CoreOS Essentials

This book will help you get up and running on using CoreOS to develop effective computing networks. You will begin with understanding the basics of CoreOS. You will then discover what etcd is and how it is used, followed by launching Docker containers with systemd and fleet. Learn how to manage clusters, read system logs, and customize with cloud-config. You will set up the deployment to production using Docker builder and a private Docker registry. You will also see how to set up and use CoreUpdate and Enterprise Registry, and get an introduction to the new App Container called rkt and the newly introduced cluster manager known as Kubernetes.

This book will equip you with all the information you need to leverage the power of CoreOS and the related containers for the effective deployment of your applications.

Who this book is written for
If you are a system administrator with experience in network virtualization and want to develop computing networks for deploying applications and servers, then this is the book for you. Knowledge of Linux operating systems and Docker is expected.

What you will learn from this book
- Install and set up CoreOS on your machine and learn more about etcd
- Write and deploy systemd and fleet units
- Customize your clustering machines with cloud-config
- Set up your Docker container's deployment
- Configure cluster Central Services and release channels
- Familiarize yourself with free as well as paid CoreOS services
- Get an overview of CoreOS rkt, a new App Container runtime
- Learn about Kubernetes and set up its cluster

Develop effective computing networks to deploy your applications and servers using CoreOS

Rimantas Mocevicius

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'CoreOS – Overview and Installation'
- A synopsis of the book’s content
- More information on CoreOS Essentials
About the Author

**Rimantas Mocevicius** is an IT professional with over 20 years of experience in Linux. He is a supporter and fan of open source software. His passion for new technologies drives him forward, and he never wants to stop learning about them.
Preface

CoreOS is a new breed of the Linux operating system and is optimized to run Linux containers, such as Docker and rkt. It has a fully automated update system, no package manager, and a fully clustered architecture.

Whether you are a Linux expert or just a beginner with some knowledge of Linux, this book will provide you with step-by-step instructions on installing and configuring CoreOS servers as well as building development and production environments. You will be introduced to the new CoreOS rkt Application Containers runtime engine and Google’s Kubernetes system, which allows you to manage a cluster of Linux containers as a single system.

What this book covers

Chapter 1, CoreOS – Overview and Installation, contains a brief CoreOS overview what CoreOS is about.

Chapter 2, Getting Started with etcd, explains what etcd is and what it can be used for.

Chapter 3, Getting Started with systemd and fleet, covers an overview of systemd. This chapter tells you what fleet is and how to use it to deploy Docker containers.

Chapter 4, Managing Clusters, is a guide to setting up and managing a cluster.

Chapter 5, Building a Development Environment, shows you how to set up the CoreOS development environment to test your Application Containers.
Preface

Chapter 6, Building a Deployment Setup, helps you set up code deployment, the Docker image builder, and the private Docker registry.

Chapter 7, Building a Production Cluster, explains the setup of the CoreOS production cluster on the cloud.

Chapter 8, Introducing CoreUpdate and Container/Enterprise Registry, has an overview of free and paid CoreOS services.

Chapter 9, Introduction to CoreOS rkt, tells you what rkt is and how to use it.

Chapter 10, Introduction to Kubernetes, teaches you how to set up and use Kubernetes.
1
CoreOS – Overview and Installation

CoreOS is often described as Linux for massive server deployments, but it can also run easily as a single host on bare-metal, cloud servers, and as a virtual machine on your computer as well. It is designed to run application containers as docker and rkt, and you will learn about its main features later in this book.

This book is a practical, example-driven guide to help you learn about the essentials of the CoreOS Linux operating system. We assume that you have experience with VirtualBox, Vagrant, Git, Bash shell scripting and the command line (terminal on UNIX-like computers), and you have already installed VirtualBox, Vagrant, and git on your Mac OS X or Linux computer, which will be needed for the first chapters. As for a cloud installation, we will use Google Cloud’s Compute Engine instances.

By the end of this book, you will hopefully be familiar with setting up CoreOS on your laptop or desktop, and on the cloud. You will learn how to set up a local computer development machine and a cluster on a local computer and in the cloud. Also, we will cover etcd, systemd, fleet, cluster management, deployment setup, and production clusters.

Also, the last chapter will introduce Google Kubernetes. This is an open source orchestration system for docker and rkt containers and allows to manage them as a single system on on compute clusters.

In this chapter, you will learn how CoreOS works and how to carry out a basic CoreOS installation on your laptop or desktop with the help of VirtualBox and Vagrant.

We will basically cover two topics in this chapter:

- An overview of CoreOS
- Installing the CoreOS virtual machine
An overview of CoreOS

CoreOS is a minimal Linux operation system built to run docker and rkt containers (application containers). By default, it is designed to build powerful and easily manageable server clusters. It provides automatic, very reliable, and stable updates to all machines, which takes away a big maintenance headache from sysadmins. And, by running everything in application containers, such setup allows you to very easily scale servers and applications, replace faulty servers in a fraction of a second, and so on.

How CoreOS works

CoreOS has no package manager, so everything needs to be installed and used via docker containers. Moreover, it is 40 percent more efficient in RAM usage than an average Linux installation, as shown in this diagram:

CoreOS utilizes an active/passive dual-partition scheme to update itself as a single unit, instead of using a package-by-package method. Its root partition is read-only and changes only when an update is applied. If the update is unsuccessful during reboot time, then it rolls back to the previous boot partition. The following image shows OS updated gets applied to partition B (passive) and after reboot it becomes the active to boot from.
The docker and rkt containers run as applications on CoreOS. Containers can provide very good flexibility for application packaging and can start very quickly—in a matter of milliseconds. The following image shows the simplicity of CoreOS. Bottom part is Linux OS, the second level is etcd/fleet with docker daemon and the top level are running containers on the server.

By default, CoreOS is designed to work in a clustered form, but it also works very well as a single host. It is very easy to control and run application containers across cluster machines with fleet and use the etcd service discovery to connect them as it shown in the following image.
CoreOS can be deployed easily on all major cloud providers, for example, Google Cloud, Amazon Web Services, Digital Ocean, and so on. It runs very well on bare-metal servers as well. Moreover, it can be easily installed on a laptop or desktop with Linux, Mac OS X, or Windows via Vagrant, with VirtualBox or VMware virtual machine support.

This short overview should throw some light on what CoreOS is about and what it can do. Let's now move on to the real stuff and install CoreOS on to our laptop or desktop machine.

**Installing the CoreOS virtual machine**

To use the CoreOS virtual machine, you need to have VirtualBox, Vagrant, and git installed on your computer.

In the following examples, we will install CoreOS on our local computer, which will serve as a virtual machine on VirtualBox.

Okay, let's get started!

**Cloning the coreos-vagrant GitHub project**

Let's clone this project and get it running.

In your terminal (from now on, we will use just the terminal phrase and use $ to label the terminal prompt), type the following command:

$$
$ git clone https://github.com/coreos/coreos-vagrant/
$$

This will clone from the GitHub repository to the coreos-vagrant folder on your computer.

**Working with cloud-config**

To start even a single host, we need to provide some config parameters in the cloud-config format via the user data file.

In your terminal, type this:

$$
$ cd coreos-vagrant
$ mv user-data.sample user-data
$$
The user data should have content like this (the coreos-vagrant Github repository is constantly changing, so you might see a bit of different content when you clone the repository):

```bash
#cloud-config
coreos:
etcd2:
    #generate a new token for each unique cluster from https://discovery.etcd.io/new
    #discovery: https://discovery.etcd.io/<token>
    # multi-region and multi-cloud deployments need to use $public_ipv4
    advertise-client-urls: http://$public_ipv4:2379
    initial-advertise-peer-urls: http://$private_ipv4:2380
    # listen on both the official ports and the legacy ports
    # legacy ports can be omitted if your application doesn't depend on them
    listen-client-urls: http://0.0.0.0:2379, http://0.0.0.0:4001
    listen-peer-urls:
fleet:
    public-ip: $public_ipv4
flannel:
    interface: $public_ipv4
units:
    - name: etcd2.service
      command: start
    - name: fleet.service
      command: start
    - name: docker-tcp.socket
      command: start
      enable: true
      content: |
          [Unit]
          Description=Docker Socket for the API
          [Socket]
          ListenStream=2375
          Service=docker.service
          BindIPv6Only=both
          [Install]
          WantedBy=sockets.target
```
CoreOS – Overview and Installation

Replace the text between the `etcd2:` and `fleet:` lines to look this:

```
etcd2:
    name: core-01
    initial-advertise-peer-urls: http://$private_ipv4:2380
    listen-peer-urls:
    initial-cluster-token: core-01_etcd
    initial-cluster: core-01=http://$private_ipv4:2380
    initial-cluster-state: new
    advertise-client-urls:
    listen-client-urls: http://0.0.0.0:2379,http://0.0.0.0:4001
fleet:
```

You can also download the latest user-data file from https://github.com/rimusz/coreos-essentials-book/blob/master/Chapter1/user-data.

This should be enough to bootstrap a single-host CoreOS VM with `etcd`, `fleet`, and `docker` running there.

We will cover `cloud-config`, `etcd` and `fleet` in more detail in later chapters.

**Startup and SSH**

It's now time to boot our CoreOS VM and log in to its console using `ssh`.

Let's boot our first CoreOS VM host. To do so, using the terminal, type the following command:

```
$ vagrant up
```

This will trigger `vagrant` to download the latest CoreOS alpha (this is the default channel set in the `config.rb` file, and it can easily be changed to beta, or stable) channel image and the `lunch` VM instance.
CoreOS VM has booted up, so let's open the `ssh` connection to our new VM using the following command:

```
$ vagrant ssh
```

It should show something like this:

```
CoreOS alpha (some version)
core@core-01 - $
```

**Downloading the example code**

You can download the example code files from your account at http://www.packtpub.com for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.
CoreOS – Overview and Installation

Perfect! Let’s verify that etcd, fleet, and docker are running there. Here are the commands required and the corresponding screenshots of the output:

$ systemctl status etcd2

```
CoreOS alpha (681.0.0)
Update Strategy: Stable

* etcd2.service - etcd2
   Loaded: loaded (/usr/lib64/systemd/system/etcd2.service; static; vendor preset: disabled)
   Drop-In: /run/systemd/system/etcd2.service.d
      |-20-cloudinit.conf

   Active: active (running) since Thu 2015-05-14 21:34:55 ; 6min 8s ago
   Main PID: 557 (etcd2)
   Memory: 2.9M
   CGroup: /system.slice/etcd2.service

May 14 21:34:55 core-01 etcd2[557]: 2015/05/14 21:34:55 etcdserver: restart member 1a71e8822544990be in cluster db58e65d91967d27c0 at commit index 1000
May 14 21:34:55 core-01 etcd2[557]: 2015/05/14 21:34:55 raft: 1a71e8822544990be become follower at term 2
May 14 21:34:55 core-01 etcd2[557]: 2015/05/14 21:34:55 raft: 1a71e8822544990be is starting a new election at term 2
May 14 21:34:55 core-01 etcd2[557]: 2015/05/14 21:34:55 raft: 1a71e8822544990be become candidate at term 3
May 14 21:34:55 core-01 etcd2[557]: 2015/05/14 21:34:55 raft: 1a71e8822544990be received vote from 1a71e8822544990be at term 3
May 14 21:34:55 core-01 etcd2[557]: 2015/05/14 21:34:55 raft: 1a71e8822544990be become leader at term 3
May 14 21:34:55 core-01 etcd2[557]: 2015/05/14 21:34:55 raft: leader 1a71e8822544990be elected leader 1a71e8822544990be at term 3
```

To check the status of fleet, type this:

$ systemctl status fleet

```
* fleet.service - fleet daemon
   Loaded: loaded (/usr/lib/systemd/system/fleet.service; static; vendor preset: disabled)
   Drop-In: /run/systemd/system/fleet.service.d
      |-20-cloudinit.conf

   Active: active (running) since Thu 2015-05-14 21:34:55 ; 6min ago
   Main PID: 509 (Fleetd)
   Memory: 7.2M
   CGroup: /system.slice/fleet.service

May 14 21:34:56 core-01 fleetd[509]: INFO manager.go:246: Writing systemd unit fleet-ui.service (5480)
May 14 21:34:56 core-01 fleetd[509]: INFO manager.go:182: Initiating systemd to reload units
May 14 21:34:56 core-01 fleetd[509]: INFO engine.go:183: Engine leadership acquired
May 14 21:34:56 core-01 fleetd[509]: INFO manager.go:127: Triggered systemd unit dockerui.service start...1005
May 14 21:34:56 core-01 fleetd[509]: INFO engine.go:183: Engine leadership acquired
May 14 21:34:56 core-01 fleetd[509]: INFO manager.go:127: Triggered systemd unit docker-ui.service start...1179
May 14 21:34:56 core-01 fleetd[509]: INFO reconcile.go:330: AgentReconciler completed tasks: type=Loading...added
May 14 21:34:56 core-01 fleetd[509]: INFO reconcile.go:330: AgentReconciler completed tasks: type=Loading...added
May 14 21:34:56 core-01 fleetd[509]: INFO reconcile.go:330: AgentReconciler completed tasks: type=Start...added
May 14 21:34:56 core-01 fleetd[509]: INFO reconcile.go:330: AgentReconciler completed tasks: type=Start...added
May 14 21:34:56 core-01 fleetd[509]: INFO reconcile.go:330: AgentReconciler completed tasks: type=Start...added
May 14 21:34:56 core-01 fleetd[509]: INFO reconcile.go:330: AgentReconciler completed tasks: type=Start...added
May 14 21:34:56 core-01 fleetd[509]: INFO reconcile.go:330: AgentReconciler completed tasks: type=Stop...added
May 14 21:34:56 core-01 fleetd[509]: INFO reconcile.go:330: AgentReconciler completed tasks: type=Stop...added
May 14 21:34:56 core-01 fleetd[509]: INFO reconcile.go:330: AgentReconciler completed tasks: type=Stop...added
May 14 21:34:56 core-01 fleetd[509]: INFO reconcile.go:330: AgentReconciler completed tasks: type=Stop...added
May 14 21:34:56 core-01 fleetd[509]: INFO reconcile.go:330: AgentReconciler completed tasks: type=Stop...added
```

Hint: Some lines were ellipsized; use -l to show in full.
To check the status of `docker`, type the following command:

```
$ docker version
```

![Docker version output]

Lovely! Everything looks fine. Thus, we've got our first CoreOS VM up and running in VirtualBox.

**Summary**

In this chapter, we saw what CoreOS is and how it is installed. We covered a simple CoreOS installation on a local computer with the help of Vagrant and VirtualBox, and checked whether `etcd`, `fleet`, and `docker` are running there.

You will continue to explore and learn about all CoreOS services in more detail in the upcoming chapters.
Where to buy this book

You can buy CoreOS Essentials from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.