SAMPLE CHAPTER

Mesos IN ACTION

Roger Ignazio

FOREWORD BY Florian Leibert

MANNING

Mesos in Action

by Roger Ignazio

Chapter 3

brief contents

3 Setting up Mesos

This chapter covers

- Deployment considerations for development and production clusters
- Installing and configuring Mesos, ZooKeeper, and Docker
- Upgrading Mesos without downtime

Regardless of whether you're using a configuration management tool (such as Puppet, Chef, or Ansible), or using SSH or Fabric to execute commands and scripts on remote systems, it's important to understand how to deploy Mesos using publicly available packages, and how to compile it from source code for customizing your deployment or building your own packages.

This chapter walks you through the installation and configuration of the Mesos master and slaves, including Apache ZooKeeper for cluster coordination, and Docker for launching containers. You'll learn about the installation and configuration of Mesos and ZooKeeper for highly available production deployments as well as for installing everything on a single node for development purposes.

Configuration management with Puppet

A good configuration management strategy is key to any well-run datacenter. Many of the instructions included in this chapter—including the installation and configuration of Mesos, ZooKeeper, and Docker—can be performed using Puppet, an open source configuration management tool (and as of this writing, the most popular of the several configuration management tools in the configuration management space). Three Puppet modules in particular can help you automate and maintain the configuration of your Mesos cluster:

- https://forge.puppetlabs.com/deric/mesos
- https://forge.puppetlabs.com/deric/zookeeper
- https://forge.puppetlabs.com/garethr/docker

Because Puppet usage is best left to the official documentation or other books already available on the topic, this chapter doesn't cover it. You can find and download Puppet from most Linux distributions' package repositories or from puppetlabs.com.

Regardless of whether or not you're planning to use a configuration management tool for the deployment of your Mesos cluster, it's still a good idea to read through this chapter and understand how the various components depend on and interact with each other.

3.1 Deploying Mesos

When deploying any new technology or system, whether it's in development, staging, or production, it's always a good idea to understand as much as you can about it. This way, you're prepared when something inevitably goes awry. The next few sections teach you about the components that make up a Mesos cluster. You'll also learn a few things to take into consideration when deploying clusters in development and in production.

3.1.1 Mesos cluster components

Let's revisit the various components and how they communicate with each other. Regardless of whether or not you're attempting to deploy Mesos in a development environment or a production environment, a cluster deployment is made up of some (or all) of the following components:

- *Required—One or more Mesos masters.* (If the number of masters is greater than 1, this must be an odd number.)
- *Required—One or more Mesos slaves.* (Generally speaking, the more nodes in a cluster, the better.)
- *Optional—A ZooKeeper ensemble consisting of one or more machines.* Required only if deploying Mesos in a highly available configuration. (If the number of ZooKeeper nodes is greater than 1, this must be an odd number.)
- *Optional—Docker Engine running on each of the Mesos slaves.*

Depending on your intended purpose, you may need to consider additional information for development and production environments. The next couple of sections cover this topic.

3.1.2 Considerations for a development environment

When installing and configuring Mesos for development purposes, it's reasonable to deploy all of the components for a Mesos deployment on a single node, opting for a simpler deployment in lieu of a highly available one. For a development environment, you need to install, configure, and deploy the following components, in the following order:

1. A single instance of ZooKeeper. Note that this is optional, and needed only if your framework needs ZooKeeper for coordination or to maintain state (as needed by highly available frameworks).
2. The Mesos master service.
3. The Mesos slave service.
4. Docker Engine (optional).

When you reach the installation and configuration of Mesos later in this chapter, install all the components on a single machine. But it's worth noting that this use case is so common for development purposes that the team over at Mesosphere, Inc. has developed a project for just this purpose: Playa Mesos.

Introducing Playa Mesos

A popular way of distributing reproducible development environments between systems or team members is to use Vagrant (www.vagrantup.com). Playa Mesos is one such development environment created and maintained by Mesosphere as an easy way to provision a Mesos cluster on a single VM, and experiment with developing frameworks, running applications (via Marathon), and running scheduled tasks (via Chronos). Out of the box, it also includes a single instance of ZooKeeper, and Docker Engine, both preinstalled and preconfigured.

> **TIP** The Playa Mesos project can be found on GitHub at https://github.com/mesosphere/playa-mesos.

Assuming you already have a working Vagrant setup (which requires virtualization software such as VirtualBox, VMware Fusion, or VMware Workstation), getting up and running in a development environment is easy:

```
$ git clone https://github.com/mesosphere/playa-mesos
$ cd playa-mesos
$ vagrant up --provision
```

The same general components and concepts apply to production environments as well, but some additional considerations need to be taken in order to deploy the various components in a highly available manner. Let's take a look at this now.

3.1.3 Considerations for a production environment

This section contains some best practices to consider for deploying the Mesos cluster. It's worth noting, however, that this isn't an extensive guide for running Mesos in production. This section is intended to help you make certain provisioning decisions at this stage of the book. Chapter 6 provides more details about running Mesos in a production environment, including logging, monitoring, and access control.

PRODUCTION DEPLOYMENT OVERVIEW

For a production environment, you'll want a minimum of three Mesos masters and three servers making up the ZooKeeper ensemble. For development purposes, you can get away with a single Mesos master running a single instance of ZooKeeper, but note that you won't have any redundancy in the system.

In figure 3.1, you can visualize the multiple Mesos masters using ZooKeeper for coordination, and the various Mesos slaves (in this case, also running Docker) communicating with the leading Mesos master.

Although ZooKeeper is illustrated as a standalone service running separate from the Mesos masters, you'll opt for a slightly simpler deployment overall and install ZooKeeper alongside Mesos on each of the Mesos master machines. Don't worry about memorizing this graphic; I'll repeat it as you make your way through the installation and configuration of the cluster.

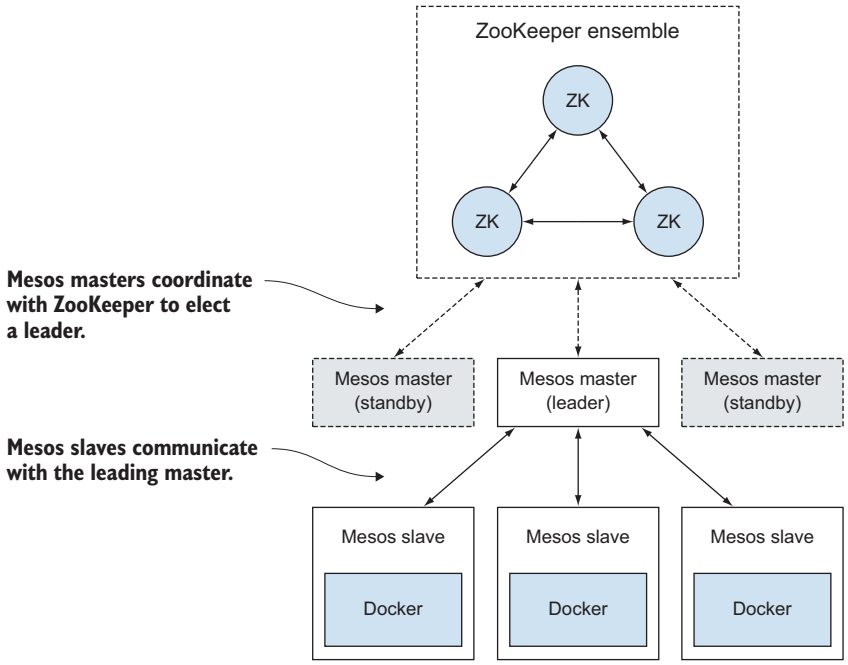


Figure 3.1 Components in a Mesos deployment

This almost goes without saying, but when deploying these services on dedicated hardware or using your virtualization or cloud provider of choice, be sure to account for redundancy at all hardware levels. If you're running in a physical datacenter, your Mesos masters and ZooKeeper servers should perhaps be placed in different racks, connected to different (or multiple) network switches, be connected to multiple power distribution units, and so forth. If you're running in a virtualized or cloud environment, ensure that the necessary policies are in place to keep the virtual machines running on different hypervisors or in different availability zones.

Considering that all of your cluster coordination will be happening through the Mesos masters and the ZooKeeper ensemble, you want to keep the single points of failure to a minimum. If you have multiple datacenters or a disaster recovery datacenter, you might even consider using them, assuming the network latency is low enough.

MESOS MASTERS

If you're planning to run a Mesos cluster spanned across multiple datacenters, it's a good idea to ensure that low network latency exists between each of the masters and datacenters. Otherwise, Mesos registry updates could fail if the leading master can't write to the standby masters' registries within the `registry_store_timeout` (default: five seconds). If you need to increase this time-out, you may consider increasing the `registry_fetch_timeout` (default: one minute) as well. Depending on your environment, you might want to consider running a separate Mesos cluster at each site and load-balancing across datacenters instead.

ZOOKEEPER ENSEMBLE

Considering that ZooKeeper is required for all coordination between Mesos masters, slaves, and frameworks, it goes without saying that it needs to be highly available for production deployments. A ZooKeeper cluster, known as an *ensemble*, needs to maintain a *quorum*, or a majority vote, within the cluster. The number of failures you're willing to tolerate depends on your environment and service-level agreements to your users, but to create an environment that tolerates F node failures, you should deploy $(2 \times F + 1)$ machines, as shown in table 3.1.

Table 3.1 Number of ZooKeeper nodes required for a quorum

ZooKeeper cluster size (number of nodes)	Quorum size	Number of machine failures tolerated
1	1	0
3	2	1
5	3	2
$2 \times F + 1$	$F + 1$	F

Because the ZooKeeper ensemble requires a majority vote to make cluster decisions, it usually makes sense to deploy ZooKeeper clusters with an odd number of nodes. Generally speaking, I recommend that you start with five ZooKeeper nodes for a production environment. This enables one of the nodes to be taken offline for maintenance and the cluster to still tolerate an unexpected failure.

> **WARNING** Before setting up a ZooKeeper ensemble in a production environment, please take the time to read the ZooKeeper Administrator's Guide at http://zookeeper.apache.org/doc/current/zookeeperAdmin.pdf.

Now, installing ZooKeeper on the same machines as the Mesos masters isn't a requirement, but it does make for simpler deployments and is generally an acceptable approach. If you're planning to deploy other software that also requires ZooKeeper, I recommend that you deploy a separate, standalone ZooKeeper ensemble for those applications and leave the Mesos ZooKeeper ensemble dedicated to serving the Mesos cluster. Later in this chapter, I'll have you install ZooKeeper on each of the Mesos masters to reduce complexity.

As I said previously, this isn't an exhaustive list of best practices, but rather some guidelines for planning the number and location of ZooKeeper nodes and Mesos masters for your deployment. Chapter 6 discusses additional production considerations for both Mesos and ZooKeeper, including logging, monitoring, and access control.

3.2 Installing Mesos and ZooKeeper

Mesos installations are supported on Linux and Mac OS X, and may work on other UNIX-like operating systems. You have two options for installing Mesos and ZooKeeper:

- Using your operating system's package manager
- Compiling the source code and installing the resulting binaries

In this section, you'll learn how to install Mesos on two of the most popular Linux distributions: Red Hat Enterprise Linux (RHEL) / CentOS 7, and Ubuntu 14.04 LTS (code name Trusty).

> **NOTE** Remember that with any on-premises or cloud deployment, you're still responsible for managing the installation, configuration, and health of services. Some cloud providers, such as Amazon Web Services (AWS), provide tools to create templates for provisioning infrastructure automatically. Because these tools are specific to the provider and not related to Mesos, this text doesn't cover them, instead opting to cover configuration of the system at the operating-system level. You should consult your provider's documentation for more information on how to use their various solutions to automate your Mesos cluster deployment.

3.2.1 Installing from packages

Mesosphere provides Mesos package repositories for several Linux distributions that are commonly found in production environments. As of this writing, they include the following operating systems and OS releases:

- RHEL / CentOS 6 and 7
- Ubuntu 12.04 through 14.04
- Debian 7 (code name Wheezy)

This example includes installation instructions for the latest release of RHEL/CentOS and the latest long-term support (LTS) release of Ubuntu. Documentation for setting up Mesosphere repositories on other supported operating systems can be found on its website at https://mesosphere.com/downloads.

RHEL / CentOS 7

First, you need to download the package that installs and configures your system for the Mesosphere repository. You can do this by running the following command:

```
$ sudo rpm -Uvh http://repos.mesosphere.io/el/7/noarch/RPMS/
➥ mesosphere-el-repo-7-1.noarch.rpm
```

After the repository has been installed, you need to install the Mesos package on the masters and slaves. On the masters, you'll also install Mesosphere's ZooKeeper packages. To do so, run the following command:

```
$ sudo yum -y install mesos-0.22.2-0.2.62.centos701406
➥ mesosphere-zookeeper
```

On the slaves, install Mesos by running this command:

```
$ sudo yum -y install mesos-0.22.2-0.2.62.centos701406
```

Installing these packages also installs any dependent packages required by Mesos and ZooKeeper. After these packages are installed, feel free to skip ahead to section 3.3.

Ubuntu 14.04 (Trusty)

To set up the package repositories on Ubuntu, you first need to fetch Mesosphere's GPG public key, which is used to sign the packages. You'll then add Mesosphere's repository to Apt's sources list, and refresh all of the package metadata on the system:

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv E56151BF
$ echo "deb http://repos.mesosphere.io/ubuntu trusty main" |
➥ sudo tee /etc/apt/sources.list.d/mesosphere.list
$ sudo apt-get update
```

After the Mesosphere key is present and the repository is available, you can install the Mesos package on the masters and slaves. On the masters, you'll also install the ZooKeeper package. To do so, run the following command:

```
$ sudo apt-get install mesos=0.22.2-0.2.62.ubuntu1404 zookeeperd
```

Now, on the slaves, install Mesos by running this command:

```
$ sudo apt-get install mesos=0.22.2-0.2.62.ubuntu1404
```

Installing these packages also installs any dependent packages required by Mesos and ZooKeeper. After these packages are installed, feel free to skip ahead to section 3.3.

3.2.2 Compiling and installing from source

Although installing Mesos from the packages provided by Mesosphere is by far the quickest way to get up and running with Mesos, I thought it best to also demonstrate how to build Mesos from source. You might consider doing this for a few reasons:

- You need to modify the Mesos build configuration or enable additional functionality.
- You prefer to build your own packages, potentially with site-specific modifications.
- You prefer to obtain the code directly from the Apache Software Foundation.

You can configure, compile, and install Mesos in various ways. In the following example, you'll use the default configuration as specified by the Mesos configure script. It's worth noting that by default, compiling Mesos also compiles a bundled version of ZooKeeper.

If you'd like to compile ZooKeeper separately or use a different version, you can do so by configuring Mesos prior to compilation time. A complete list of the options you can pass to the configure script can be found with the Mesos project documentation at http://mesos.apache.org/documentation/latest/configuration/.

> **NOTE** Although I've done my best to capture the prerequisite steps and walk you through the build process, these instructions are subject to change for future releases. For the most up-to-date instructions, see http://mesos.apache.org/gettingstarted/.

In short, compiling Mesos requires the following:

- A 64-bit Linux distribution
- GNU Compiler Collection (GCC) 4.4 or newer, or Clang 3.3 or newer

> **TIP** Starting with Mesos 0.23, the minimum compiler version has been upgraded to GCC 4.8 and Clang 3.5 for all supported platforms. For more information, see https://issues.apache.org/jira/browse/MESOS-2604.

Because the number of Linux distributions is always growing, it isn't feasible for me (or the Mesos authors) to provide instructions on how to set up the environment to compile Mesos for all of them. For parity with the package installation instructions I covered in the previous section, the next two sections include instructions for compiling Mesos on RHEL / CentOS 7 and Ubuntu 14.04 LTS (Trusty).

RHEL / CentOS 7 Prerequisites

Fortunately, RHEL 7 includes GCC 4.8.3 as part of its base package offering. This enables you to compile Mesos version 0.22.2, and ensure that you're able to compile version 0.23 and beyond, without requiring you to upgrade the compiler.

All of the dependencies required to compile Mesos are available via RHEL's Yum package manager, except one: Boto, a Python module that provides an interface to the AWS API. You'll need to install this one optional package by using Python's `easy_install` utility, which is included with RHEL.

To install the dependencies required to compile Mesos on RHEL / CentOS 7, run the following commands:

```
$ sudo yum -y groupinstall "Development Tools"

$ sudo yum -y install subversion-devel java-1.8.0-openjdk-devel zlib-devel
➥ libcurl-devel openssl-devel cyrus-sasl-devel cyrus-sasl-md5 apr-devel
➥ apr-util-devel maven python-devel

$ sudo easy_install boto
```

After these development dependencies are installed, feel free to skip to the upcoming "Compiling" section.

Ubuntu 14.04 (Trusty) Prerequisites

Fortunately, Ubuntu 14.04 includes GCC 4.8.4 as part of its base package offering. This enables you to compile Mesos version 0.22.2, and ensure that you're able to compile version 0.23 and beyond, without requiring you to upgrade the compiler.

All of the dependencies required to compile Mesos are available via Ubuntu's Apt package manager, including Boto, the optional Python module that provides an interface to the AWS API.

To install the dependencies required to compile Mesos on Ubuntu 14.04, run the following commands:

```
$ sudo apt-get update

$ sudo apt-get -y install build-essential openjdk-7-jdk python-dev
➥ python-boto libcurl4-nss-dev libsasl2-dev maven libapr1-dev libsvn-dev
```

After these development dependencies are installed, proceed to the next section to compile Mesos and ZooKeeper.

COMPILING

After you have all the development tools needed to compile Mesos, you can download a Mesos release and begin the compilation process. As mentioned previously, you have many options to configure Mesos at compilation time. For this example, you'll accept most of the defaults, but you'll specify the option `--prefix="/usr/local/mesos"` to ensure that all Mesos-related files are kept in a single directory. Feel free to modify this to suit your preferences or environment.

Run the following commands to download, configure, and compile Mesos 0.22.2:

```
$ curl -L -O https://www.apache.org/dist/mesos/0.22.2/mesos-0.22.2.tar.gz
$ tar zxf mesos-0.22.2.tar.gz
$ cd mesos-0.22.2
$ mkdir build && cd $_
$ ../configure --prefix="/usr/local/mesos"
$ make
```

> **TIP** You can speed up compilation time, increasing the number of jobs that Make launches simultaneously, by using the `-j` argument. For example: on a machine with four CPU cores, you can use `make -j4`. Note that each job requires about 2 GB of memory.

Depending on the number of cores available on your system, compiling Mesos could take several minutes. After it has finished building, you might want to run the included test suite. This next step is optional, but it's a good idea if you intend to use any of the example frameworks included with the Mesos source code.

To run the test suite, run the following command:

```
$ make check
```

You'll observe output throughout the test suite, and at the end you should see a test summary resembling the following:

```
[==========] 539 tests from 86 test cases ran. (260794 ms total)
[  PASSED  ] 539 tests.
```

Now that you've compiled Mesos and ensured that the tests are passing on your system, you can go ahead and install Mesos.

INSTALLING MESOS

After Mesos has finished building, you can run the following command to install Mesos:

```
$ sudo make install
```

By specifying `--prefix="/usr/local/mesos"` when running the configure script earlier in the build process, Mesos will install itself into the /usr/local/mesos directory.

Let's take a look at the subdirectories that are now present in the installation directory, as well as their purposes and contents:

- bin/ contains command-line tools for interacting with a Mesos cluster. Some of these tools include mesos-local, mesos-execute, and mesos-ps.
- etc/mesos/ contains configuration files for the Mesos cluster.
- include/ contains the various C++ header files used for interfacing with Mesos.
- lib/ contains native Mesos libraries, such as libmesos.so.
- libexec/mesos/ contains Mesos helper binaries and scripts.
- sbin/ contains several scripts used to start and stop Mesos masters and slaves. Some of these include mesos-master, mesos-slave, and mesos-daemon.sh.
- share/mesos/webui/master/static/ contains the static code for the Mesos web interface.

It's worth noting that because you should be able to compile Mesos on any Linux distribution (provided you have the necessary development tools), it isn't feasible for the Mesos project maintainers to include service wrapper scripts for all Linux operating systems. But as mentioned previously, a set of scripts in the Mesos sbin/ directory enables you to start masters and slaves, both in the foreground and as daemons. You should be able to write a simple wrapper around the included scripts to start, stop, and restart the `mesos-master` and `mesos-slave` services using your operating system's service manager or your service manager of choice.

> **TIP** The mesos-init-wrapper script, which is included with the Mesos packages provided by Mesosphere, is open source and can be found in the following GitHub repository: https://github.com/mesosphere/mesos-deb-packaging/blob/master/mesos-init-wrapper.

INSTALLING ZOOKEEPER

When you compiled Mesos in the build/ directory, you also compiled a version of ZooKeeper that the Mesos maintainers bundle in the Mesos source tree. The compiled ZooKeeper release is located within the build/3rdparty directory, and is now ready to be installed.

Chances are you'll want to relocate ZooKeeper somewhere a little more permanent, so let's go ahead and copy it alongside the Mesos installation in /usr/local:

```
$ sudo cp -rp 3rdparty/zookeeper-3.4.5 /usr/local/
$ sudo chown -R root:root /usr/local/zookeeper-3.4.5
```

Located within the ZooKeeper distribution are several scripts in the bin/ directory that assist you with starting and stopping the ZooKeeper cluster. Specifically, you'll want to pay attention to zkServer.sh, as this script will allow you to start, stop, and restart your Zookeeper cluster.

As was mentioned previously when you were installing Mesos, it's not feasible for the ZooKeeper project maintainers to maintain service scripts for all of the operating

systems ZooKeeper can run on. Some init scripts for RPM-based and Deb-based operating systems are located in the src/packages/ directory of the ZooKeeper distribution. But otherwise, you should be able to write a small wrapper script for your operating system's service manager, or your service manager of choice, using the zkServer.sh script.

Now that you've downloaded, compiled, and installed both Mesos and ZooKeeper, you're ready to learn about the various configuration options—and the methods for specifying those options—available for both Mesos and ZooKeeper.

3.3 Configuring Mesos and ZooKeeper

Now that you understand the components that make up a Mesos deployment and have installed Mesos and ZooKeeper, you need to configure everything so you can fire up the services and start using your Mesos cluster.

The ways you'll configure ZooKeeper and Mesos will differ slightly based on whether you chose to install from packages or compile from source in the preceding section. But not to worry—you'll learn about both as you work your way through this section.

3.3.1 ZooKeeper configuration

First and foremost, you'll begin by configuring ZooKeeper, which is required for Mesos cluster coordination and leader election. Figure 3.2 shows ZooKeeper's role in the cluster you're deploying.

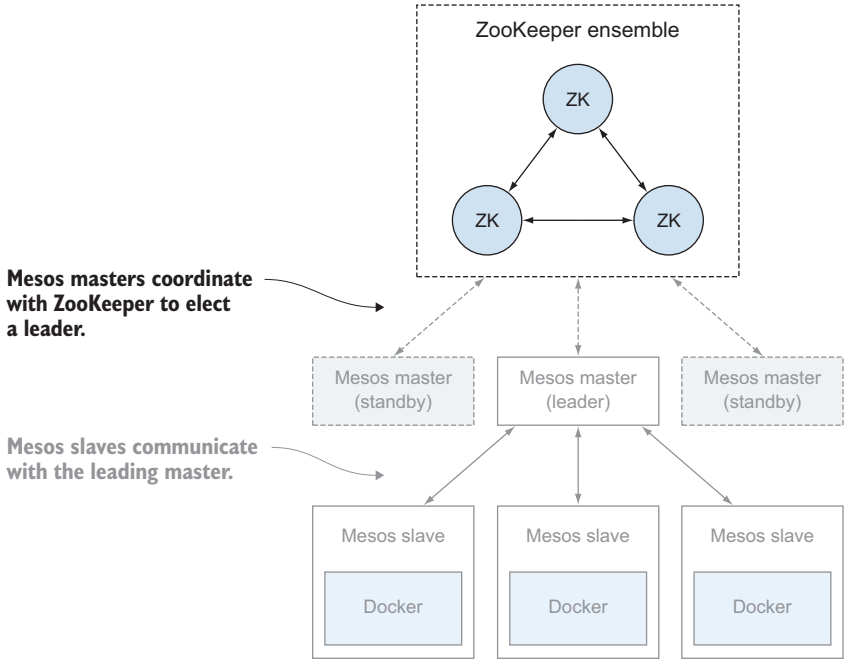


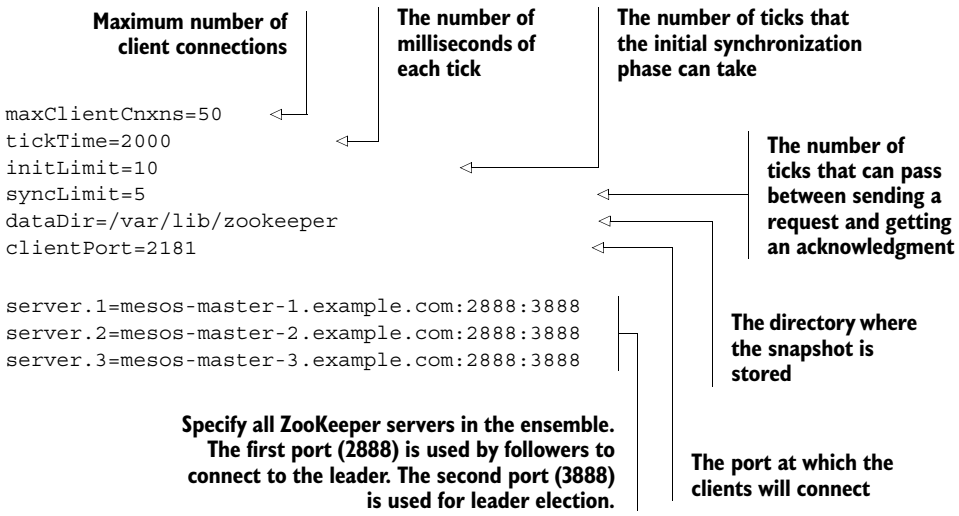
Figure 3.2 Mesos masters coordinate leader election using a ZooKeeper ensemble.

The location of the configuration file, zoo.cfg, varies depending on whether you installed ZooKeeper from packages or compiled it as part of Mesos:

- For package-based installations, the configuration file is located at /etc/zookeeper/conf/zoo.cfg.
- For source-based installations (assuming you placed the ZooKeeper distribution at /usr/local/zookeeper-3.4.5 as instructed), the configuration file is located at /usr/local/zookeeper-3.4.5/conf/zoo.cfg.

The following listing provides basic settings needed to get a ZooKeeper 3.4.*x* cluster up and running.

Listing 3.1 Basic ZooKeeper configuration



In addition to creating the zoo.cfg configuration file, you'll need to assign each node in the ZooKeeper ensemble a unique ID. You'll notice in this listing that the servers are numbered: `server.1`, `server.2`, and so on. On each of those machines, create the file myid within ZooKeeper's conf/ directory. The file's only contents should be an integer between 1 and 255, representing the unique ID for that particular ZooKeeper node.

NOTE The ID number configured in the myid file on a given ZooKeeper node must correspond with the ID number it's given in zoo.cfg.

STARTING THE SERVICES

Although I've covered only the bare-minimum ZooKeeper configuration required, it's enough to bring up the machines in the ensemble and allow them to serve clients. Let's go ahead and start the ZooKeeper service on each of the masters:

- For package-based installations—`service zookeeper start`
- For source-based installations—`/usr/local/zookeeper-3.4.5/bin/zkServer.sh start`

When the service is up and running, you can ensure that the ZooKeeper server is in a healthy state by using Netcat to send it a health-check command. You should be able to send ZooKeeper the `ruok` command, and it should respond with `imok`:

```
$ echo ruok | nc 127.0.0.1 2181
imok
```

If that last check worked, fantastic! ZooKeeper is up and running and ready to serve your Mesos cluster. Chapter 6 covers monitoring the ZooKeeper cluster a bit more. But for now, the ZooKeeper ensemble is up and running, so let's proceed to configure Mesos.

> **TIP** For additional configuration options that might apply to your environment, please consult the ZooKeeper Administrator's Guide at http://zookeeper.apache.org/doc/current/zookeeperAdmin.pdf.

3.3.2 Mesos configuration

Now that the ZooKeeper ensemble is up and ready to provide services for your Mesos cluster, let's discuss how to go about configuring Mesos.

The configuration provided here is enough to get a cluster ready to handle distributed workloads, but it won't go into all the configuration options that Mesos provides. You should refer to http://mesos.apache.org/documentation/latest/configuration for the latest documentation and configuration options.

CONVENTIONS

Multiple conventions exist for configuring a Mesos deployment:

- *File-based*—When using Mesos packages provided by Mesosphere, configuration values can exist in files on disk, with each file named after the configuration option. Some examples of this include /etc/mesos/zk and /etc/mesos-slave/attributes/rack.
- *Environment-based*—For both package-based and source-based installations, configuration values can exist as environment variables that are read when the `mesos-master` or `mesos-slave` services start. They can already be part of the environment, or in a shell script that's sourced before the services are started. For example: `MESOS_zk="zk://..."`

- *Command-line arguments*—Configuration values can also be passed in as arguments to the `mesos-master` and `mesos-slave` binaries and service scripts. For example: `mesos-master --zk=zk://...`

Chances are you'll be interacting with Mesos more as a service and less on the command line, so this section explains configuring the masters and slaves by using the file-based and environment-based methods. But it's worth noting that all the configuration options can be specified as command-line arguments, if you so desire.

To determine the locations of the configuration files for your installation method and operating system, take a look at table 3.2. This table assumes that for compiled installations, you installed Mesos to /usr/local/mesos.

Table 3.2 Mesos configuration file locations

Operating system	Installation method	Configuration method	Configuration locations
RHEL and CentOS 7	Packages (Mesosphere)	File	/etc/mesos/ /etc/mesos-master/ /etc/mesos-slave/
RHEL and CentOS 7	Packages (Mesosphere)	Environment	/etc/default/mesos /etc/default/mesos-master /etc/default/mesos-slave
Ubuntu 14.04	Packages (Mesosphere)	File	/etc/mesos/ /etc/mesos-master/ /etc/mesos-slave/
Ubuntu 14.04	Packages (Mesosphere)	Environment	/etc/default/mesos /etc/default/mesos-master /etc/default/mesos-slave
All	Source	Environment	/usr/local/mesos/etc/mesos/mesos-master-env.sh /usr/local/mesos/etc/mesos/mesos-slave-env.sh

Please refer to the paths in this table for the configuration method of your choosing. The following sections cover common options for Mesos master and slave configuration without necessarily referencing the configuration files (or their locations) directly.

MASTER CONFIGURATION

You must use several required configuration settings for the proper functioning of the Mesos masters: the ZooKeeper URL, the size of the Mesos master quorum, and the Mesos master's working directory. Figure 3.3 shows where you're at in the deployment process: configuring the Mesos masters.

The `--zk` option sets the ZooKeeper URL, which is used for leader election and coordination among the masters. This option is used in highly available deployments

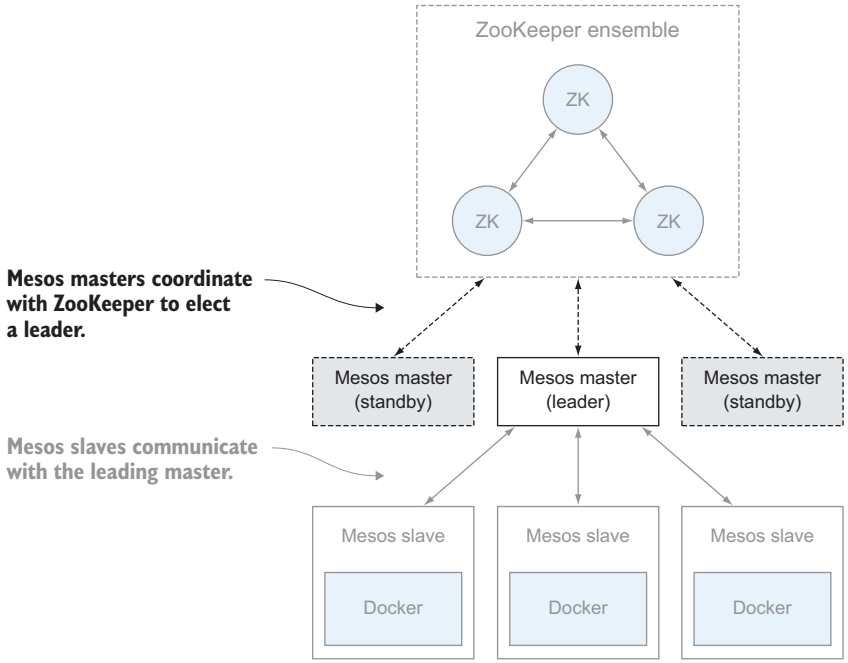


Figure 3.3 Mesos masters use the previously configured ZooKeeper ensemble for leader election.

and isn't required if you're running Mesos in standalone mode. The ZooKeeper URL uses the following convention:

```
zk://mesos-master-1.example.com:2181,
➥ mesos-master-2.example.com:2181,
➥ mesos-master-3.example.com:2181/mesos
```

If you have authentication enabled on the ZooKeeper cluster, you can either specify a username and password in the URL itself, or reference a file on disk that contains the ZooKeeper URL with your authentication details:

```
zk://username:password@host1:2181,host2:2181,host3:2181/mesos
file:///path/to/zk_url
```

If you set up the ZooKeeper cluster by following the instructions in this chapter, don't worry about authentication for now. Chapter 6 covers securing your Mesos and ZooKeeper installations.

The `--quorum` option establishes the majority of the Mesos masters in the cluster, and is used for the Mesos replicated registry. Much like the ZooKeeper quorum, it should be set according to table 3.3, where *N* is the value of the `quorum` option.

Table 3.3 Number of Mesos masters required for a quorum

Number of Mesos masters	Mesos master quorum	Number of machine failures tolerated
1	1	0
3	2	1
5	3	2
$2 \times N - 1$	N	$N - 1$

WARNING When configuring the `quorum` option, it's important to ensure that the number of Mesos masters in the cluster matches table 3.3. Any additional masters could violate the quorum and corrupt the replicated log. Chapter 6 provides additional information on how to safely increase and decrease the Mesos master quorum size.

In the context of the Mesos masters, the `--work_dir` option specifies the location on disk that Mesos will use for its own replicated log. There's no default, so this must be specified. Usually a path such as /var/lib/mesos is a good option.

Although it's not required, it's a great idea to set the `--log_dir` option as well. This ensures that Mesos will log events to disk, so that you can more easily debug your Mesos cluster going forward. The packages provided by Mesosphere already set `log_dir` to /var/log/mesos.

TIP I recommend setting two additional configuration options: `--hostname` and `--ip`. Setting these options ensures that the hostname used by the Mesos service, and the IP address for the Mesos service to communicate on, are configured properly and not autodiscovered. Setting values for these two options becomes even more important on machines with more than one network interface.

Based on the preceding examples, here's what the mesos-master-env.sh script might look like on a source-based installation of Mesos:

```
export MESOS_zk=zk://mesos-master-1.example.com:2181,
➥ mesos-master-2.example.com:2181,mesos-master-3.example.com:2181/mesos
export MESOS_quorum=2
export MESOS_work_dir=/var/lib/mesos
export MESOS_log_dir=/var/log/mesos
```

After you've applied a basic configuration, all that's left to do is to start the `mesos-master` service by running the following command:

```
$ sudo service mesos-master start
```

It's also a good idea to disable the `mesos-slave` service on the masters:

- On RHEL / CentOS 7—`sudo systemctl disable mesos-slave.service`
- On Ubuntu 14.04—`echo "manual" | sudo tee /etc/init/mesos-slave.override`

You should now be able to open a web browser and connect to one of the Mesos masters by visiting http://mesos-master-1.example.com:5050, replacing the hostname with one of your own Mesos masters' hostnames or IP addresses. If you connect to a Mesos master that isn't the current leading master, you'll be redirected to the leader automatically.

SLAVE CONFIGURATION

You need to set several configuration options for the proper functioning of a Mesos slave, specifically the ZooKeeper URL and the slave's working directory. Figure 3.4 shows where you're at in the deployment process: configuring the Mesos slaves.

The `--master` option sets the ZooKeeper URL, which is used by the Mesos slave to detect the leading Mesos master and connect to the cluster. The ZooKeeper URL uses the following convention:

```
zk://mesos-master-1.example.com:2181,
➥ mesos-master-2.example.com:2181,
➥ mesos-master-3.example.com:2181/mesos
```

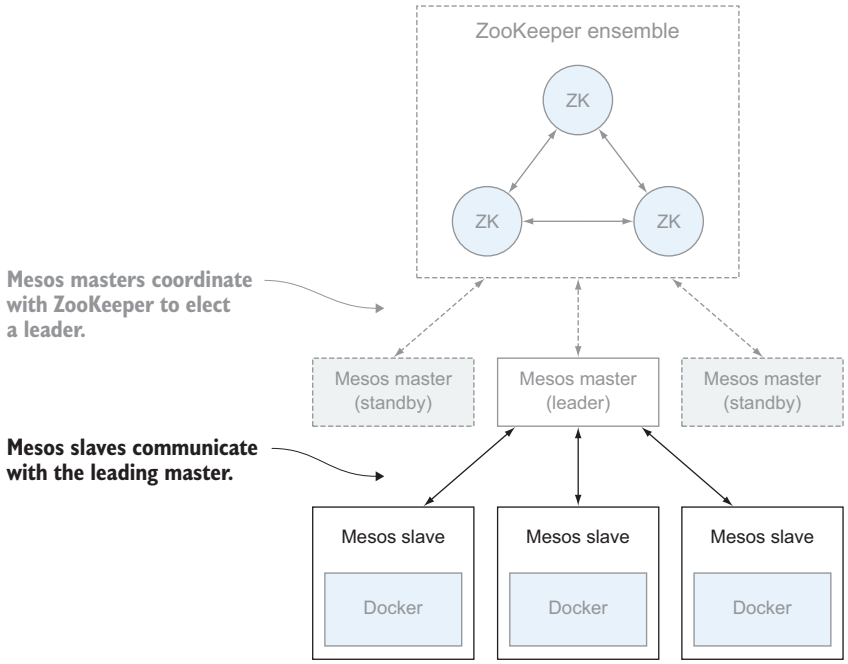


Figure 3.4 Mesos slaves use ZooKeeper to detect—and register with—the leading Mesos master.

When you have authentication enabled on the ZooKeeper cluster, you can either specify a username and password in the URL itself, or reference a file on disk that contains the ZooKeeper URL with your authentication details:

```
zk://username:password@host1:2181,host2:2181,host3:2181/mesos
file:///path/to/zk_url
```

> **NOTE** If you're using the Mesos packages provided by Mesosphere, you should set the ZooKeeper URL for the Mesos slave daemon in the file /etc/mesos/zk, not in /etc/mesos-slave/master.

In the context of the Mesos slaves, the `--work_dir` option specifies the location on disk that Mesos frameworks will use for their working directories and sandboxes. There's no default, so this must be specified. Usually a path such as /var/lib/mesos is a good option. It's also a good idea to ensure that the partition that this directory resides on isn't mounted with the `noexec` option, as /tmp often is.

Although it's not required, it's a great idea to set the `--log_dir` option as well. This ensures that Mesos will log events to disk so that you can more easily debug your Mesos cluster going forward. The packages provided by Mesosphere already set `--log_dir` to /var/log/mesos.

Based on the preceding examples, here's what the mesos-slave-env.sh script might look like on a source-based installation of Mesos:

```
export MESOS_master=zk://mesos-master-1.example.com:2181,
➥ mesos-master-2.example.com:2181,mesos-master-3.example.com:2181/mesos
export MESOS_work_dir=/var/lib/mesos
export MESOS_log_dir=/var/log/mesos
```

After you've applied a basic configuration, all that's left to do is to start the `mesos-slave` service by running the following command:

```
$ sudo service mesos-slave start
```

It's also a good idea to disable the `mesos-master` service on the slaves:

- On RHEL 7—`sudo systemctl disable mesos-master.service`
- On Ubuntu—`echo "manual" | sudo tee /etc/init/mesos-master.override`

If you navigate to the web interface for the Mesos master, you should now notice that the newly configured slave has appeared and is advertising its resources to the cluster.

As mentioned earlier, this is the bare-minimum configuration required to get a Mesos cluster up and running. For an extensive list of configuration options for both the masters and slaves, please consult the official Mesos documentation at http://mesos.apache.org/documentation/latest/configuration. Chapter 4 discusses additional configuration options, such as slave attributes and resources.

3.4 Installing and configuring Docker

Because multiple applications' workloads can run simultaneously on any given Mesos slave, each executor runs inside a container. In addition to the Mesos native containerizer (which uses Linux control groups), you also have the option of launching containers by using Docker. This section explains a bit more about deploying Docker, including installation and some of the commonly used configuration options.

Figure 3.5 shows where you're at in the deployment process: installing and configuring Docker on the Mesos slaves.

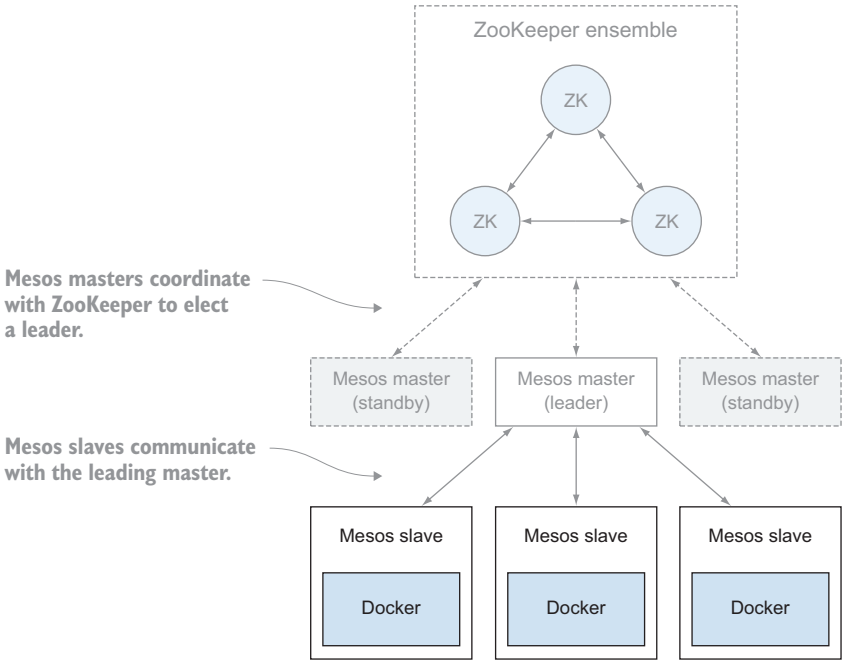


Figure 3.5 Mesos slaves can launch Docker containers, in addition to using Linux control groups.

If you aren't interested in launching Docker containers on your Mesos cluster just yet, no worries; you can always come back to this section. But it's worth knowing that later chapters in this book include examples that involve launching Docker containers on a Mesos cluster.

3.4.1 Installation

As I mentioned in chapter 1, custom executors must be available on each Mesos slave. Although Docker is a Mesos containerizer, it is no exception: the Docker daemon

must be installed, configured, and running on each Mesos slave before you can launch Docker containers on the cluster.

RHEL / CentOS 7

Red Hat includes Docker packages in the "extras" channel (repository), so you don't need to install any additional repositories in order to install the package. You can use the Yum package manager to install it:

```
$ sudo yum -y install docker
```

As of Docker 1.5.0, the service doesn't automatically start on RHEL 7. To start services, run the following command:

```
$ sudo service docker start
```

> **If the Docker service fails to start on RHEL / CentOS 7**
>
> The Docker daemon might fail to start if you haven't installed the latest software updates. Specifically, if the package `device-mapper-event-libs` is older than version 1.02.90, you'll see the following cryptic error message:
>
> ```
> /usr/bin/docker: relocation error: /usr/bin/docker:
> symbol dm_task_get_info_with_deferred_remove,
> version Base not defined in file libdevmapper.so.1.02
> with link time reference
> ```
>
> To resolve this issue, install the latest patches for RHEL by running `yum update`, or upgrade this particular package with `yum update device-mapper-event-libs`.

Ubuntu 14.04 (Trusty)

The Apt repository for Docker is available only by using HTTPS. If your system doesn't already have the `apt-transport-https` package installed, you'll need to install it first:

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https
```

Next, fetch Docker's package-signing key, and add the Docker repository to Apt's sources list:

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv A88D21E9

$ echo "deb https://get.docker.com/ubuntu docker main" |
➥ sudo tee /etc/apt/sources.list.d/docker.list

$ sudo apt-get update
```

Finally, install the package for Docker Engine:

```
$ sudo apt-get -y install docker-engine
```

As of Docker 1.6.0, the Docker service will automatically be started after the package is installed. You can verify this by running the following command:

```
$ sudo service docker status
```

3.4.2 Configuration

If you need to make any configuration changes to the Docker daemon, such as configuring a proxy or specifying certain DNS servers, now is a good time. Because Docker works well using the default configuration, this section doesn't go into Docker's configuration in too much detail. But I will cover some of the more common configuration settings for both RHEL / CentOS 7 and Ubuntu 14.04.

> **TIP** You can find a complete list of Docker's configuration settings on its website at https://docs.docker.com/engine/reference/commandline/daemon/.

If your environment doesn't require the use of an HTTP proxy and you have a properly working DNS deployment, feel free to skip ahead to the next section.

RHEL / CentOS 7

On RHEL / CentOS 7, Docker's configuration file is located at /etc/sysconfig/docker. The following code snippet demonstrates using an HTTP proxy for connecting to Docker Hub, enabling SELinux support in Docker itself, and specifying external DNS servers:

```
http_proxy="http://127.0.0.1:3128/"
OPTIONS="--selinux-enabled --dns 8.8.8.8 --dns 8.8.4.4"
```

Specify the path to an HTTP proxy.

Pass additional options to the Docker daemon.

If you made any changes to the configuration file, be sure to restart the Docker service:

```
$ sudo service docker restart
```

Ubuntu 14.04 (Trusty)

On Ubuntu 14.04, Docker's configuration file is located at /etc/default/docker. The following code snippet demonstrates using an HTTP proxy for connecting to Docker Hub, enabling SELinux support in Docker itself, and specifying external DNS servers:

```
DOCKER_OPTS="--dns 8.8.8.8 --dns 8.8.4.4"
export http_proxy="http://127.0.0.1:3128/"
```

Use DOCKER_OPTS to modify the options passed to the Docker daemon.

Specify the path to an HTTP proxy.

If you made any changes to the configuration file, be sure to restart the Docker service:

```
$ sudo service docker restart
```

Now that you have the Docker service running and any environment-specific configurations applied, let's look at the small configuration changes you need to make to Mesos in order for it to launch Docker containers.

3.4.3 Configuring Mesos slaves for Docker

With Docker up and operational, you'll need to modify the list of containerizers available to the Mesos slave and increase the executor registration time-out. By increasing the executor registration time-out, Docker is given more time to fetch images from Docker Hub (or from your private Docker Registry) before Mesos thinks something failed. Here, you'll set this value to `5mins`. Assuming you're using the file-based configuration that the Mesosphere packages allow for, the configuration will look something like the following:

```
$ echo "docker,mesos" | sudo tee /etc/mesos-slave/containerizers
$ echo "5mins" | sudo tee /etc/mesos-slave/executor_registration_timeout
```

Now restart the `mesos-slave` service to apply the changes:

```
$ sudo service mesos-slave restart
```

> **NOTE** Unlike the default `mesos` containerizer, the `docker` containerizer in Mesos doesn't enforce a disk quota (as of Mesos 0.22.2). For more information, see https://issues.apache.org/jira/browse/MESOS-2502.

3.5 Upgrading Mesos

In contrast to distributed systems that set up statically partitioned clusters or that use only a single master, the fault-tolerant architecture of Mesos allows upgrades to occur on both the masters and the slaves without any cluster downtime. The Mesos API even provides a means to notify frameworks that a failure or leader election occurred, allowing them to take appropriate action if needed. This section explains the process for upgrading both Mesos masters and slaves.

> **NOTE** Prior to Mesos reaching version 1.0, the Mesos maintainers support only $N + 1$ upgrades. If you're running Mesos 0.22 and wish to upgrade to 0.24, you'll need to upgrade to 0.23 first. When performing upgrades, it's always a good idea to consult the latest release notes and upgrade documentation on the Mesos project page. This documentation is located at http://mesos.apache.org/documentation/latest/upgrades.

As with any piece of software, eventually a time comes when we must upgrade it. Whether it's for new features or to patch security vulnerabilities, software upgrades

are usually associated with scheduled downtime. In contrast, Mesos can be upgraded without taking the cluster offline.

This section discusses the procedure for upgrading both the Mesos masters and the slaves, which is largely the same as the Mesos installation you performed previously, with a few additional considerations. Chapter 4 explains how Mesos makes this possible.

3.5.1 Upgrading Mesos masters

As the number of Mesos masters will be much smaller than the number of Mesos slaves, it's a good idea to use some sort of configuration management or orchestration tool to ensure that the entire master quorum isn't upgraded at the same time. Using a configuration management tool such as Puppet or Chef, you can ensure that each individual master is upgraded only on a set schedule. Using an orchestration tool such as Ansible or Fabric, you can ensure that upgrades are serially executed across multiple masters.

To upgrade the Mesos masters, you need to take the following steps:

1. Upgrade the Mesos binaries and restart the master daemon.
2. Upgrade the schedulers to use the new Mesos native library, JAR, or egg.
3. Restart the schedulers.

3.5.2 Upgrading Mesos slaves

Using features known as *slave recovery* and *checkpointing*, Mesos allows for slaves to upgrade themselves without interrupting running tasks. When the `mesos-slave` service is stopped and the Mesos binaries are upgraded, the executors—and their tasks—will continue running. If they finish before the Mesos slave daemon comes back online, they'll wait the `recovery_timeout`, a configurable slave option that defaults to 15 minutes. But if this threshold is exceeded, any executors waiting to connect to the Mesos slave process will self-terminate.

TIP Chapter 4 covers slave recovery and checkpointing in more detail.

When performing rolling upgrades on your Mesos infrastructure, it's good to have some sort of configuration management or orchestration tooling in place. This will ensure that only a percentage of the cluster is upgrading at any given time in case it becomes necessary to stop the upgrade, or if the upgrade coincides with other scheduled maintenance (for example, applying security patches). To upgrade the Mesos slaves, you need to take the following steps:

1. Upgrade the Mesos binaries and restart the slave daemon.
2. Upgrade the executors to use the new Mesos native library, JAR, or egg (if needed).

3.6 Summary

Following the examples provided in this chapter, you should have a properly functioning, highly available Mesos and ZooKeeper cluster that's ready to execute tasks and launch Docker containers. Here are a few things to remember:

- The Mesos masters use Apache ZooKeeper for leader election and coordination. Mesos slaves and schedulers also use ZooKeeper to detect the leading Mesos master.
- For highly available deployments, you should deploy at least three machines running ZooKeeper and three machines to serve as Mesos masters. For a simplified deployment, you can run the ZooKeeper services on the same machines as the Mesos masters, but you might want to separate them if you plan on using other software that depends on ZooKeeper.
- You should install and configure ZooKeeper, followed by Mesos masters, followed by Mesos slaves.
- Docker is a popular option for running distributed applications within a datacenter. Although Mesos has native Docker support out of the box, it still needs to be installed separately. Mesos requires slight configuration to enable Docker support.
- Both the Mesos masters and slaves can be upgraded without incurring any cluster downtime.
- Puppet, an open source configuration management tool, can automate the installation and configuration of your Mesos, ZooKeeper, and Docker infrastructure. The Puppet Forge contains modules to manage each of the components in a Mesos deployment.

The next chapter presents more Mesos fundamentals, including how Mesos implements resource isolation, slave resources, and fault tolerance.

PROGRAMMING

Mesos IN ACTION

Roger Ignazio

Modern datacenters are complex environments, and when you throw Docker and other container-based systems into the mix, there's a great need to simplify. Mesos is an open source cluster management platform that transforms the whole datacenter into a single pool of compute, memory, and storage resources that you can allocate, automate, and scale as if you're working with a single supercomputer.

Mesos in Action introduces readers to the Apache Mesos cluster manager and the concept of application-centric infrastructure. Filled with helpful figures and hands-on instructions, this book guides you from your first steps creating a highly-available Mesos cluster through deploying applications in production and writing native Mesos frameworks. You'll learn how to scale to thousands of nodes, while providing resource isolation between processes using Linux and Docker containers. You'll also learn practical techniques for deploying applications using popular key frameworks.

What's Inside

- Spinning up your first Mesos cluster
- Scheduling, resource administration, and logging
- Deploying containerized applications with Marathon, Chronos, and Aurora
- Writing Mesos frameworks using Python

Readers need to be familiar with the core ideas of datacenter administration and need a basic knowledge of Python or a similar programming language.

Roger Ignazio is an experienced systems engineer with a focus on distributed, fault-tolerant, and scalable infrastructure. He is currently a technical lead at Mesosphere.

"You would be hard-pressed to find a better guide than Roger Ignazio and a better book than *Mesos in Action*."
—From the Foreword by Florian Leibert, Mesosphere

"Helps to illuminate best practices and avoid hidden pitfalls when deploying Apache Mesos."
—Marco Massenzio, Apple

"Ignazio knows his stuff, but more importantly he knows how to explain it."
—Morgan Nelson, getaroom.com

"You will not only learn Mesos, you will learn a whole ecosystem."
—Thomas Peklak, Emakina CEE

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit www.manning.com/books/mesos-in-action

ISBN-13: 978-1-61729-292-7
ISBN-10: 1-61729-292-3
54499
9 781617 292927

MANNING $44.99 / Can $51.99 [INCLUDING eBOOK]