are also synced.Use this option multiple times on a DC/OS Enterprise cluster to sync both DC/OS Enterprise and DC/OS Open Source tests.

-te, --test-env
With this flag set, environment variables are set and the command is run in the integration test directory. This means that "pytest" will run the integration tests.

--env <env>
Set environment variables in the format "<KEY>=<VALUE>"

--node <node>
A reference to a particular node to run the command on. This can be one of: The node's IP address, the node's VM name, a reference in the format "<role>_<number>". These details be seen with `minidcos vagrant inspect`. [default: master_0]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

Arguments

NODE_ARGS
Required argument(s)

send-file

Send a file to a node or multiple nodes.

```
minidcos vagrant send-file [OPTIONS] SOURCE DESTINATION
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--node `<node>`
A reference to a particular node to run the command on. This can be one of: The node's IP address, the node's VM name, a reference in the format "<role>_<number>". These details be seen with `minidcos vagrant inspect`. [default: master_0]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

Arguments

SOURCE
Required argument

DESTINATION
Required argument

sync

Sync files from a DC/OS checkout to master nodes.

This syncs integration test files and bootstrap files.

`DCOS_CHECKOUT_DIR` should be set to the path of clone of an open source DC/OS or DC/OS Enterprise repository.

By default the `DCOS_CHECKOUT_DIR` argument is set to the value of the `DCOS_CHECKOUT_DIR` environment variable.

If no `DCOS_CHECKOUT_DIR` is given, the current working directory is used.

This makes an assumption that all DC/OS Enterprise and DC/OS OSS integration tests are in the top level `packages/dcos-integration-test` directory.

```
minidcos vagrant sync [OPTIONS] [DCOS_CHECKOUT_DIR]
```

Options

-c, --cluster-id `<cluster_id>`
The ID of the cluster to use. [default: default]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

Arguments

DCOS_CHECKOUT_DIR
Optional argument

Environment variables

DCOS_CHECKOUT_DIR

Provide a default for *DCOS_CHECKOUT_DIR*

upgrade

Upgrade a cluster to a given version of DC/OS.

```
minidcos vagrant upgrade [OPTIONS] INSTALLER
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--extra-config <extra_config>
The path to a file including DC/OS configuration YAML. The contents of this file will be added to add to a default configuration.

--variant <variant>
Choose the DC/OS variant. If the variant does not match the variant of the given installer, an error will occur. Using "auto" finds the variant from the installer. Finding the variant from the installer takes some time and so using another option is a performance optimization.

Options auto|oss|enterprise

--workspace-dir <workspace_dir>
Creating a cluster can use approximately 2 GB of temporary storage. Set this option to use a custom "workspace" for this temporary storage. See https://docs.python.org/3/library/tempfile.html#tempfile.gettempdir for details on the temporary directory location if this option is not set.

--security-mode <security_mode>
The security mode to use for a DC/OS Enterprise cluster. This overrides any security mode set in `--extra-config`.

Options disabled|permissive|strict

--wait-for-dcos
Wait for DC/OS after creating the cluster. This is equivalent to using "minidcos vagrant wait" after this command.

--license-key <license_key>
This is ignored if using open source DC/OS. If using DC/OS Enterprise, this defaults to the value of the *DCOS_LICENSE_KEY_PATH* environment variable.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

--genconf-dir <files_to_copy_to_genconf_dir>
Path to a directory that contains additional files for the DC/OS installer. All files from this directory will be copied to the "genconf" directory before running the DC/OS installer.

Arguments

INSTALLER
Required argument

Environment variables

DCOS_LICENSE_KEY_PATH

Provide a default for *--license-key*

wait

Wait for DC/OS to start.

```
minidcos vagrant wait [OPTIONS]
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--superuser-username <superuser_username>
The superuser username is needed only on DC/OS Enterprise clusters. [default: bootstrapuser]

--superuser-password <superuser_password>
The superuser password is needed only on DC/OS Enterprise clusters. [default: deleteme]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

web

Open the browser at the web UI.

Note that the web UI may not be available at first. Consider using `minidcos vagrant wait` before running this command.

```
minidcos vagrant web [OPTIONS]
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

CHAPTER

THREE

AWS

The *minidcos aws* CLI allows you to create and manage open source DC/OS and DC/OS Enterprise clusters on AWS EC2 instances.

A typical CLI workflow for open source DC/OS may look like the following. Install the CLI (see *Installation*), then create and manage a cluster:

```
# Fix issues shown by minidcos aws doctor
$ minidcos aws doctor
$ minidcos aws create https://downloads.dcos.io/dcos/stable/dcos_generate_config.sh --variant oss
default
$ minidcos aws wait
$ minidcos aws run --test-env --sync-dir /path/to/dcos/checkout pytest -k test_tls
...
# Get onto a node
$ minidcos aws run bash
[master-0]# exit
$ minidcos aws destroy
```

Each of these and more are described in detail below.

3.1 Requirements

3.1.1 Amazon Web Services

An Amazon Web Services account with sufficient funds must be available.

The AWS credentials for the account must be present either in the environment as environment variables or in the default file system location under `~/.aws/credentials` with a AWS profile in the environment referencing those credentials.

The Mesosphere internal AWS tool maws automatically stores account specific temporary AWS credentials in the default file system location and exports the corresponding profile into the environment. After logging in with maws clusters can be launched using the AWS backend.

For CI deployments long lived credentials are preferred. It is recommended to use the environment variables method for AWS credentials in that case.

The environment variables are set as follows:

```
$ export AWS_ACCESS_KEY_ID=<aws_access_key_id>
$ export AWS_SECRET_ACCESS_KEY=<aws_secret_access_key>
```

The EC2 instances launched by the AWS backend will bring about costs in the order of 24 ct per instance, assuming the fixed cluster lifetime of two hours and `m4.large` EC2 instances.

3.1.2 `ssh`

The `ssh` command must be available.

3.1.3 Operating System

The AWS backend has been tested on macOS and on Linux.

It is not expected that it will work out of the box with Windows, see issue QUALITY-1771.

If your operating system is not supported, it may be possible to use Vagrant, or another Linux virtual machine.

3.1.4 `doctor` command

minidcos aws comes with the *doctor* command. Run this command to check your system for common causes of problems.

3.2 Creating a Cluster

To create a cluster you first need the link to a DC/OS installer.

These can be found on the releases page.

DC/OS Enterprise is also supported. Ask your sales representative for installers.

Creating a cluster is possible with the *create* command. This command allows you to customize the cluster in many ways.

The command returns when the DC/OS installation process has started. To wait until DC/OS has finished installing, use the *wait* command.

To use this cluster, it is useful to find details using the *inspect* command.

3.2.1 DC/OS Enterprise

There are multiple DC/OS Enterprise-only features available in *create*.

The only extra requirement is to give a valid license key, for DC/OS 1.11+. See *create* for details on how to provide a license key.

Ask your sales representative for DC/OS Enterprise installers.

For, example, run the following to create a DC/OS Enterprise cluster in strict mode:

```
$ minidcos aws create $DCOS_ENTERPRISE_URL       --variant enterprise       --license-key /path/to/li
```

The command returns when the DC/OS installation process has started. To wait until DC/OS has finished installing, use the *wait* command.

See *create* for details on this command and its options.

3.3 Cluster IDs

Clusters have unique IDs. Multiple commands take `--cluster-id` options. Specify a cluster ID in *create*, and then use it in other commands. Any command which takes a `--cluster-id` option defaults to using "default" if no cluster ID is given.

3.4 Running commands on Cluster Nodes

It is possible to run commands on a cluster node in multiple ways. These include using *run* and `ssh`.

3.4.1 Running commands on a cluster node using run

It is possible to run the following to run a command on an arbitrary master node.

```
$ minidcos aws run systemctl list-units
```

See *run* for more information on this command.

3.4.2 Running commands on a cluster node using `ssh`

One SSH key allows access to all nodes in the cluster. See this SSH key's path and the IP addresses of nodes using *inspect*.

3.5 Getting on to a Cluster Node

Sometimes it is useful to get onto a cluster node. To do this, you can use any of the ways of *Running commands on Cluster Nodes*.

For example, to use *run* to run `bash` to get on to an arbitrary master node:

```
$ minidcos aws run bash
```

3.6 Destroying Clusters

There are two commands which can be used to destroy clusters. These are *destroy* and *destroy-list*.

Either destroy a cluster with *destroy*:

```
$ minidcos aws destroy
default
$ minidcos aws destroy --cluster-id pr_4033_strict
pr_4033_strict
```

or use *destroy-list* to destroy multiple clusters:

```
$ minidcos aws destroy-list pr_4033_strict pr_4019_permissive
pr_4033_strict
pr_4019_permissive
```

To destroy all clusters, run the following command:

```
$ minidcos aws destroy-list $(dcos-aws list)
pr_4033_strict
pr_4019_permissive
```

3.7 Running Integration Tests

The *run* command is useful for running integration tests.

To run integration tests which are developed in the a DC/OS checkout at `/path/to/dcos`, you can use the following workflow:

```
$ minidcos aws create       --variant oss       https://downloads.dcos.io/dcos/stable/dcos_generate_c
$ minidcos aws wait
$ minidcos aws run --test-env --sync-dir /path/to/dcos/checkout pytest -k test_tls.py
```

There are multiple options and shortcuts for using these commands. See *run* for more information on this command.

3.8 Viewing the Web UI

To view the web UI of your cluster, use the *web* command. To see the web UI URL of your cluster, use the *inspect* command.

Before viewing the UI, you may first need to configure your browser to trust your DC/OS CA, or choose to override the browser protection.

3.9 Using a Custom CA Certificate

On DC/OS Enterprise clusters, it is possible to use a custom CA certificate. See the Custom CA certificate documentation for details. It is possible to use *create* to create a cluster with a custom CA certificate.

1. Create or obtain the necessary files:

   `dcos-ca-certificate.crt`, `dcos-ca-certificate-key.key`, and `dcos-ca-certificate-chain.crt`.

2. Put the above-mentioned files into a directory, e.g. `/path/to/genconf/`.

3. Create a file containing the "extra" configuration.

   create takes an `--extra-config` option. This adds the contents of the specified YAML file to a minimal DC/OS configuration.

   Create a file with the following contents:

   ```
   ca_certificate_path: genconf/dcos-ca-certificate.crt
   ca_certificate_key_path: genconf/dcos-ca-certificate-key.key
   ca_certificate_chain_path: genconf/dcos-ca-certificate-chain.crt
   ```

4. Create a cluster.

   ```
   $ minidcos aws create      $DCOS_ENTERPRISE_URL      --variant enterprise      --genconf-dir /pa
   ```

5. Verify that everything has worked.

   See Verify installation for steps to verify that the DC/OS Enterprise cluster was installed properly with the custom CA certificate.

3.10 CLI Reference

3.10.1 minidcos aws

Manage DC/OS clusters on AWS.

```
minidcos aws [OPTIONS] COMMAND [ARGS]...
```

create

Create a DC/OS cluster.

DC/OS Enterprise

DC/OS Enterprise clusters require different configuration variables to DC/OS OSS. For example, enterprise clusters require the following configuration parameters:

`superuser_username`, `superuser_password_hash`, `fault_domain_enabled`, `license_key_contents`

These can all be set in `--extra-config`. However, some defaults are provided for all but the license key.

The default superuser username is `bootstrapuser`. The default superuser password is `deleteme`. The default `fault_domain_enabled` is `false`.

`license_key_contents` must be set for DC/OS Enterprise 1.11 and above. This is set to one of the following, in order:

* The `license_key_contents` set in `--extra-config`. * The contents of the path given with `--license-key`. * The contents of the path set in the `DCOS_LICENSE_KEY_PATH` environment variable.

If none of these are set, `license_key_contents` is not given.

```
minidcos aws create [OPTIONS] INSTALLER_URL
```

Options

--custom-tag `<custom_tag>`
Add tags to EC2 instances in the format "<TAG_KEY>:<TAG_VALUE>".

--variant `<variant>`
Choose the DC/OS variant. If the variant does not match the variant of the given installer URL, an error will occur. [required]

Options oss|enterprise

--wait-for-dcos
Wait for DC/OS after creating the cluster. This is equivalent to using "minidcos aws wait" after this command. "minidcos aws wait" has various options available and so may be more appropriate for your use case.

--masters `<masters>`
The number of master nodes. [default: 1]

--agents `<agents>`
The number of agent nodes. [default: 1]

--extra-config `<extra_config>`
The path to a file including DC/OS configuration YAML. The contents of this file will be added to add to a default configuration.

--public-agents `<public_agents>`
The number of public agent nodes. [default: 1]

--aws-instance-type `<aws_instance_type>`
The AWS instance type to use. [default: m4.large]

--aws-region `<aws_region>`
The AWS region to use. [default: us-west-2]

--linux-distribution `<linux_distribution>`
The Linux distribution to use on the nodes. [default: centos-7]

Options centos-7|coreos

--workspace-dir `<workspace_dir>`
Creating a cluster can use approximately 2 GB of temporary storage. Set this option to use a custom "workspace" for this temporary storage. See https://docs.python.org/3/library/tempfile.html#tempfile.gettempdir for details on the temporary directory location if this option is not set.

--license-key `<license_key>`
This is ignored if using open source DC/OS. If using DC/OS Enterprise, this defaults to the value of the *DCOS_LICENSE_KEY_PATH* environment variable.

--genconf-dir `<files_to_copy_to_genconf_dir>`
Path to a directory that contains additional files for the DC/OS installer. All files from this directory will be copied to the "genconf" directory before running the DC/OS installer.

--security-mode `<security_mode>`
The security mode to use for a DC/OS Enterprise cluster. This overrides any security mode set in `--extra-config`.

Options disabled|permissive|strict

--copy-to-master `<copy_to_master>`
Files to copy to master nodes before installing DC/OS. This option can be given multiple times. Each option should be in the format /absolute/local/path:/remote/path.

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

-c, --cluster-id `<cluster_id>`
A unique identifier for the cluster. Use the value "default" to use this cluster for other commands without specifying –cluster-id.

--enable-selinux-enforcing
With this flag set, SELinux is set to enforcing before DC/OS is installed on the cluster.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

Arguments

INSTALLER_URL
Required argument

Environment variables

DCOS_LICENSE_KEY_PATH

Provide a default for *--license-key*

destroy

Destroy a cluster.

```
minidcos aws destroy [OPTIONS]
```

Options

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

--aws-region <aws_region>
The AWS region to use. [default: us-west-2]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

destroy-list

Destroy clusters.

To destroy all clusters, run `minidcos aws destroy $(minidcos aws list)`.

```
minidcos aws destroy-list [OPTIONS] [CLUSTER_IDS]...
```

Options

--aws-region <aws_region>
The AWS region to use. [default: us-west-2]

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

Arguments

CLUSTER_IDS
Optional argument(s)

doctor

Diagnose common issues which stop this CLI from working correctly.

```
minidcos aws doctor [OPTIONS]
```

Options

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

inspect

Show cluster details.

```
minidcos aws inspect [OPTIONS]
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--aws-region <aws_region>
The AWS region to use. [default: us-west-2]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

install

Install DC/OS on a provisioned AWS cluster.

```
minidcos aws install [OPTIONS] INSTALLER_URL
```

Options

--variant <variant>
Choose the DC/OS variant. If the variant does not match the variant of the given installer URL, an error will occur. [required]

Options osslenterprise

--wait-for-dcos
Wait for DC/OS after creating the cluster. This is equivalent to using "minidcos aws wait" after this command. "minidcos aws wait" has various options available and so may be more appropriate for your use case.

--extra-config <extra_config>
The path to a file including DC/OS configuration YAML. The contents of this file will be added to add to a default configuration.

--aws-region <aws_region>
The AWS region to use. [default: us-west-2]

--workspace-dir `<workspace_dir>`
Creating a cluster can use approximately 2 GB of temporary storage. Set this option to use a custom "workspace" for this temporary storage. See https://docs.python.org/3/library/tempfile.html#tempfile.gettempdir for details on the temporary directory location if this option is not set.

--license-key `<license_key>`
This is ignored if using open source DC/OS. If using DC/OS Enterprise, this defaults to the value of the *DCOS_LICENSE_KEY_PATH* environment variable.

--genconf-dir `<files_to_copy_to_genconf_dir>`
Path to a directory that contains additional files for the DC/OS installer. All files from this directory will be copied to the "genconf" directory before running the DC/OS installer.

--security-mode `<security_mode>`
The security mode to use for a DC/OS Enterprise cluster. This overrides any security mode set in `--extra-config`.

Options disabled|permissive|strict

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

-c, --cluster-id `<cluster_id>`
A unique identifier for the cluster. Use the value "default" to use this cluster for other commands without specifying –cluster-id.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

Arguments

INSTALLER_URL
Required argument

Environment variables

DCOS_LICENSE_KEY_PATH

Provide a default for *`--license-key`*

list

List all clusters.

```
minidcos aws list [OPTIONS]
```

Options

--aws-region `<aws_region>`
The AWS region to use. [default: us-west-2]

provision

Provision an AWS cluster to install DC/OS.

```
minidcos aws provision [OPTIONS]
```

Options

--custom-tag <custom_tag>
Add tags to EC2 instances in the format "<TAG_KEY>:<TAG_VALUE>".

--masters <masters>
The number of master nodes. [default: 1]

--agents <agents>
The number of agent nodes. [default: 1]

--public-agents <public_agents>
The number of public agent nodes. [default: 1]

--aws-instance-type <aws_instance_type>
The AWS instance type to use. [default: m4.large]

--aws-region <aws_region>
The AWS region to use. [default: us-west-2]

--linux-distribution <linux_distribution>
The Linux distribution to use on the nodes. [default: centos-7]

Options centos-7|coreos

--workspace-dir <workspace_dir>
Creating a cluster can use approximately 2 GB of temporary storage. Set this option to use a custom "workspace" for this temporary storage. See https://docs.python.org/3/library/tempfile.html#tempfile.gettempdir for details on the temporary directory location if this option is not set.

--copy-to-master <copy_to_master>
Files to copy to master nodes before installing DC/OS. This option can be given multiple times. Each option should be in the format /absolute/local/path:/remote/path.

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

-c, --cluster-id <cluster_id>
A unique identifier for the cluster. Use the value "default" to use this cluster for other commands without specifying –cluster-id.

--enable-selinux-enforcing
With this flag set, SELinux is set to enforcing before DC/OS is installed on the cluster.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

run

Run an arbitrary command on a node or multiple nodes.

To use special characters such as single quotes in your command, wrap the whole command in double quotes.

```
minidcos aws run [OPTIONS] NODE_ARGS...
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--dcos-login-uname <dcos_login_uname>
The username to set the `DCOS_LOGIN_UNAME` environment variable to. [default: bootstrapuser]

--dcos-login-pw <dcos_login_pw>
The password to set the `DCOS_LOGIN_PW` environment variable to. [default: deleteme]

--sync-dir <sync_dir>
The path to a DC/OS checkout. Part of this checkout will be synced to all master nodes before the command is run. The bootstrap directory is synced if the checkout directory variant matches the cluster variant.Integration tests are also synced.Use this option multiple times on a DC/OS Enterprise cluster to sync both DC/OS Enterprise and DC/OS Open Source tests.

-te, --test-env
With this flag set, environment variables are set and the command is run in the integration test directory. This means that "pytest" will run the integration tests.

--env <env>
Set environment variables in the format "<KEY>=<VALUE>"

--aws-region <aws_region>
The AWS region to use. [default: us-west-2]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--node <node>
A reference to a particular node to run the command on. This can be one of: The node's public IP address, The node's private IP address, the node's EC2 instance ID, a reference in the format "<role>_<number>". These details be seen with `minidcos aws inspect`. [default: master_0]

Arguments

NODE_ARGS
Required argument(s)

send-file

Send a file to a node or multiple nodes.

```
minidcos aws send-file [OPTIONS] SOURCE DESTINATION
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--node <node>
A reference to a particular node to run the command on. This can be one of: The node's public IP address, The node's private IP address, the node's EC2 instance ID, a reference in the format "<role>_<number>". These details be seen with `minidcos aws inspect`. [default: master_0]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--aws-region <aws_region>
The AWS region to use. [default: us-west-2]

Arguments

SOURCE
Required argument

DESTINATION
Required argument

sync

Sync files from a DC/OS checkout to master nodes.

This syncs integration test files and bootstrap files.

`DCOS_CHECKOUT_DIR` should be set to the path of clone of an open source DC/OS or DC/OS Enterprise repository.

By default the `DCOS_CHECKOUT_DIR` argument is set to the value of the `DCOS_CHECKOUT_DIR` environment variable.

If no `DCOS_CHECKOUT_DIR` is given, the current working directory is used.

This makes an assumption that all DC/OS Enterprise and DC/OS OSS integration tests are in the top level `packages/dcos-integration-test` directory.

```
minidcos aws sync [OPTIONS] [DCOS_CHECKOUT_DIR]
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--aws-region <aws_region>
The AWS region to use. [default: us-west-2]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

Arguments

DCOS_CHECKOUT_DIR
Optional argument

Environment variables

DCOS_CHECKOUT_DIR

Provide a default for *DCOS_CHECKOUT_DIR*

upgrade

Upgrade a cluster to a given version of DC/OS.

```
minidcos aws upgrade [OPTIONS] INSTALLER_URL
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--extra-config <extra_config>
The path to a file including DC/OS configuration YAML. The contents of this file will be added to add to a default configuration.

--variant <variant>
Choose the DC/OS variant. If the variant does not match the variant of the given installer URL, an error will occur. [required]

Options oss|enterprise

--aws-region <aws_region>
The AWS region to use. [default: us-west-2]

--workspace-dir <workspace_dir>
Creating a cluster can use approximately 2 GB of temporary storage. Set this option to use a custom "workspace" for this temporary storage. See https://docs.python.org/3/library/tempfile.html#tempfile.gettempdir for details on the temporary directory location if this option is not set.

--security-mode <security_mode>
The security mode to use for a DC/OS Enterprise cluster. This overrides any security mode set in `--extra-config`.

Options disabled|permissive|strict

--wait-for-dcos
Wait for DC/OS after creating the cluster. This is equivalent to using "minidcos aws wait" after this command. "minidcos aws wait" has various options available and so may be more appropriate for your use case.

--license-key <license_key>
This is ignored if using open source DC/OS. If using DC/OS Enterprise, this defaults to the value of the *DCOS_LICENSE_KEY_PATH* environment variable.

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

--genconf-dir <files_to_copy_to_genconf_dir>
Path to a directory that contains additional files for the DC/OS installer. All files from this directory will be copied to the "genconf" directory before running the DC/OS installer.

Arguments

INSTALLER_URL
Required argument

Environment variables

DCOS_LICENSE_KEY_PATH

Provide a default for *--license-key*

wait

Wait for DC/OS to start.

```
minidcos aws wait [OPTIONS]
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--superuser-username <superuser_username>
The superuser username is needed only on DC/OS Enterprise clusters. [default: bootstrapuser]

--superuser-password <superuser_password>
The superuser password is needed only on DC/OS Enterprise clusters. [default: deleteme]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

--aws-region <aws_region>
The AWS region to use. [default: us-west-2]

--enable-spinner, --no-enable-spinner
Whether to show a spinner animation. This defaults to true if stdout is a TTY.

web

Open the browser at the web UI.

Note that the web UI may not be available at first. Consider using `minidcos aws wait` before running this command.

```
minidcos aws web [OPTIONS]
```

Options

-c, --cluster-id <cluster_id>
The ID of the cluster to use. [default: default]

--aws-region <aws_region>
The AWS region to use. [default: us-west-2]

-v, --verbose
Use verbose output. Use this option multiple times for more verbose output.

CHAPTER

FOUR

INSTALLATION

There are multiple installation options. Choose one which works on your operating system.

Contents

4.1 Homebrew or Linuxbrew

Install Homebrew (macOS) or Linuxbrew (Linux). Then install the latest stable version:

```
$ brew install python
$ brew postinstall python
$ brew install https://raw.githubusercontent.com/|github-owner|/|github-repository|/master/|brewfile-
```

To upgrade from an older version, run the following command:

```
$ brew install python
$ brew postinstall python
$ brew upgrade https://raw.githubusercontent.com/|github-owner|/|github-repository|/master/|brewfile-
```

4.2 With Python (`pip`)

Requires Python 3.5.2+. To avoid interfering with your system's Python, we recommend using a virtualenv.

Check the Python version:

```
$ python3 --version
```

On Fedora, install Python development requirements:

```
$ dnf install -y git python3-devel
```

On Ubuntu, install Python development requirements:

```
$ apt update -y &&  apt install -y software-properties-common &&  apt upgrade -y python-apt &&  add-a
$ pip3 install setuptools
```

Install dependencies, preferably in a virtual environment. If you are not in a virtualenv, you may have to use `sudo` before the following command, or `--user` after `install`.

```
$ pip3 install --upgrade git+https://github.com/|github-owner|/|github-repository|.git@|release|
```

4.3 Linux Package

```
$ curl --fail -L https://github.com/|github-owner|/|github-repository|/releases/download/|release|/mi
```

4.4 Check that your system is ready to go

Run *minidcos docker*'s *doctor* to confirm that your system is ready to go for the *Docker*.

Run *minidcos aws*'s *doctor* to confirm that your system is ready to go for the *AWS*.

Run *minidcos vagrant*'s *doctor* to confirm that your system is ready to go for the *Vagrant*.

4.5 Uninstall

To uninstall the miniDC/OS, use one of the following methods.

For `pip` installations:

```
$ pip3 uninstall -y dcos-e2e
```

For Homebrew or Linuxbrew installations:

```
$ # --force uninstalls all versions which have been installed.
$ brew uninstall |brewfile-stem| --force
```

For installations from pre-built packages:

```
$ rm -f /usr/local/bin/minidcos
```

CHAPTER

FIVE

REFERENCE

5.1 Installation

There are multiple installation options. Choose one which works on your operating system.

Contents

5.1.1 Homebrew or Linuxbrew

Install Homebrew (macOS) or Linuxbrew (Linux). Then install the latest stable version:

```
$ brew install python
$ brew postinstall python
$ brew install https://raw.githubusercontent.com/|github-owner|/|github-repository|/master/|brewfile-
```

To upgrade from an older version, run the following command:

```
$ brew install python
$ brew postinstall python
$ brew upgrade https://raw.githubusercontent.com/|github-owner|/|github-repository|/master/|brewfile-
```

5.1.2 With Python (`pip`)

Requires Python 3.5.2+. To avoid interfering with your system's Python, we recommend using a virtualenv.

Check the Python version:

```
$ python3 --version
```

On Fedora, install Python development requirements:

```
$ dnf install -y git python3-devel
```

On Ubuntu, install Python development requirements:

```
$ apt update -y &&  apt install -y software-properties-common &&  apt upgrade -y python-apt &&  add-a
$ pip3 install setuptools
```

Install dependencies, preferably in a virtual environment. If you are not in a virtualenv, you may have to use `sudo` before the following command, or `--user` after `install`.

```
$ pip3 install --upgrade git+https://github.com/|github-owner|/|github-repository|.git@|release|
```

5.1.3 Linux Package

```
$ curl --fail -L https://github.com/|github-owner|/|github-repository|/releases/download/|release|/mi
```

5.1.4 Check that your system is ready to go

Run *minidcos docker*'s *doctor* to confirm that your system is ready to go for the *Docker*.

Run *minidcos aws*'s *doctor* to confirm that your system is ready to go for the *AWS*.

Run *minidcos vagrant*'s *doctor* to confirm that your system is ready to go for the *Vagrant*.

5.2 Versioning, Support and API Stability

miniDC/OS aims to work with DC/OS OSS and DC/OS Enterprise `master` branches. These are moving targets. For this reason, CalVer is used as a date at which the repository is last known to have worked with DC/OS OSS and DC/OS Enterprise is the main versioning use.

As well as `master`, miniDC/OS supports the following versions of DC/OS:

- DC/OS 1.11
- DC/OS 1.10
- DC/OS 1.9 (limited support, see *DC/OS 1.9 and below*)

Other versions may work but are not tested.

See GitHub for releases.

There is no guarantee of API stability at this point. All backwards incompatible changes will be documented in the *Changelog*.

5.2.1 DC/OS 1.9 and below

Installers for DC/OS 1.9 and below require a version of `sed` that is not compatible with the BSD sed that ships with macOS. *minidcos docker*'s *doctor* command includes a check for compatible `sed` versions.

Modify the installer

The following command replaces an installer named `dcos_generate_config.sh` with a slightly different installer that works with the default `sed` on macOS.

```
$ sed      -e 'H;1h;$!d;x'      -e "s/sed '0,/sed '1,/"      dcos_generate_config.sh > dcos_generate_
$ mv dcos_generate_config.sh.bak dcos_generate_config.sh
```

Change the local version of `sed`

It is possible to use unmodified installers if we use `GNU sed` as the system's default `sed`. This may have unforeseen side-effects. This requires Homebrew to be installed.

```
$ brew install gnu-sed --with-default-names
```

5.3 Contributing

Contributions to this repository must pass tests and linting.

Contents

5.3.1 Install Contribution Dependencies

Requires Python 3.5.2+. To avoid interfering with your system's Python, we recommend using a virtualenv.

Check the Python version:

```
$ python3 --version
```

On Fedora, install Python development requirements:

```
$ dnf install -y git python3-devel
```

On Ubuntu, install Python development requirements:

```
$ apt update -y &&  apt install -y software-properties-common &&  apt upgrade -y python-apt &&  add-a
$ pip3 install setuptools
```

Install dependencies, preferably in a virtual environment. If you are not in a virtualenv, you may have to use `sudo` before the following command, or `--user` after `install`.

```
$ pip3 install --editable '.[dev]'
```

Optionally install the following tools for linting and interacting with Travis CI:

```
$ gem install travis --no-rdoc --no-ri
```

Spell checking requires `enchant`. This can be installed on macOS, for example, with Homebrew:

```
$ brew install enchant
```

and on Ubuntu with `apt`:

```
$ apt install -y enchant
```

Linting Bash requires shellcheck: This can be installed on macOS, for example, with Homebrew:

```
$ brew install shellcheck
```

and on Ubuntu with `apt`:

```
$ apt-get install -y shellcheck
```

5.3.2 Linting

Install Contribution Dependencies.

Run lint tools:

```
$ make lint
```

These can be run in parallel with:

```
$ make lint --jobs --output-sync=target
```

To fix some lint errors, run the following:

```
$ make fix-lint
```

5.3.3 Tests for this package

To run the full test suite, set environment variables for DC/OS Enterprise installer URLs:

```
$ export EE_MASTER_INSTALLER_URL=https://...
$ export EE_1_9_INSTALLER_URL=https://...
$ export EE_1_10_INSTALLER_URL=https://...
$ export EE_1_11_INSTALLER_URL=https://...
$ export EE_1_12_INSTALLER_URL=https://...
$ export EE_1_13_INSTALLER_URL=https://...
```

Download dependencies which are used by the tests:

```
$ python admin/download_installers.py
```

A license key is required for some tests:

```
$ cp /path/to/license-key.txt /tmp/license-key.txt
```

Run `pytest`:

```
$ pytest
```

To run the tests concurrently, use pytest-xdist. For example:

```
$ pytest -n 2
```

5.3.4 Documentation

Run the following commands to build and open the documentation:

```
$ make docs
$ make open-docs
```

5.3.5 Reviews

Ask Adam Dangoor if you are unsure who to ask for help from.

5.3.6 CI

Linting and some tests are run on Travis CI. See `.travis.yml` for details on the limitations. To check if a new change works on CI, unfortunately it is necessary to change `.travis.yml` to run the desired tests.

Most of the CLI functionality is not covered by automated tests. Changes should take this into consideration.

Rotating license keys

DC/OS Enterprise requires a license key. Mesosphere uses license keys internally for testing, and these expire regularly. A license key is encrypted and used by the Travis CI tests.

To update this link use the following command, after setting the `LICENSE_KEY_CONTENTS` environment variable.

This command will affect all builds and not just the current branch.

We do not use encrypted secret files in case the contents are shown in the logs.

We do not add an encrypted environment variable to `.travis.yml` because the license is too large.

```
$ travis env set --repo |github-owner|/|github-repository| LICENSE_KEY_CONTENTS $LICENSE_KEY_CONTENTS
```

Updating the DC/OS Enterprise installer links

Private links to DC/OS Enterprise installers are used by Travis CI.

To update these links use the following commands, after setting the following environment variables:

- `EE_MASTER_INSTALLER_URL`
- `EE_1_9_INSTALLER_URL`
- `EE_1_10_INSTALLER_URL`
- `EE_1_11_INSTALLER_URL`
- `EE_1_12_INSTALLER_URL`
- `EE_1_13_INSTALLER_URL`

```
$ travis env set --repo |github-owner|/|github-repository| EE_MASTER_INSTALLER_URL $EE_MASTER_INSTALL
$ travis env set --repo |github-owner|/|github-repository| EE_1_9_INSTALLER_URL $EE_1_9_INSTALLER_URL
$ travis env set --repo |github-owner|/|github-repository| EE_1_10_INSTALLER_URL $EE_1_10_INSTALLER_U
$ travis env set --repo |github-owner|/|github-repository| EE_1_11_INSTALLER_URL $EE_1_11_INSTALLER_U
$ travis env set --repo |github-owner|/|github-repository| EE_1_12_INSTALLER_URL $EE_1_12_INSTALLER_U
$ travis env set --repo |github-owner|/|github-repository| EE_1_13_INSTALLER_URL $EE_1_13_INSTALLER_U
```

Updating the Amazon Web Services credentials

Private credentials for Amazon Web Services are used by Travis CI.

To update the credentials use the following commands, after setting the following environment variables:

- `AWS_ACCESS_KEY_ID`
- `AWS_SECRET_ACCESS_KEY`

```
$ travis env set --repo |github-owner|/|github-repository| AWS_ACCESS_KEY_ID $AWS_ACCESS_KEY_ID
$ travis env set --repo |github-owner|/|github-repository| AWS_SECRET_ACCESS_KEY $AWS_SECRET_ACCESS_K
```

Currently credentials are taken from the OneLogin Secure Notes note `dcos-e2e integration testing AWS credentials`.

Parallel builders

Travis CI has a maximum test run time of 50 minutes. In order to avoid this and to see failures faster, we run multiple builds per commit. We run almost one builder per test. Some tests are grouped as they can run quickly.

5.3.7 Goals

Avoid flakiness

For timeouts, err on the side of a much longer timeout than necessary.

Do not access the web while running tests.

Parallelizable Tests

The tests in this repository and using this harness are slow. This harness must not get in the way of parallelization efforts.

Logging

End to end tests are notoriously difficult to get meaning from. To help with this, an "excessive logging" policy is used here.

Robustness

Narrowing down bugs from end to end tests is hard enough without dealing with the framework's bugs. This repository aims to maintain high standards in terms of coding quality and quality enforcement by CI is part of that.

5.3.8 Version Policy

This repository aims to work with DC/OS OSS and DC/OS Enterprise `master` branches. These are moving targets. For this reason, CalVer is used as a date at which the repository is last known to have worked with DC/OS OSS and DC/OS Enterprise is the main versioning use.

5.3.9 Release Process

See *Release Process*.

5.3.10 Updating vendored packages

Various repositories, such as DC/OS Test Utils and DC/OS Launch are vendored in this repository. To update DC/OS Test Utils or DC/OS Launch:

Update the SHAs in `admin/update_vendored_packages.py`.

The following creates a commit with changes to the vendored packages:

```
$ python admin/update_vendored_packages.py
```

5.4 Changelog

Contents

- *2019.04.18.0*
- *2019.04.08.1*
- *2019.04.08.0*
- *2019.04.02.1*
- *2019.04.02.0*
- *2019.03.28.0*
- *2019.03.27.0*
- *2019.03.23.0*
- *2019.03.22.1*
- *2019.03.22.0*
- *2019.03.13.0*
- *2019.02.17.1*
- *2019.02.17.0*
- *2019.02.16.0*
- *2019.01.29.1*
- *2019.01.29.0*
- *2019.01.27.1*
- *2019.01.27.0*
- *2019.01.10.0*
- *2019.01.05.0*
- *2018.12.10.0*
- *2018.12.05.0*
- *2018.12.01.1*
- *2018.12.01.0*
- *2018.11.22.0*
- *2018.11.21.0*
- *2018.11.20.1*
- *2018.11.20.0*
- *2018.11.16.2*
- *2018.11.16.1*
- *2018.11.16.0*
- *2018.11.09.1*
- *2018.11.09.0*
- *2018.11.07.1*
- *2018.11.07.0*

- *2018.06.28.0*
- *2018.06.26.0*
- *2018.06.20.0*
- *2018.06.18.0*
- *2018.06.14.1*
- *2018.06.14.0*
- *2018.06.12.1*
- *2018.06.12.0*
- *2018.06.05.0*
- *2018.05.29.0*
- *2018.05.24.2*
- *2018.05.24.1*
- *2018.05.21.0*
- *2018.05.17.0*
- *2018.05.15.0*
- *2018.05.14.0*
- *2018.05.10.0*
- *2018.05.02.0*
- *2018.04.30.2*
- *2018.04.29.0*
- *2018.04.25.0*
- *2018.04.19.0*
- *2018.04.11.0*
- *2018.04.02.1*
- *2018.04.02.0*
- *2018.03.26.0*
- *2018.03.07.0*
- *2018.02.28.0*
- *2018.02.27.0*
- *2018.02.23.0*
- *2018.01.25.0*
- *2018.01.22.0*
- *2017.12.11.0*
- *2017.12.08.0*
- *2017.11.29.0*

5.4.1 Next

5.4.2 2020.08.14.0

- Updated `google-api-python-client` to `v1.7.12`.
- Added support for CentOS 8 and Flatcar when using Docker.

5.4.3 2020.05.26.0

- Changed the default version of `dcos-vagrant-box` to `0.10`.
- Updated dependencies.
- Drop support for RHEL in `minidcos aws`.

5.4.4 2019.10.11.0

- Update vendored `dcos-test-utils` dependency.

5.4.5 2019.10.10.0

- Bump `dcos-test-utils` dependency.

5.4.6 2019.08.28.0

- Fix issue which prevented CoreOS nodes starting on Docker.

5.4.7 2019.06.19.0

- Renamed `Cluster.run_integration_tests`'s `test_host` parameter to `node`.
- Renamed `Cluster.run_integration_tests`'s `pytest_command` parameter to `args`.
- Renamed `Cluster.run_integration_tests` to `Cluster.run_with_test_environment`.
- Split `Cluster` and `Node` `upgrade` and `install_dcos` functions into `*_dcos_from_[path|url]` functions.
- Enable larger log sizes and lower memory uses by moving `Docker` backend logs to the host disk.

5.4.8 2019.06.15.0

- Add `minidcos aws destroy` and `minidcos aws destroy-list` commands.

5.4.9 2019.06.10.0

- Fix error "No module named 'keyring.util.escape'".
- Added `Cluster.upgrade_dcos` and `minidcos upgrade` commands.
- Replaced `Cluster.install_dcos_from_url` and `Cluster.install_dcos_from_path` with `Cluster.install_dcos` which takes a URL or Path.
- Replaced `Node.install_dcos_from_url` and `Node.install_dcos_from_path` with `Node.install_dcos` which takes a URL or Path.
- Fix `minidcos inspect` commands for legacy versions of DC/OS.
- Add options to enable or disable spinner animations in `minidcos`.

5.4.10 2019.06.07.0

- Get DC/OS build information from a `Node` object after installation.
- Added option to `minidcos aws create` and `minidcos aws provision` to choose the AWS instance type.
- Changed the default version of Docker installed on `minidcos docker` clusters to `18.06.3-ce`.
- Added `Node.upgrade_dcos_from_path`.
- Added `minidcos docker upgrade`.

5.4.11 2019.06.03.0

- Added options to choose the amount of memory given to each VM.
- Fixed a bug which prevented `minidcos vagrant` from working when a VM existed with a space in the name.
- Fixed a bug which prevented `minidcos vagrant` from working in some situations when the `$HOME` environment variable is not set.

5.4.12 2019.05.24.1

- Add a `minidcos docker doctor` check which fails when using `Boot2Docker`.

5.4.13 2019.05.24.0

5.4.14 2019.05.23.1

- Fix issue with `minidcos vagrant` which prevented node access via SSH.
- Change `minidcos` default credentials for DC/OS Enterprise clusters from `admin/admin` to `bootstrapuser/deleteme`.

5.4.15 2019.05.23.0

- Download a file or directory from a `Node`.
- Improve efficiency of installing DC/OS with `create` on `minidcos docker` and `minidcos aws`.
- Allow the use of a `MINIDCOS_NODE_DOCKER_VERSION` environment variable to set the version of Docker inside `minidcos docker` nodes.

5.4.16 2019.04.29.0

- Remove use of `select` which is not supported on Windows.
- `minidcos docker clean` will no longer clean up containers which are started from now on by the tooling to create a custom macOS network.

5.4.17 2019.04.25.0

5.4.18 2019.04.23.1

- The `wait_for_dcos_oss` and `wait_for_dcos_ee` methods no longer log output of node poststart check command run.
- The `Node.run` method logs the command that is going to execute with `debug` level if `output` is configured to `LOG_AND_CAPTURE`.
- The `Node.run` method no longer logs `stderr` when `Output.CAPTURE` is used.
- The `Node.run` method no longer merges `stderr` into `stdout` when `Output.LOG_AND_CAPTURE` is used.

5.4.19 2019.04.23.0

- The library no longer configures logger handler. Applications using `dcos_e2e` library that were relying on logging being printed to stdout should configure `logging` on its own.

5.4.20 2019.04.18.2

5.4.21 2019.04.18.1

- Add new commands to `minidcos vagrant` and `minidcos aws` to provision a bare cluster (`provision`) and install DC/OS on a bare cluster (`install`).

5.4.22 2019.04.18.0

- Improve the spinner while waiting for `minidcos` commands.
- Add `send-file` commands to `minidcos` subcommands.
- Remove `--env` option on `minidcos docker inspect`.
- Custom backends must now specify the base config in the `ClusterBackend` rather than the `ClusterManager`.
- Add new commands to `minidcos docker` to provision a bare cluster (`provision`) and install DC/OS on a bare cluster (`install`).

5.4.23 2019.04.08.1

- Add `minidcos vagrant clean` command to clean up left over VMs.

5.4.24 2019.04.08.0

- Allow multiple `--node` options on `run` commands to run a command on multiple nodes.
- `minidcos vagrant list` will now not show clusters with aborted VMs. A VM is aborted for example if the host is shut down.

5.4.25 2019.04.02.1

- Make various `minidcos` commands faster by using caching.

5.4.26 2019.04.02.0

- Remove warnings about YAML when running `minidcos vagrant`.

5.4.27 2019.03.28.0

- Change names of install functions for custom backends to remove `_with_bootstrap_node`.

5.4.28 2019.03.27.0

- Allow login after `minidcos docker wait` on DC/OS OSS 1.13.

5.4.29 2019.03.23.0

5.4.30 2019.03.22.1

5.4.31 2019.03.22.0

5.4.32 2019.03.13.0

5.4.33 2019.02.17.1

5.4.34 2019.02.17.0

5.4.35 2019.02.16.0

- Mount `/sys/fs/cgroup` into Docker agents by default.
- Add options to the CLI and library to disable mounting `/sys/fs/cgroup`.

5.4.36 2019.01.29.1

5.4.37 2019.01.29.0

5.4.38 2019.01.27.1

- Stop mounting `/sys/fs/cgroup` into Docker agents.

5.4.39 2019.01.27.0

- Add more `minidcos docker doctor` checks.

5.4.40 2019.01.10.0

- Fix issue where "RuntimeError: cannot join current thread" was shown.

5.4.41 2019.01.05.0

- Backwards incompatible change: Move `ClusterBackend` and `ClusterManager` to `dcos_e2e.base_classes`.

5.4.42 2018.12.10.0

5.4.43 2018.12.05.0

- Limit UUID in the cluster ID to 5 characters to avoid problems with Docker.

5.4.44 2018.12.01.1

5.4.45 2018.12.01.0

- Ship type hints for other packages to use.

5.4.46 2018.11.22.0

- Allow `-h` instead of `--help` on all CLI commands.

5.4.47 2018.11.21.0

5.4.48 2018.11.20.1

- Allow multiple `--sync-dir` options to be given to `run` commands.

5.4.49 2018.11.20.0

- Rename `build_artifact` and related variables to "installer".
- If syncing a DC/OS OSS repository to a DC/OS Enterprise cluster, only Open Source tests are synced.

5.4.50 2018.11.16.2

5.4.51 2018.11.16.1

- Backwards incompatible change: Changed CLI commands from `dcos-docker` to `minidcos docker` and alike.

5.4.52 2018.11.16.0

- Add a `dcos-docker doctor` check for systemd.
- Add a progress bar for `doctor` commands.
- Log subprocess output unicode characters where possible.

5.4.53 2018.11.09.1

- Backwards incompatible change: Change `--no-test-env` to `--test-env` on `run` commands, with the opposite default.

5.4.54 2018.11.09.0

- Fix an issue which caused incompatible version errors between `keyring` and `SecretStore` dependencies.

5.4.55 2018.11.07.1

5.4.56 2018.11.07.0

- Add `dcos-docker create-loopback-sidecar` and `dcos-docker destroy-loopback-sidecar` commands to provide unformatted block devices to DC/OS.
- Add `dcos-docker clean` command to clean left over artifacts.
- Backwards incompatible change: Changed names of VPN containers on macOS.

5.4.57 2018.10.17.1

5.4.58 2018.10.17.0

- Fix an issue which stopped the SSH transport from working on CLIs.

5.4.59 2018.10.16.0

- Remove `log_output_live` parameters on various functions in favor of new `output` options.
- **`Node.__init__`'s `ssh_key_path` parameter now expects a path to an SSH key file with specific permissions.** See the documentation for this class for details.

5.4.60 2018.10.13.0

5.4.61 2018.10.12.2

5.4.62 2018.10.12.1

5.4.63 2018.10.12.0

- The `docker-exec` transport uses interactive mode only when running in a terminal.

5.4.64 2018.10.11.3

5.4.65 2018.10.11.2

5.4.66 2018.10.11.1

5.4.67 2018.10.11.0

- Show full path on `download-artifact` downloads.
- Default to downloading to the current directory for `download-artifact` downloads.
- Use a TTY on CLI run commands only if Stdin is a TTY.

5.4.68 2018.10.10.0

- Fix issues which stopped pre-built Linux binaries from working.

5.4.69 2018.09.25.0

- `wait_for_dcos_oss` and `wait_for_dcos_ee` now raise a custom `DCOSTimeoutError` if DC/OS has not started within one hour.

5.4.70 2018.09.06.0

- The `--variant` option is now required for the `dcos-aws` CLI.
- Added the ability to install on Linux from a pre-built binary.
- Add the ability to do a release to a fork.

5.4.71 2018.08.31.0

- Fix using macOS with no custom network.

5.4.72 2018.08.28.0

- Support for CoreOS on the AWS backend.
- Fix an issue which prevented the Vagrant backend from working.

5.4.73 2018.08.22.0

- Improve diagnostics when creating a Docker-backed cluster with no running Docker daemon.

5.4.74 2018.08.13.0

- Add instructions for uninstalling DC/OS E2E.

5.4.75 2018.08.03.0

- Pin `msrestazure` pip dependency to specific version to avoid dependency conflict.

5.4.76 2018.07.31.0

- Add a `dcos-docker doctor` check that relevant Docker images can be built.

5.4.77 2018.07.30.0

- Add Red Hat Enterprise Linux 7.4 support to the AWS backend.

5.4.78 2018.07.27.0

- Fix bug which meant that a user could not log in after `dcos-docker wait` on DC/OS Open Source clusters.
- Backwards incompatible change: Remove `files_to_copy_to_installer` from `Cluster.__init__` and add `files_to_copy_to_genconf_dir` as an argument to `Cluster.install_dcos_from_path` as well as `Cluster.install_dcos_from_url`.
- Add `files_to_copy_to_genconf_dir` as an argument to `Node.install_dcos_from_path` and `Node.install_dcos_from_url`.

5.4.79 2018.07.25.0

- Add the capability of sending a directory to a `Node` via `Node.send_file`.
- Add `ip_detect_path` to the each `ClusterBackend` as a property and to each install DC/OS function as a parameter.

5.4.80 2018.07.23.1

5.4.81 2018.07.23.0

- Add an initial `dcos-aws` CLI.

5.4.82 2018.07.22.1

- Add `dcos-docker download-artifact` and `dcos-vagrant download-artifact`.

5.4.83 2018.07.22.0

- Add `verbose` option to multiple commands.

5.4.84 2018.07.16.0

- Add `virtualbox_description` parameter to the `Vagrant` backend.
- Change the default transport for the Docker backend to `DOCKER_EXEC`.

5.4.85 2018.07.15.0

- Add a `--one-master-host-port-map` option to `dcos-docker create`.

5.4.86 2018.07.10.0

- Execute `node-poststart` checks in `Cluster.wait_for_dcos` and `Cluster.wait_for_dcos_ee`.
- Add `dcos-vagrant doctor` checks.

5.4.87 2018.07.03.5

- Add a `--network` option to the `dcos-docker` CLI.

5.4.88 2018.07.03.0

- Add a `dcos-vagrant` CLI.

5.4.89 2018.07.01.0

- Renamed Homebrew formula. To upgrade from a previous version, follow Homebrew's linking instructions after upgrade instructions.

5.4.90 2018.06.30.0

- Add a `Vagrant` backend.

5.4.91 2018.06.28.2

- Add a `aws_instance_type` parameter to the `AWS` backend.

5.4.92 2018.06.28.0

- Compare `Node` objects based on the `public_ip_address` and `private_ip_address`.

5.4.93 2018.06.26.0

- Add a `network` parameter to the `Docker` backend.

5.4.94 2018.06.20.0

- Add platform-independent DC/OS installation method from `Path` and URL on `Node`.

5.4.95 2018.06.18.0

- Add `dcos-docker doctor` check for a version conflict between systemd and Docker.
- Allow installing DC/OS by a URL on the Docker backend, and a cluster `from_nodes`.

5.4.96 2018.06.14.1

- Add `Cluster.remove_node`.

5.4.97 2018.06.14.0

- Add Ubuntu support to the Docker backend.
- Add `aws_key_pair` parameter to the AWS backend.
- Fix Linuxbrew installation on Ubuntu.

5.4.98 2018.06.12.1

- Add a `--wait` flag to `dcos-docker create` to also wait for the cluster.

5.4.99 2018.06.12.0

- `dcos-docker create` now creates clusters with the `--cluster-id` "default" by default.

5.4.100 2018.06.05.0

- Change `Node.default_ssh_user` to `Node.default_user`.
- Add a `docker exec` transport to `Node` operations.
- Add a `--transport` options to multiple `dcos-docker` commands.

5.4.101 2018.05.29.0

- Do not pin `setuptools` to an exact version.

5.4.102 2018.05.24.2

- Add `--env` option to `dcos-docker run`.

5.4.103 2018.05.24.1

- Make `xfs_info` available on nodes, meaning that preflight checks can be run on nodes with XFS.
- Fix `dcos-docker doctor` for cases where `df` produces very long results.

5.4.104 2018.05.21.0

- Show a formatted error rather than a traceback if Docker cannot be connected to.
- Custom backends' must now implement a `base_config` method.
- Custom backends' installation methods must now take `dcos_config` rather than `extra_config`.
- `Cluster.install_dcos_from_url` and `Cluster.install_dcos_from_path` now take `dcos_config` rather than `extra_config`.

5.4.105 2018.05.17.0

- Add a `--variant` option to `dcos-docker create` to speed up cluster creation.

5.4.106 2018.05.15.0

- Add a `test_host` parameter to `Cluster.run_integration_tests`.
- Add the ability to specify a node to use for `dcos-docker run`.

5.4.107 2018.05.14.0

- Show IP address in `dcos-docker inspect`.

5.4.108 2018.05.10.0

- Expose the SSH key location in `dcos-docker inspect`.
- Make network created by `setup-mac-network` now survives restarts.

5.4.109 2018.05.02.0

- Previously not all volumes were destroyed when destroying a cluster from the CLI or with the `Docker` backend. This has been resolved. To remove dangling volumes from previous versions, use `docker volume prune`.
- Backwards incompatible change: `mount` parameters to `Docker.__init__` now take a `list` of `docker.types.Mounts`.
- Docker version 17.06 or later is now required for the CLI and for the `Docker` backend.

5.4.110 2018.04.30.2

- Added `dcos-docker destroy-mac-network` command.
- Added a `--force` parameter to `dcos-docker setup-mac-network` to override files and containers.

5.4.111 2018.04.29.0

- Added `dcos-docker setup-mac-network` command.

5.4.112 2018.04.25.0

- Logs from dependencies are no longer emitted.
- The `dcos-docker` CLI now gives more feedback to let you know that things are happening.

5.4.113 2018.04.19.0

- The AWS backend now supports DC/OS 1.9.
- The Docker backend now supports having custom mounts which apply to all nodes.
- Add `custom-volume` parameter (and similar for each node type) to `dcos-docker create`.

5.4.114 2018.04.11.0

- Add an AWS backend to the library.
- Add ability to control which labels are added to particular node types on the `Docker` backend.
- Add support for Ubuntu on the `Docker` backend.

5.4.115 2018.04.02.1

- Add a new `dcos-docker doctor` check for suitable `sed` for DC/OS 1.9.
- Support `cluster.run_integration_tests` on DC/OS 1.9.

5.4.116 2018.04.02.0

- Add support for DC/OS 1.9 on Linux hosts.
- `dcos-docker doctor` returns a status code of `1` if there are any errors.
- Add a new `dcos-docker doctor` check for free space in the Docker root directory.

5.4.117 2018.03.26.0

- Add a `dcos-docker doctor` check that a supported storage driver is available.
- Fix error with using Docker version *v17.12.1-ce* inside Docker nodes.
- Fix race condition between installing DC/OS and SSH starting.
- Remove support for Ubuntu on the Docker backend.

5.4.118 2018.03.07.0

- Fix public agents on DC/OS 1.10.
- Remove options to use Fedora and Debian in the `Docker` backend nodes.
- Fix the Ubuntu distribution on the `Docker` backend.
- Add support for Docker `17.12.1-ce` on nodes in the `Docker` backend.
- Exceptions in `create` in the CLI point towards the `doctor` command.
- Removed a race condition in the `doctor` command.
- `dcos-docker run` now exits with the return code of the command run.
- `dcos-docker destroy-list` is a new command and `dcos-docker destroy` now adheres to the common semantics of the CLI.

5.4.119 2018.02.28.0

- Add `Vagrantfile` to run DC/OS E2E in a virtual machine.
- Add instructions for running DC/OS E2E on Windows.
- Allow relative paths for the build artifact.

5.4.120 2018.02.27.0

- Backwards incompatible change: Move `default_ssh_user` parameter from `Cluster` to `Node`. The `default_ssh_user` is now used for `Node.run`, `Node.popen` and `Node.send_file` if `user` is not supplied.

5.4.121 2018.02.23.0

- Add `linux_distribution` parameter to the `Docker` backend.
- Add support for CoreOS in the `Docker` backend.
- Add `docker_version` parameter to the `Docker` backend.
- The fallback Docker storage driver for the `Docker` backend is now `aufs`.
- Add `storage_driver` parameter to the `Docker` backend.
- Add `docker_container_labels` parameter to the `Docker` backend.
- Logs are now less cluttered with escape characters.
- Documentation is now on Read The Docs.
- Add a Command Line Interface.
- Vendor `dcos_test_utils` so `--process-dependency-links` is not needed.
- Backwards incompatible change: `Cluter`'s `files_to_copy_to_installer` argument is now a `List` of `Tuples` rather than a `Dict`.
- Add a `tty` option to `Node.run` and `Cluster.run_integration_tests`.

5.4.122 2018.01.25.0

- Backwards incompatible change: Change the default behavior of `Node.run` and `Node.popen` to quote arguments, unless a new `shell` parameter is `True`. These methods now behave similarly to `subprocess.run`.
- Add custom string representation for `Node` object.
- Bump `dcos-test-utils` for better diagnostics reports.

5.4.123 2018.01.22.0

- Expose the `public_ip_address` of the SSH connection and the `private_ip_address` of its DC/OS component on `Node` objects.
- Bump `dcos-test-utils` for better diagnostics reports.

5.4.124 2017.12.11.0

- Replace the extended `wait_for_dcos_ee` timeout with a preceding `dcos-diagnostics` check.

5.4.125 2017.12.08.0

- Extend `wait_for_dcos_ee` timeout for waiting until the DC/OS CA cert can be fetched.

5.4.126 2017.11.29.0

- Backwards incompatible change: Introduce separate `wait_for_dcos_oss` and `wait_for_dcos_ee` methods. Both methods improve the boot process waiting time for the corresponding DC/OS version.
- Backwards incompatible change: `run_integration_tests` now requires users to call `wait_for_dcos_oss` or `wait_for_dcos_ee` beforehand.

5.4.127 2017.11.21.0

- Remove `ExistingCluster` backend and replaced it with simpler `Cluster.from_nodes` method.
- Simplified the default configuration for the Docker backend. Notably this no longer contains a default `superuser_username` or `superuser_password_hash`.
- Support `custom_agent_mounts` and `custom_public_agent_mounts` on the Docker backend.

5.4.128 2017.11.15.0

- Remove `destroy_on_error` and `destroy_on_success` from `Cluster`. Instead, avoid using `Cluster` as a context manager to keep the cluster alive.

5.4.129 2017.11.14.0

- Backwards incompatible change: Rename `DCOS_Docker` backend to `Docker` backend.
- Backwards incompatible change: Replace `generate_config_path` with `build_artifact` that can either be a `Path` or a HTTP(S) URL string. This allows for supporting installation methods that require build artifacts to be downloaded from a HTTP server.
- Backwards incompatible change: Remove `run_as_root`. Instead require a `default_ssh_user` for backends to `run` commands over SSH on any cluster `Node` created with this backend.
- Backwards incompatible change: Split the DC/OS installation from the ClusterManager `__init__` procedure. This allows for installing DC/OS after `Cluster` creation, and therefore enables decoupling of transferring files ahead of the installation process.
- Backwards incompatible change: Explicit distinction of installation methods by providing separate methods for `install_dcos_from_path` and `install_dcos_from_url` instead of inspecting the type of `build_artifact`.
- Backwards incompatible change: `log_output_live` is no longer an attribute of the `Cluster` class. It may now be passed separately as a parameter for each output-generating operation.

5.4.130 2017.11.02.0

- Added `Node.send_file` to allow files to be copied to nodes.
- Added `custom_master_mounts` to the DC/OS Docker backend.
- Backwards incompatible change: Removed `files_to_copy_to_masters`. Instead, use `custom_master_mounts` or `Node.send_file`.

5.4.131 2017.10.04.0

- Added Apache2 license.
- Repository moved to `https://github.com/dcos/dcos-e2e`.
- Added `run`, which is similar to `run_as_root` but takes a `user` argument.
- Added `popen`, which can be used for running commands asynchronously.

5.4.132 2017.08.11.0

- Fix bug where `Node reprs` were put into environment variables rather than IP addresses. This prevented some integration tests from working.

5.4.133 2017.08.08.0

- Fixed issue which prevented `files_to_copy_to_installer` from working.

5.4.134 2017.08.05.0

- The Enterprise DC/OS integration tests now require environment variables describing the IP addresses of the cluster. Now passes these environment variables.

5.4.135 2017.06.23.0

- Wait for 5 minutes after diagnostics check.

5.4.136 2017.06.22.0

- Account for the name of `3dt` having changed to `dcos-diagnostics`.

5.4.137 2017.06.21.1

- Support platforms where `$HOME` is set as `/root`.
- `Cluster.wait_for_dcos` now waits for CA cert to be available.

5.4.138 2017.06.21.0

- Add ability to specify a workspace.
- Fixed issue with DC/OS Docker files not existing in the repository.

5.4.139 2017.06.20.0

- Vendor DC/OS Docker so a path is not needed.
- If `log_output_live` is set to `True` for a `Cluster`, logs are shown in `wait_for_dcos`.

5.4.140 2017.06.19.0

- More storage efficient.
- Removed need to tell `Cluster` whether a cluster is an enterprise cluster.
- Removed need to tell `Cluster` the `superuser_password`.
- Added ability to set environment variables on remote nodes when running commands.

5.4.141 2017.06.15.0

- Initial release.

5.5 Release Process

5.5.1 Outcomes

- A new `git` tag available to install.
- A release on GitHub.
- An updated Homebrew recipe.
- A changed Vagrantfile.
- Linux binaries.
- The new version title in the changelog.

5.5.2 Prerequisites

- `python3` on your `PATH` set to Python 3.5+.
- Docker available and set up for your user.
- `virtualenv`.
- Push access to this repository.
- Trust that `master` is ready and high enough quality for release. This includes the `Next` section in `CHANGELOG.rst` being up to date.

5.5.3 Perform a Release

1. Get a GitHub access token:

   Follow the GitHub instructions for getting an access token.

2. Set environment variables to GitHub credentials, e.g.:

   ```
   $ export GITHUB_TOKEN=75c72ad718d9c346c13d30ce762f121647b502414
   ```

3. Perform a release:

   ```
   $ export GITHUB_OWNER=|github-owner|
   $ curl https://raw.githubusercontent.com/"$GITHUB_OWNER"/dcos-e2e/master/admin/release.sh |
   ```

INDEX

Symbols

C

N

S