



Kubernetes Management Design Patterns

With Docker, CoreOS Linux, and Other
Platforms

—
Deepak Vohra

Apress®

Kubernetes Management Design Patterns

With Docker, CoreOS Linux,
and Other Platforms



Deepak Vohra

Apress®

Kubernetes Management Design Patterns: With Docker, CoreOS Linux, and Other Platforms

Deepak Vohra
White Rock, British Columbia
Canada

ISBN-13 (pbk): 978-1-4842-2597-4
DOI 10.1007/978-1-4842-2598-1

ISBN-13 (electronic): 978-1-4842-2598-1

Library of Congress Control Number: 2017930815

Copyright © 2017 by Deepak Vohra

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

“Docker and the Docker logo are trademarks or registered trademarks of Docker, Inc. in the United States and/or other countries. Docker, Inc. and other parties may also have trademark rights in other terms used herein.”

Kubernetes is Apache License 2.0 software. Docker is Apache License 2.0 software. CoreOS is Apache License 2.0 software.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Cover image designed by Freepik.
Managing Director: Welmoed Spahr
Lead Editor: Steve Anglin
Development Editor: Matthew Moodie
Technical Reviewer: Massimo Nardone
Coordinating Editor: Mark Powers
Copy Editor: James A. Compton
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit <http://www.apress.com/us/services/rights-permission>.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers for download or cloning at Github via the book's product page, located at www.apress.com/9781484225974. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

Contents at a Glance

About the Author	xiii
About the Technical Reviewer	xv
Introduction	xvii
■ Part I: Platforms	1
■ Chapter 1: Kubernetes on AWS.....	3
■ Chapter 2: Kubernetes on CoreOS on AWS	23
■ Chapter 3: Kubernetes on Google Cloud Platform	49
■ Part II: Administration and Configuration.....	89
■ Chapter 4: Using Multiple Zones.....	91
■ Chapter 5: Using the Tectonic Console	117
■ Chapter 6: Using Volumes	135
■ Chapter 7: Using Services	153
■ Chapter 8: Using Rolling Updates	171
■ Chapter 9: Scheduling Pods on Nodes.....	199
■ Chapter 10: Configuring Compute Resources.....	237
■ Chapter 11: Using ConfigMaps	257
■ Chapter 12: Using Resource Quotas	279
■ Chapter 13: Using Autoscaling	299
■ Chapter 14: Configuring Logging.....	309

■ Part III: High Availability	333
■ Chapter 15: Using an HA Master with OpenShift	335
■ Chapter 16: Developing a Highly Available Website	355
Index	393

Contents

About the Author	xiii
About the Technical Reviewer	xv
Introduction	xvii
■ Part I: Platforms	1
■ Chapter 1: Kubernetes on AWS.....	3
Problem	3
Solution	3
Overview	3
Setting the Environment.....	4
Configuring AWS.....	7
Starting the Kubernetes Cluster	11
Testing the Cluster	17
Configuring the Cluster	18
Stopping the Cluster.....	21
Summary.....	22
■ Chapter 2: Kubernetes on CoreOS on AWS	23
Problem	23
Solution	23
Overview	24
Setting the Environment.....	25
Configuring AWS Credentials.....	25
Installing Kube-aws.....	25

Setting Up Cluster Parameters	27
Creating a KMS Key	28
Setting Up an External DNS Name.....	29
Creating the Cluster	29
Creating an Asset Directory	29
Initializing the Cluster CloudFormation.....	30
Rendering Contents of the Asset Directory.....	30
Customizing the Cluster.....	31
Validating the CloudFormation Stack	34
Launching the Cluster CloudFormation	34
Configuring DNS	35
Accessing the Cluster.....	39
Testing the Cluster	41
Summary.....	47
■ Chapter 3: Kubernetes on Google Cloud Platform	49
Problem	49
Solution	49
Overview	49
Setting the Environment.....	50
Creating a Project on Google Cloud Platform	50
Enabling Permissions.....	55
Enabling the Compute Engine API	56
Creating a VM Instance	62
Connecting to the VM Instance.....	66
Reserving a Static External IP Address.....	67
Creating a Kubernetes Cluster	67
Creating a Kubernetes Application and Service	71
Stopping the Cluster.....	75
Using Kubernetes with Google Container Engine	77

Creating a Google Container Cluster.....	77
Connecting to the Google Cloud Shell	80
Configuring kubectl	80
Testing the Kubernetes Cluster.....	81
Summary.....	87
■ Part II: Administration and Configuration.....	89
■ Chapter 4: Using Multiple Zones.....	91
Problem	91
Solution	92
Overview	93
Setting the Environment.....	93
Initializing a CloudFormation.....	95
Configuring cluster.yaml for Multiple Zones.....	95
Launching the CloudFormation	99
Configuring External DNS.....	100
Running a Kubernetes Application	101
Using Multiple Zones on AWS.....	103
Summary.....	116
■ Chapter 5: Using the Tectonic Console	117
Problem	117
Solution	117
Overview	118
Setting the Environment.....	118
Downloading the Pull Secret and the Tectonic Console Manifest	120
Installing the Pull Secret and the Tectonic Console Manifest.....	122
Accessing the Tectonic Console	123
Using the Tectonic Console	124
Removing the Tectonic Console.....	134
Summary.....	134

■ Chapter 6: Using Volumes	135
Problem	135
Solution	135
Overview	136
Setting the Environment.....	137
Creating an AWS Volume	139
Using an awsElasticBlockStore Volume	141
Creating a Git Repo	145
Using a gitRepo Volume.....	149
Summary.....	152
■ Chapter 7: Using Services	153
Problem	153
Solution	154
Overview	154
Setting the Environment.....	155
Creating a ClusterIP Service.....	156
Creating a NodePort Service	159
Creating a LoadBalancer Service	166
Summary.....	170
■ Chapter 8: Using Rolling Updates	171
Problem	171
Solution	171
Overview	172
Setting the Environment.....	173
Rolling Update with an RC Definition File.....	174
Rolling Update by Updating the Container Image.....	177
Rolling Back an Update	184
Using Only Either File or Image	186

Multiple-Container Pods.....	186
Rolling Update to a Deployment.....	186
Summary.....	198
■ Chapter 9: Scheduling Pods on Nodes.....	199
Problem.....	199
Solution.....	199
Overview.....	200
Defining a Scheduling Policy.....	200
Setting the Environment.....	202
Using the Default Scheduler.....	203
Scheduling Pods without a Node Selector.....	213
Setting Node Labels.....	213
Scheduling Pods with a Node Selector.....	214
Setting Node Affinity.....	220
Setting requiredDuringSchedulingIgnoredDuringExecution.....	222
Setting preferredDuringSchedulingIgnoredDuringExecution.....	229
Summary.....	236
■ Chapter 10: Configuring Compute Resources.....	237
Problem.....	237
Solution.....	237
Overview.....	238
Types of Compute Resources.....	239
Resource Requests and Limits.....	240
Quality of Service.....	242
Setting the Environment.....	243
Finding Node Capacity.....	244
Creating a Pod with Resources Specified.....	245
Limit on Number of Pods.....	252

Overcommitting Resource Limits	254
Reserving Node Resources	254
Summary	256
■ Chapter 11: Using ConfigMaps	257
Problem	257
Solution	257
Overview	257
Kubectl create configmap Command	258
Setting the Environment.....	258
Creating ConfigMaps from Directories	259
Creating ConfigMaps from Files	266
Creating ConfigMaps from Literal Values	270
Consuming a ConfigMap in a Volume.....	274
Summary.....	277
■ Chapter 12: Using Resource Quotas	279
Problem.....	279
Solution	279
Overview	280
Setting the Environment.....	281
Defining Compute Resource Quotas.....	282
Exceeding Compute Resource Quotas	284
Defining Object Quotas.....	288
Exceeding Object Quotas.....	290
Defining Best-Effort Scope Quotas.....	294
Summary.....	298
■ Chapter 13: Using Autoscaling	299
Problem.....	299
Solution	299

Overview	300
Setting the Environment.....	300
Running a PHP Apache Server Deployment	302
Creating a Service	302
Creating a Horizontal Pod Autoscaler	303
Increasing Load	306
Summary	308
■ Chapter 14: Configuring Logging.....	309
Problem	309
Solution	309
Overview	310
Setting the Environment.....	311
Getting the Logs Generated by Default Logger.....	311
Docker Log Files.....	313
Cluster-Level Logging with Elasticsearch and Kibana	314
Starting a Replication Controller.....	315
Starting Elastic Search	318
Starting Fluentd to Collect Logs	322
Starting Kibana.....	324
Summary.....	331
■ Part III: High Availability.....	333
■ Chapter 15: Using an HA Master with OpenShift.....	335
Problem	335
Solution	335
Overview	336
Setting the Environment.....	336
Installing the Credentials.....	338
Installing the Network Manager	339

- Installing OpenShift via Ansible on the Client Machine 339
- Configuring Ansible 342
- Running the Ansible Playbook..... 346
- Testing the Cluster 347
- Testing the High Availability 349
- Summary..... 353
- Chapter 16: Developing a Highly Available Website 355**
 - Problem 355
 - Solution 355
 - Overview 356
 - Setting the Environment..... 357
 - Creating CloudFormations..... 357
 - Configuring External DNS..... 361
 - Creating a Kubernetes Service..... 362
 - Creating an AWS Route 53 Service..... 367
 - Creating a Hosted Zone 368
 - Configuring Name Servers on a Domain Name 369
 - Creating Record Sets..... 374
 - Testing High Availability 384
 - Summary..... 392
- Index..... 393**

About the Author



Deepak Vohra is a consultant and a principal member of the NuBean software company. Deepak is a Sun-certified Java programmer and Web component developer. He has worked in the fields of XML, Java programming, and Java EE for over seven years. Deepak is the coauthor of *Pro XML Development with Java Technology* (Apress, 2006). Deepak is also the author of *JDBC 4.0* and *Oracle JDeveloper for J2EE Development, Processing XML Documents with Oracle JDeveloper 11g, EJB 3.0 Database Persistence with Oracle Fusion Middleware 11g, and Java EE Development in Eclipse IDE* (Packt Publishing). He also served as the technical reviewer on *WebLogic: The Definitive Guide* (O'Reilly Media, 2004) and *Ruby Programming for the Absolute Beginner* (Cengage Learning PTR, 2007).

About the Technical Reviewer



Massimo Nardone has more than 22 years of experiences in Security, Web/Mobile development, Cloud and IT Architecture. His true IT passions are security and Android.

He has been programming and teaching how to program with Android, Perl, PHP, Java, VB, Python, C/C++ and MySQL for more than 20 years.

He holds a Master of Science degree in Computing Science from the University of Salerno, Italy.

He has worked as a Project Manager, Software Engineer, Research Engineer, Chief Security Architect, Information Security Manager, PCI/SCADA Auditor and Senior Lead IT Security/Cloud/SCADA Architect for many years.

Technical skills include: Security, Android, Cloud, Java, MySQL, Drupal, Cobol, Perl, Web and Mobile development, MongoDB, D3, Joomla, Couchbase, C/C++, WebGL, Python, Pro Rails, Django CMS, Jekyll, Scratch, etc.

He currently works as Chief Information Security Office (CISO) for Cargotec Oyj.

He worked as visiting lecturer and supervisor for exercises at the Networking Laboratory of the Helsinki University of Technology (Aalto University). He holds four international patents (PKI, SIP, SAML and Proxy areas).

Massimo has reviewed more than 40 IT books for different publishing company and he is the coauthor of *Pro Android Games* (Apress, 2015).

Introduction

Docker was made available as open source in March 2013 and has become the most commonly used containerization platform. Kubernetes was open-sourced in June 2014 and has become the most widely used container cluster manager. The first stable version of CoreOS Linux was made available in July 2014 and since has become the most commonly used operating system for containers. My first book, *Kubernetes Microservices with Docker* (Apress, 2016), is an introduction to creating microservices with Kubernetes and Docker. This book, *Kubernetes Management Design Patterns*, takes container cluster management to the next level and discusses all or most aspects of administering and configuring Kubernetes on CoreOS and applying suitable design patterns such as ConfigMaps, autoscaling, resource quotas, and high availability. Kubernetes is a cluster manager for Docker and rkt containers, but this book discusses Kubernetes in the context of Docker only. A cluster manager for Docker containers is needed because the Docker engine by itself lacks some functionality, such as the ability to scale a cluster of containers, schedule pods on nodes, or mount a certain type of storage (such as an AWS Volume or Github repo) as volumes. Docker Engine 1.12 integrates the Docker Swarm cluster manager and Docker Swarm does overcome some of the earlier limitations of Docker by providing replication, load balancing, fault tolerance, and service discovery, but Kubernetes provides some features suitable for developing object-oriented applications. The Pod abstraction is the atomic unit of deployment in Kubernetes. A Pod may consist of one or more containers. Co-locating containers has several advantages as containers in a Pod share the same networking and filesystem and run on the same node. Docker Swarm does not support autoscaling directly. While Docker Swarm is Docker native, Kubernetes is more production-ready having been used in production at Google for more than 15 years.

Kubernetes Design Patterns

A software design pattern is a general reusable solution to a commonly occurring problem within a given context in software design.

Wikipedia

A Docker image includes instructions to package all the required software and dependencies, set the environment variables, and run commands, and it is a reusable encapsulation of software for modular design. The atomic unit of modular container service in Kubernetes is a *pod*, which is a group of containers with a common filesystem and networking. The Kubernetes pod abstraction enables design patterns for containerized applications similar to object oriented design patterns. Pod, service, replication controller, deployment, and ConfigMap are all types of Kubernetes objects. Further, because containers interact with each other over HTTP, making use of a commonly available data format such as JSON, Kubernetes design

patterns are language and platform independent. Containers provide some of the same benefits as software objects such as modularity or packaging, abstraction and reuse. Kubernetes has described three classes or types of patterns.

- Management design patterns
- Patterns involving multiple cooperating containers running on the same node
- Patterns involving containers running across multiple nodes

Some of the benefits of modular containers are as follows:

- The container boundary is an encapsulation or abstraction boundary that can be used to build modular, reusable components.
- The reusable containers may be shared between different applications and agile developer teams.
- Containers speed application development.
- Containers are suitable for agile team development.
- Containers can be used to encapsulate a best design or implementation.
- Containers provide separation of concerns

The design patterns are introduced in the publication *Design Patterns For Container-Based Distributed Systems*, by Brendan Burns and David Oppenheimer (<https://www.usenix.org/node/196347>). In this book we shall be using some of these and other design patterns.

Kubernetes Architecture

A Kubernetes cluster consists of a single *master node* (unless a high-availability master is used, which is not the default) and one or more *worker nodes* with Docker installed on each node. The following components run on each master node:

- *etcd* to store the persistent state of the master including all configuration data. A high-availability etcd cluster can also be used.
- An *API Server* to serve up the Kubernetes REST API for Kubernetes objects (pods, services, replication controllers, and others).
- *Scheduler* to bind unassigned pods on nodes.
- *Controller manager* performs all cluster level operations such as create and update service endpoints, discover, manage and monitor nodes. The replication controller is used to scale pods in a cluster.

The following components are run on each worker node:

- *kubelet* to manage the pods (including containers), Docker images, and volumes. The kubelet is managed from the API Server on the master node.
- *kube-proxy* is a network proxy and load balancer to serve up services.

The Kubernetes architecture is shown in Figure I-1.

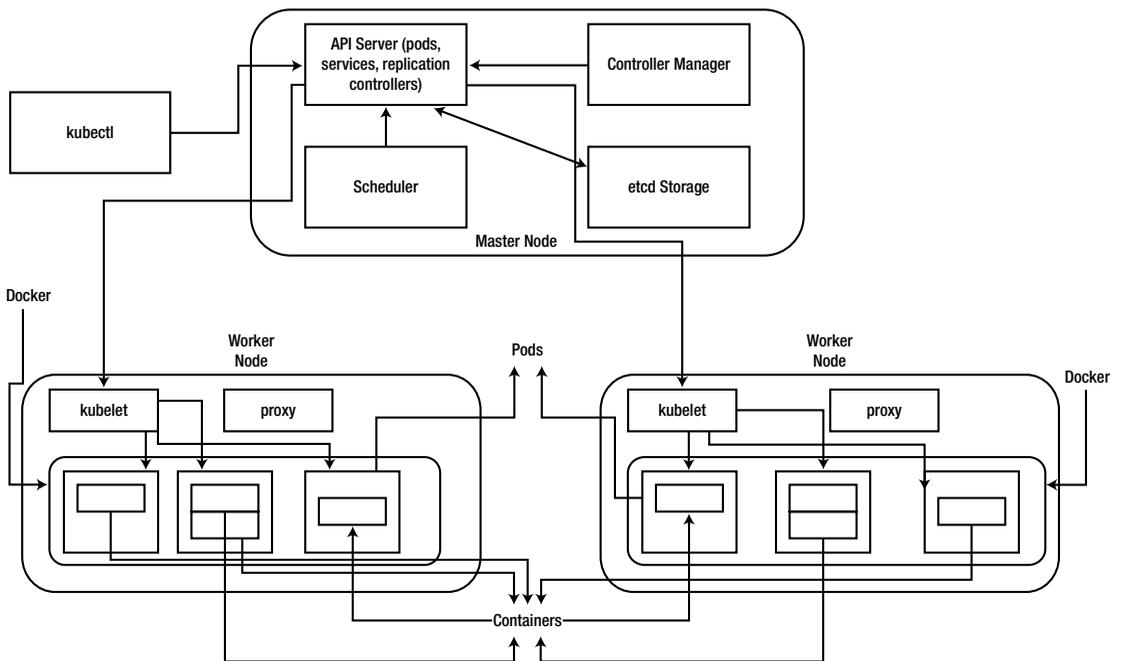


Figure I-1. Kubernetes Architecture

Why CoreOS?

CoreOS is the most widely used Linux OS designed for containers, not just Docker containers but also rkt (an implementation of the APP Container spec) containers. Docker and rkt are pre-installed on CoreOS out-of-the-box. CoreOS supports most cloud providers including Amazon Web Services (AWS) Elastic Compute Cloud (EC2), Google Cloud Platform, and virtualization platforms such as VMWare and VirtualBox. CoreOS provides Cloud-Config for declaratively configuring for OS items such as network configuration (flannel), storage (etcd), and user accounts. CoreOS provides a production-level infrastructure for containerized applications including automation, security and scalability. CoreOS has been leading the drive for container industry standards and in fact founded appc. CoreOS is not only the most widely used operating system for containers but also the most advanced container registry, Quay. CoreOS provides server security with Distributed Trusted Computing. CoreOS also provides Tectonic Enterprise for enterprise-level workloads without operational overhead and an out-of-the-box Kubernetes cluster and a user-friendly dashboard.

Chapter Description

In Chapter 1 we shall install Kubernetes on Amazon Web Services (AWS), create a sample deployment and service, and subsequently invoke the service. Kubernetes installation on AWS requires almost no configuration to spin-up a multi-node cluster.

In Chapter 2 we shall install Kubernetes on CoreOS, which is the main platform we shall use for most of the chapters. We'll first create an AWS EC2 instance from Amazon Linux AMI, which has the AWS Command Line Interface (CLI) preinstalled. We'll then SSH log in to the EC2 instance and install Kube-aws. Then we will launch a CloudFormation for a Kubernetes cluster with one controller node and three worker nodes and SSH log in to the controller instance and install kubectl binaries to access the API server.

In Chapter 3 we shall discuss Google Cloud Platform for Kubernetes. First, create a project and a VM instance. Subsequently connect to the VM instance to create a Kubernetes cluster and test a sample application.

In Chapter 4 we shall use multiple zones to create an AWS CloudFormation for a Kubernetes cluster.

Chapter 5 introduces the Tectonic Console for managing Kubernetes applications deployed on CoreOS.

Chapter 6 is on volumes. We demonstrate using volumes with two types of volumes:

awsElasticBlockStore volume and gitRepo volume.

Chapter 7 is on using services. We shall create sample services for three kinds of services supported by Kubernetes: ClusterIP, NodePort and LoadBalancer.

In Chapter 8 we shall discuss rolling updates. A rolling update is the mechanism by which a running replication controller can be updated to a newer image or specification while it is running.

In Chapter 9 we introduce the scheduling policy used by Kubernetes to schedule pods on nodes. We discuss the various options including using a NodeSelector, and setting node affinity.

Chapter 10 is on allocating compute resources to applications. The two supported compute resources are CPU and memory. We shall discuss setting resource requests and limits and also how Kubernetes provides a quality of service by guaranteeing a preset level of resources.

Chapter 11 is on ConfigMaps, which are maps of configuration properties that may be used in pods and replication controller definition files to set environment variables, command arguments and such.

Chapter 12 is on setting resource quotas on namespaces for constraining resource usage in a namespace. Resource quotas are useful in team development (different teams have different requirements) and different phases of application which have different resource requirements such as development, testing, and production.

Chapter 13 is on autoscaling, which is suitable for production workloads that can fluctuate. Autoscaling of a deployment, replica set, or replication controller scales the number of pods in a cluster automatically when the load fluctuates.

Chapter 14 is on logging. The default logger is discussed in addition to cluster-level logging using Elasticsearch, Fluentd, and Kibana.

In Chapter 15 OpenShift, a PaaS platform for Kubernetes, is discussed to create a high availability master Kubernetes cluster using Ansible. Ansible is an automation platform for application deployment, configuration management, and orchestration.

In Chapter 16 a high availability web site is developed using AWS Route 53 for DNS failover.

PART I



Platforms

CHAPTER 1



Kubernetes on AWS

Kubernetes is a cluster manager for Docker (and rkt) containers. The Introduction outlines its basic architecture and relationship to CoreOS and Amazon Web Services (AWS). In this chapter we'll spin up a basic cluster without configuration.

■ **Note** *Kubernetes Microservices with Docker* (Apress, 2016) covers installing Kubernetes on single-node and multi-node clusters.

Problem

Installing Kubernetes by installing its individual components (Docker, Flannel, Kubelet, and Service Proxy) separately is an involved process that requires many commands to be run and files to be configured.

Solution

AWS provides a legacy tool called `kube-up.sh` to spin up a Kubernetes cluster without requiring any configuration. Only an AWS account, the AWS Command Line Interface (CLI), and access to the AWS APIs are required. Kubernetes and other tools such as Elasticsearch (used to index and store logs), Heapster (used to analyze compute resource usage), Kibana (a GUI dashboard used to view the logs), KubeDNS (used to resolve DNS names for services), Kubernetes-dashboard, Grafana (used for metrics visualization), and InfluxDB are all installed with a single command.

Overview

In this chapter we will create a multi-node cluster (consisting of one master and multiple minions) on Amazon Elastic Compute Cloud (EC2) using the AWS Command Line Interface. The stages are as follows:

- Setting the Environment

- Starting a Cluster

- Testing the Cluster

- Configuring the Cluster

- Stopping the Cluster

Setting the Environment

Because we’re using Amazon EC2, an AWS account is required. Also, to configure AWS you need to obtain security credentials. Select Security Credentials for a user account. In the Your Security Credentials screen, select the Access Keys node and click Create New Access Key as shown in Figure 1-1 to create a new access key.

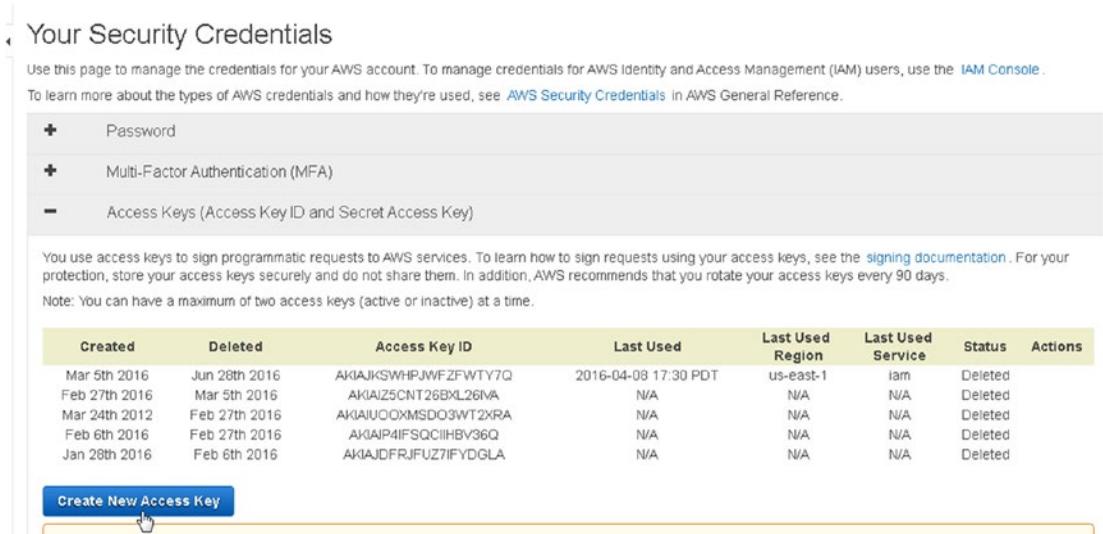


Figure 1-1. Creating a new access key

A new security access key is created and the Access Key ID and Secret Access Key are listed.

Copy the Access Key ID and Secret Access Key to be used later to configure AWS. The Access Key ID and Secret Access Key will be different for different users.

```
AWS_ACCESS_KEY_ID AKIAISQVxxxxxxxxxxxxxxxxx
AWS_SECRET_ACCESS_KEY VuJD5gDxxxxxxxxxxxxxxxxx
```

Because the AWS Command Line Interface is required, create an EC2 instance of the Amazon Linux Amazon Machine Image (AMI), which has the AWS CLI preinstalled. Click on Launch Instance as shown in Figure 1-2 to create a new instance.



Figure 1-2. Launching an EC2 instance

In the next screen, select the Amazon Linux AMI (64 bit) as shown in Figure 1-3.

Step 1: Choose an Amazon Machine Image (AMI) Cancel and Exit

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace, or you can select one of your own AMIs.

Quick Start |< < 1 to 25 of 25 AMIs > >|

AMI Name	Description	Root device type	Virtualization type	Architecture
Amazon Linux Amazon Linux AMI 2016.03.3 (HVM), SSD Volume Type - ami-8886aa05	The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.	ebs	hvm	64-bit
Red Hat Red Hat Enterprise Linux 7.2 (HVM), EBS General Purpose (SSD) Volume Type	Red Hat Enterprise Linux version 7.2 (HVM), EBS General Purpose (SSD) Volume Type	ebs	hvm	64-bit
SUSE Linux SUSE Linux Enterprise Server 12 SP1 (HVM), SSD Volume Type - ami-b7b4fedd	SUSE Linux Enterprise Server 12 Service Pack 1 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.	ebs	hvm	64-bit

Figure 1-3. Selecting Amazon Linux AMI

For the Instance Type, select a relatively large Instance size (m4.xlarge) as shown in Figure 1-4, because the default (Free Tier) micro size may not provide sufficient memory or disk space to install Kubernetes. Some of the instance types such as m4.xlarge may only be launched in a virtual private cloud (VPC). When you are ready, click Next: Configure Instance Details.

Step 2: Choose an Instance Type

Instance Type	Instance Class	VCPU	Memory (GiB)	Storage	Network	Availability	
<input type="checkbox"/>	General purpose	m4.large	2	8	EBS only	Yes	Moderate
<input checked="" type="checkbox"/>	General purpose	m4.xlarge	4	16	EBS only	Yes	High
<input type="checkbox"/>	General purpose	m4.2xlarge	8	32	EBS only	Yes	High
<input type="checkbox"/>	General purpose	m4.4xlarge	16	64	EBS only	Yes	High
<input type="checkbox"/>	General purpose	m4.10xlarge	40	160	EBS only	Yes	10 Gigabit
<input type="checkbox"/>	General purpose	m3.medium	1	3.75	1 x 4 (SSD)	-	Moderate
<input type="checkbox"/>	General purpose	m3.large	2	7.5	1 x 32 (SSD)	-	Moderate
<input type="checkbox"/>	General purpose	m3.xlarge	4	15	2 x 40 (SSD)	Yes	High
<input type="checkbox"/>	General purpose	m3.2xlarge	8	30	2 x 80 (SSD)	Yes	High
<input type="checkbox"/>	Compute optimized	c4.large	2	3.75	EBS only	Yes	Moderate

Cancel Previous Review and Launch Next: Configure Instance Details

Figure 1-4. Choosing an instance type

Specify the instance details such as Network VPC and Subnet as shown in Figure 1-5. When finished, click Next: Add Storage.

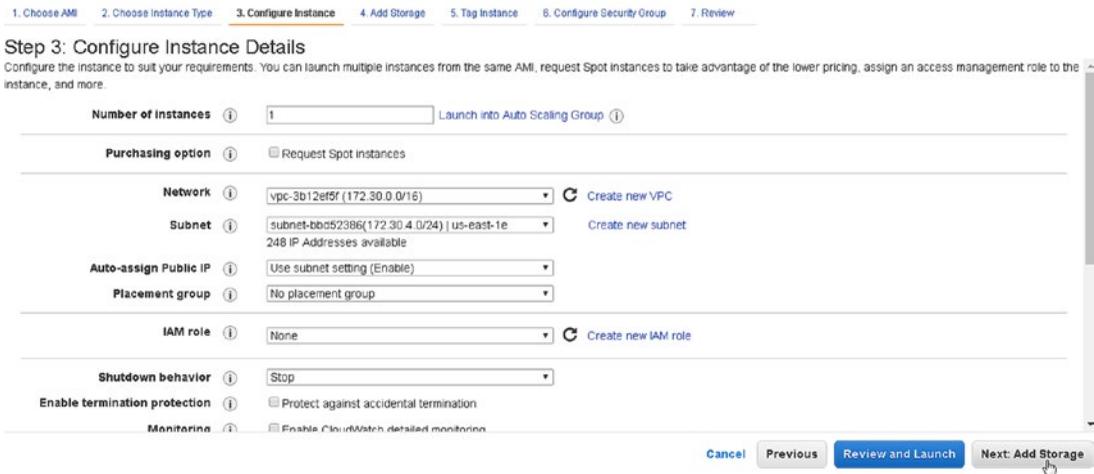


Figure 1-5. Configuring instance details

A new EC2 instance is created. Obtain the Public DNS for the instance as shown in Figure 1-6.

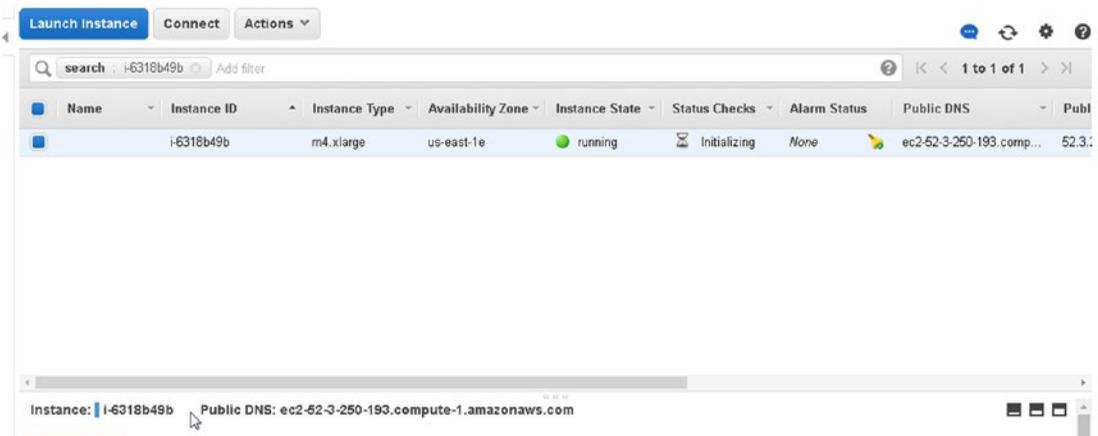


Figure 1-6. The public DNS

Using the private key that was specified when the instance was created, SSH log in to the instance:

```
ssh -i "docker.pem" ec2-user@ec2-52-3-250-193.compute-1.amazonaws.com
```

The Amazon Linux command prompt is displayed as shown in Figure 1-7.

```

ec2-user@ip-172-30-4-159:~
File Edit View Search Terminal Help
[root@localhost ~]# ssh -i "docker.pem" ec2-user@ec2-52-3-250-193.compute-1.ama
zonaws.com
The authenticity of host 'ec2-52-3-250-193.compute-1.amazonaws.com (52.3.250.193
)' can't be established.
RSA key fingerprint is ad:09:a9:d7:65:03:4f:fe:ba:f7:a9:ad:83:e4:64:9a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-52-3-250-193.compute-1.amazonaws.com,52.3.250.19
3' (RSA) to the list of known hosts.

  _ | ( _ | _ )
  _ | ( _ | /   Amazon Linux AMI
  _ |\ _ | _ |

https://aws.amazon.com/amazon-linux-ami/2016.03-release-notes/
1 package(s) needed for security, out of 1 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-30-4-159 ~]$ █

```

Figure 1-7. Amazon Linux AMI command prompt

Configuring AWS

When a Kubernetes cluster is started on AWS EC2, a new VPC is created for the master and minion nodes. The number of VPCs that may be created in an AWS account has a limit, which can vary for different users. Before starting the cluster, delete the VPCs that are not being used so that the limit is not reached when a new VPC is created. To begin, select VPC in the AWS Services as shown in Figure 1-8.

History



VPC



EC2



IAM



Console Home



Device Farm

All AWS Services

Compute

Storage & Content Delivery

Database

Networking

Developer Tools

Management Tools

Security & Identity

Analytics

Internet of Things

Mobile Services

Application Services

Enterprise Applications

Game Development

Figure 1-8. *Selecting the VPC console*

Click on Start VPC Wizard as shown in Figure 1-9 to list and delete VPCs if required.

Resources ↻

[Start VPC Wizard](#) [Launch EC2 Instances](#)

Note: Your Instances will launch in the US East (N. Virginia) region.

You are using the following Amazon VPC resources in the US East (N. Virginia) region:

2 VPCs	2 Internet Gateways
8 Subnets	3 Route Tables
2 Network ACLs	0 Elastic IPs
0 VPC Peering Connections	0 Endpoints
0 Nat Gateways	80 Security Groups
1 Running Instance	0 VPN Connections
0 Virtual Private Gateways	0 Customer Gateways

VPN Connections

Amazon VPC enables you to use your own isolated resources within the AWS cloud, and then connect those resources directly to your own datacenter using industry-standard encrypted IPsec VPN connections.

[Create VPN Connection](#)

Service Health

Current Status	Details
✓ Amazon VPC - US East (N. Virginia)	Service is operating normally
✓ Amazon EC2 - US East (N. Virginia)	Service is operating normally

[View complete service health details](#)

Additional Information

[VPC Documentation](#)
[All VPC Resources](#)
[Forums](#)
[Report an Issue](#)

Figure 1-9. Starting the VPC Wizard

The VPCs already available are listed as shown in Figure 1-10.

Name	VPC ID	State	VPC CIDR	DHCP options set	Route table	Network ACL	Tenancy	Def
kubernetes-vpc	vpc-e6bb0781	available	172.20.0.0/16	dopt-2d345a49 kubern...	rtb-aebfb8c9	acl-bbb1d7dc	Default	No
vpc-3b12e5f	vpc-3b12e5f	available	172.30.0.0/16	dopt-09b9476c	rtb-cb369af	acl-a92ad2cd	Default	No

Figure 1-10. Available VPCs

To delete a VPC, select the VPC and click Actions ► Delete VPC, as shown in Figure 1-11.



Figure 1-11. Selecting Actions ► Delete VPC

In the confirmation screen that appears, click Yes, Delete. If the VPC is not associated with any instance, the VPC should start to be deleted as shown in Figure 1-12.

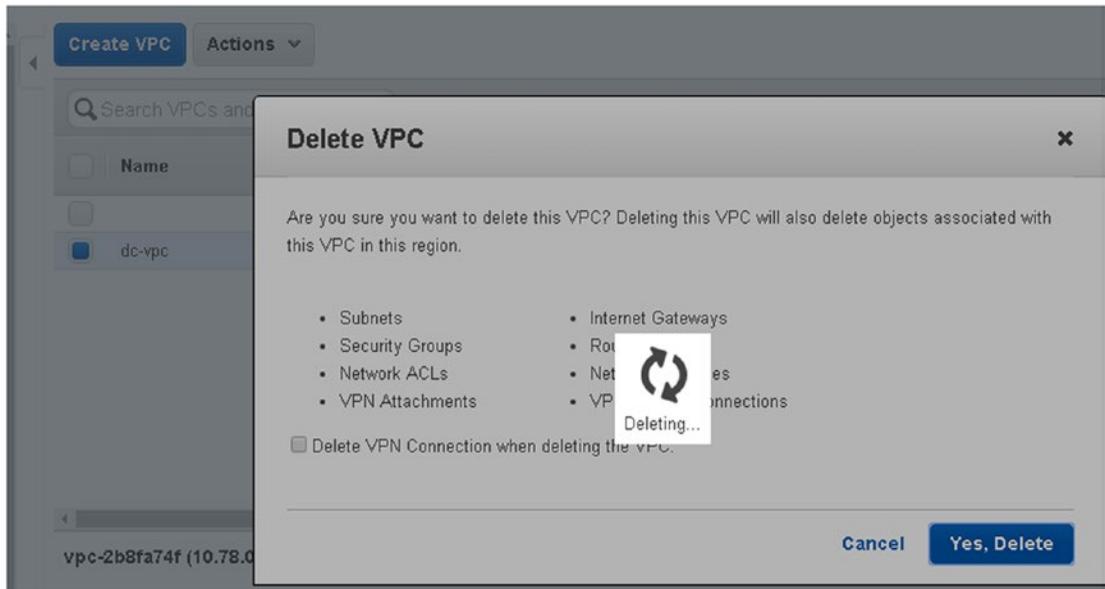


Figure 1-12. Deleting a VPC

If a VPC is associated with any instance, then it is not deletable and the Yes, Delete button is unavailable, as shown in Figure 1-13.

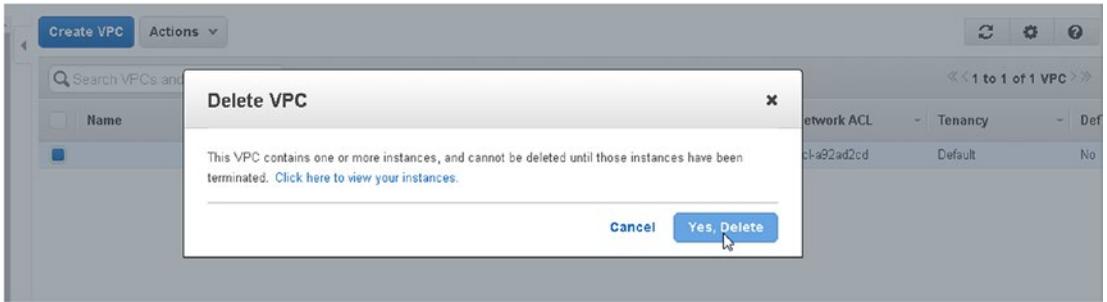


Figure 1-13. The message for a nondeletable VPC

Next, configure AWS on the Amazon Linux instance using the following command:

```
aws configure
```

When prompted, specify the AWS Access Key ID and AWS Access Key. Also specify the default region name (us-east-1) and the default output format (json) as shown in Figure 1-14.

```
[ec2-user@ip-172-30-4-159 ~]$ aws configure
AWS Access Key ID [None]: AKIAISQVSTC
AWS Secret Access Key [None]: VuJD5gDzCBZ2v
Default region name [None]: us-east-1
Default output format [None]: json
[ec2-user@ip-172-30-4-159 ~]$
```

Figure 1-14. Configuring AWS

Starting the Kubernetes Cluster

Now that you have configured AWS, run the following command to install Kubernetes:

```
export KUBERNETES_PROVIDER=aws; wget -q -O - https://get.k8s.io | bash
```

This command starts the Kubernetes installation process as shown in Figure 1-15.

```
[ec2-user@ip-172-30-4-159 ~]$ aws configure
AWS Access Key ID [None]: AKIAISQVSTC
AWS Secret Access Key [None]: VuJD5g
Default region name [None]: us-east-1
Default output format [None]: json
[ec2-user@ip-172-30-4-159 ~]$ export KUBERNETES_PROVIDER=aws; wget -q -O - https://get.k8s.io | bash
```

Figure 1-15. Installing Kubernetes

The preceding command invokes the `cluster/kube-up.sh` script, which further invokes the `cluster/aws/util.sh` script using the configuration specified in the `cluster/aws/config-default.sh` script. One master and four minions are started on Debian 8 (jessie) as shown in Figure 1-16. The cluster initialization is started subsequently.

```

Sleeping for 3 seconds...
Waiting for instance i-04978291 to be running (currently pending)
Sleeping for 3 seconds...
Waiting for instance i-04978291 to be running (currently pending)
Sleeping for 3 seconds...
[master running]
Attaching IP 50.112.79.71 to instance i-04978291
Attaching persistent data volume (vol-c026ee75) to master
2016-06-28T18:42:25.708Z /dev/sdb i-04978291 attaching v
ol-c026ee75
cluster "aws_kubernetes" set.
user "aws_kubernetes" set.
context "aws_kubernetes" set.
switched to context "aws_kubernetes".
user "aws_kubernetes-basic-auth" set.
Wrote config for aws_kubernetes to /home/ec2-user/.kube/config
Creating minion configuration
Creating autoscaling group
 0 minions started; waiting
 4 minions started; ready
Waiting for cluster initialization.

This will continually check to see if the API for kubernetes is reachable.
This might loop forever if there was some uncaught error during start
up.

.....█

```

Figure 1-16. Starting master and minions

The cluster is started and validated, and the components installed are listed. The URLs at which the Kubernetes master, Elasticsearch, Heapster, and other services are running are listed as shown in Figure 1-17. The directory path at which the Kubernetes binaries are installed is also listed.