

# Designing AWS Environments

Architect large-scale cloud infrastructures with AWS



Packt

[www.packt.com](http://www.packt.com)

Mitesh Soni and Wayde Gilchrist

# Designing AWS Environments

Architect large-scale cloud infrastructures with AWS

**Mitesh Soni**  
**Wayde Gilchrist**

**Packt**

BIRMINGHAM - MUMBAI

# Designing AWS Environments

Copyright © 2018 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

**Commissioning Editor:** Vijn Boricha

**Acquisition Editor:** Rahul Nair

**Content Development Editor:** Deepti Thore

**Technical Editor:** Sayali Thanekar

**Copy Editor:** Safis Editing

**Project Coordinator:** Kinjal Bari

**Proofreader:** Safis Editing

**Indexer:** Mariammal Chettiar

**Graphics:** Jisha Chirayil

**Production Coordinator:** Nilesh Mohite

First published: September 2018

Production reference: 1290918

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-78953-554-9

[www.packtpub.com](http://www.packtpub.com)



[mapt.io](http://mapt.io)

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

## Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

## Packt.com

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at [www.packt.com](http://www.packt.com) and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at [customercare@packtpub.com](mailto:customercare@packtpub.com) for more details.

At [www.packt.com](http://www.packt.com), you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

# Contributors

## About the authors

**Mitesh Soni** is an avid learner with 10 years' experience in the IT industry. He is an SCJP, SCWCD, and VCP. He is IBM Urbancode- and IBM Bluemix-certified, and is also a Certified Jenkins Engineer. He loves DevOps and cloud computing, and he also has an interest in programming in Java. He finds design patterns fascinating and believes that a picture is worth a thousand words. He occasionally contributes to clean-clouds and e-Tutorials World websites. He loves to play with his kids, fiddle with his camera, and take photographs at Indroda Park.

*Dedicated to Shreyu, Jigi, Parents, Grand Parents, Priyanka, Varsha, Radhika+Mukund, Mayur, Ashish, Navrang, Dharmesh, Vinay Kher, Yohan, Rohini, Teachers, Anupama+Mihir and Priyanka+Hemant, Sourabh, Gowri, Sudeep, Aishwarya, and Rohan.*

**Wayde Gilchrist** started moving customers of his IT consulting business into the cloud and away from traditional hosting environments back in 2010. In addition to consulting, he delivers AWS training for Fortune 500 companies, government agencies, and international consulting firms. When he is not out visiting customers, he is delivering training virtually from his home in Florida.

## About the reviewer

**Sunil Gulabani** is a software engineer based in India. He is currently working on Java EE and the AWS cloud platform. He is also a cloud evangelist who helps IT professionals to leverage the AWS cloud platform for their business needs. He has insightful knowledge on designing microservices, system architecture and integration, data modeling, relational databases, and NoSQL, so as to enable applications to achieve high throughput.

## Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit [authors.packtpub.com](http://authors.packtpub.com) and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

# Table of Contents

<b>Preface</b>	1
<b>Chapter 1: Installation and Setup</b>	5
<b>Opening an AWS account</b>	6
<b>The AWS Management Console</b>	14
<b>Summary</b>	23
<b>Chapter 2: Launching an EC2 Instance</b>	24
<b>EC2 instance types</b>	24
General purpose instance	24
T3s – burstable general-purpose instance type	25
T2s – burstable general-purpose instance type	25
M5	26
M4	27
Compute optimized	28
C5	28
C4	29
Memory-optimized	30
X1e	30
X1	31
R5	31
R4	32
z1d	33
Accelerated computing – general purpose GPU instances	34
P3	34
P2	34
G3	35
F1	36
Storage-optimized instance types	36
H1	36
I3	37
D2	38
<b>Launching the instance</b>	39
<b>EC2 storage options</b>	43
Instance storage	43
Elastic block storage	44
General purpose SSD	45
Provisioned IOPS SSD	45
Throughput optimized HDD	45
<b>Security groups</b>	46
<b>AMIs</b>	52
Quick start	53

Community AMIs	55
AWS marketplace	57
My AMIs	59
<b>Summary</b>	61
<b>Chapter 3: Logging in to EC2 Instances</b>	62
<b>Key pairs</b>	62
<b>Logging in to Linux instances</b>	67
<b>Logging in to Windows instances</b>	78
<b>Summary</b>	85
<b>Chapter 4: Networking on AWS</b>	86
<b>CIDR</b>	86
IPv4	87
Valid private IP address ranges	88
EC2 IP addressing	90
Private IP addresses	90
Public IP addresses	91
Elastic IP addresses	91
Elastic network interface (ENI)	92
<b>Subnets and route tables</b>	93
What are subnets?	94
Route tables	94
Difference between public and private subnets	96
NAT instance	97
<b>Summary</b>	100
<b>Chapter 5: Creating a VPC</b>	101
<b>Getting started with VPCs</b>	101
Classic EC2s	102
EC2s in a VPC	102
The default VPC	103
<b>Creating a VPC demo</b>	108
Create VPC using Wizard	108
<b>Connecting to a VPC</b>	126
Internet gateway	127
Software VPN	128
Virtual gateway	128
Direct connect	129
VPC peering	130
<b>Securing your VPC</b>	130
NACLs	130
Bastion instances	135
<b>Highly available architectures</b>	136
Availability zones	136
Elastic load balancer	138

*Table of Contents*

---

Load balancing stateful applications	138
Auto scaling	140
<b>Summary</b>	154
<b>Other Books You May Enjoy</b>	155
<b>Index</b>	158

# Preface

**Amazon Web Services (AWS)** provides trusted, cloud-based solutions to help you meet your business needs. Running your solutions on AWS can help you get your applications up and running faster while providing the security necessary to meet your compliance requirements.

This book begins by familiarizing you with the key capabilities to architect and host applications, websites, and services on AWS. We'll explain the available options for virtual instances and demonstrate launching and connecting to them. Using practical examples, you will be able to design and deploy networking and hosting solutions for large deployments. Finally, the book focuses on security and the important elements of scalability and high availability.

## Who this book is for

This book is for new and aspiring individuals who are gearing up for a solutions architect role. You'll also find this useful if you're an IT professional or DevOps engineer who is preparing to design and deploy large solutions on AWS. No experience with AWS is required.

## What this book covers

Chapter 1, *Installation and Setup*, helps us understand how to sign up for a free AWS account, and how to use the Management Console.

Chapter 2, *Launching an EC2 Instance*, provides us with information on how to launch an EC2 instance, and during that process, we will learn about AMIs, instance types, storage options, and security groups.

Chapter 3, *Logging in to EC2 Instances*, teaches us about key pairs, which we will then use to authenticate an SSH to Linux instances. Finally, we will use them to decrypt the administrator password and remote desktop to a Windows instance.

Chapter 4, *Networking on AWS*, covers designating private IP address ranges for your VPC. We also cover the three types of IP address used for EC2s, as well as elastic network interfaces, which hold the instance's network attributes for IP connections. We discuss subnets and route tables, and how a route in the route table can make a subnet public or private. Then, we'll talk about NAT instances and NAT gateways, to give instances and private subnets access to the internet.

Chapter 5, *Creating a VPC*, covers classic EC2s and EC2s in a VPC, using the default VPC, creating your own VPC with the VPC wizard or from scratch, connecting to your VPC, and securing your VPC with network ACLs, bastions, and NAT instances. Finally, we cover making your architecture highly available by means of multiple availability zones, elastic load balancing, and auto scaling.

## To get the most out of this book

Before starting to read the book, basic knowledge of networking concepts, basic knowledge of cloud computing, cloud service models, and cloud deployment models, basic knowledge of Linux and Windows operating systems would be useful.

## Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: [https://www.packtpub.com/sites/default/files/downloads/9781789535549\\_ColorImages.pdf](https://www.packtpub.com/sites/default/files/downloads/9781789535549_ColorImages.pdf).

## Conventions used

There are a number of text conventions used throughout this book.

**CodeInText:** Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "A /16 is a typical size, and this gives your VPC 65536 private IP addresses it can use."

Any command-line input or output is written as follows:

```
$ sudo yum update
```

**Bold:** Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "We could, of course, switch the **Volume Type** to **Provisioned IOPS SSD (io1)**, or use the cheaper **Magnetic (standard)** storage."



Warnings or important notes appear like this.



Tips and tricks appear like this.

## Get in touch

Feedback from our readers is always welcome.

**General feedback:** If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at [customercare@packtpub.com](mailto:customercare@packtpub.com).

**Errata:** Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit [www.packt.com/submit-errata](http://www.packt.com/submit-errata), selecting your book, clicking on the Errata Submission Form link, and entering the details.

**Piracy:** If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at [copyright@packt.com](mailto:copyright@packt.com) with a link to the material.

**If you are interested in becoming an author:** If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit [authors.packtpub.com](http://authors.packtpub.com).

## Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit [packt.com](http://packt.com).

# 1

# Installation and Setup

You will need an AWS account in order to follow the examples in this book. Since the account is free to open, you need not worry about payment for now. However, the free tier has defined usage limits for specific services. If your usage of these services exceeds the monthly quota for 12 months, you will need to pay the pay-as-you-go AWS service rates.

In this chapter, we will describe the process of opening an AWS account, including adding your payment information. Then we will discuss the free tier and demonstrate all the things you can do for free in the first year and beyond. Finally, we will introduce you to the AWS Management Console.

Here are the topics we'll cover:

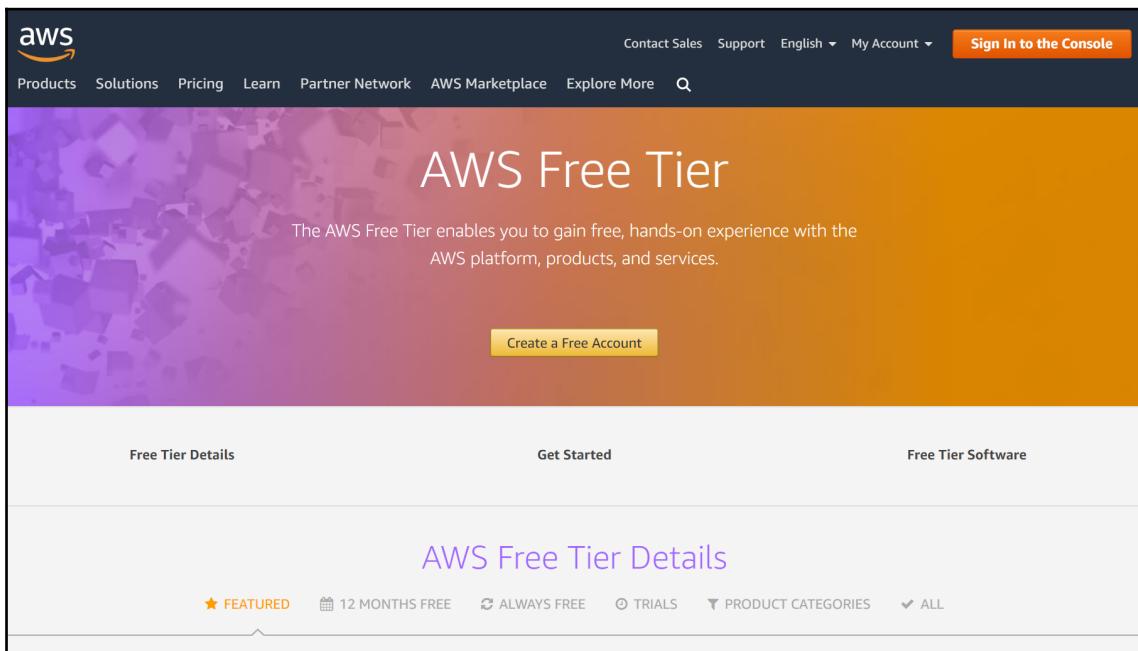
- Creating an AWS account
- The free tier
- The AWS Management Console

Let's begin by showing you how to open your AWS account.

# Opening an AWS account

In this section, we're going to describe how to sign up for an AWS account and add your payment information:

1. Begin by visiting <https://aws.amazon.com/free/>:



2. You can see there are different categories available in the free-tier:

- **FEATURED**
- **12 MONTHS FREE**
- **ALWAYS FREE**
- **TRIALS**
- **PRODUCT CATEGORIES**

## AWS Free Tier Details

★ FEATURED    12 MONTHS FREE    ALWAYS FREE    TRIALS    PRODUCT CATEGORIES    ALL

**12 months free and always free products**

AWS Free Tier includes offers that expire 12 months following sign up and others that never expire.

[Learn more »](#)

**COMPUTE**  
**Amazon EC2**

**750 Hours**  
per month

Resizable compute capacity in the Cloud

[Learn more about Amazon EC2 »](#)

[EXPAND DETAILS ^](#)

**ANALYTICS**  
**Amazon QuickSight**

**1 GB**  
of SPICE capacity

Fast, easy-to-use, cloud-powered business analytics service at 1/10th the cost of traditional BI solutions

[Learn more about Amazon QuickSight »](#)

[EXPAND DETAILS ^](#)

**DATABASE**  
**Amazon RDS**

**750 Hours**  
per month of db.t2.micro database usage (applicable DB engines)

Managed Relational Database Service for MySQL, PostgreSQL, MariaDB, Oracle BYOL, or SQL Server

[Learn more about Amazon RDS »](#)

**STORAGE & CONTENT DELIVERY**  
**Amazon S3**

**5 GB**  
of standard storage

Secure, durable, and scalable object storage infrastructure

[Learn more about Amazon S3 »](#)

**COMPUTE**  
**AWS Lambda**

**1 Million**  
free requests per month

Compute service that runs your code in response to events and automatically manages the compute resources

[Learn more about AWS Lambda »](#)



You can find more details about **AWS Free Tier (Non-expiring Offers)** for products such as **Amazon DynamoDB**, **Amazon Cognito**, **Amazon CodeCommit**, **AWS Lambda**, and so on can be found on <https://aws.amazon.com/free/>. You will also get to know about **AWS Free Tier (12 Month Introductory Period)** offers for products such as **Elastic Compute Cloud (EC2)**, **Amazon Simple Storage Service (S3)**, **Amazon Elastic File System (EFS)**, **Amazon CloudFront** and so on. Furthermore you can also check what **AWS Free Tier (12 Month Introductory Period)** offers for products such as **Amazon Relational Database Service (RDS)**, **Amazon CloudDirectory**, **AWS IOT**, and so on, on the same page.

3. Click **Create an AWS Account**, or **Create a Free Account** (either button will take you to the same place).
4. Select the tier with the **12 Months Free** option.
5. Now enter your email address, and choose a password and AWS account name. Click on **Continue**.
6. Fill in your contact information by selecting an **Account type**. You will probably want to make this a personal account, so click **Personal**. Now enter your **Full name** again, select your **Country/Region**, and enter your **Address** and your **Phone number**:

## Contact Information

*All fields are required.*

Please select the account type and complete the fields below with your contact details.

Account type ?

Professional  Personal

Full name

cleanclouds

Phone number

Country/Region

United States ▾

Address

Street, P.O. Box, Company Name, c/o

Apartment, suite, unit, building, floor, etc.

Provide **City**, and **Postal code**, details check the agreement details, and click on **Create Account and Continue**. You probably should review this first, and when you are satisfied, then click **Create Account and Continue**.

7. Next, enter your credit card number. Even though this is a free account, you still need to provide billing information. Once you have filled in your credit card information, click on **Continue**. Provide **Payment Information** and **Authenticate Transaction** details:

The screenshot shows the 'Payment Information' step of the AWS account creation process. At the top, there's a note: 'Please type your payment information so we can verify your identity. We will not charge you unless your usage exceeds the [AWS Free Tier Limits](#). Review [frequently asked questions](#) for more information.' Below this, there's an info icon with a message: 'As part of our card verification process we will charge INR 2 on your card when you click the "Secure Submit" button below. This will be refunded once your card has been validated. Your bank may take 3-5 business days to show the refund. Mastercard/Visa customers may be redirected to your bank website to authorize the charge.' The form fields include: 'Credit/Debit card number' (input field), 'Expiration date' (two dropdown menus for month and year), and 'Cardholder's name' (input field).

8. To verify your information, you need to let AWS call you. So, enter your phone number, and then click **Call Me Now**:

## Phone Verification

AWS will call you immediately using an automated system. When prompted, enter the 4-digit number from the AWS website on your phone keypad.

**Provide a telephone number**

Please enter your information below and click the "Call Me Now" button.

Country/Region code

▾

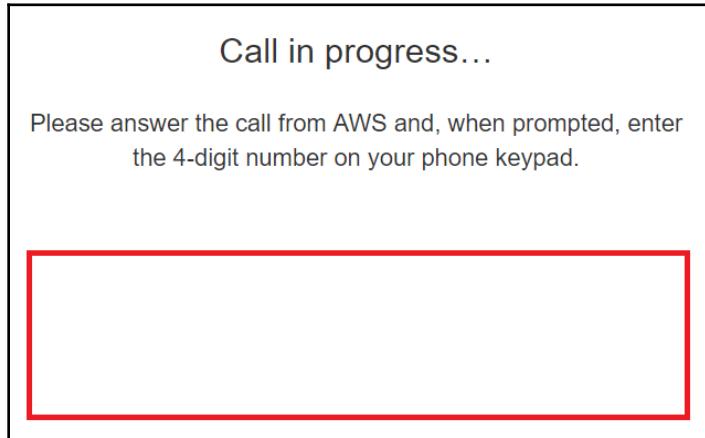
Phone number      Ext

Security Check

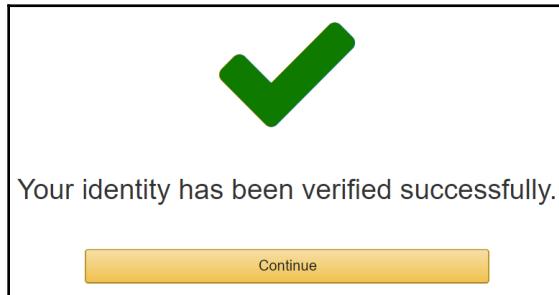


*Please type the characters as shown above*

9. You will receive a phone call giving you a PIN number, which you need to enter on the screen:



10. After your identity is verified, click **Continue**:



11. After the phone call, and your identity has been verified, click **Continue** the select your support plan:

## Select a Support Plan

AWS offers a selection of support plans to meet your needs. Choose the support plan that best aligns with your AWS usage. [Learn more](#)

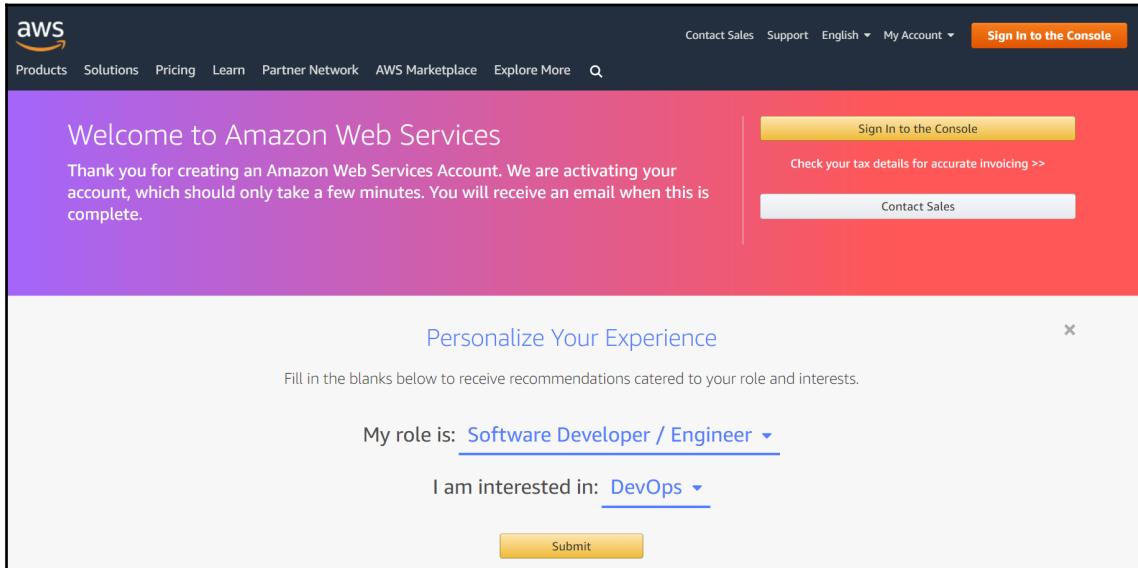
Basic Plan	Developer Plan	Business Plan
 Free	 From \$29/month	 From \$100/month
<ul style="list-style-type: none"><li>Included with all accounts</li><li>24/7 self-service access to forums and resources</li><li>Best practice checks to help improve security and performance</li><li>Access to health status and notifications</li></ul>	<ul style="list-style-type: none"><li>For early adoption, testing and development</li><li>Email access to AWS Support during business hours</li><li>1 primary contact can open an unlimited number of support cases</li><li>12-hour response time for nonproduction systems</li></ul>	<ul style="list-style-type: none"><li>For production workloads &amp; business-critical dependencies</li><li>24/7 chat, phone, and email access to AWS Support</li><li>Unlimited contacts can open an unlimited number of support cases</li><li>1-hour response time for production systems</li></ul>

**Need Enterprise level support?**

Contact your account manager for additional information on running business and mission critical-workloads on AWS (starting at \$15,000/month). [Learn more](#)

© 2018 Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.  
[Privacy Policy](#) | [Terms of Use](#) | [Sign Out](#)

12. The next screen lets you **Personalize Your Experience**. Select a role and the service you're interested in services:

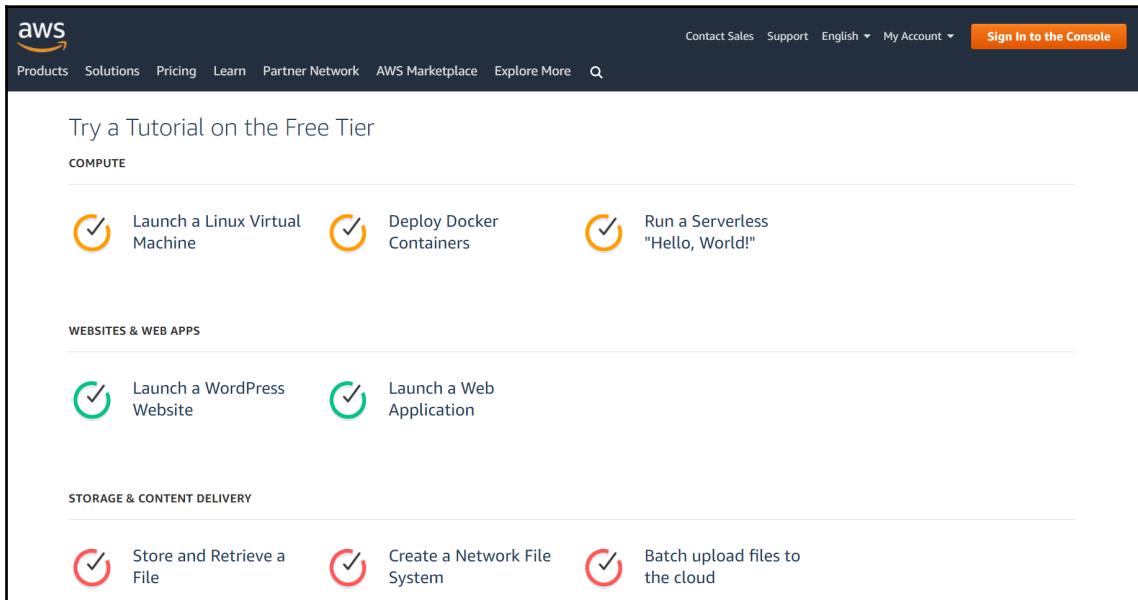


Congratulations, you now have an AWS account! A confirmation message will be sent to your email address.

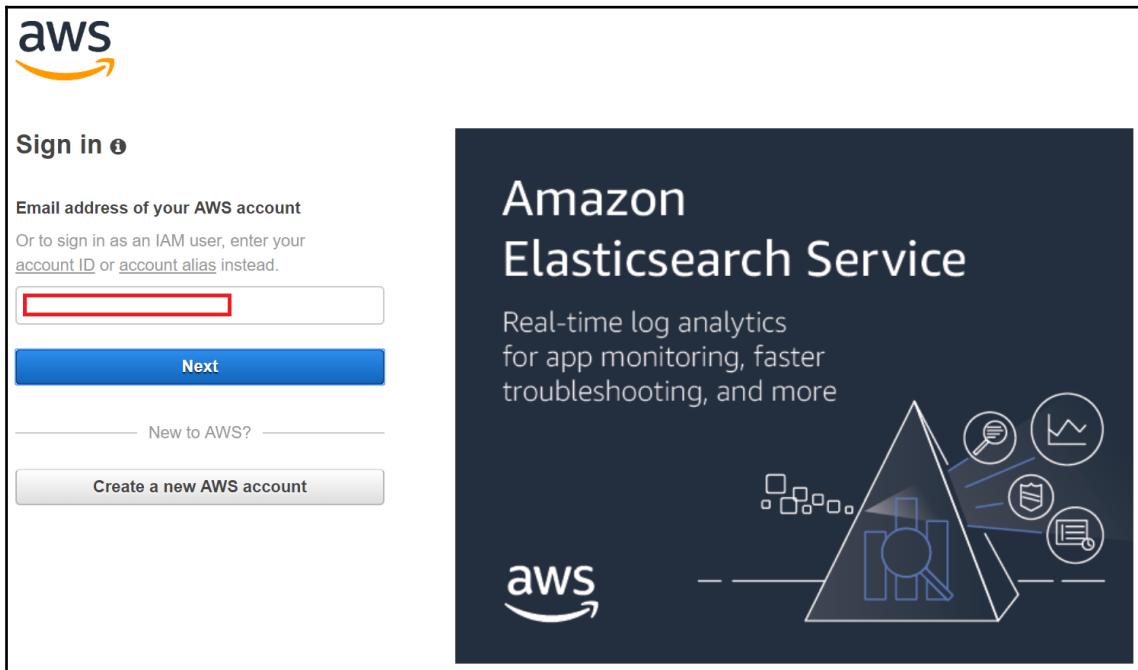
# The AWS Management Console

Now that you have an AWS account, you're ready to begin using the Management Console. In this section, we're going to describe three ways to log into the Console, navigating, viewing your build, and switching between regions.

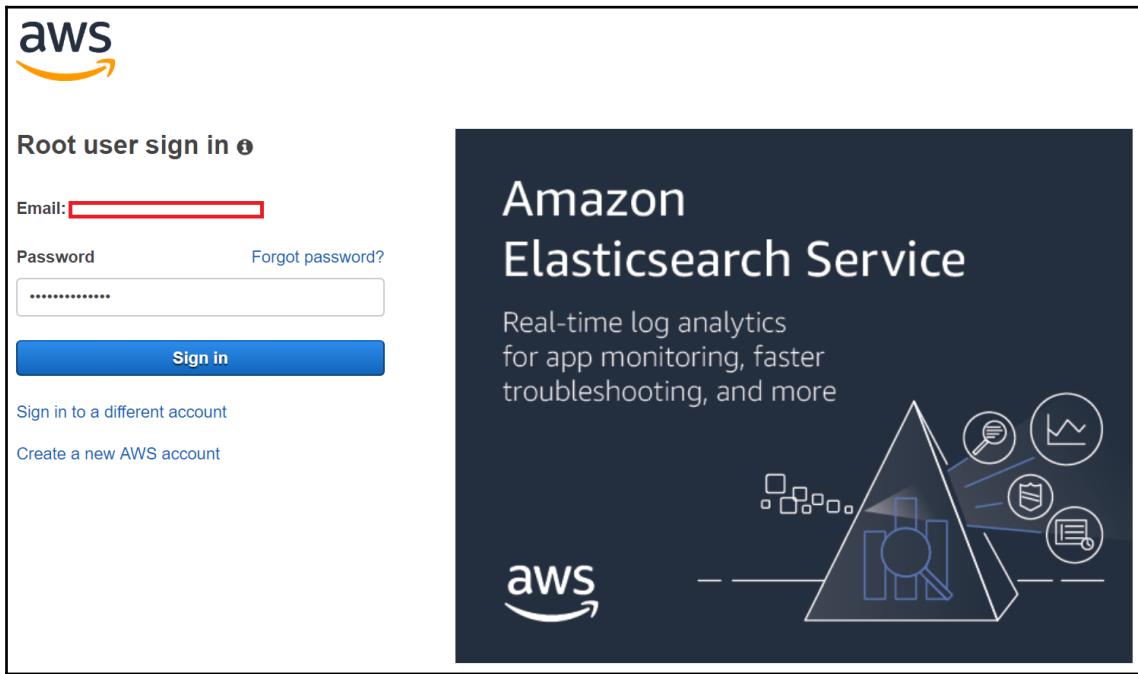
You can log into the console by visiting <https://aws.amazon.com/> and clicking **Sign In to the Console**:



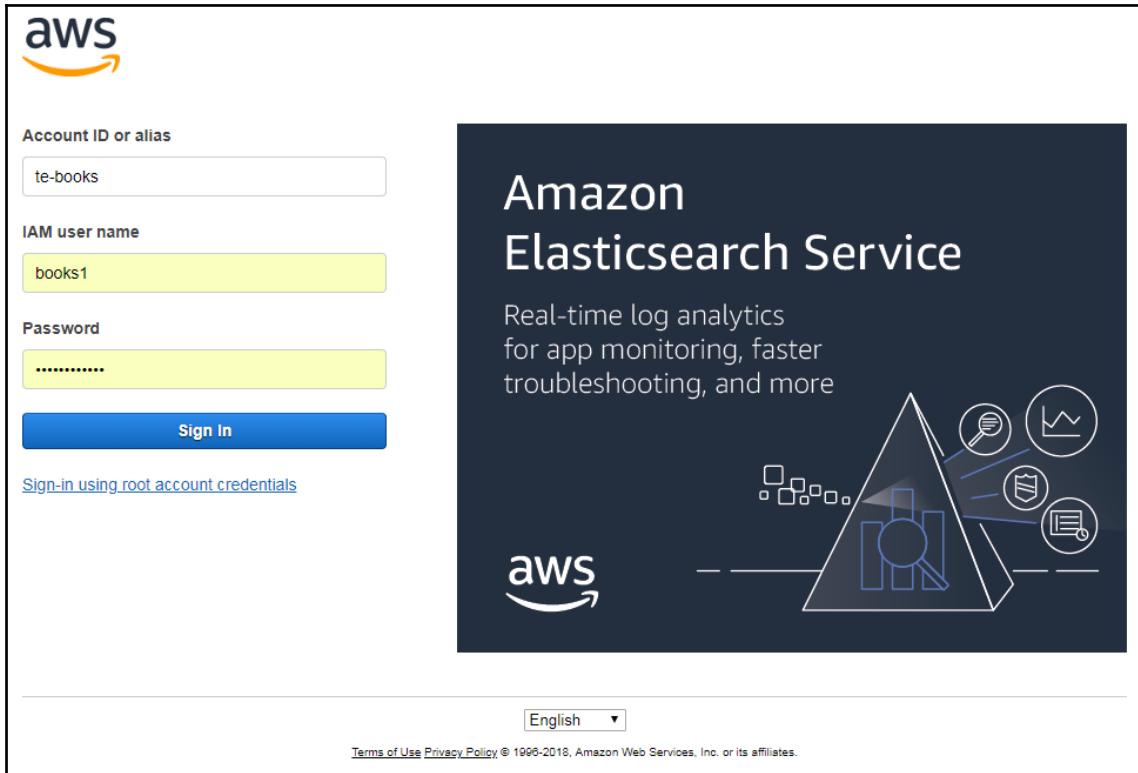
If your screen looks like the following screenshot, then you are signing in with your root account credentials. Simply enter an e-mail address and click on **Next**:



Enter the password you used when you created your AWS account, and click the **Sign In** button:



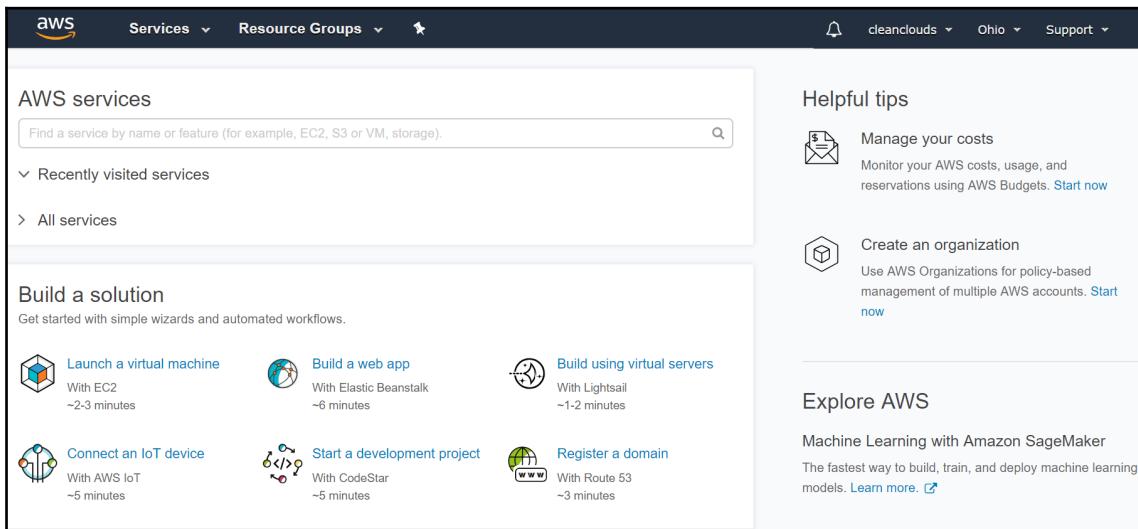
If you are signing in as an IAM user, your screen will look like the following:



IAM users are created in the console to grant login and management permissions to others. If you would rather log in as your root account, click the link under **Sign In** that reads **Sign-in using root account credentials**.

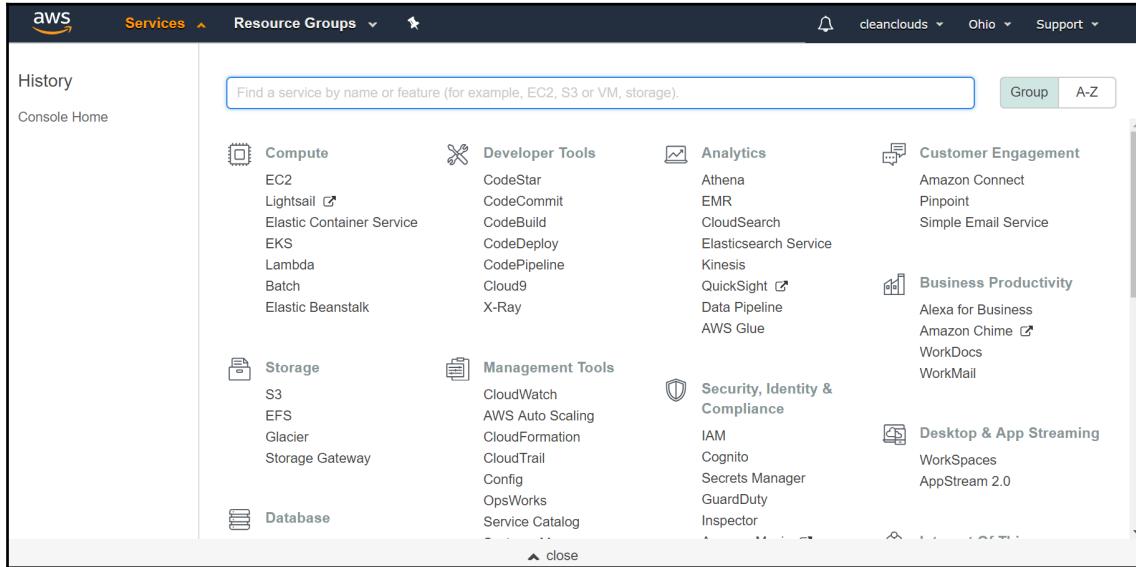
Otherwise, click **Sign In** after you enter your username and password.

After you log in, you will see the dashboard:



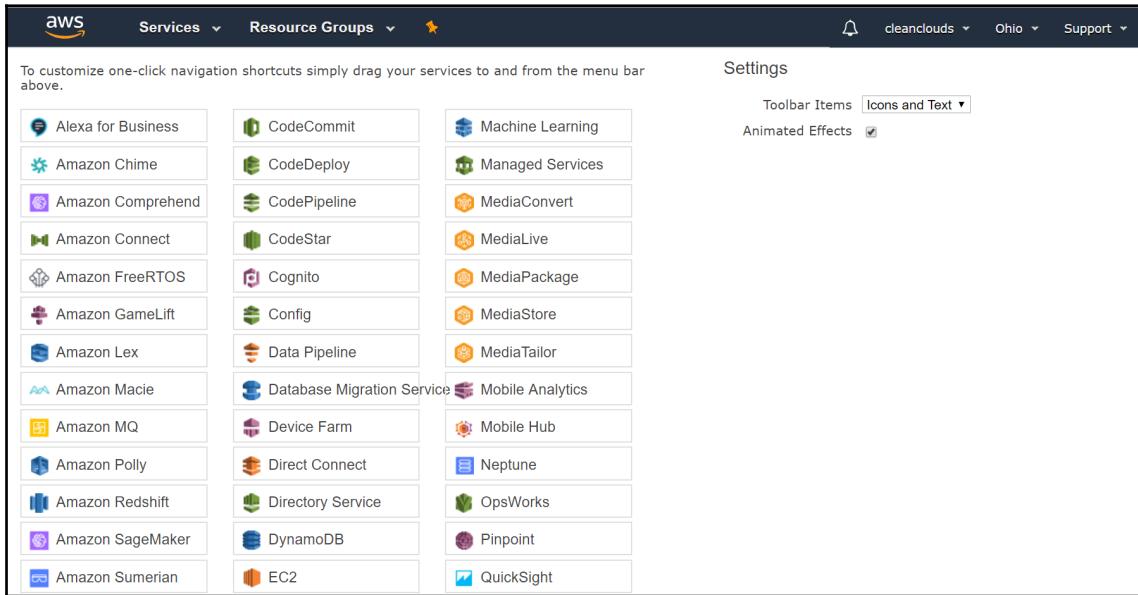
The dashboard is a quick way to locate AWS services.

The **Services** menu is another way to locate AWS services:

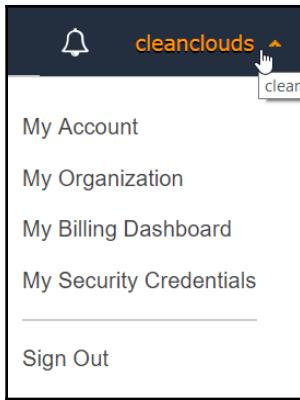


It includes a recent **History** of services used, and more detailed descriptions of services.

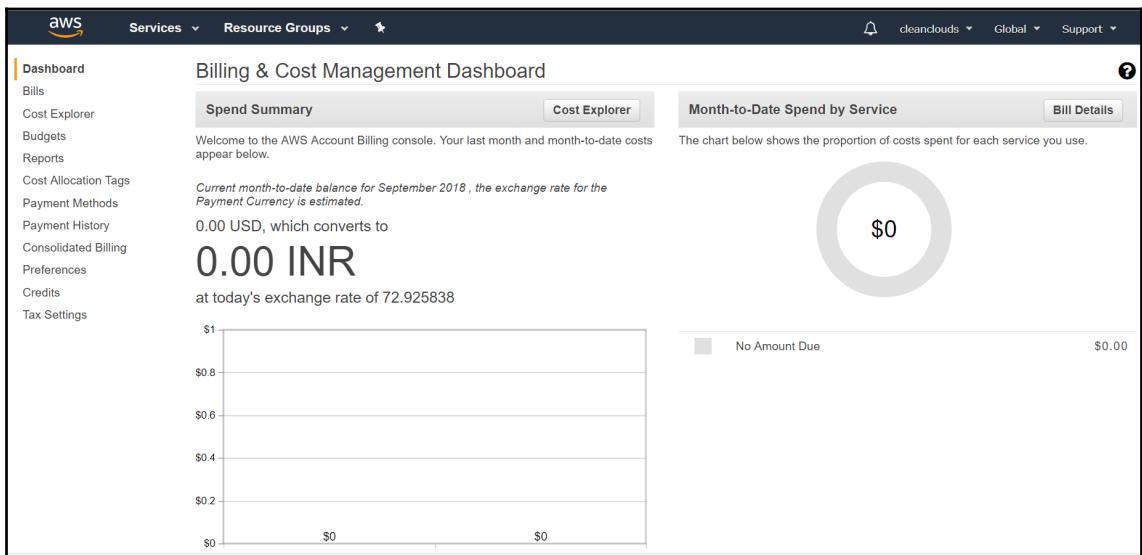
Next, under the Edit icon, any services that you use frequently can be dragged to the top bar, where you can easily find them:



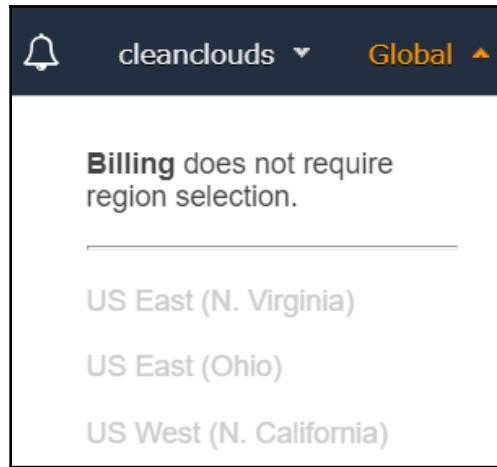
If you click on your name, you will see a submenu that will allow you to access your billing information:



This is where you find your AWS charges for this month the last month, and also a projection of next month's charges:



Next to your name, you will see a drop down for region selection:



Some services, such as billing, IAM, CloudFront, and Route 53, do not require you to select a region. However, most other services will require you to select a region from this drop-down.

You can close the account from **My Billing Dashboard**:

The screenshot shows the AWS Management Console with the 'Account Settings' page open. The left sidebar lists various account management options like Dashboard, Bills, Cost Explorer, etc. The main content area is titled 'Account Settings' and contains sections for 'Contact Information', 'Alternate Contacts', 'Configure Security Challenge Questions', 'IAM User and Role Access to Billing Information', 'Account Contract Information', 'Communication Preferences', 'Manage AWS Support Plans', 'GovCloud (US)', and 'Close Account'. Each section has an 'Edit' link to its right.



AWS free tier does not automatically expire at the end of your 12-month term. However, Microsoft Windows Server 2008 R2 with SQL Server Web, Microsoft Windows Server 2008 R2 with SQL Server Standard, Microsoft Windows 2008 R2 64-bit for Cluster Instances, and Microsoft Windows 2008 R2 SQL Server 64-bit for Cluster Instances are not eligible for the free tier. Currently, the free tier is not available in the China (Beijing) region.

## Summary

In this chapter, we described how to open a free AWS account. We also discussed the AWS feature that you can use for free. We introduced you to the AWS Management Console. In Chapter 2, *Launching an EC2 Instance*, we'll discuss virtual servers on AWS, known as EC2.

# 2

# Launching an EC2 Instance

In this chapter, we will see how to launch an EC2 instance using your own AWS account. We will discuss some of the choices you will have to make when you launch an instance, including choosing **Amazon Machine Images (AMIs)**, an instance type and size, storage options, and configuring your security group.

The topics that we will cover are as follows:

- Instance types
- Storage
- Security groups
- AMIs

## EC2 instance types

In this section, we're going to take a look at selecting the instance type and size as we continue launching our instance. We'll discuss the general types their features, and some good use cases for each.

### General purpose instance

The first instance we're going to discuss is a general purpose instance type known as T3s and T2s.

## T3s – burstable general-purpose instance type

T3 instances are good candidates for the development environment, micro-services, and so on.

Here are some of the features of T3:

- Based on Intel® Xeon® Scalable processors and the AWS Nitro System
- They provide consistent baseline performance

Before we move on, we should note that instance types are designated by a letter followed by a number, which is the generation of the instance type:

Model	vCPU	CPU credits/hour	Mem (GiB)	Storage
t3.nano	2	6	0.5	EBS-only
t3.micro	2	12	1	EBS-only
t3.small	2	24	2	EBS-only
t3.medium	2	24	4	EBS-only
t3.large	2	36	8	EBS-only
t3.xlarge	4	96	16	EBS-only
t3.2xlarge	8	192	32	EBS-only

## T2s – burstable general-purpose instance type

For proper workloads, such as development environments, T2 instances should never run out of credits. These are the only types of burstable CPU capacity allocation.

Here are some of the advantages of T2:

- Free tier-eligible (t2.micro)
- T2s are the least expensive instance type
- Instead of dedicated CPU capacity, these instances share CPU capacity with other instances
- They operate at a low baseline CPU performance, and accumulate credits when idle
- When they need to use the CPU, they can burst up to the full utilization of the virtual CPUs until they exhaust their credits

Before we move on, we should note that instance types are designated by a letter followed by a number, which is the generation of the instance type:

Model	vCPU	CPU Credits / hour	Mem (GiB)	Storage
t2.nano	1	3	0.5	EBS-only
t2.micro	1	6	1	EBS-only
t2.small	1	12	2	EBS-only
t2.medium	2	24	4	EBS-only
t2.large	2	36	8	EBS-only
t2.xlarge	4	54	16	EBS-only
t2.2xlarge	8	81	32	EBS-only

So, T2 designates the second generation of the T instances.

Another general-purpose instance type is the M's. These are the latest ones.

## M5

M5 instances are utilized for enterprise applications, data processing tasks, and small and medium size databases.

Here are some of the advantages of M5:

- They provide enhanced networking—up to 25 Gbps network bandwidth
- They provide instance storage using EBS
- They provide larger instance sizes

You can see the example here for the different type of m5 models:

Model	vCPU	Mem (GiB)	Instance storage (GiB)	Dedicated EBS bandwidth (Mbps)
m5.large	2	8	EBS-only	Up to 3,500
m5.xlarge	4	16	EBS-only	Up to 3,500
m5.2xlarge	8	32	EBS-only	Up to 3,500
m5.4xlarge	16	64	EBS-only	3500
m5.12xlarge	48	192	EBS-only	7000
m5.24xlarge	96	384	EBS-only	14000
m5d.large	2	8	1 x 75 NVMe SSD	Up to 3,500
m5d.xlarge	4	16	1 x 150 NVMe SSD	Up to 3,500
m5d.2xlarge	8	32	1 x 300 NVMe SSD	Up to 3,500
m5d.4xlarge	16	64	2 x 300 NVMe SSD	3500
m5d.12xlarge	48	192	2 x 900 NVMe SSD	7000
m5d.24xlarge	96	384	4 x 900 NVMe SSD	14000

## M4

M4s are very suitable for small-and mid-sized databases and many web applications.

Here are some of the advantages of M4:

- M4s use the advanced Broadwell or Haswell architecture microprocessor
- They are EBS optimized, which means they will provide consistent network connectivity to EBS volumes
- They also support enhanced networking, which provides higher bandwidth, higher number of packets per second, and consistently lower latencies between instances
- They do not provide local instance storage

Here you can see list of m4 models with its details:

Model	vCPU	Mem (GiB)	SSD storage (GB)	Dedicated EBS bandwidth (Mbps)
m4.large	2	8	EBS-only	450
m4.xlarge	4	16	EBS-only	750
m4.2xlarge	8	32	EBS-only	1000
m4.4xlarge	16	64	EBS-only	2000
m4.10xlarge	40	160	EBS-only	4000
m4.16xlarge	64	256	EBS-only	10000

## Compute optimized

The first compute optimized instance we're going to look at is the C5s.

### C5

C5 instances are utilized for machine learning, deep learning, batch processing, HPC, and so on.

Here are some of its features:

- They provide low cost and high-performance workloads
- They provide enhanced networking—up to 25 Gbps network bandwidth
- They provide instance storage using EBS
- They provide larger instance sizes

Lets take a look at c5 models and its specifications:

Model	vCPU	Mem (GiB)	Instance storage (GiB)	Dedicated EBS bandwidth (Mbps)
c5.large	2	4	EBS-only	Up to 3,500
c5.xlarge	4	8	EBS-only	Up to 3,500
c5.2xlarge	8	16	EBS-only	Up to 3,500
c5.4xlarge	16	32	EBS-only	3500
c5.9xlarge	36	72	EBS-only	7000
c5.18xlarge	72	144	EBS-only	14000
c5d.large	2	4	1 x 50 NVMe SSD	Up to 3,500
c5d.xlarge	4	8	1 x 100 NVMe SSD	Up to 3,500
c5d.2xlarge	8	16	1 x 200 NVMe SSD	Up to 3,500
c5d.4xlarge	16	32	1 x 400 NVMe SSD	3500
c5d.9xlarge	36	72	1 x 900 NVMe SSD	7000
c5d.18xlarge	72	144	2 x 900 NVMe SSD	14000

## C4

C4 instances are utilized for high-performance front end fleets, analytics, and batch processing.

Here are some of its features:

- They use the most powerful Haswell processors
- They are EBS-optimized and support enhanced networking and clustering
- They do not come with any instance storage

Here is a list of c4 models:

Model	vCPU	Mem (GiB)	Storage	Dedicated EBS bandwidth (Mbps)
c4.large	2	3.75	EBS-only	500
c4.xlarge	4	7.5	EBS-only	750
c4.2xlarge	8	15	EBS-only	1000
c4.4xlarge	16	30	EBS-only	2000
c4.8xlarge	36	60	EBS-only	4000

## Memory-optimized

Next up are the memory optimized instance types.

### X1e

X1e instances are utilized in memory databases and high-performance databases.

Some of its features are:

- Comparatively less costly
- By default EBS-optimized.

Lets take a look at x1e models with its specifications:

Model	vCPU	Mem (GiB)	SSD storage (GB)	Dedicated EBS bandwidth (Mbps)
x1e.xlarge	4	122	1 x 120	500
x1e.2xlarge	8	244	1 x 240	1000
x1e.4xlarge	16	488	1 x 480	1750
x1e.8xlarge	32	976	1 x 960	3500
x1e.16xlarge	64	1952	1 x 1,920	7000
x1e.32xlarge	128	3904	2 x 1,920	14000

## X1

X1 instances are for in-memory databases, SAP HANA, and big data processing.

Some of its features are as follows:

- Haswell processors
- SSD-based instance storage
- Lowest price per GB of RAM.

Here is an example of x1 instance model:

Model	vCPU	Mem (GiB)	SSD storage (GB)	Dedicated EBS bandwidth (Mbps)
x1.16xlarge	64	976	1 x 1,920	7000
x1.32xlarge	128	1952	2 x 1,920	14000

## R5

R5 instances are utilized for enterprise applications, in-memory databases, and big data analytics.

Here are some of its features:

- They use the Intel Xeon Platinum 8000 series
- No virtualization overhead

Lets take a look at some of the R5 instances:

Model	vCPU	Mem (GiB)	Networking perf.	SSD storage (GB)
r5.large	2	16	Up to 10 Gigabit	EBS-only
r5.xlarge	4	32	Up to 10 Gigabit	EBS-only
r5.2xlarge	8	64	Up to 10 Gigabit	EBS-only
r5.4xlarge	16	128	Up to 10 Gigabit	EBS-only
r5.12xlarge	48	384	10 Gigabit	EBS-only
r5.24xlarge	96	768	25 Gigabit	EBS-only
r5d.large	2	16	Up to 10 Gigabit	1 x 75 NVMe SSD
r5d.xlarge	4	32	Up to 10 Gigabit	1 x 150 NVMe SSD
r5d.2xlarge	8	64	Up to 10 Gigabit	1 x 300 NVMe SSD
r5d.4xlarge	16	128	Up to 10 Gigabit	2 x 300 NVMe SSD
r5d.12xlarge	48	384	10 Gigabit	2 x 900 NVMe SSD
r5d.24xlarge	96	768	25 Gigabit	4 x 900 NVMe SSD

## R4

R4 instances are utilized for enterprise applications, in-memory databases, and Hadoop clusters.

Here are some of its features:

- They have Broadwell processors
- Support DDR4 memory
- Enhanced networking

Below you can find R4 instances product details:

Model	vCPU	Mem (GiB)	Networking perf.	SSD storage (GB)
r4.large	2	15.25	Up to 10 Gigabit	EBS-only
r4.xlarge	4	30.5	Up to 10 Gigabit	EBS-only
r4.2xlarge	8	61	Up to 10 Gigabit	EBS-only
r4.4xlarge	16	122	Up to 10 Gigabit	EBS-only
r4.8xlarge	32	244	10 Gigabit	EBS-only
r4.16xlarge	64	488	25 Gigabit	EBS-only

## z1d

z1d instances are utilized for design automation and relational database workloads.

Here are some of its features:

- They provide high compute capacity and a high memory
- No virtualization overhead
- Fastest cloud instances

Lets look at z1d product details:

Model	vCPU	Mem (GiB)	Networking perf.	SSD storage (GB)
z1d.large	2	16	Up to 10 Gigabit	1 x 75 NVMe SSD
z1d.xlarge	4	32	Up to 10 Gigabit	1 x 150 NVMe SSD
z1d.2xlarge	8	64	Up to 10 Gigabit	1 x 300 NVMe SSD
z1d.3xlarge	12	96	Up to 10 Gigabit	1 x 450 NVMe SSD
z1d.6xlarge	24	192	10 Gigabit	1 x 900 NVMe SSD
z1d.12xlarge	48	384	25 Gigabit	2 x 900 NVMe SSD

## Accelerated computing – general purpose GPU instances

If your applications can benefit from GPU acceleration, G2s come with NVIDIA GPUs with an onboard video encoder. One example is the Intel Xeon E5-2670 (Sandy Bridge) processors. These have SSD-based instance storage, and 3D application streaming and video encoding.

### P3

P3 instances are utilized for high-performance computing and speech recognition. They are general purpose GPU instances.

Below you can find P3 instance model details:

Model	GPUs	vCPU	Mem (GiB)	GPU mem (GiB)	GPU P2P
p3.2xlarge	1	8	61	16	-
p3.8xlarge	4	32	244	64	NVLink
p3.16xlarge	8	64	488	128	NVLink

### P2

P2 instances are utilized for machine learning and high-performance databases.

Here are some of its features:

- These instances are by default EBS-optimized
- They support enhanced networking

Below you can find P2 instance model details:

Model	GPUs	vCPU	Mem (GiB)	GPU memory (GiB)
p2.xlarge	1	4	61	12
p2.8xlarge	8	32	488	96
p2.16xlarge	16	64	732	192

## G3

G3 instances are utilized for 3D rendering, application streaming, and video encoding.

Here are some of its features:

- Optimized for graphics-intensive usage
- Enhanced networking using the elastic network adapter

Below you can find G3 instance model details:

Model	GPUs	vCPU	Mem (GiB)	GPU memory (GiB)
g3.4xlarge	1	16	122	8
g3.8xlarge	2	32	244	16
g3.16xlarge	4	64	488	32

## F1

F1 instances are utilized for video processing and analytic:

Here are some of its features:

- They provide customizable hardware acceleration with field programmable gate arrays
- Enhanced networking

Below you can find F1 instance model details

Model	FPGAs	vCPU	Mem (GiB)	SSD storage (GB)	Networking performance
f1.2xlarge	1	8	122	470	Up to 10 Gigabit
f1.16xlarge	8	64	976	4 x 940	25 Gigabit

## Storage-optimized instance types

There are three storage optimized instance types

## H1

H1 instances are utilized for big data workload clusters, distributed file systems, and network file systems.

Here are some of its features:

- H1 instances provides high disk throughput
- Enhanced Networking with ENA

Please see the details of H1 instances here:

Model	vCPU	Mem (GiB)	Networking performance	Storage (GB)
<b>h1.2xlarge</b>	<b>8</b>	<b>32</b>	<b>Up to 10 Gigabit</b>	<b>1 x 2,000 HDD</b>
<b>h1.4xlarge</b>	<b>16</b>	<b>64</b>	<b>Up to 10 Gigabit</b>	<b>2 x 2,000 HDD</b>
<b>h1.8xlarge</b>	<b>32</b>	<b>128</b>	<b>10 Gigabit</b>	<b>4 x 2,000 HDD</b>
<b>h1.16xlarge</b>	<b>64</b>	<b>256</b>	<b>25 Gigabit</b>	<b>8 x 2,000 HDD</b>

## I3

I3 instances are utilized for NoSQL databases and in-memory databases.

Here are some of its features:

- It provides Broadwell processors
- **Elastic network adapter (ENA)-based enhanced networking**

Refer to the below I3 instance details:

Model	vCPU	Mem (GiB)	Networking performance	Storage (TB)
<b>i3.large</b>	<b>2</b>	<b>15.25</b>	<b>Up to 10 Gigabit</b>	<b>1 x 0.475 NVMe SSD</b>
<b>i3.xlarge</b>	<b>4</b>	<b>30.5</b>	<b>Up to 10 Gigabit</b>	<b>1 x 0.95 NVMe SSD</b>
<b>i3.2xlarge</b>	<b>8</b>	<b>61</b>	<b>Up to 10 Gigabit</b>	<b>1 x 1.9 NVMe SSD</b>
<b>i3.4xlarge</b>	<b>16</b>	<b>122</b>	<b>Up to 10 Gigabit</b>	<b>2 x 1.9 NVMe SSD</b>
<b>i3.8xlarge</b>	<b>32</b>	<b>244</b>	<b>10 Gigabit</b>	<b>4 x 1.9 NVMe SSD</b>
<b>i3.16xlarge</b>	<b>64</b>	<b>488</b>	<b>25 Gigabit</b>	<b>8 x 1.9 NVMe SSD</b>
<b>i3.metal</b>	<b>72*</b>	<b>512</b>	<b>25 Gigabit</b>	<b>8 x 1.9 NVMe SSD</b>

## D2

D2 instances are utilized for MapReduce and Hadoop-distributed computing, distributed file systems and network file systems.

Here are some of its features:

- They provide high disk throughput
- They provide high-performance at launch time
- They provide enhanced networking

Refer to the below D2 instance details:

Model	vCPU	Mem (GiB)	Storage (GB)
<b>d2.xlarge</b>	<b>4</b>	<b>30.5</b>	<b>3 x 2000 HDD</b>
<b>d2.2xlarge</b>	<b>8</b>	<b>61</b>	<b>6 x 2000 HDD</b>
<b>d2.4xlarge</b>	<b>16</b>	<b>122</b>	<b>12 x 2000 HDD</b>
<b>d2.8xlarge</b>	<b>36</b>	<b>244</b>	<b>24 x 2000 HDD</b>



For more details refer to, <https://aws.amazon.com/ec2/instance-types/>

# Launching the instance

Now that we have learned about the instance types, let's continue launching our instance:

The screenshot shows the AWS Launch Wizard interface for launching an EC2 instance. The top navigation bar includes the AWS logo, Services dropdown, Resource Groups dropdown, and account information (cleanclouds, Ohio, Support). Below the navigation is a progress bar with steps 1. Choose AMI (highlighted in orange), 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group, and 7. Review. A 'Cancel and Exit' button is on the right.

**Step 1: Choose an Amazon Machine Image (AMI)**

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.

A search bar at the top says "Search for an AMI by entering a search term e.g. "Windows"".

**Quick Start**

- My AMIs
- AWS Marketplace
- Community AMIs
- Free tier only ⓘ

**Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-0b59bfac6be064b78**

Amazon Linux Free tier eligible 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.

Root device type: ebs Virtualization type: hvm

**Select**

**Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0cf31d971a3ca20d6**

Amazon Linux Free tier eligible Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.

Root device type: ebs Virtualization type: hvm

**Select**

**Red Hat Enterprise Linux 7.5 (HVM), SSD Volume Type - ami-03291866**

Red Hat Free tier eligible Red Hat Enterprise Linux version 7.5 (HVM), EBS General Purpose (SSD) Volume Type

**Select**

At the bottom, there are links for Feedback, English (US), and footer links: © 2008 - 2018, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved., Privacy Policy, Terms of Use.

In the second step for launching our instance, we need to select the instance type and size.

The screenshot shows the AWS EC2 instance creation wizard at Step 2: Choose Instance Type. The 't2.micro' instance is selected in the list, which is highlighted with a green background and labeled 'Free tier eligible'. The table includes columns for Family, Type, vCPUs, Memory (GiB), Instance Storage (GB), EBS-Optimized Available, Network Performance, and IPv6 Support. The 't2.micro' row has a checked checkbox in the first column. Other visible rows include t2.nano, t2.small, and t2.medium. At the bottom right of the table are buttons for 'Cancel', 'Previous', 'Review and Launch' (which is highlighted in blue), and 'Next: Configure Instance Details'.

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
General purpose	<b>t2.micro</b> Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes

To stay on the free tier, let's select a **t2.micro**. This is a general purpose instance that will provide one virtual CPU and, 1 GB of memory, and will not have any instance storage.

Click on **Next: Configure Instance Details** to go to the next step:

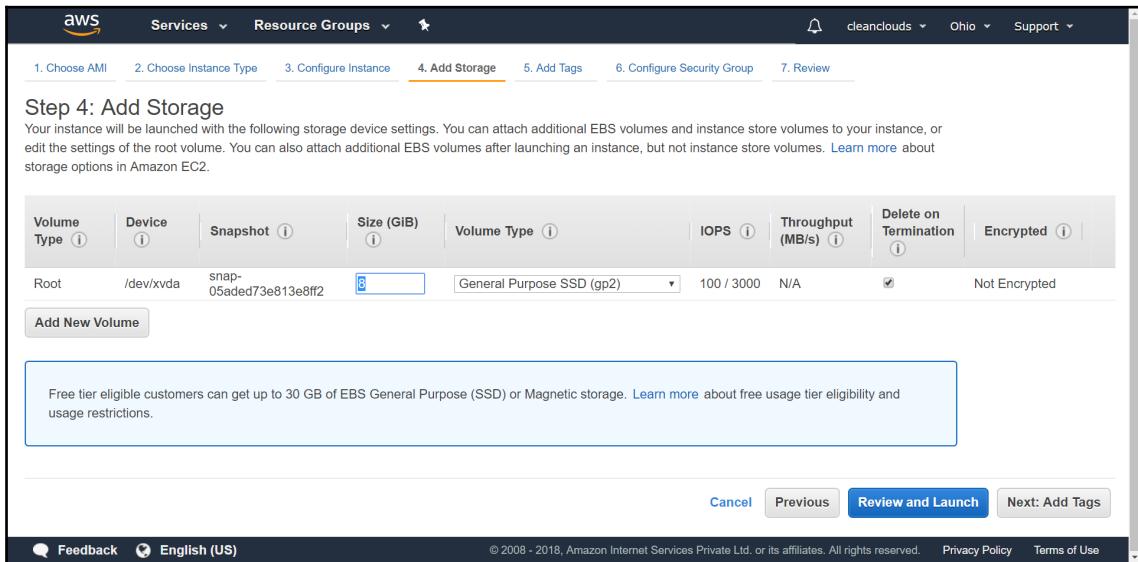
The screenshot shows the AWS EC2 Launch Wizard at Step 3: Configure Instance Details. The page title is "Step 3: Configure Instance Details". It includes a sub-instruction: "Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more." Below this, there are several configuration fields:

- Number of instances:** Set to 1.
- Purchasing option:** An unchecked checkbox for "Request Spot instances".
- Network:** Set to "vpc-48023d20 (default)". Includes a "Create new VPC" button.
- Subnet:** Set to "No preference (default subnet in any Availability Zone)". Includes a "Create new subnet" button.
- Auto-assign Public IP:** Set to "Enable".
- Placement group:** An unchecked checkbox for "Add instance to placement group".
- IAM role:** Set to "None". Includes a "Create new IAM role" button.
- Shutdown behavior:** Set to "Stop".

At the bottom right, there are buttons for "Cancel", "Previous", "Review and Launch" (which is highlighted in blue), and "Next: Add Storage".

On this page, we're pretty much going to leave everything as the default settings.

However, we do want to make sure that we get a public IP address for our instance so we can log in to it. So, choose **Enable** on **Auto-assign Public IP**, then click **Next: Add Storage**:



The next decision we have to make is the size and type of storage volumes to attach to our instance. So, in the next section we will discuss EC2 storage options.

# EC2 storage options

In the previous section, we looked at the available instance types. In this section, we're going to look at the options for filesystem storage for our instance. We will cover the available options, which include local instance storage and network-attached elastic block storage.

## Instance storage

Instance storage is SSD, or hard disk storage, that is installed in the host machine. Since it is local, you can get high disk throughput and high input/output per second, known as IOPS. However, one big caveat is the ephemeral, temporary nature of instance storage. When your instance is running, it is occupying a slot in the hypervisor of a host server. So, it has access to its share of the host machine's local storage.

However, when an instance is stopped or terminated, it is removed from the hypervisor slot, and that frees the slot for another instance to occupy it. So, to make sure that no one will be able to access your data, the local storage for your instance is wiped, and any data that you wrote to the volume is gone.

You may be thinking that you could just back up the files before you stop your instance. However, this may not be possible if your instance is impaired and that is the reason you're trying to stop it. So the bottom line is, don't store any important data on these volume, unless the data is also replicated elsewhere. You also don't get to directly provision the amount of instance storage that you want.

You are allocated a certain amount of instance storage, depending upon the size of the instance.

So then, what can you use this instance storage for? Well, they're good for swapping temporary files for sure, or data that is replicated elsewhere. For example, your application can be run from these as long as the source code is stored in a source code repository and your application is stateless.

This means that the state or session data is stored in a separate storage system, such as a NoSQL database. They're also fine for boot volumes, but they are not as convenient to use as EBS volumes. But one thing is that you can't take a snapshot of them, and so making an AMI with instance storage requires an extra step that's known as bundling. This involves copying the files up to S3 first. Also, you cannot stop and start an instance store back instance, because when you do, you completely wipe the boot volume. So, you have to terminate them and then relaunch. Finally, AWS does not offer full volume encryption for instance storage. Because of the limitations and ephemeral nature of instance storage, AWS offers a more durable alternative: **Elastic Block Storage (EBS)**.

This storage is not located on the host machine; it is on separate hardware elsewhere in the same availability zone.

## Elastic block storage

EBS volumes are attached to your instance via the network. Because of this, they have a separate lifecycle and can persist even after you terminate your instance, so you can freely stop and start EBS-backed instances. To further improve their durability, the data is mirrored on two devices. Since they are network-connected to your instance, you should ideally use them with instances that are EBS-optimized, which will give you a more consistent disk throughput. You can provision the size of your EBS volumes however you want, from 1 GB up to 16 TB. You can also attach multiple EBS volumes to a single instance, and also RAID them if you like. However, like a physical disk drive, they can only be attached to one instance at a time.

So, if you need a shareable filesystem like an AZ, then you should consider AWS's elastic filesystem service. One really nice feature is the ability to take snapshots of EBS volumes. These are very durably stored in the S3 storage service. You can restore or create new EBS volumes from snapshots. You can also enable full volume encryption, which will encrypt your data at rest, and also encrypt the snapshots.

## General purpose SSD

The most common type of EBS volume is general purpose SSD. Because of the price drops in solid state storage, they are fairly cost effective and deliver good performance. You may remember from the previous section how T instances accumulate credits and burst when they need CPU capacity. Well, general purpose EBS volumes do a similar thing with IOPS.

They are also good for storage for databases, and particularly ones that are in development or test environments. However, if you want consistent IOPS for EBS volumes, then you can upgrade to provisioned IOPS SSD.

## Provisioned IOPS SSD

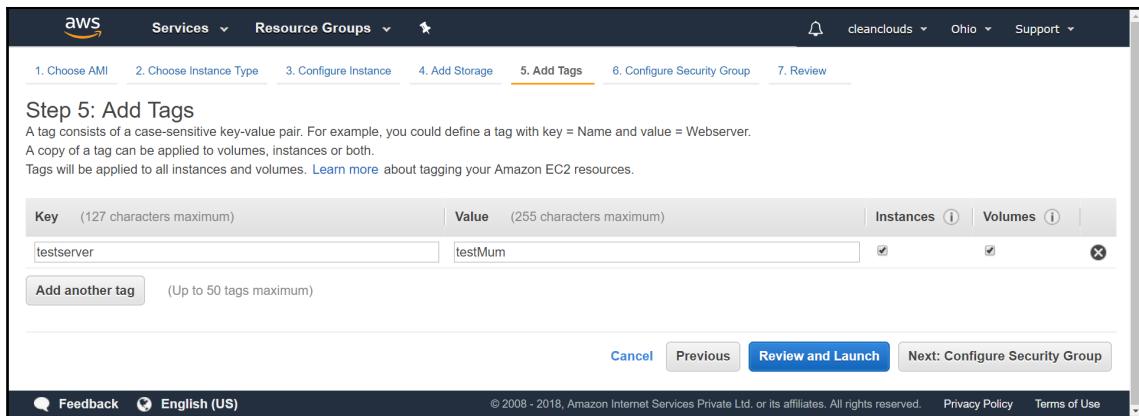
You can provision a consistent amount of IOPS from these volumes without bursting.

There's an extra cost for the amount of IOPS you provision, so it only makes sense to use these in production environments or for things like databases, which really benefit from the high and consistent I/O.

## Throughput optimized HDD

Even though SSD is the new standard, AWS offers hard disk EBS volumes for a cheaper price than the SSD ones. These are optimized for sequential I/O and cannot be used for boot volumes. They offer a burstable throughput, similar to the general purpose SSD.

Notice that there is already a default route volume here of 8 GB general purpose SSD. We could, of course, switch the **Volume Type** to **Provisioned IOPS SSD (io1)**, or use the cheaper **Magnetic (standard)** storage. However, we get up to 30 GB of EBS general purpose SSD storage on the free tier, so we should definitely use that. We can also **Add New Volume** at this time if we would like, or we can add one later on. So, let's just leave it as the default and click **Next: Add Tags**:



What is the use of tagging? Tagging instances are an optional step, and you can use them to give a name for this particular instance. So, let's put Packt Training Instance as our **Value**, and then click the **Next: Configure Security Group** button. So, now we are ready to learn about security groups in our next section.

## Security groups

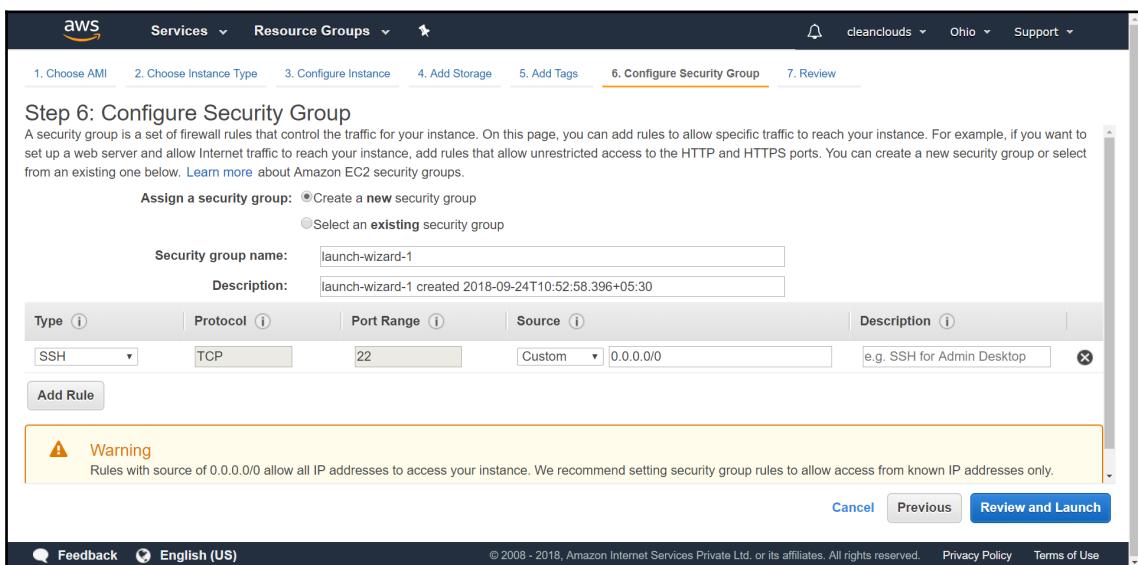
In the previous section, we discussed the available storage options for EC2 instances. The next step to launching our instance is to create a security group. In this section, we're going to show you how to create a security group, add rules, and complete the launch of our instance.

Now we've reached step 6 of launching our instance: configuring the security group.

Security groups allow us to protect our instance with firewall rules. We can allow traffic into our instance by protocol and source. Note that we have to have a security group associated with our instance. A security group could be associated with more than one instance.

So, for example, we can create a single security group for all of our web server instances if we would like. All traffic is implicitly denied by default. So, if you don't specify a rule for a particular protocol, then that traffic is going to be blocked. On the screen, we are adding rules for the inbound traffic. We don't need to worry about outbound, because security groups are stateful, which means that responses to requests will always be allowed. You can edit your security groups later and specify outbound rules as well if you like. However, you usually don't need to do this because there is already a default rule to allow all traffic outbound. Another thing I should mention is that when you modify your security group rules, the changes take place instantly.

There are two options shown on the following screen:



We can either **Create a new security group**, which already has a default rule for SSH traffic, or **Select an existing group** that we created previously. Well, let's go and **Create a new security group**.

Let's start by giving our **Security group a name**. So, let's call it a WebServerSG, and put in a description: SG for Web Servers:

<b>Assign a security group:</b>	<input checked="" type="radio"/> Create a <b>new</b> security group <input type="radio"/> Select an <b>existing</b> security group
<b>Security group name:</b>	WebServerSG
<b>Description:</b>	SG for Web Servers

In order to SSH into our instance, we have to have the SSH protocol allowed. So, there's already a rule in place for that. Now the **Source** here though is listed as **Anywhere**, or it could be **Custom** or **My IP**. You should lock this down. By saying **Anywhere**, it puts an IP address range of `0.0.0.0/0`, which means the entire internet. Since that's not very secure, there's a warning right at the bottom of the page that says, Rules with source of `0.0.0.0` allow all IP addresses to access your instance.

So, instead, you should really put something like **My IP**, or if you know the range of the office where you work, put that in:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	My IP 123.252.235.122/32	e.g. SSH for Admin Desktop

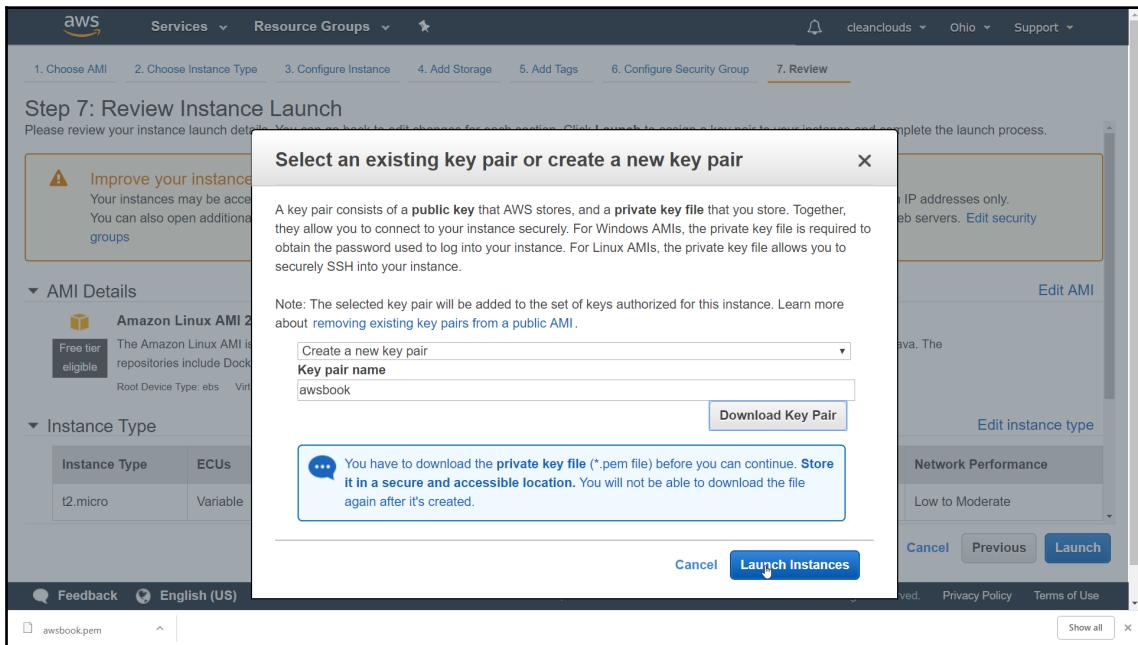
So, that locks it down a lot better. Now since this is going to be for a web server, we need to allow **HTTP** traffic:

Select the type of traffic you want to allow, and you then can select **HTTP**. That will fill in port 80. This we do want to allow from anywhere on the internet, and let's do the same thing for **HTTPS**. While we are doing this, notice all the protocols that are there by default. Plus, you can specify your own custom TCP rules, UDP, and ICMP rules, and specify the port numbers that you want to allow. So, now we've got **SSH** from **My IP** mentioned, and **HTTP** and **HTTPS** from the internet. That should do it.

So, let's click **Review and Launch**:

This page allows us to review the settings for our instance. You can see the **AMI Details**, the **Instance Type**, the **Security Groups**, and the **Storage and Tags** all on the same page.

If you're happy with what we got, then click the **Launch** button. Now, you should see a popup, asking us to choose a **key pair**:



Key pairs are credentials that allow us to log in to our instance. We don't have any created, so we're going to have to create a new one. So, choose **Create a new key pair**.

Give it a name, `awsbook` , and then click **Download Key Pair**. This will save a file to our Downloads folder on our desktop. To complete the launch of our instance, just click **Launch Instances**.

Click on the **View Instances** button down at the bottom, and you'll be able to see the instance being created:

The screenshot shows the AWS EC2 Instances page. On the left sidebar, under the 'INSTANCES' section, 'Instances' is selected. In the main content area, there is a table with one row. The row details a single instance with the following information:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
	i-0899a2dcf09ae1b3d	t2.micro	us-east-2b	running	Initializing	None	ec2-18-1

Below the table, there are tabs for 'Description', 'Status Checks', 'Monitoring', and 'Tags'. The 'Description' tab is active. Under 'Description', the following details are shown:

Instance ID	Public DNS (IPv4)
i-0899a2dcf09ae1b3d	ec2-18-1.2.compute.amazonaws.com

Instance state	IPv4 Public IP
running	[REDACTED]

Instance type	IPv6 IPs
t2.micro	-

Elastic IPs	Private DNS
	ip-124.us-east-2.compute.internal

Availability zone	Private IPs
us-east-2b	[REDACTED]

Security groups	Secondary private IPs
launch-wizard-1 . view inbound rules . view outbound rules	

Scheduled events	VPC ID
No scheduled events	vpc-48023d20

AMI ID	Subnet ID
amzn-ami-hvm-	subnet-1c049966

At the bottom of the page, there are links for 'Feedback', 'English (US)', and copyright information: '© 2008 - 2018, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.' and links to 'Privacy Policy' and 'Terms of Use'.

Notice the **Instance State** will be pending initially, and then, after about a minute, this will switch to running:

The screenshot shows the AWS EC2 Dashboard. On the left sidebar, 'Instances' is selected under the 'Instances' section. In the main content area, there is a table with one row. The row details an instance with the following information:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
i-0899a2dcf09ae1b3d	t2.micro	us-east-2b	running	2/2 checks ...	None	ec2-18-1	

Below the table, the instance's public DNS is listed as `ec2-[REDACTED]114.us-east-2.compute.amazonaws.com`. The 'Description' tab is selected, showing detailed configuration settings:

- Instance ID: i-0899a2dcf09ae1b3d
- Instance state: running
- Instance type: t2.micro
- Elastic IPs:
- Availability zone: us-east-2b
- Security groups: launch-wizard-1, view inbound rules, view outbound rules
- Scheduled events: No scheduled events
- AMI ID: amzn-ami-hvm-
- Public DNS (IPv4): ec2-[REDACTED]114.us-east-2.compute.amazonaws.com
- IPv4 Public IP: [REDACTED]
- IPv6 IPs: -
- Private DNS: ip-172-[REDACTED]us-east-2.compute.internal
- Private IPs: 172-[REDACTED]
- Secondary private IPs: -
- VPC ID: vpc-48023d20
- Subnet ID: subnet-1c049966

## AMIs

An AMI is an image of the filesystem for your boot volume. A base AMI contains an operating system. However, you can launch AMIs that have applications already installed and pre-configured.

These are available from the AWS marketplace or the AWS community, or you can launch custom AMIs that you configure for yourself.

## Quick start

When you click on **Quick Start**, you will get the following screen:

The screenshot shows the AWS Quick Start interface for launching an EC2 instance. The top navigation bar includes the AWS logo, Services dropdown, Resource Groups dropdown, a notification bell, user 'cleanclouds', region 'Ohio', and Support dropdown. Below the navigation is a progress bar with steps: 1. Choose AMI (highlighted in orange), 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group, and 7. Review. A 'Cancel and Exit' button is on the right.

**Step 1: Choose an Amazon Machine Image (AMI)**

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.

A search bar at the top says "Search for an AMI by entering a search term e.g. "Windows"".

The main area displays a list of AMIs:

- Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-0b59bfac6be064b78**  
Free tier eligible  
Select 64-bit
- Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0cf31d971a3ca20d6**  
Free tier eligible  
Select 64-bit
- Red Hat Enterprise Linux 7.5 (HVM), SSD Volume Type - ami-03291866**  
Free tier eligible  
Select 64-bit

On the left, a sidebar titled "Quick Start" has three tabs: "My AMIs", "AWS Marketplace", and "Community AMIs". A checkbox labeled "Free tier only" is checked. At the bottom, there are "Feedback", "English (US)", copyright notice ("© 2008 - 2018, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved."), and links for "Privacy Policy" and "Terms of Use".

You can select the **Free tier only** checkbox to filter the available AMIs.

These include Linux varieties such as Red Hat, Ubuntu, SUSE, and Amazon's own Linux variety, which is optimized to the AWS environment and most closely resembles Red Hat. You can also select the Windows Server operating system, some with SQL Server already installed.



If you choose one of these, or Linux varieties such as Red Hat which have a software license fee, the licensing will be added to the hourly charge for the instance.

When selecting an AMI, some of the features to note are the architecture, 32- or 64-bit, the root volume type, EBS or instance store, and the virtualization type, HVM or para virtual:

The screenshot shows the AWS Launch Wizard interface for launching an EC2 instance. The top navigation bar includes the AWS logo, Services dropdown, Resource Groups dropdown, a notification bell, user 'cleanclouds', region 'Ohio', and Support dropdown. Below the navigation is a progress bar with steps 1 through 7: 1. Choose AMI (highlighted in orange), 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group, 7. Review. A 'Cancel and Exit' button is on the right. The main content area is titled 'Step 1: Choose an Amazon Machine Image (AMI)'. It lists two AMIs:

- Microsoft Windows Server 2016 Base - ami-0ca3e3965ada31684**: Windows, Free tier eligible, 64-bit. Root device type: ebs, Virtualization type: hvm. Includes a note about Amazon RDS and a 'Launch a database using RDS' button.
- Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0f65671a86f061fcf**: Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>). Free tier eligible, 64-bit. Root device type: ebs, Virtualization type: hvm.

At the bottom are 'Feedback' and 'English (US)' buttons, a copyright notice (© 2008 - 2018, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.), and links to 'Privacy Policy' and 'Terms of Use'.

## Community AMIs

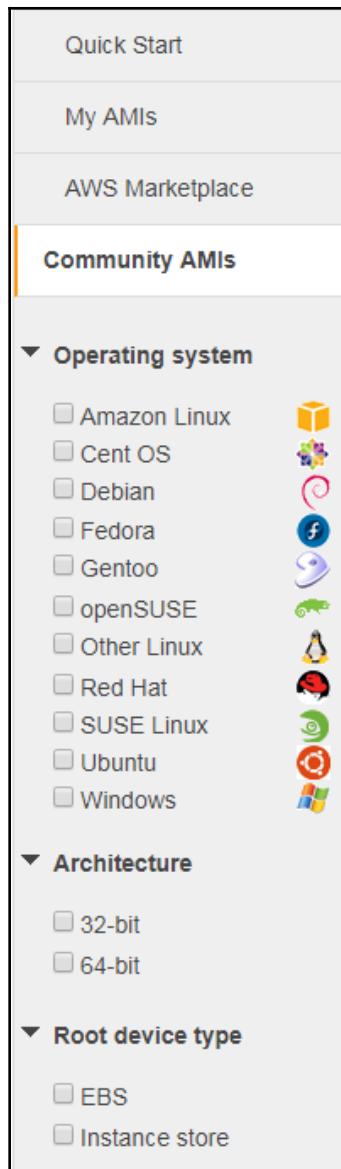
To get an idea of all the options, click on the **Community AMIs** tab:

The screenshot shows the AWS Step 1: Choose an Amazon Machine Image (AMI) interface. The 'Community AMIs' tab is selected. On the left, there's a sidebar with 'Operating system' dropdowns for various Linux distributions like Amazon Linux, Cent OS, Debian, etc. The main area lists several AMIs:

- amzn2-ami-hvm-2.0.20180810-x86\_64-gp2** - ami-0cf31d971a3ca20d6 (64-bit)
- RHEL-7.5\_HVM\_GA-20180322-x86\_64-1-Hourly2-GP2** - ami-03291866 (64-bit)
- suse-sles-15-v20180816-hvm-ssd-x86\_64** - ami-0eb9f58db22854f8f (64-bit)
- ubuntu/images/hvm-ssd/ubuntu/bionic-18.04-amd64-server-20180912** - ami-0f65671a86ff61fcd (Select button)

Each item shows its AMI ID, name, provider (e.g., Amazon Linux 2 AMI), root device type (ebs), and virtualization type (hvm). There are 'Select' buttons next to each entry.

To get the maximum access to the instance memory, you should use the 64-bit AMIs, unless you have some software library that doesn't run well on 64-bit architectures:



The choice of storage for the root volume depends upon the required durability of the data you put on it. If you don't plan on writing any critical data to the boot volume, then you could use **Instance store**, which is much faster than **EBS** volumes.

However, if you stop your instance for any reason, you will need to relaunch it from the AMI, because instance storage is wiped when the instance stops, so you are effectively terminating it. EBS storage persists with a separate lifecycle from the instance, so you can freely stop and start EBS-backed instances, and all of your data is retained. The ability to stop and start your instance is a really nice feature, because an instance that is impaired can usually be fixed by just stopping and starting it. So, my recommendation is to generally choose **EBS** for your root device type for most purposes, unless you need a very fast disk I/O to your root volume.

In the early days of the cloud, **paravirtualization**, or **PV**, was believed to perform faster than a **hardware virtual machine (HVM)**. However, this is no longer the case, and PV virtualization cannot make use of hardware extensions, such as enhanced networking. So, choose HVM whenever possible. Be cautious before you choose an **AMI from the community AMIs tab**. For safety, choose one that says **Provided by Amazon**, or another source that you trust.

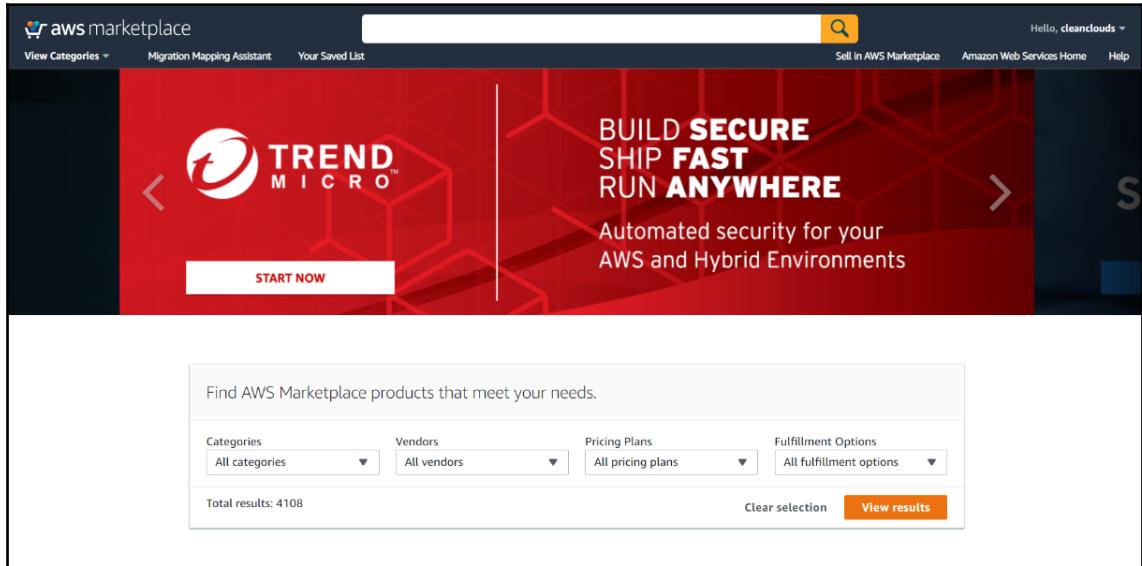
## AWS marketplace

Third-party software vendors make their software available as pre-built AMIs. You can find these under the **AWS Marketplace** tab:

The screenshot shows the AWS Marketplace interface. On the left, there's a sidebar with tabs for 'My AMIs', 'AWS Marketplace' (which is selected), and 'Community AMIs'. Below these are sections for 'Categories' with links to 'All Categories', 'Infrastructure Software (2311)', 'Developer Tools (572)', and 'Business Software (1074)'. The main content area is titled 'Step 1: Choose an Amazon Machine Image (AMI)'. It features a heading 'awsmarketplace' with a shopping cart icon. Below this, a paragraph explains that users can find and buy software from trusted vendors like SAP, Zend, Microsoft, and many open source offerings. It also mentions that users can view Marketplace products they are currently subscribed to by visiting 'Your Software' in the AWS Marketplace. A 'Featured Software' section displays four items: 'Barracuda CloudGen Firewall for AWS - ...' (Rating: ★★★★☆, Sold by Barracuda Networks, Inc.), 'vSRX Next Generation Firewall' (Sold by Juniper Networks, Inc., Starting from \$0.60/hr or from \$4,599/yr (12% savings) for software), 'Matillion ETL for Amazon Redshift' (Rating: ★★★★★, Sold by Matillion, Starting from \$1.37/hr or from \$9,950/yr (17% savings) for software), and 'Trend Micro Deep Security' (Rating: ★★★★★, Sold by Trend Micro, Starting from \$0.01 per host/hr for software usage). At the bottom, there are links for 'Feedback', 'English (US)', and copyright information: '© 2008 - 2018, Amazon Internet Services Private Ltd, or its affiliates. All rights reserved.' followed by 'Privacy Policy' and 'Terms of Use'.

There are thousands to choose from, from well-known and respected software companies such as SAP, Barracuda, and Oracle.

For the complete selection, visit the marketplace at <https://aws.amazon.com/marketplace>:



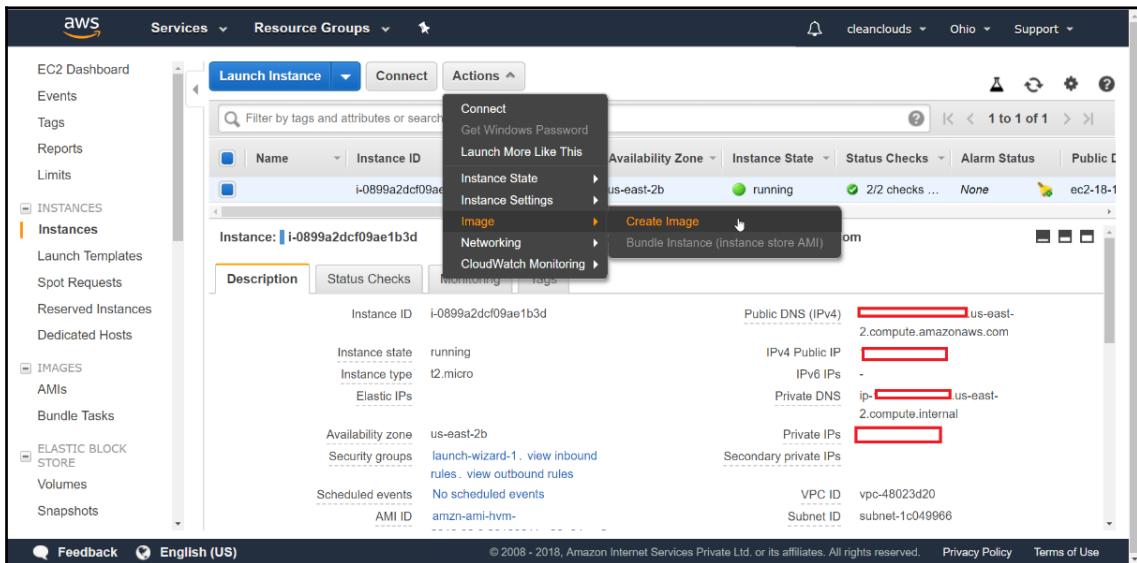
Some of these add the software license to the hourly rate, while others use a bring-your-own license model, in which you need to purchase the license separately.

## My AMIs

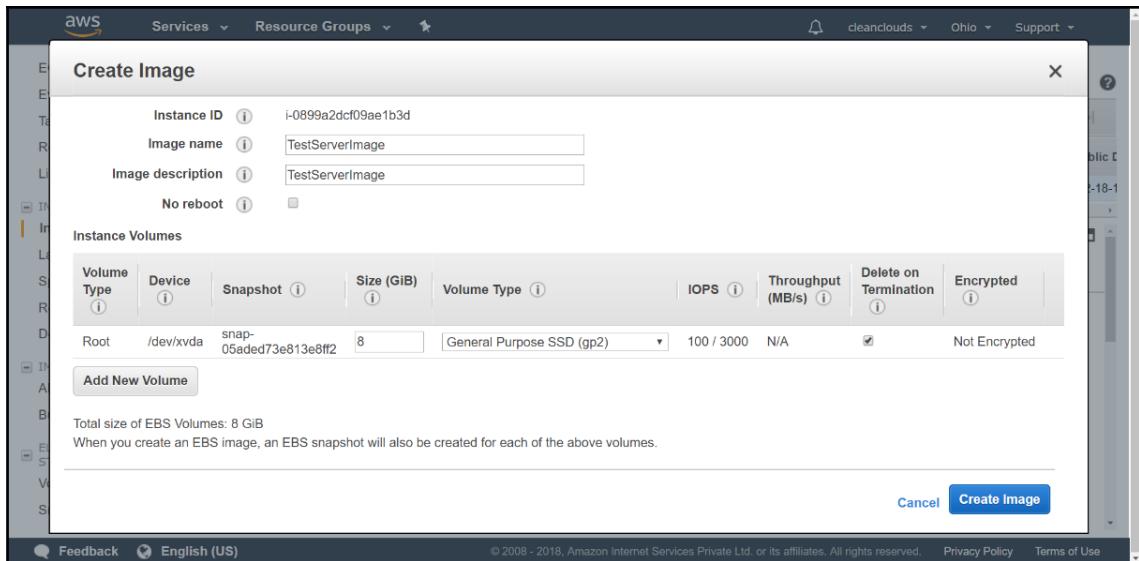
Finally, you can launch any AMI that you create yourself. These are listed under the **My AMIs** tab.

Creating an AMI is a very simple process. It involves launching an EC2 instance from a base AMI; logging in to the operating system; and then adding your own configuration, downloading and installing software, and creating an image from the running instance:

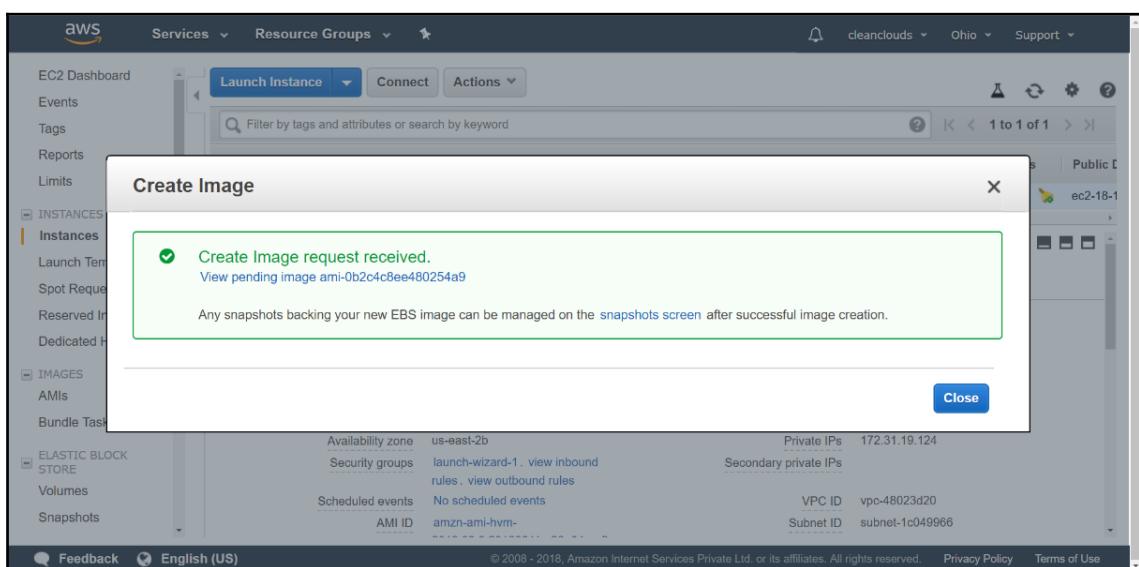
1. Click on **Actions | Image | Create Image**:



## 2. Review Instance Volume:



## 3. Click Close:



4. Go to the **My AMI** section and verify our newly created image:

The screenshot shows the AWS Management Console interface for launching an EC2 instance. The top navigation bar includes 'Services', 'Resource Groups', and a user profile 'cleanclouds'. Below the navigation is a step-by-step wizard: '1. Choose AMI', '2. Choose Instance Type', '3. Configure Instance', '4. Add Storage', '5. Add Tags', '6. Configure Security Group', and '7. Review'. The 'Step 1: Choose an Amazon Machine Image (AMI)' page is displayed. A search bar at the top says 'Search for an AMI by entering a search term e.g. "Windows"'. On the left, a sidebar lists 'Quick Start', 'My AMIs' (which is selected and highlighted in orange), 'AWS Marketplace', and 'Community AMIs'. Under 'Ownership', there are two options: 'Owned by me' (with a checked checkbox) and 'Shared with me' (with an unchecked checkbox). The main content area shows a single AMI entry: 'TestServerImage - ami-0b2c4c8ee480254a9'. It includes a small icon of a server, the name 'TestServerImage', and details: 'Root device type: ebs', 'Virtualization type: hvm', and 'Owner: 511173568473'. To the right of the AMI entry is a blue 'Select' button. At the bottom of the page are links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'.

Launch an instance from your image and verify the previously installed settings.

## Summary

In this chapter, we learned about AMIs and where to locate them. We learned about the five general categories of instance types, their features, and suitable workloads for each. We discussed the storage options for EC2 instances, and showed how to attach these volumes when launching an instance.

In Chapter 3, *Logging in to EC2 Instances*, we're going to learn more about the key pairs and logging in to EC2 instances, and the difference between Linux instances and Windows.

# 3

# Logging in to EC2 Instances

In Chapter 2, *Launching an EC2 Instance*, we covered the process of launching EC2 instances. In this chapter, we will cover how to connect to them.

The topics that we will cover are as follows:

- Key pairs
- Logging in to Linux instances
- Adding Linux users
- Logging in to Windows instances

Key pairs consist of a public key and a private key, and are used to securely connect to your instances. The process is different for Linux and Windows instances. We will discuss ways to generate the keys, and then use them to log in to your instances.

## Key pairs

We will first cover key pairs. You must have a key pair before you launch an instance. We're going to generate a key pair and associate it with an instance.

AWS uses RSA asymmetric public-key cryptography to secure the login information for your instance. Each pair consists of a public key, used to encrypt data, and a private key, used to decrypt data.

When you launch a Windows instance, a random administrator password is automatically created and then encrypted with the public key. You must present the private part of the key to decrypt the password. Then, use a remote desktop client to connect to the instance, and log in as administrator using the password. For Linux instances, AWS will create a Linux user named `EC2-user`, or on Ubuntu, a user called `Ubuntu`. The public key is copied automatically into a file called `authorized_keys`, in the `SSH` directory, in the user's home: `~/.ssh/authorized_keys`. When the user attempts to log in, the server uses the public key to encrypt a challenge message, and sends it to the user's SSH client.

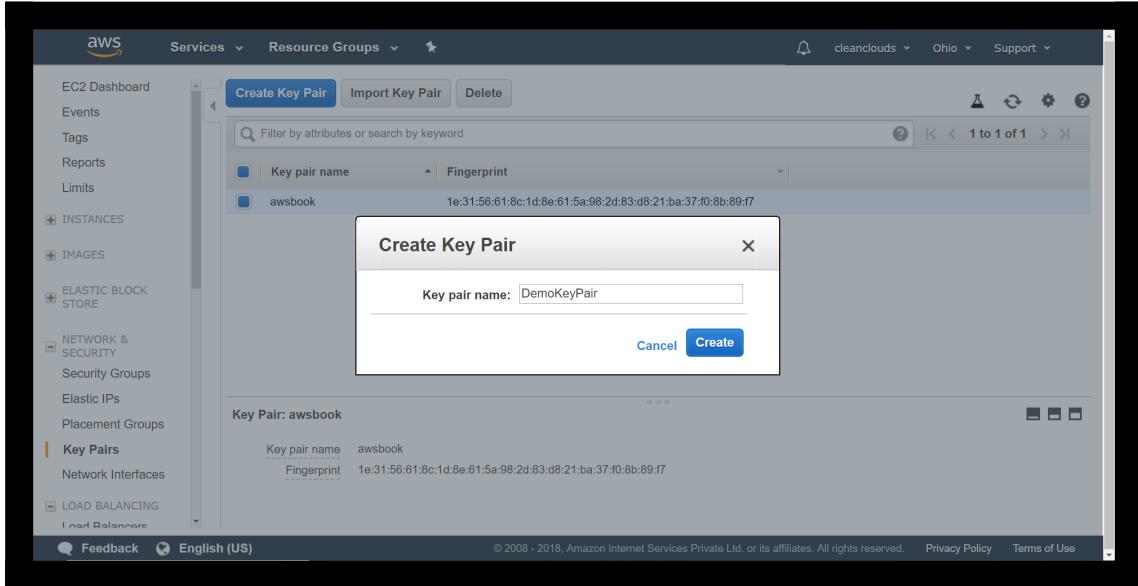
If the client is able to decrypt this message with the private key, then the user is authenticated and can log in. It's important to note that AWS never stores the private part of the key pair. This must be safely stored on the client machine.

Key pairs are very easy to create in the management console. Just go to **EC2**.

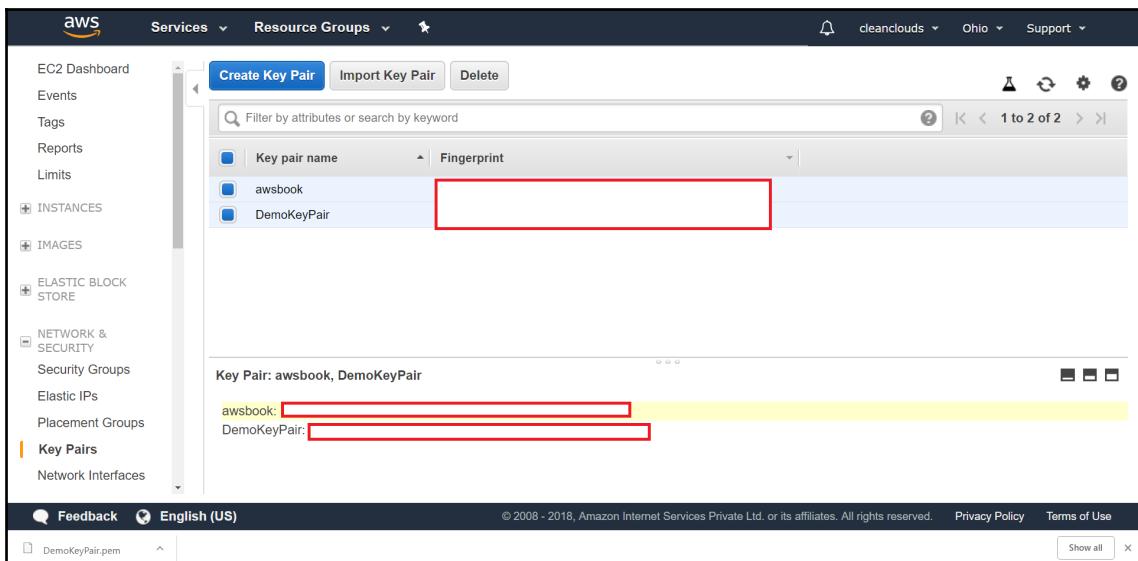
Then, in the left navigation section, go down to **Key Pairs**:

The screenshot shows the AWS Management Console interface for the EC2 service. The left sidebar has a tree view with categories like EC2 Dashboard, Instances, Images, and Network & Security. Under Network & Security, the 'Key Pairs' option is selected and highlighted with an orange bar. The main content area is titled 'Key Pairs' and shows a table with one row. The row contains a checkbox, the key pair name 'awsbook', and its fingerprint '1e:31:56:61:8c:1d:8e:61:5a:98:2d:83:d8:21:ba:37:f0:8b:89:f7'. There are buttons for 'Create Key Pair', 'Import Key Pair', and 'Delete' at the top of the table. A search bar is also present above the table. The bottom of the screen shows standard AWS footer links for Feedback, English (US), Privacy Policy, and Terms of Use.

Click the **Create Key Pair** button, and give it a name:



Click on **Create** and it will download the key:

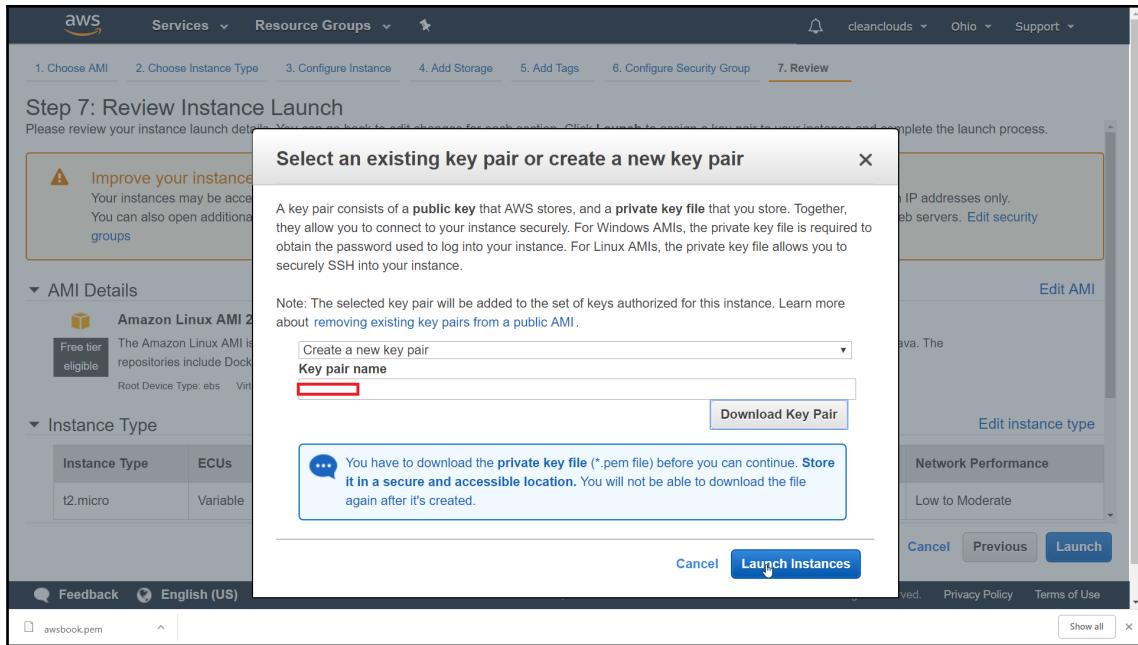


Here, we have added DemoKeyPair:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpaIBAAKCAQEAjjlayH7kkjHpc2qxCpT91AJ45R0rxidxCTLzHE84LGIjP0QXNkmEACbLxiJLS
HfTBksww9rQLY61Wsqef77rJMJh1+bGPsRZwIVjvD7/2CU9dDcgRzvyAo2k5YR3dxHA9e95LGiA
PL2XF7BZSkbsgW0719solQTCMRpm5hkBUB0mCVpCh/ytFBm2DfwRqPD4Hyjew+zDZ2/yn2jjdHMT
ks719rfnudglgHN/dBCtWS7Ac0JIYqEevGxOkoTxs8ojY+8ZcEZntXpho+XQFnv2HfxXpRx1AzEr
v+pr0iA6vSYFnZ2BzJyC1112vTiBl1i1XC2jqGVxuoxetjM+dk5MJxwIDAQABAcIBABzdbRxmDJPT
R3MiQz5hKNoaUnkFTgaBNn+GjMzrm3tQQdGKzu7LhvJ0pWKxtSvmR3nDUT0s580ThtCQ+8Eh4rL
xbSjpLaWeTTGncBV9Oz0We2dRpLoisaT4tSmNuzWkJgZezlVGdm7rOTAcM7PaC64cb3Uf/8GkTcQ
CbPzDx4oDtZF/YlQRyBJlhE/Cjp+iQilZkVC+MBfxTx5QBE7Tn4Yt746YsMV7D0mKV6Iw7AHqOgN
K7c71DdLudJ0nc2rLWrEhPnrvXYL0NiAvle3lm95SJU1LXSDSz3MjypuyvToSBkzhGA31iQ8AKRH
HF1c2ltYe0twR/F1dd64HBCJXVECGYEAv/HM38S1bLSkDq6t+2VvR82Yi2mXd6hGICnOyNlp07zK
GqVviKqi2uGb4g9zH15dWzSAzFAA3i5eaPjTKRranLdFm3z2V+0WvDQZiTUqb2+DvEKqTYmvx2W3
apEsWjDhQKqJ6DTEk8XpgMkNQHvRHGctY+0UsNfrEHyZjywFfkCgYEAv/VtXXMr6pfiFSj6wGs
CUNVBzg0RLIfkjWUAvYfyZuN45HBruyNHNFqMAAz940Hosrjq9XR8D2uaW/ESzz9D2nbrEm3bZ1K
ZY0FtkB0w4gvxikTgWuYTN6gxF485DrU1KBSwaalOpSpNYUorx5B+1WJxTdsfp1FgBEa8cT9Db8C
gYEAIORWjTCeGk0qxB3oj2bSGbOzm5oPIJouj7nBdJU6NkCqkx5x0TKBO0Gz/yGtpHcAdc+YHB7i
v5KqWos7bhFs4GAFEMOqnULF+CvgGm8EeL07YsTY2Lvd1YPBsAshaqF5xLjb6fPYTtvTkLnS7anQ
b3KVGhAHNw+H6D6QNkIKA2EcgYAJjGdimgpysU3by/7wXoYtdTE7is2+w4SpVH8d4wvYlzb+RKEO
UCQOV1BVGD2FcZjykbtCPWIzCLfV4OVtsCVA+GAxQh5wRg0onOuJ/J4wUs+W2T699KT1AUdvWyt
RHTgOy3PR7Xsf1EFu60/m3NQ1o532gFhIkOUXcgQcY0XewKBgQCa5+I9rqheo3HXNrYY5O93n613
d9k7BgPTmX3yRifIgVImWnjgXtrscpoudHjOBKNANegjhtDqi7E5cLXFxxEiKHXvg1MtNcCwcNH
zQr+d/Oi1UtXvzJg3pJtz+L8Ob0giEgxrAth3fNzGPttJxMZxOy7mvNjUpScMqM9hYjJqQ==
-----END RSA PRIVATE KEY-----
```

The private part of the key pair will be automatically downloaded to your desktop, and this is what it looks like. The public part of the key is saved by Amazon.

You'll recall from the previous Chapter 2, *Launching an EC2 Instance*, that the last step when launching an instance is choosing a key pair:



As you can see, we have a popup that allows us to choose an existing key pair, and we can choose from the ones that we've already created. You have to acknowledge that you have access to this key by checking the box shown in the previous screenshot. Now, you can launch your instance by clicking the **Launch Instances** button.



Note that you can use this same key to launch multiple instances, or even all of your instances if you like.

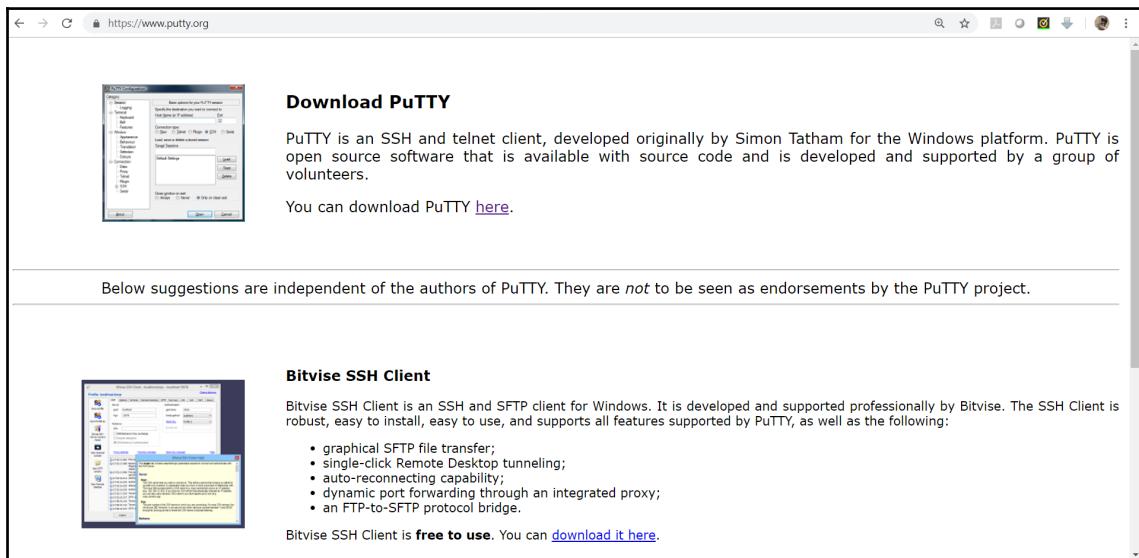
In the next section, we will cover how to log in to Linux EC2 instances.

# Logging in to Linux instances

In the previous section, we looked at EC2 key pairs, and associating them with instances. In this section, we're going to use a key pair to log in to a Linux instance from Windows. Then, we will add more Linux users and generate key pairs for them to use.

To connect and log in to Linux instances, we use the SSH2 protocol. We don't use a password, because we provide the private key for authentication. If you are connecting from a Windows laptop or desktop, you will need an SSH client. There have been rumors that Microsoft is planning on adding an SSH client to PowerShell, but that hasn't happened yet. There are many good free SSH clients you can use.

For this example, we're going to use PuTTY. You can download PuTTY by going to <https://putty.org/> and then clicking the link on the download page:



The screenshot shows a web browser window with the URL <https://www.putty.org>. The main content is the "Download PuTTY" page. It features a screenshot of the PuTTY configuration interface, which includes fields for host name, port, and session name. Below the screenshot, the title "Download PuTTY" is displayed, followed by a brief description of what PuTTY is and where to download it. A note below states that suggestions are independent of the authors. At the bottom, there is another screenshot of a software interface titled "Bitvise SSH Client" with a list of its features.

**Download PuTTY**

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers.

You can download PuTTY [here](#).

Below suggestions are independent of the authors of PuTTY. They are *not* to be seen as endorsements by the PuTTY project.

**Bitvise SSH Client**

Bitvise SSH Client is an SSH and SFTP client for Windows. It is developed and supported professionally by Bitvise. The SSH Client is robust, easy to install, easy to use, and supports all features supported by PuTTY, as well as the following:

- graphical SFTP file transfer;
- single-click Remote Desktop tunneling;
- auto-reconnecting capability;
- dynamic port forwarding through an integrated proxy;
- an FTP-to-SFTP protocol bridge.

Bitvise SSH Client is **free to use**. You can [download it here](#).

You're going to need `putty.exe` to SSH, but if you're going to be working with Linux instances from Windows, you'll probably also need `puttogen.exe`, `pageant.exe`, and `psftp.exe`. So, you should probably just download the ZIP file containing all of the binaries:

The screenshot shows a web browser window displaying the PuTTY download page at <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>. The page title is "Download PuTTY: latest release (0.70)". It includes links for Home, FAQ, Feedback, Licence, Updates, Mirrors, Keys, Links, Team, and a download section for Stable, Snapshot, Docs, Changes, and Wishlist. A note states that the page contains download links for the latest released version of PuTTY (0.70, released on 2017-07-08). It also mentions that new releases will update the page. Below this, it says that release versions are reasonably likely to work well but may not be the most up-to-date. A "Package files" section highlights the MSI ('Windows Installer') and Unix source archive options.

**Package files**

You probably want one of these. They include all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

**MSI ('Windows Installer')**

32-bit:	<a href="#">putty-0.70-installer.msi</a>	(or by <a href="#">FTP</a> )	( <a href="#">signature</a> )
64-bit:	<a href="#">putty-64bit-0.70-installer.msi</a>	(or by <a href="#">FTP</a> )	( <a href="#">signature</a> )

**Unix source archive**

.tar.gz:	<a href="#">putty-0.70.tar.gz</a>	(or by <a href="#">FTP</a> )	( <a href="#">signature</a> )
----------	-----------------------------------	------------------------------	-------------------------------

First, you will need the public IP address of the instance. So from the dashboard, click on EC2 and then **Running Instances**:

The screenshot shows the AWS EC2 Dashboard. On the left sidebar, under the 'INSTANCES' section, there is a link labeled 'Running Instances'. The main content area displays a summary of resources in the US East (Ohio) region:

Value	Description
1	Running Instances
0	Elastic IPs
0	Dedicated Hosts
0	Snapshots
1	Volumes
0	Load Balancers
2	Key Pairs
2	Security Groups
0	Placement Groups

Below this summary, there is a callout box with the text: "Learn more about the latest in AWS Compute from AWS re:Invent 2017 by viewing the EC2 Videos." A blue button labeled "Launch Instance" is visible. To the right, there is a sidebar titled "Account Attributes" and another titled "Additional Information".

Then, make sure your instance is selected, and then go down to the bottom and copy **IPv4 Public IP**, under the **Description** tab:

The screenshot shows the AWS EC2 Dashboard. On the left sidebar, under the 'INSTANCES' section, there is a single instance listed: i-0899a2dcf09ae1b3d. The main content area displays the details for this instance. At the top, there is a 'Description' tab which is currently selected, followed by 'Status Checks', 'Monitoring', and 'Tags'. The 'Description' tab contains the following information:

Attribute	Value	Attribute	Value
Instance ID	i-0899a2dcf09ae1b3d	Public DNS (IPv4)	ec2-18-191-182-114.us-east-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	18.191.182.114
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-19-124.us-east-2.compute.internal
Availability zone	us-east-2b	Private IPs	172.31.19.124
Security groups	launch-wizard-1, view inbound rules, view outbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-48023d20
AMI ID	amzn-ami-hvm-	Subnet ID	subnet-1c049966

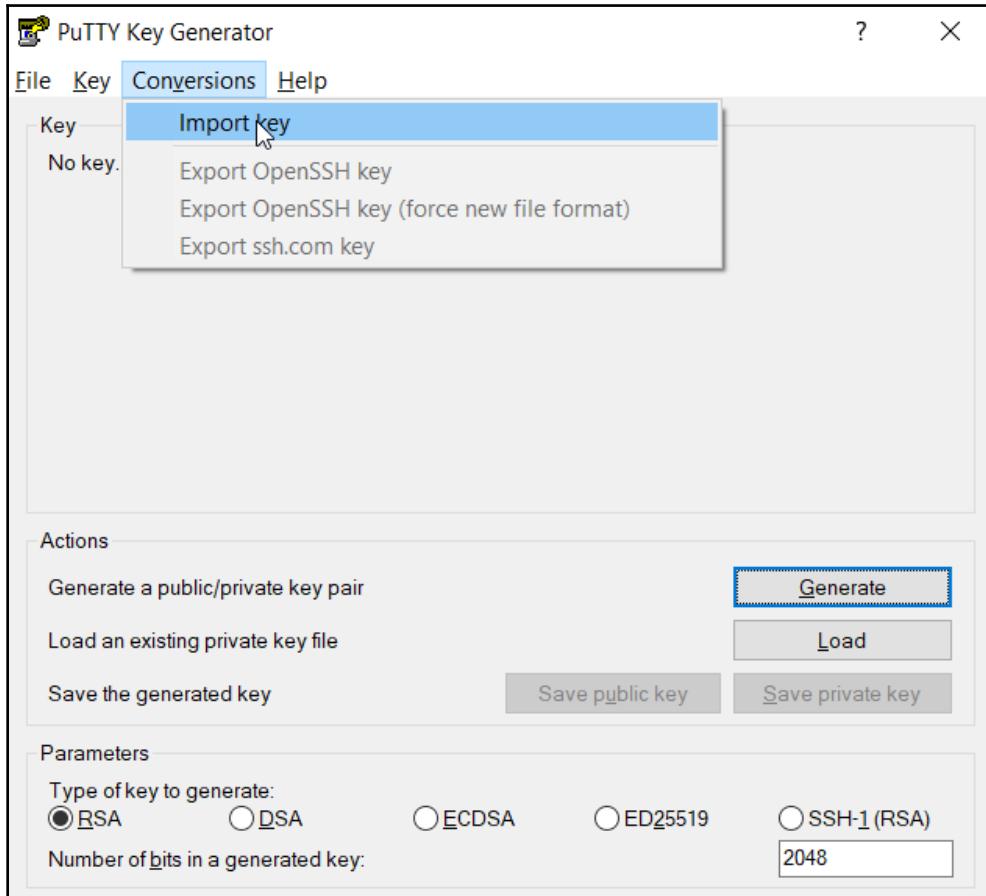
At the bottom of the page, there are links for 'Feedback', 'English (US)', '© 2008 - 2018, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.', 'Privacy Policy', and 'Terms of Use'.

Copy that into the clipboard, and now you have the public IP address for the instance. The last thing we need to do before connecting is to convert our private key file into a format for PuTTY to use.

So, go to your putty folder and launch PUTTYGEN:

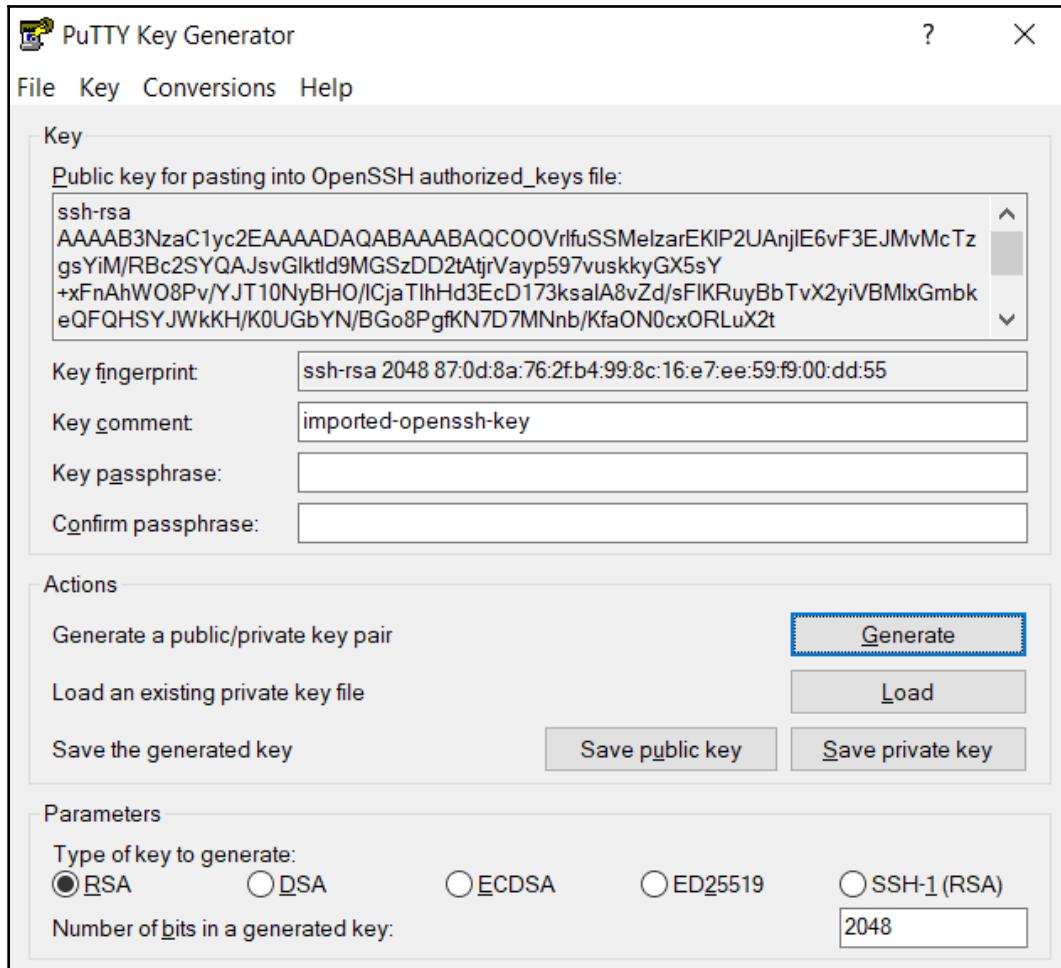
<input type="checkbox"/> Name	Date modified	Type	Size
LICENCE	04-07-2017 19:31	File	2 KB
pageant.exe	04-07-2017 19:34	Application	307 KB
plink.exe	04-07-2017 19:34	Application	603 KB
pscp.exe	04-07-2017 19:34	Application	613 KB
psftp.exe	04-07-2017 19:34	Application	629 KB
putty.chm	04-07-2017 19:31	Compiled HTML H...	277 KB
putty.exe	04-07-2017 19:34	Application	835 KB
<input checked="" type="checkbox"/> puttygen.exe	04-07-2017 19:35	Application	398 KB
README.txt	04-07-2017 19:30	TXT File	2 KB
website	04-07-2017 19:30	Internet Shortcut	1 KB

Go to the **Conversions** menu and choose **Import key**:



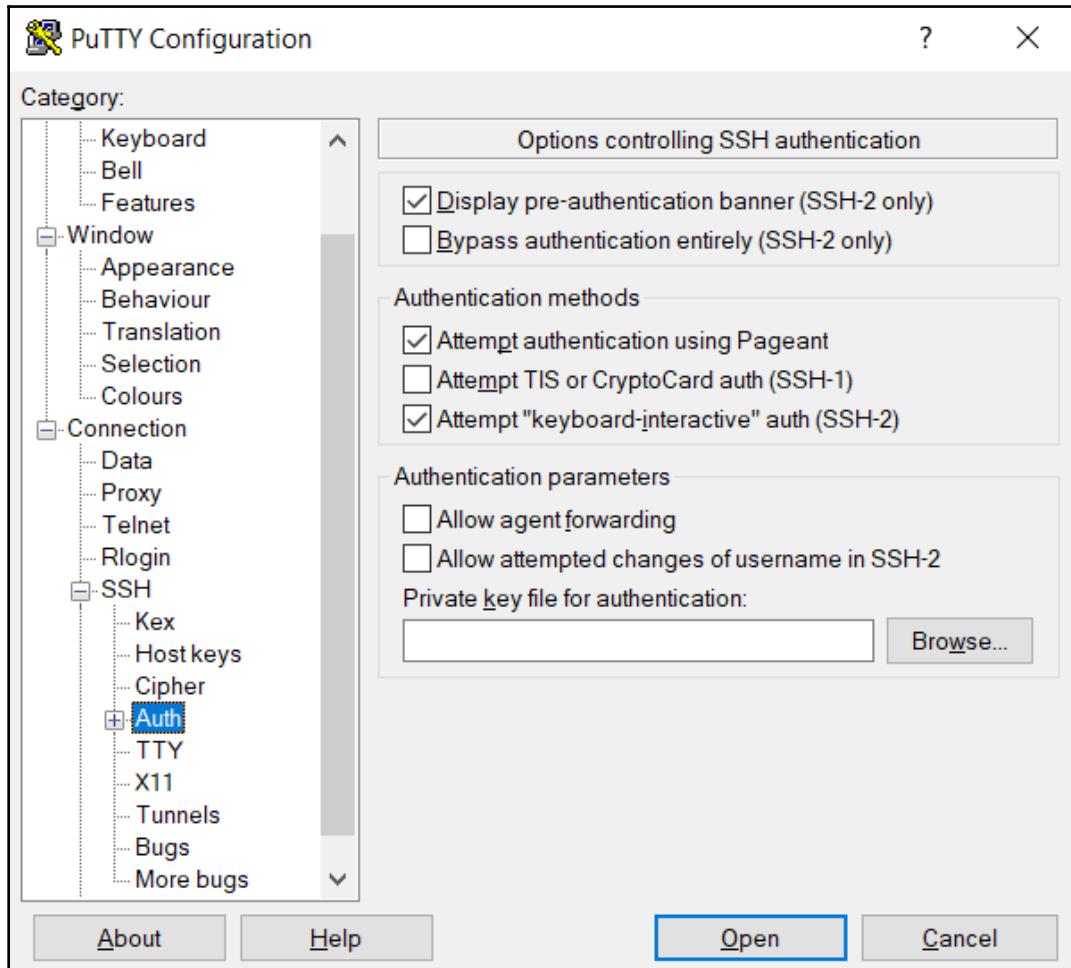
Now, select the key pair file that we downloaded previously and choose **Open**.

This will convert the key into a format that PuTTY can use:



Click on **Save private key**, and save it without a passphrase. Make sure you save it as a .ppk file. Now, we are ready to launch PuTTY and SSH into our instance.

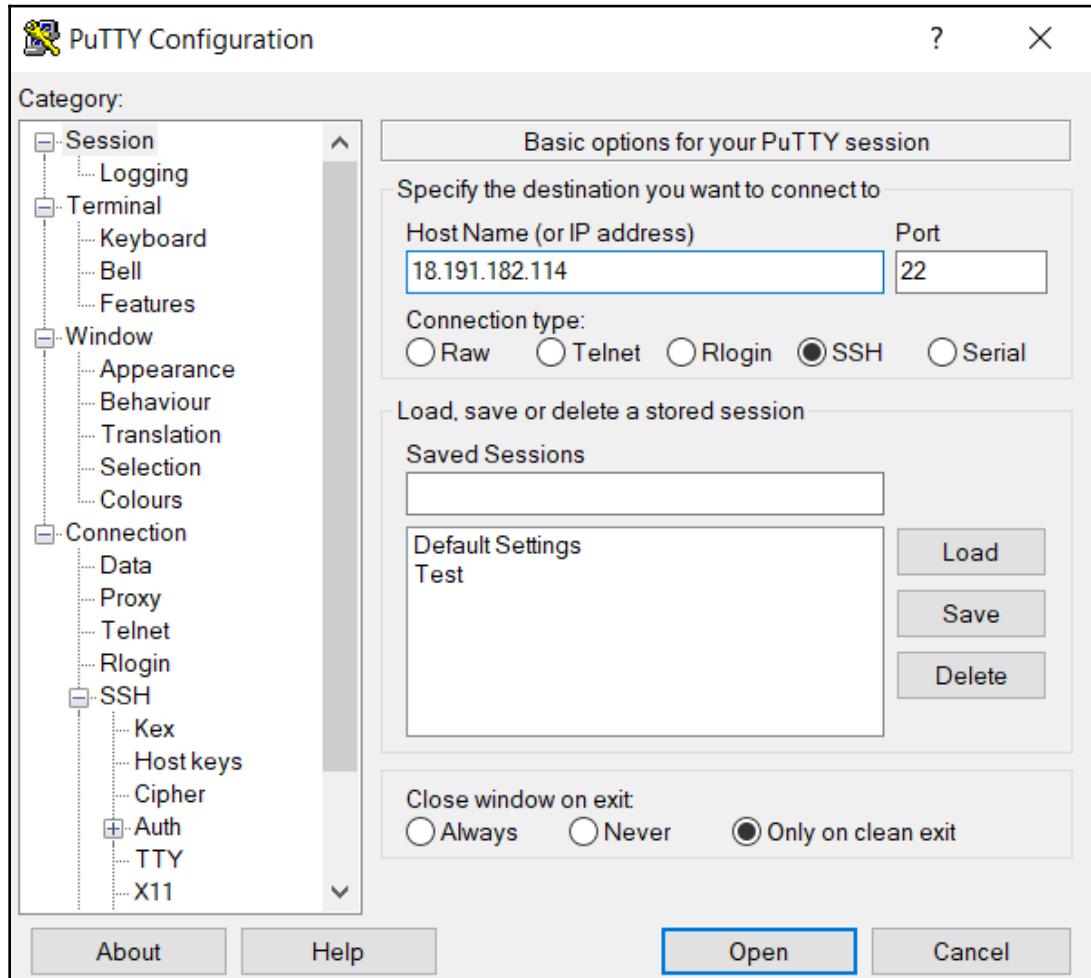
Open PuTTY, and under the **Connection** category, choose **SSH** and then **Auth**:



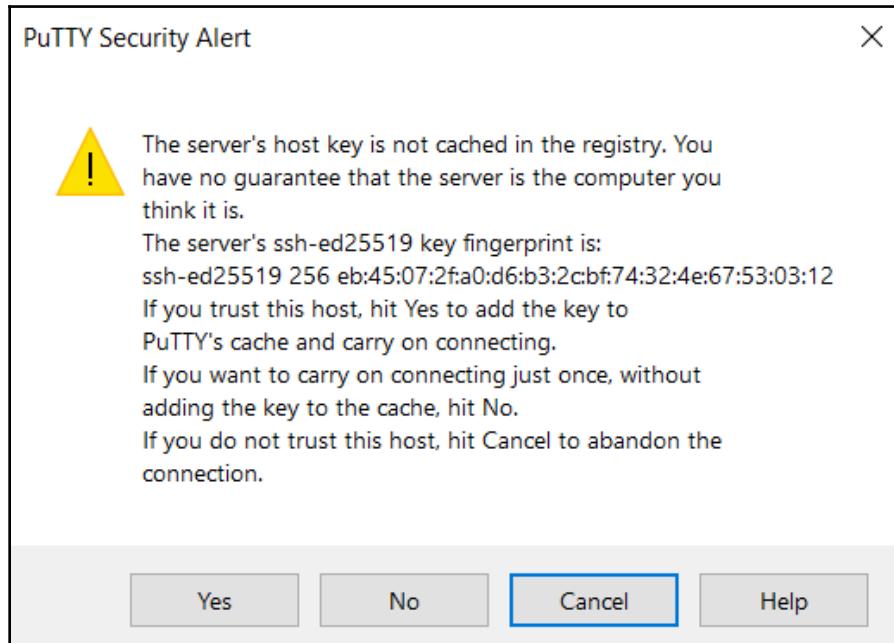
Browse... to the .ppk file and then click **Open**

Now, go back up to the top and choose **Session**.

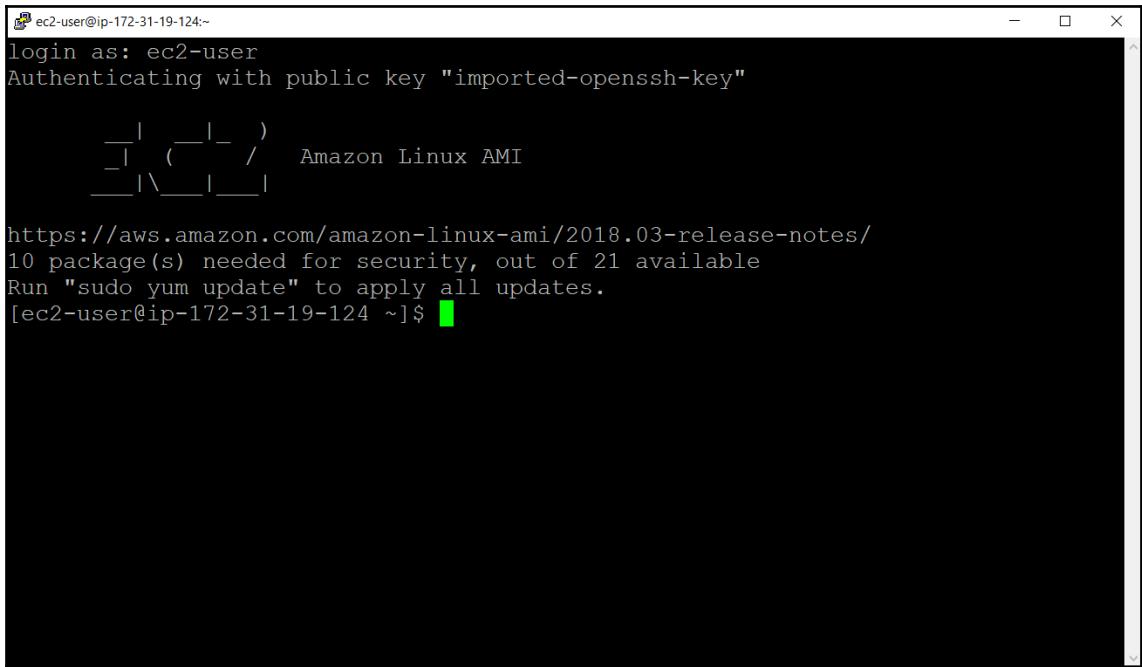
In the **Host Name (or IP address)** field, enter `ec2-user@` and the IP address of the EC2 instance that we saved before. If this was an Ubuntu instance, you would enter `ubuntu` instead of `ec2-user`:



Finally, click the **Open** button to begin the SSH session. Choose **Yes** at the dialog box:



Now, we are logged in to our SSH session:



The screenshot shows a terminal window titled "ec2-user@ip-172-31-19-124:~". The session starts with "login as: ec2-user" and "Authenticating with public key "imported-openssh-key"". It then displays the Amazon Linux AMI logo, which is a stylized tree icon. Below the logo, it says "Amazon Linux AMI". The terminal then shows the URL "https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/" followed by a message indicating "10 package(s) needed for security, out of 21 available". It also suggests running "sudo yum update" to apply all updates. The command prompt "[ec2-user@ip-172-31-19-124 ~]\$" is visible at the bottom.

This user has `sudo` privileges, so just to test that, we can enter the following command:

```
$ sudo yum update
```

This will update the Linux packages:

```
ec2-user@ip-172-31-19-124:~$ sudo apt update
[sudo] password for ec2-user:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Updating: kernel
  x86_64 4.14.70-67.55.amzn1
  amzn-updates 21 M
Updating:
amazon-ssm-agent  x86_64 2.3.68.0-1.amzn1
aws-cli           noarch 1.15.83-1.49.amzn1
e2fsprogs         x86_64 1.43.5-2.42.amzn1
e2fsprogs-libs   x86_64 1.43.5-2.42.amzn1
gnupg2            x86_64 2.0.28-2.33.amzn1
java-1.7.0-openjdk x86_64 1:1.7.0.191-2.6.15.4.82.amzn1
kernel-tools      x86_64 4.14.70-67.55.amzn1
krb5-libs          x86_64 1.15.1-19.43.amzn1
libcom_err         x86_64 1.43.5-2.42.amzn1
libss              x86_64 1.43.5-2.42.amzn1
libxml2             x86_64 2.9.1-6.3.52.amzn1
libxml2-python27    x86_64 2.9.1-6.3.52.amzn1
ntp                x86_64 4.2.8p12-1.39.amzn1
ntpdate            x86_64 4.2.8p12-1.39.amzn1
openssh            x86_64 7.4p1-16.71.amzn1
openssh-clients    x86_64 7.4p1-16.71.amzn1
openssh-server     x86_64 7.4p1-16.71.amzn1
openssl             x86_64 1:1.0.2k-12.110.amzn1
procps             x86_64 3.2.8-45.16.amzn1
python27-botocore  noarch 1.10.82-1.67.amzn1
  amzn-updates 4.5 M
Installing for dependencies:
copy-jdk-configs   noarch 3.3-10.3.amzn1
  amzn-updates 21 k
fuse-libs           x86_64 2.9.4-1.17.amzn1
  amzn-main    98 k

Transaction Summary
=====
Install 1 Package (+2 Dependent packages)
Upgrade 20 Packages

Total download size: 87 M
Is this ok [y/d/N]: 
```

Input y to update packages.

## Logging in to Windows instances

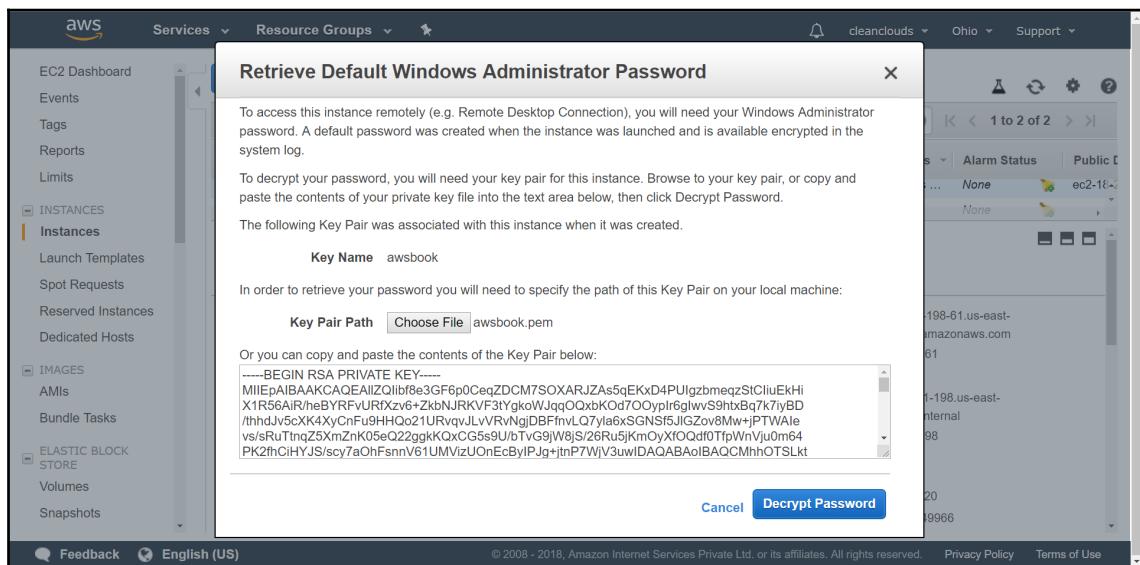
In the previous section, we saw how to connect to Linux instances using a private key for authentication. In this section, we're going to see how the process works for Windows instances.

For Windows, we use the remote desktop protocol, and log in as administrator. We will obtain the administrator password, then we will log in to the instance from Windows. When AWS launches in a Windows instance, it creates a random password for the administrator user. The public key associated with the instance is used to encrypt the password. The encrypted password can only be decrypted with the private key portion of the key pair.

You can decrypt the password easily in the management console.

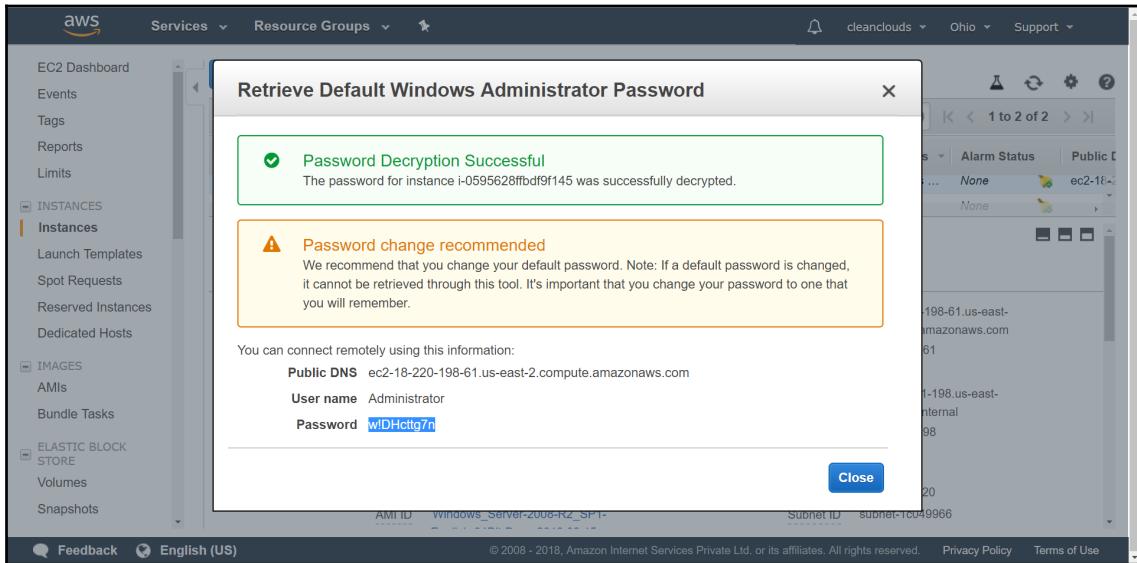
Go to **EC2**, go to **Running Instances**, select the instance, and under the **Actions** menu choose **Get Windows Password**.

Notice that it gives the name of the key pair that was used when the instance was launched:



In this case, it was awsbook. So, we need to select the private key.

Then, click **Decrypt Password**:



Now, we have the password for the administrator login. Let's make a note of the **Public DNS** and **Password**. Copy that and place it on the notepad.

Note that, in order to connect, there has to be a rule in our security group that allows remote desktop. So, let's check the security group and add that rule.

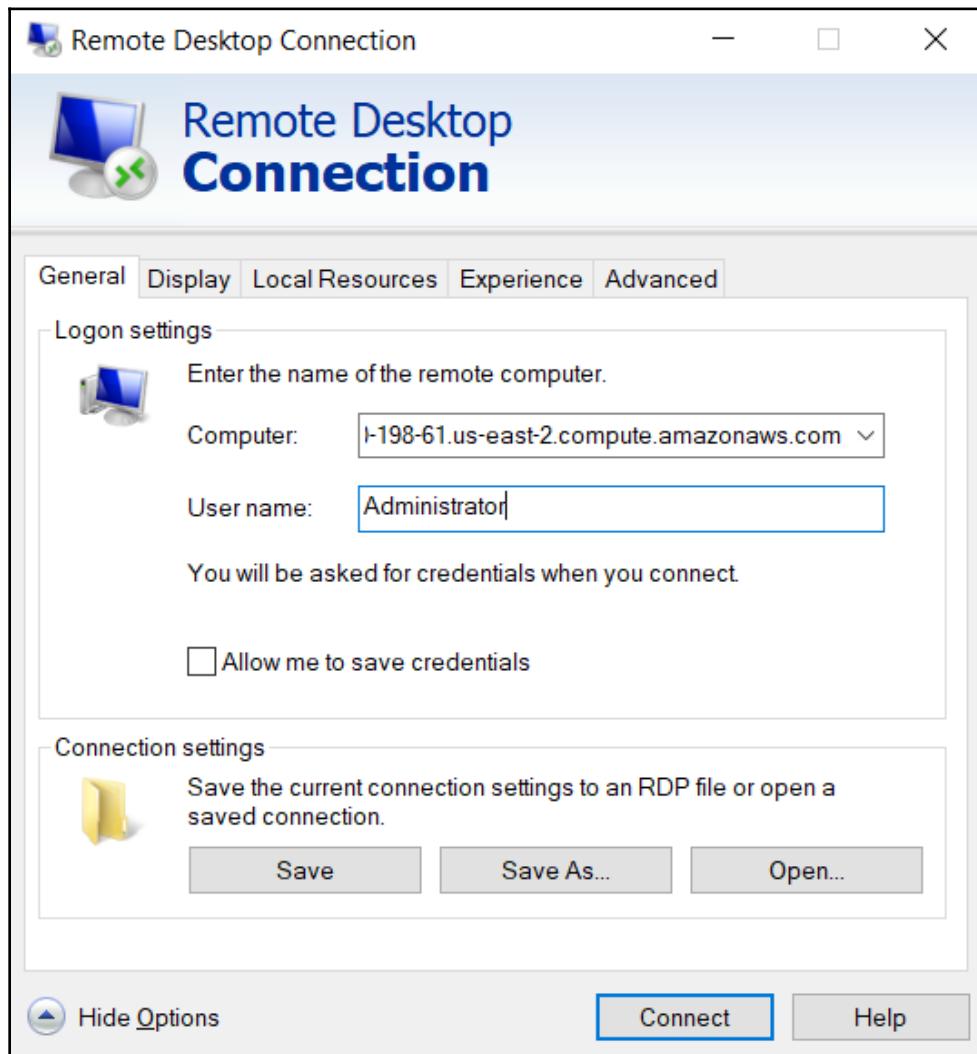
Select RDP, which will already fill in the Port Range 3389:

The screenshot shows the AWS EC2 Security Groups interface. On the left, there's a sidebar with navigation links like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Launch Templates, Spot Requests, Reserved Instances, Dedicated Hosts, Images, AMIs, and Elastic Block Store. The main area shows a table of security groups. One row is selected, showing details for a security group named 'sg-08e1cb4f6c23cc3ef' associated with 'launch-wizard-2' and 'vpc-48023d20'. Below the table, under 'Security Group: sg-08e1cb4f6c23cc3ef', there are tabs for Description, Inbound, Outbound, and Tags. The Inbound tab is selected, showing a single rule: Type: RDP, Protocol: TCP, Port Range: 3389, Source: 0.0.0.0/0. There's also an 'Edit' button. At the bottom, there are links for Feedback, English (US), and a footer with copyright information.

Select your IP address and click on **Save**.

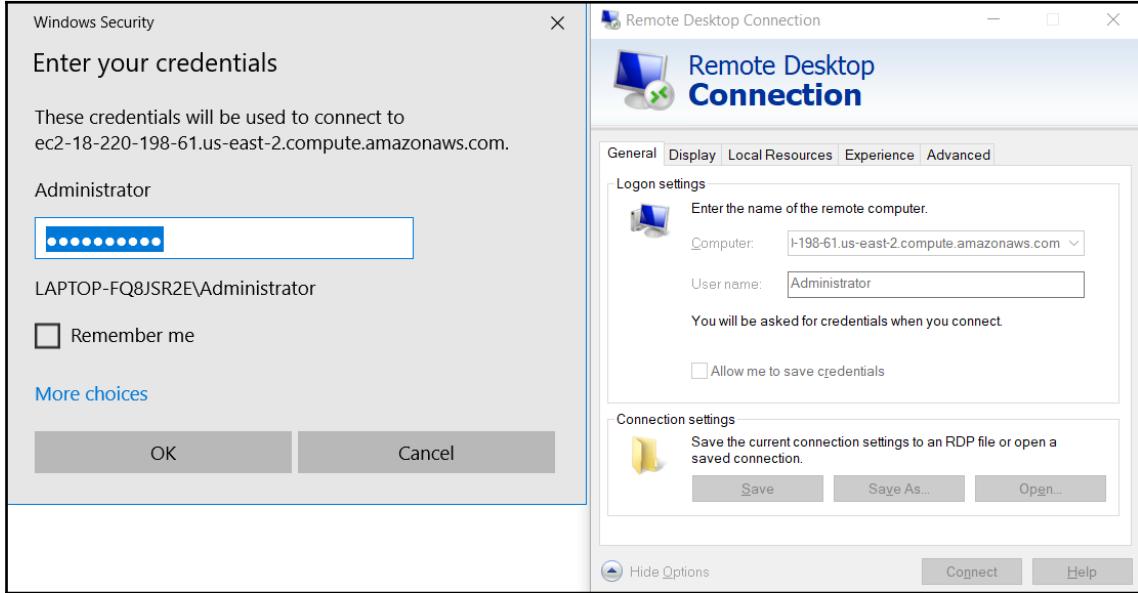
Now, we should be ready to RDP into the instance.

For Windows, we'll just search for the remote desktop client, and enter our connection information. The **User name** should be Administrator:

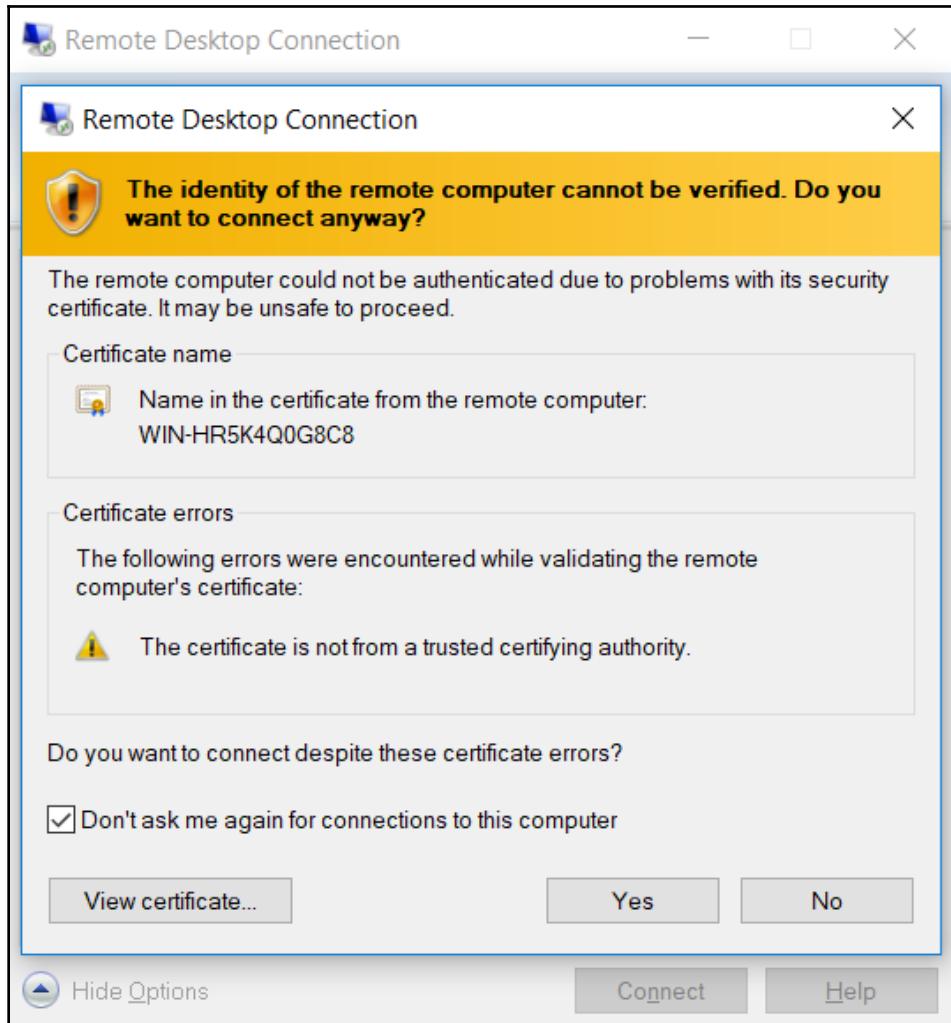


Copy the previously saved password into the clipboard and choose **Connect**.

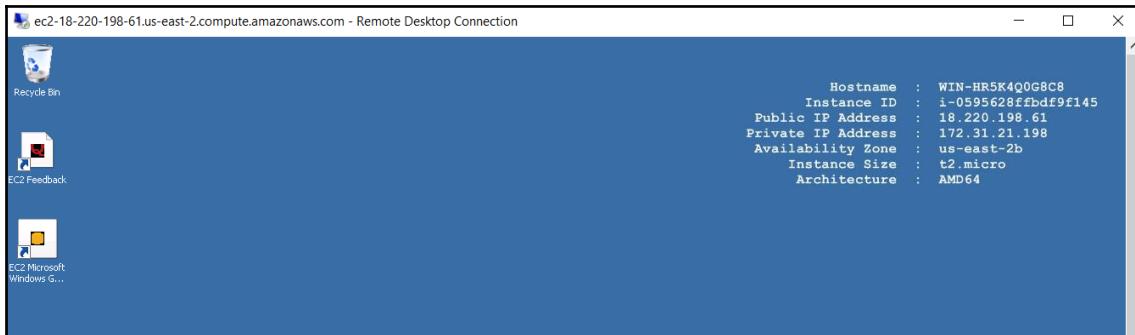
Enter the password and click **OK**:



Verify the identity of the instance:



Now, we have a remote desktop connection to the EC2 instance:



## Summary

In this chapter, we learned about key pairs, how to generate them, and how to associate them with an EC2 instance. We have logged in to a Linux instance, from Windows. We saw how to connect to a Windows instance.

In Chapter 4, *Networking on AWS*, we will cover networking on AWS, including IP addressing, subnets, and routing tables.

# 4

# Networking on AWS

In this chapter, we will explain how to define the IP address block for your virtual private cloud, also known as a VPC. We will start by explaining the **Classless Inter-Domain Routing (CIDR)** notation, which IP addresses are available for you to use for your VPC, and then the three types of IP address on AWS: public, private, and elastic. Next, we will discuss dividing your VPC into one or more subnets, and finally defining routes for your network traffic in route tables.

The main topics that we will cover are as follows:

- CIDR notation
- Private, public, and Elastic IP addresses
- Subnets
- Route tables

Now, let's move on to the first section in this chapter, CIDR .

## CIDR

When you work with VPCs, subnets, security groups, and network access control lists, you will often need to specify IP address ranges. CIDR provides a succinct method, known as CIDR notation, for defining IP address blocks.

In this section, we're going to briefly explain CIDR notation and then review the valid IP address ranges you can use when defining your VPC and subnets. We'll also discuss IP addresses that are reserved by AWS, and why this is important when you're sizing your subnets.

## IPv4

VPCs on AWS use version 4 of the Internet Protocol, known as IPv4. We can assign the IPv6 CIDR block to our VPC, and assign IPv6 CIDR blocks to our subnets. An IPv4 address is defined using 32 bits, broken up into four parts (4x8 bits). Each part is 8 bits, and so their values range from 0 to 255. So, an IP address can be anywhere from 0.0.0.0 to 255.255.255.255. This means that the total number of IP addresses available globally is 2 to the 32 power, or 4,294,967,296, so basically just under 4.3 billion. CIDR notation makes it easy to specify a range of IP addresses by appending a slash and the number of fixed bits, from 0 to 32, after the address. It is easiest to explain this by an example (/24, /28).



By default, Amazon EC2 and Amazon VPC use the IPv4 addressing protocol

For a single IP address, we need to provide all 32 bits, so an example of a single IP would be 192.168.55.31/32. If instead we wanted to define a range of IP addresses, we would use a number less than 32 for the prefix length. When we use /24, for example, we are saying that the first 24 bits are fixed, and the remaining 8 bits are unspecified. This would give us an IP address block consisting of 256 addresses:

192.168.55.0/24 - 192.168.55.0 → 192.168.55.255 (256 addresses)

For /16, there would be 32 minus 16, which equals 16 unspecified bits, and so this represents 65,536 addresses:

192.168.0.0/16 - 192.168.0.0 → 192.168.255.255 (65,536 addresses)

As a final example, /20 would leave 12 bits unspecified, or 4,096 addresses:

192.168.96.0/20 - 192.168.96.0 → 192.168.111.255 (4,096 addresses)

the following is an example of the available range for IP addresses, Subnet Masks, and IP Quantity:

CIDR Block	IP Range	Subnet Mask	IP Quantity
<b>10.0.0.0/32</b>	<b>10.0.0.0 - 10.0.0.0</b>	<b>255.255.255.255</b>	<b>1</b>
<b>10.0.0.0/31</b>	<b>10.0.0.0 - 10.0.0.1</b>	<b>255.255.255.254</b>	<b>2</b>
<b>10.0.0.0/30</b>	<b>10.0.0.0 - 10.0.0.3</b>	<b>255.255.255.252</b>	<b>4</b>
<b>10.0.0.0/29</b>	<b>10.0.0.0 - 10.0.0.7</b>	<b>255.255.255.248</b>	<b>8</b>
<b>10.0.0.0/28</b>	<b>10.0.0.0 - 10.0.0.15</b>	<b>255.255.255.240</b>	<b>16</b>
<b>10.0.0.0/27</b>	<b>10.0.0.0 - 10.0.0.31</b>	<b>255.255.255.224</b>	<b>32</b>
<b>10.0.0.0/26</b>	<b>10.0.0.0 - 10.0.0.63</b>	<b>255.255.255.192</b>	<b>64</b>
<b>10.0.0.0/25</b>	<b>10.0.0.0 - 10.0.0.127</b>	<b>255.255.255.128</b>	<b>128</b>
<b>10.0.0.0/24</b>	<b>10.0.0.0 - 10.0.0.255</b>	<b>255.255.255.0</b>	<b>256</b>



Here is a link to one of the many online calculators that will convert CIDR notation to starting and ending IP addresses: <http://www.ipaddressguide.com/cidr>.

## Valid private IP address ranges

For VPCs, we need to specify private IP address blocks. The Internet Assigned Numbers Authority (IANA) has reserved certain IP address ranges to be used for private IPs.

The following are few examples:

10.0.0.0	-	10.255.255.255	(16,777,216 addresses)
172.16.0.0	-	172.31.255.255	(1,048,576 addresses)
192.168.0.0	-	192.168.255.255	(65,536 addresses)

So, the CIDR blocks for your VPCs need to fit into these ranges.

You can find some of them defined by RFC1918, as follows:

<https://tools.ietf.org/html/rfc1918>

On AWS, VPCs can be made as small as /28, which is 16 addresses, all the way up to /16, which is 65536 addresses.

However, think twice before you create a VPC as small as /28. You will have to create at least one subnet in your VPC, and for every subnet in your VPC, five addresses are reserved by AWS. These are the first four and last one in the subnet block. So, that leaves only 11 IP addresses for your /28 VPC:

IP Addresses	Reserved for
10.0.0.0	Network address
10.0.0.1	Reserved by AWS for the VPC router
10.0.0.2	Reserved by AWS
10.0.0.3	Reserved by AWS for future use
10.0.0.255	Network broadcast address

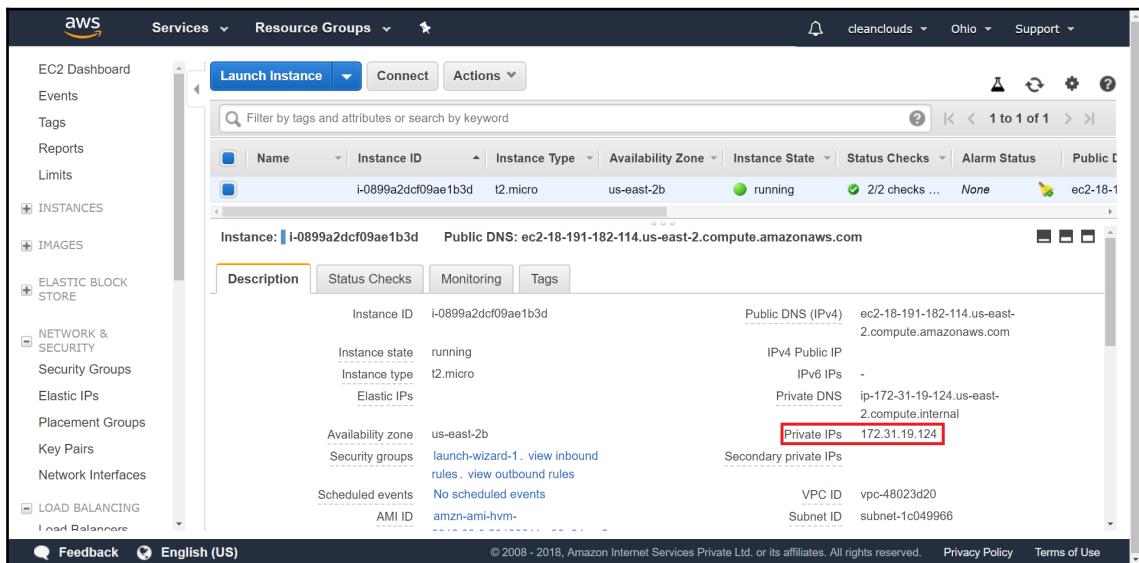
This completes our discussion of CIDR notation. In the next section, we will cover the types of IP address you can assign to your EC2 instances, and we will introduce Elastic Network Interface.

# EC2 IP addressing

In this section, we're going to continue our discussion of private IP address and the two types of public IP address you can use for your EC2 instances. We are also going to discuss Elastic Network Interface, which are the virtual network cards attached to your instance.

## Private IP addresses

Every instance in your VPC must be placed inside a subnet. A random available private IP address from the subnet range is automatically assigned to each instance. Private IP addresses are not routable from the internet:



The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with navigation links: EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES, IMAGES, ELASTIC BLOCK STORE, NETWORK & SECURITY (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), and LOAD BALANCING (Load Balancers). The main area displays a table with one row for an instance. The instance details are as follows:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
i-0899a2dcf09ae1b3d	t2.micro	us-east-2b	running	2/2 checks ...	None	ec2-18-1...	ec2-18-1...

Below the table, there are tabs for Description, Status Checks, Monitoring, and Tags. Under the Description tab, detailed information is shown:

Instance ID	Public DNS (IPv4)
i-0899a2dcf09ae1b3d	ec2-18-191-182-114.us-east-2.compute.amazonaws.com

Under the Status Checks tab, it shows:

Instance state	running
Instance type	t2.micro
Elastic IPs	

Under the Monitoring tab, it shows:

Availability zone	us-east-2b
Security groups	launch-wizard-1. view inbound rules, view outbound rules

Under the Tags tab, it shows:

Scheduled events	No scheduled events
AMI ID	amzn-ami-hvm-

On the right side, there are sections for Public DNS (IPv4), IPv6 IPs, Private DNS, and Private IPs. The Private IPs section shows "172.31.19.124" with a red box around it. Below that, it says "Secondary private IPs". At the bottom, it shows VPC ID and Subnet ID.

If you want your instance to be reachable from the internet, then you will need a publicly routable IP address assigned to the instance's Elastic Network Interface. There are two types of public IP address that can be assigned to your instance.

## Public IP addresses

AWS owns a large but finite number of public IP addresses. When you launch an instance, you can choose to have a public IP randomly assigned to it. You only need a public IP address if you want the instance to be reachable from the internet, so it is optional. When you create a subnet, which we will do later, you can set the subnet so that instances that launch in it will get a public IP automatically. However, when launching an instance, you can choose to override that and not receive a public IP.

There is no cost for assigning public IPs to your instance; however, to follow best security practices, you shouldn't get a public IP for your instance if you don't need it. One of the nice features of EC2s is that you can stop them whenever you want to stop the billing. However, when you stop your instance, AWS will take back your public IP address. When you start your instance up again, it will be assigned a new public IP address, different from the original. This will cause a problem if applications that connect to your instance have hardcoded the old address. Also, you would have to update any DNS records that point to the old address. Even if you don't plan on stopping your instance, sometimes you have to if the instance becomes impaired:

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with links like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Images, Elastic Block Store, Network & Security, Load Balancing, and Feedback. The main area has tabs for Launch Instance, Connect, and Actions. A search bar at the top says "Filter by tags and attributes or search by keyword". Below it, a table lists one instance: i-0899a2dcf09ae1b3d, t2.micro, us-east-2b, running, 2/2 checks, None, and Public DNS ec2-18-191-182-114.us-east-2.compute.amazonaws.com. Below the table, there are tabs for Description, Status Checks, Monitoring, and Tags. Under the Description tab, detailed information is shown in a grid format:

Instance ID	i-0899a2dcf09ae1b3d	Public DNS (IPv4)	ec2-18-191-182-114.us-east-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	18.191.182.114
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-19-124.us-east-2.compute.internal
Availability zone	us-east-2b	Private IPs	172.31.19.124
Security groups	launch-wizard-1. view inbound rules, view outbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-48023d20
AMI ID	amzn-ami-hvm-	Subnet ID	subnet-1c049966

To avoid this problem, AWS has a second type of public IP address, known as an Elastic IP.

## Elastic IP addresses

A public IP is given to an instance when you launch or start it, but an Elastic IP is independent of the instance lifecycle.

You can, at any time, ask for an Elastic IP address. It is given to your account, not a specific instance. One of the great things about an Elastic IP is that you can associate it with an instance that already exists, even if it is already running. And if you stop the instance, it does not become dissociated from the instance. So, you can freely stop and start EC2 instances with Elastic IPs, and they will retain the same public address. By default, you can request up to five Elastic IPs per region. If this isn't enough, you can request more by contacting AWS support. One Elastic IP is free per instance.



However, if your instance is not in the running state, you'll be charged an hourly fee. So that means that, when you stop an instance, you'll be charged a small fee per hour for the Elastic IP that is associated with it. Also, if you have Elastic IPs in your account that are not associated with a running instance, you will be charged.

## Elastic network interface (ENI)

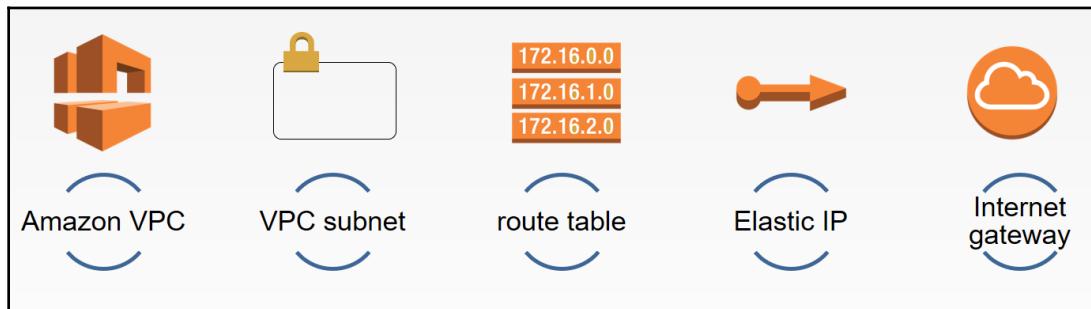
Each EC2 instance has one or more Elastic Network Interface, or ENIs, attached, and the IP addresses are actually assigned to these. You can think of them as virtual network cards.

Each instance has one ENI, designated `eth0`, attached by default, known as the primary ENI. The IPs given to your instance at launch are assigned to the primary ENI. The primary ENI cannot be detached from the instance. However, you can create additional ENIs, and attach them as secondary network interfaces to an instance. This doesn't increase the overall network capacity of the instance, but you can detach a secondary ENI and move it to another instance. This will actually move the IP addresses with it. So if you need to replace an impaired instance, or just want to replace it with a new type, you can launch a new instance and move the secondary ENI over from the old instance. This will move the associated IP addresses, so you won't have to update DNS or application config files. The ENI will have one primary private IP address, and you can add one or more secondary private IP addresses. All of these would of course come from your subnet's CIDR block. You can choose to associate an Elastic IP for each of the private IPs.

However, only one Elastic IP per instance is free. You will have to pay a small hourly rate for more than one Elastic IP per instance. The regular public IP address is also a property of the ENI, but if you associate an Elastic IP, the public IP will be returned to AWS. Security groups are also associated with the ENI, and are not directly associated with the instance. Each ENI will have a unique MAC address. This is useful if you have a software license tied to a specific MAC address. If you want to move the application to a different instance, you can move the ENI, and that will move the MAC address. Finally, ENIs have an attribute known as the source destination check flag. Enabled by default, this blocks network traffic that isn't destined for your instance. You can disable this check if you want your instance to do network address translation, routing, or serve as a firewall. In the next section, we will describe subnets in more detail and introduce route tables.

## Subnets and route tables

In the previous section, we learned about private, public, and Elastic IP addresses, and how they are assigned to the ENIs attached to an instance. Following are a few components that are important in the Amazon VPC:



In this section, we're going to discuss subnets and route tables, and how a route in the route table can make a subnet public or private. Then, we'll talk about NAT instances and NAT gateways, to give instances and private subnets access to the internet.



Do you need to create a VPC the moment you create your account? The answer is, No. A default VPC is available on the Amazon VPC. If you delete the default VPC then you cannot restore it. You need to contact AWS Support.

## What are subnets?

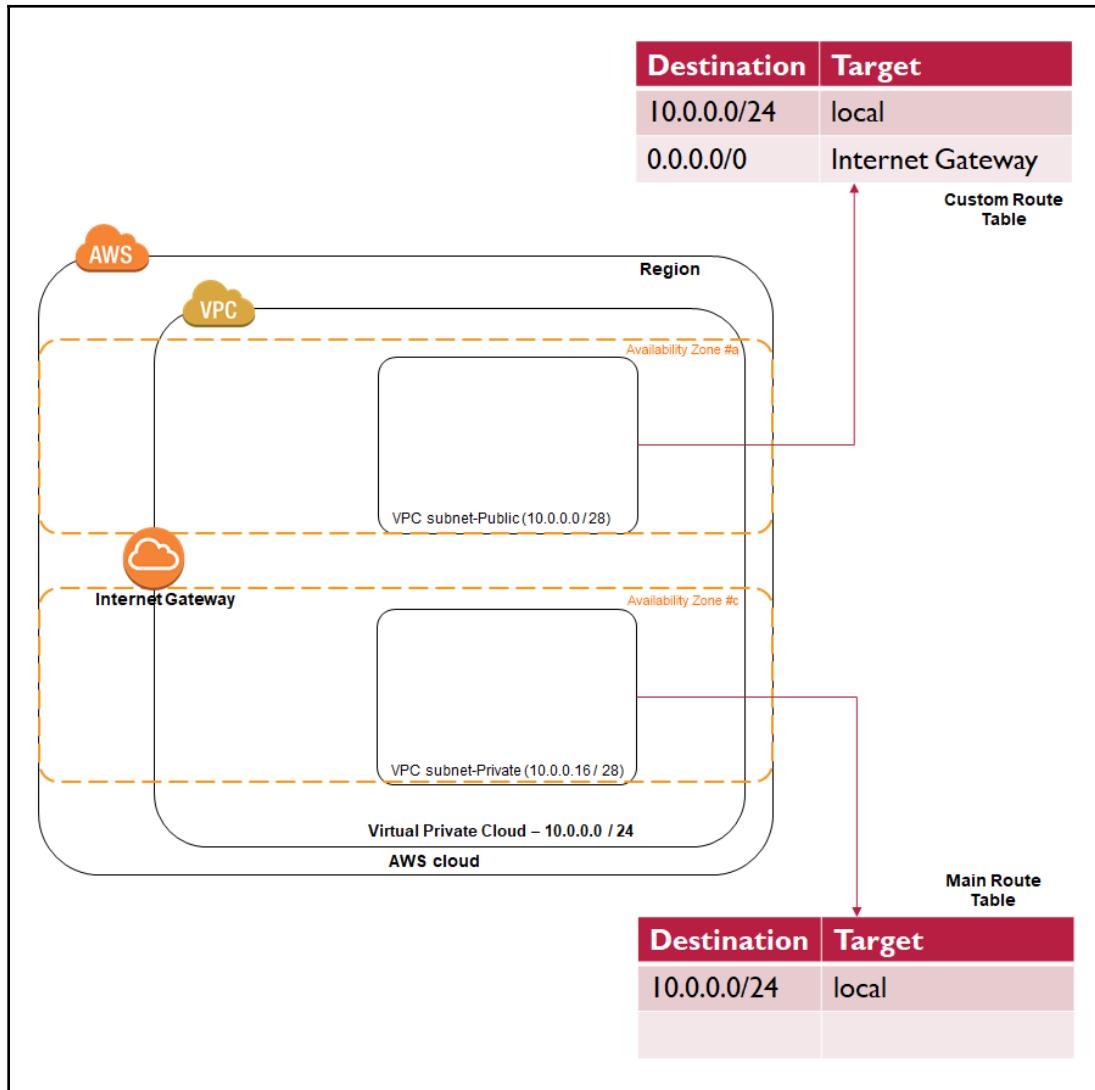
Subnets are separate portions of the VPC. EC2 instances must be launched into a subnet, so at least one subnet is required for a VPC. To create a subnet, you have to define a CIDR block for it, which is a subset of the VPC's CIDR block. Every subnet in a VPC has to have a unique range of addresses. There can't be any overlap, or else there would be a duplicate IP address in your network.

## Route tables

Now, let's talk about route tables. For instances to communicate, there has to be a route that defines a path for the network traffic to take. A route consists of a destination, designated as a CIDR address block, and a target, which defines the path to take. An example of a route would be a local route that allows communications to all instances in the local network, in our case a VPC:

Destination	Target
10.0.0.0/16	Local
192.168.0.0/20	pcx-1234abcd
0.0.0.0/0	igw-1a2b3c4d

To communicate with instances in another VPC, you could define a VPC peer connection and define the peer connection ID as the target. To access the internet, you could define an internet gateway as a target. Every VPC will have a default route table with a single local route; this defines a route so instances inside a VPC can communicate with one another:



You cannot remove this route, but you can customize the table by adding additional routes. You can create additional route tables and associate them with your subnets. This allows you to create what are known as public and private subnets.

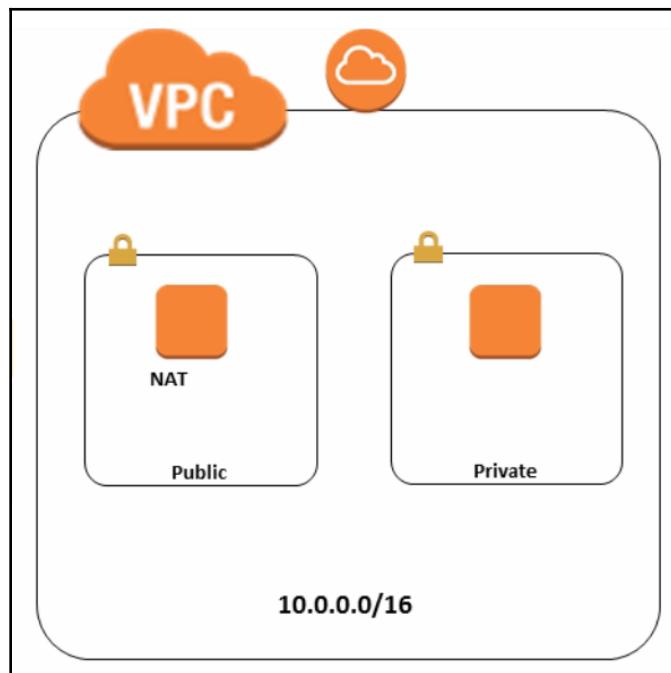


Each subnet can have only one route table. Multiple subnets can be assigned to a single route table.

## Difference between public and private subnets

A public subnet will be used for instances that need a public IP to be accessible from the internet. The way we make it a public subnet is to associate a custom route table and add a route to the internet, with an internet gateway as a target. Notice that the internet is defined with CIDR notation as  $0\text{.}0\text{.}0\text{.}0/0$ . An internet gateway must first be attached to your VPC. If you don't have an internet gateway attached to your VPC, none of your instances will be able to reach the internet.

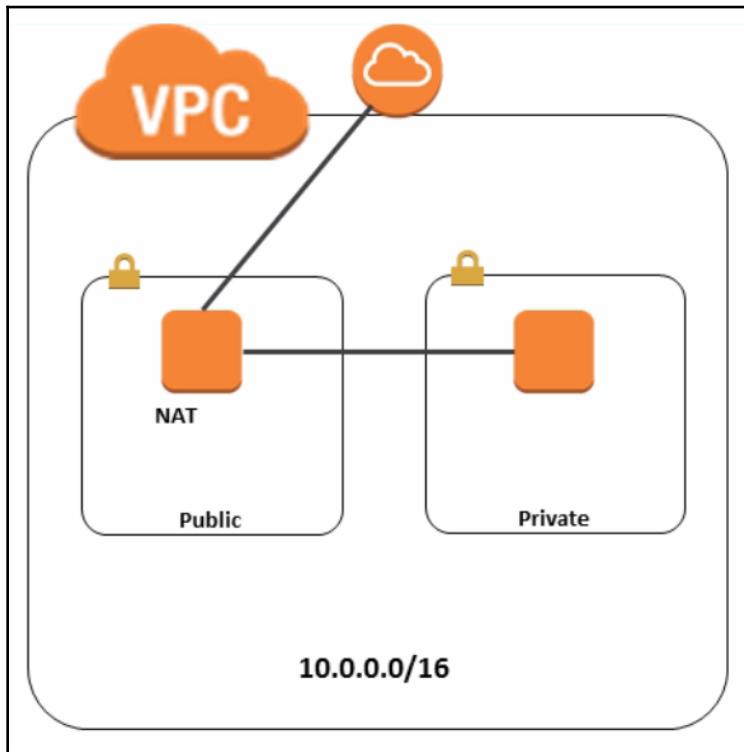
A private subnet is used for instances that do not need to be directly reachable from the internet. For the best security, it's important to keep backend instances and databases private, so those would go in a private subnet. A private subnet would be associated with a route table that does not have a route to the internet through the internet gateway. Even though private instances will not be directly reachable from the internet, we will probably need to provide a way for them to get out to the internet, to download software patches and to access external services such as AWS's DynamoDB, a NoSQL database. To do this, we can configure an instance in a public subnet to do Network Address Translation, or NAT:



## NAT instance

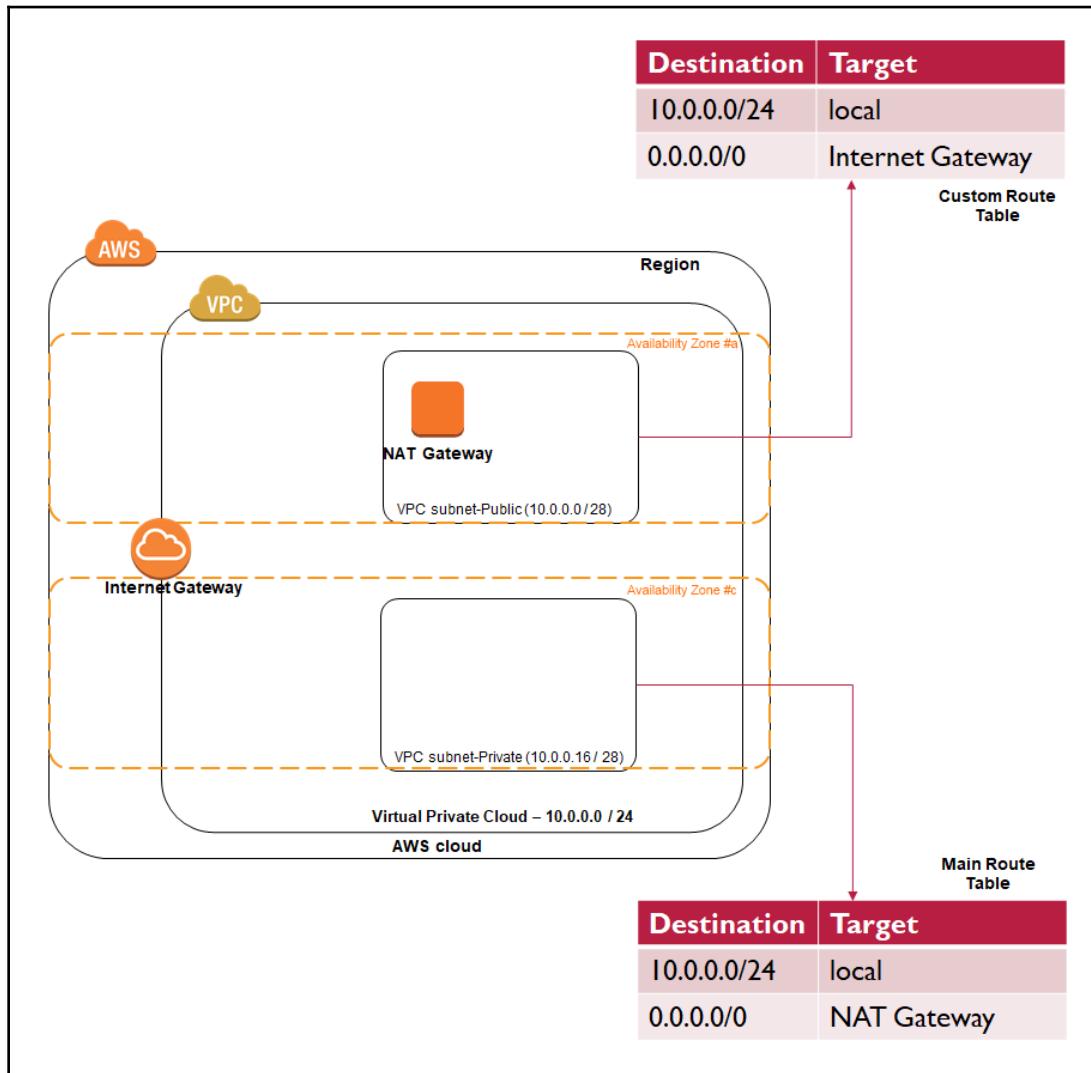
An NAT instance will forward outbound requests for private instances to the internet, and send the responses back to the correct instance.

You can easily launch an NAT instance by using an AMI with the correct network configuration:



You can find these on the community's AMI tab in the console when you're launching the instance. You will also need to add a route in the custom route table for the private subnet. For this route, you will specify `0.0.0.0/0` for the destination, and the NAT instance ID for the target. There are a few things to watch out for when you use an NAT instance. One is that it is a single point of failure, so you will need to use some automation to monitor the NAT instance, and recover it or failover to a second NAT. Another issue is that the network bandwidth will be limited by the instance type and size. A better option is to use AWS's NAT gateway service.

AWS will manage the availability of the NAT gateway so you don't have to, and it provides bursting network bandwidth up to 10 GB/s:



## Summary

In this chapter on networking in AWS, we first explained CIDR notation, which we will use to define IP address ranges for our VPC and subnets. Then we covered public, private, and Elastic IP addresses. We explained how these are assigned to Elastic Network Interface, and how they could be moved from one instance to another. Next, we explained dividing our VPC into subnets and using route tables to make subnets public, or to give instances in private subnets a path to the internet through a NAT instance or NAT gateway.

In the next chapter, we will create a VPC from scratch, and add additional security for our subnets with network access control lists. Then, we will describe ways to connect to our VPC.

# 5 Creating a VPC

In Chapter 4, *Networking on AWS*, we discussed networking on AWS, which laid the foundation for being able to create our own VPCs. We discussed IP addressing, subnets, and route tables. In this chapter, we will learn several methods to build, secure, and connect to a VPC. First, we're going to look at classic EC2s, which are instances that are launched outside of a VPC. Then, we'll talk about the VPC that AWS already creates for you, the default VPC. Next, we'll demonstrate creating a VPC, using the VPC Wizard, and then creating one from scratch. After that, we'll talk about several ways to connect to the instances in your VPC, and then we'll make your VPCs more secure by introducing network access control lists and Bastion instances.

Finally, we'll discuss making your architectures highly available by leveraging multiple availability zones, load balancing, and auto scaling.

The main topics that we will cover are as follows:

- VPC EC2s versus classic EC2s
- The default VPC
- Creating a VPC
- Connecting to a VPC
- Securing your VPC
- High availability

## Getting started with VPCs

In this section, we will begin with a little history lesson by talking about classic EC2s, and comparing them with EC2s that are launched in a VPC.

## Classic EC2s

EC2s were first introduced by AWS back in 2006. Back then, there was only one big public network in which to launch your instances. Every instance was automatically assigned a public and private IP address, controlled by AWS. If you stopped your instance for any reason, AWS took back your IPs, and when you started it up again, you got new ones.

Since every instance had a public IP address, they were all essentially public. So you had to rely on security groups to restrict access to your databases and other instances that you wanted to keep private, and the security groups only allowed you to specify inbound rules. All outbound traffic was always allowed. In 2009, AWS launched VPCs and encouraged customers to launch instances in these virtual private networks, instead of in the big public network. EC2s that are not launched in a VPC, are today called classic EC2s. These include RDS instances, which also had to be launched in the public network. In 2013 AWS declared that all accounts created after December 4th 2013, would have to launch their instances in VPCs. However, AWS accounts created before that date are grandfathered, and can still launch classic EC2 and RDS instances today.

## EC2s in a VPC

If you, or one of your customers, have one of these older AWS accounts, migrating the classic instances to a VPC is highly recommended. Some of the advantages are the ability to specify your own private IP address ranges, and keep the private IPs associated, when you stop and start your instances. You can choose not to get a public IP address, and also put your instance in a private subnet with no route to the internet. This is a powerful extra layer of security, in addition to security groups.

Security groups in a VPC allow you to create rules for outbound as well as inbound traffic, and by dividing your VPC into subnets, you can control traffic in to and out of your subnet with NACLs and custom route tables.

## The default VPC

VPCs cannot span regions, so AWS creates a default VPC in each region.

AWS Account supports EC2 instances in a VPC only. The default VPC is available in the Amazon VPC. If you delete the default VPC, then you cannot restore it. You need to contact AWS Support.

Click on Services | Go to Networking & Content Delivery section | Click on VPC | Click on VPC Dashboard:

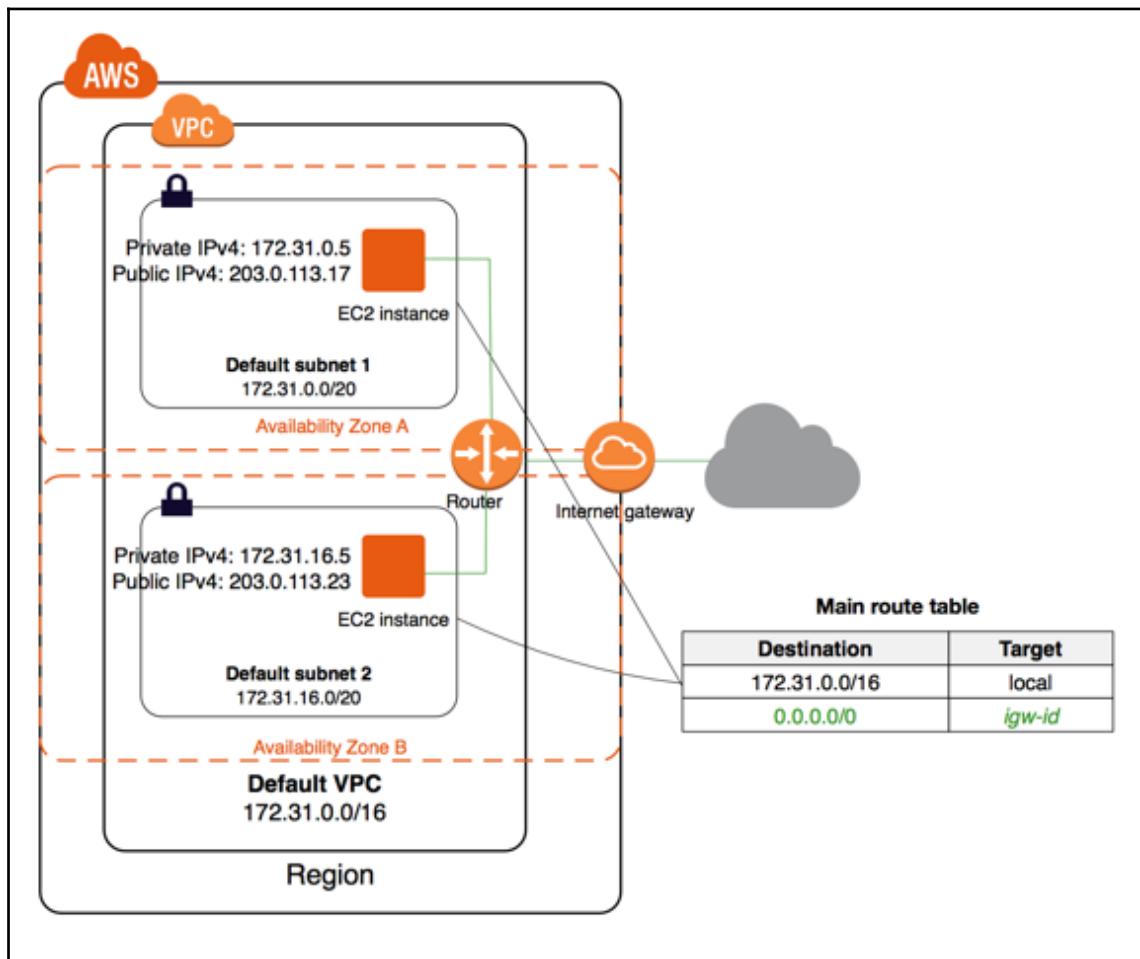


Figure reference :<http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/default-vpc.html>

The default VPC contains following:

- VPC with a size /16 IPv4 CIDR block (172.31.0.0/16); this means 65,536 private IPv4 addresses
- Default subnet /20 in each Availability Zone; it means 4,096 addresses per subnet
- One internet gateway
- A main route table for default VPC
- Default security group associated with your default VPC
- Default network access control list (ACL)

The screenshot shows the AWS VPC Dashboard. On the left sidebar, under 'Your VPCs', 'Your VPCs' is selected. In the main content area, a table lists one VPC entry: 'vpc-48023d20' with status 'available', IPv4 CIDR '172.31.0.0/16', DHCP options set 'dopt-6a495902', and route table 'rtb-b46ca8df'. Below the table, the 'Summary' tab is selected, displaying detailed information: VPC ID: vpc-48023d20, State: available, IPv4 CIDR: 172.31.0.0/16, IPv6 CIDR: (empty), DHCP options set: dopt-6a495902, Route table: rtb-b46ca8df. To the right, network ACL details are shown: Network ACL: acl-7f72e89f, Tenancy: Default, DNS resolution: yes, DNS hostnames: yes.

Verify the VPC ID, State, IPv4 CIDR, Route Table, ACL and so on.

A subnet can be defined as a section of a VPC's IP address range where you can place groups of isolated compute resources.

Click on the Subnet link in the left sidebar in the VPC Dashboard:

The screenshot shows the AWS VPC Dashboard. On the left sidebar, under the 'Subnets' section, there is a link labeled 'Route Tables'. The main content area displays a table of subnets. One subnet, 'subnet-1c049966', is selected. Below the table, there is a detailed view of this subnet's configuration.

Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4	IPv6 CIDR
subnet-1c049966	vpc-48023d20	available	172.31.16.0/20	4091	-	-
subnet-2edadfd46	vpc-48023d20	available	172.31.0.0/20	4091	-	-
subnet-32d5117e	vpc-48023d20	available	172.31.32.0/20	4091	-	-

**Subnet: subnet-1c049966**

Description	Flow Logs	Route Table	Network ACL	Tags
Subnet ID: subnet-1c049966	VPC: vpc-48023d20	State: available	IPv4 CIDR: 172.31.16.0/20	
Available IPv4 Addresses: 4091	Availability Zone: us-east-2b	IPv6 CIDR: -	Route Table: rtb-b46ca8df	
	Network ACL: acl-f7f2e89f		Default subnet: Yes	
	Auto-assign public IPv4 address: Yes		Auto-assign IPv6 address: No	

Route Tables helps to define subnets that need to be routed to the Internet gateway, the virtual private gateway, or other instances:

The screenshot shows the AWS Route Tables dashboard. On the left sidebar, under the 'Route Tables' section, there is a link labeled 'Internet Gateways'. The main content area displays a table of route tables. One route table, 'rtb-b46ca8df', is selected. Below the table, there is a detailed view of its routes.

Name	Route Table ID	Explicitly Associated	Main	VPC
rtb-b46ca8df	vpc-48023d20	0 Subnets	Yes	

**rtb-b46ca8df**

Summary	Routes	Subnet Associations	Route Propagation	Tags
	<b>Edit</b>			
	View: All rules			
Destination	Target	Status	Propagated	
172.31.0.0/16	local	Active	No	
0.0.0.0/0	<a href="#">igw-e25e878a</a>	Active	No	

Internet Gateway allows connection to the public Internet from an Amazon VPC:

The screenshot shows the AWS VPC Dashboard. On the left sidebar, 'Internet Gateways' is selected. In the main area, there is a table with one row. The columns are Name, ID, State, and VPC. The row contains: igw-e25e878a, igw-e25e878a, attached, and vpc-48023d20. Below the table, a detailed view for 'Internet gateway: igw-e25e878a' is shown with tabs for Description and Tags. Under the Description tab, it shows ID: igw-e25e878a, State: attached, Attached VPC ID: vpc-48023d20, and Attached Subnet ID: subnet-00000000. At the bottom of the screen, there are links for Feedback, English (US), and various AWS terms like Privacy Policy and Terms of Use.

NAT Gateway represents a highly available and managed **Network Address Translation (NAT)** service for resources in a private subnet to access the Internet. NAT Gateway is created in Public subnet.

An Elastic IP address is a public static IPv4 address so you can access the resource.

One Network ACL is created and associated with the default VPC:

The screenshot shows the AWS VPC Dashboard. On the left sidebar, under 'Virtual Private Cloud' > 'Your VPCs', there is a list of resources: Subnets, Route Tables, Internet Gateways, Egress Only Internet Gateways, DHCP Options Sets, Elastic IPs, Endpoints, Endpoint Services, and NAT Gateways. The main content area displays a single Network ACL named 'acl-f7f2e89f'. The table shows the following details:

Name	Network ACL ID	Associated With	Default	VPC
acl-f7f2e89f	3 Subnets	Yes		vpc-48023d20

Below the table, it says 'Allows inbound traffic. Because network ACLs are stateless, you must create inbound and outbound rules.' A 'Summary' tab is selected, followed by 'Inbound Rules' (which is active), 'Outbound Rules', 'Subnet Associations', and 'Tags'. Under 'Edit' and 'View: All rules', the table shows two rules:

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	ALL Traffic	ALL	ALL	0.0.0.0/0	ALLOW
*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY

At the bottom, there are links for 'Feedback', 'English (US)', '© 2008 - 2018, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.', 'Privacy Policy', and 'Terms of Use'.

The default VPC will have a minimum of two, but depending on the region, could have up to five public subnets, one per availability zone. We'll discuss availability zones in a later section on high availability, but for now what you need to know, is that AWS builds its data centers in small clusters, known as **availability zones (AZs)** for short. Each AZ is separated by tens of miles, so if a natural or other disaster should strike, at most only one AZ should be affected.

Every region has at least two availability zones, but the largest regions have up to five. Subnets cannot span availability zones, so your default VPC will have a subnet in each AZ. These will be public subnets, and have the size of /20. The VPC will have an internet gateway attached, and a route in the route table to the internet through the gateway. As you can see, this is a very basic VPC, and it doesn't provide private subnets. However, you can customize it and add those if you want.

In the next section, we will demonstrate how to create your own VPC.

## Creating a VPC demo

In the previous section, we learned about the difference between classic EC2s and EC2s in a VPC. In this section, we will demonstrate two methods to create your own VPC. The first method will be using the VPC Wizard, which can give us a VPC with private subnets, and even an NAT instance.

We can create an Amazon VPC in two ways:

1. A VPC the with Wizard:
  - Single Public Subnet
  - Public and Private Subnets
  - Public and Private Subnets and Hardware VPN Access
  - Private Subnet Only and Hardware VPN Access
2. A custom VPC without using the Wizard

## Create VPC using Wizard

Creating a VPC using Wizard is the easiest way to create VPC.

Click on **Services** | Go to **Networking & Content Delivery** section | Click on **VPC** | Click on **Start VPC Wizard** on VPC Dashboard.

To create a new VPC, click on Launch VPC Wizard:

The screenshot shows the AWS VPC Dashboard. On the left sidebar, there's a list of VPC-related services: Your VPCs, Subnets, Route Tables, Internet Gateways, Egress Only Internet Gateways, DHCP Options Sets, Elastic IPs, Endpoints, Endpoint Services, and NAT Gateways. At the top center, there are two buttons: "Launch VPC Wizard" (highlighted with a red box) and "Launch EC2 Instances". Below these buttons, a note says "Note: Your Instances will launch in the US East (Ohio) region." To the right, there's a "Service Health" section showing "Amazon EC2 - US East (Ohio)" with "Service is operating normally". The main area is titled "Resources by Region" and shows counts for various resources: VPCs (1), Subnets (3), Route Tables (1), Internet Gateways (1), and Egress-only Internet (0). There are also sections for Nat Gateways, VPC Peering Connections, Network ACLs, Security Groups, and VPN Connections.

There are four possible options. The first one is a very basic VPC, with a single public subnet:

The screenshot shows the "Step 1: Select a VPC Configuration" wizard. On the left, there are three options: "VPC with a Single Public Subnet" (selected and highlighted with a blue border), "VPC with Public and Private Subnets", and "VPC with Public and Private Subnets and Hardware VPN Access". The main content area describes the selected option: "Your instances run in a private, isolated section of the AWS cloud with direct access to the Internet. Network access control lists and security groups can be used to provide strict control over inbound and outbound network traffic to your instances." It also states that it "Creates: A /16 network with a /24 subnet. Public subnet instances use Elastic IPs or Public IPs to access the Internet." To the right, there's a diagram showing a cloud icon labeled "Internet, S3, DynamoDB, SNS, SQS, etc." connected to a square icon labeled "Public Subnet" with the text "Amazon Virtual Private Cloud" below it. A "Select" button is located at the bottom of this section. At the bottom of the page, there are links for "Feedback", "English (US)", "Privacy Policy", and "Terms of Use".

The second one, is a VPC with a public instance, containing an NAT instance or NAT gateway, and a private subnet:

The screenshot shows the 'Step 1: Select a VPC Configuration' page. On the left, there are four options: 'VPC with a Single Public Subnet', 'VPC with Public and Private Subnets' (which is selected and highlighted in blue), 'VPC with Public and Private Subnets and Hardware VPN Access', and 'VPC with a Private Subnet Only and Hardware VPN Access'. The main content area describes the selected configuration: 'In addition to containing a public subnet, this configuration adds a private subnet whose instances are not addressable from the Internet. Instances in the private subnet can establish outbound connections to the Internet via the public subnet using Network Address Translation (NAT)'. Below this is a 'Creates:' section: 'A /16 network with two /24 subnets. Public subnet instances use Elastic IPs to access the Internet. Private subnet instances access the Internet via Network Address Translation (NAT). (Hourly charges for NAT devices apply.)'. To the right is a diagram showing the 'Amazon Virtual Private Cloud' with a 'Public Subnet' and a 'Private Subnet'. A 'NAT' device is connected between them. An arrow points from the 'Public Subnet' to the 'Internet, S3, DynamoDB, SNS, SQS, etc.' cloud. A 'Select' button is at the bottom right of the main content area. At the bottom of the page are links for 'Feedback', 'English (US)', and 'Privacy Policy'.

The third one adds a virtual private gateway, which allows you to make a hardware VPN connection, back to your data center:

The screenshot shows the 'Step 1: Select a VPC Configuration' page. The same four options are present: 'VPC with a Single Public Subnet', 'VPC with Public and Private Subnets', 'VPC with Public and Private Subnets and Hardware VPN Access' (selected and highlighted in blue), and 'VPC with a Private Subnet Only and Hardware VPN Access'. The main content area describes the selected configuration: 'This configuration adds an IPsec Virtual Private Network (VPN) connection between your Amazon VPC and your data center - effectively extending your data center to the cloud while also providing direct access to the Internet for public subnet instances in your Amazon VPC'. Below this is a 'Creates:' section: 'A /16 network with two /24 subnets. One subnet is directly connected to the Internet while the other subnet is connected to your corporate network via IPsec VPN tunnel. (VPN charges apply.)'. To the right is a diagram showing the 'Amazon Virtual Private Cloud' with a 'Public Subnet' and a 'Private Subnet'. A 'VPN' connection is shown between the 'Private Subnet' and a 'Corporate Data Center' represented by a server icon. An arrow points from the 'Public Subnet' to the 'Internet, S3, DynamoDB, SNS, SQS, etc.' cloud. A 'Select' button is at the bottom right of the main content area. At the bottom of the page are links for 'Feedback', 'English (US)', and 'Privacy Policy'.

The fourth one takes away the public subnet, and just leaves the private subnet with a hardware VPN connection to your data center:

The screenshot shows the 'Step 1: Select a VPC Configuration' page. On the left, there are four options:

- VPC with a Single Public Subnet
- VPC with Public and Private Subnets
- VPC with Public and Private Subnets and Hardware VPN Access
- VPC with a Private Subnet Only and Hardware VPN Access** (this option is selected)

To the right of the options, there is a detailed description and a diagram:

**Description:**

Your instances run in a private, isolated section of the AWS cloud with a private subnet whose instances are not addressable from the Internet. You can connect this private subnet to your corporate data center via an IPsec Virtual Private Network (VPN) tunnel.

**Creates:**

A /16 network with a /24 subnet and provisions an IPsec VPN tunnel between your Amazon VPC and your corporate network. (VPN charges apply.)

**Select** button (disabled for the first three options)

**Diagram:**

```
graph TD; A[Amazon Virtual Private Cloud Subnet] --- B[VPN]; B --- C[Corporate Data Center]
```

The diagram illustrates a connection between an 'Amazon Virtual Private Cloud Subnet' (represented by a box containing three squares) and a 'Corporate Data Center' (represented by a server icon). An arrow labeled 'VPN' connects the two.

At the bottom of the page, there are links for Feedback, English (US), Copyright notice (© 2008 - 2018, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.), Privacy Policy, and Terms of Use.

Let's pick the second one for now, which will create a public subnet, a private subnet, and an NAT. For the VPC **IPv4 CIDR block**, let's just leave it as `10.0.0.0/16`, and let's name the VPC `TrainingDemo`.

Let's also leave the public subnet CIDR block as it is, `10.0.0.0/24`. For now, let's not choose an availability zone, and one will be randomly assigned. Let's call the public subnet Public subnet, for the private subnet we will leave it `10.0.1.0/24`, and will call the private subnet Private subnet:

The screenshot shows the 'Step 2: VPC with Public and Private Subnets' configuration page. The 'IPv4 CIDR block' is set to `10.0.0.0/16` (65531 IP addresses available). The 'IPv6 CIDR block' is set to 'No IPv6 CIDR Block'. The 'VPC name' is 'TrainingDemo'. The 'Public subnet's IPv4 CIDR' is `10.0.0.0/24` (251 IP addresses available), with an 'Availability Zone' of 'No Preference'. The 'Public subnet name' is 'Public subnet'. The 'Private subnet's IPv4 CIDR' is `10.0.1.0/24` (251 IP addresses available), with an 'Availability Zone' of 'No Preference'. The 'Private subnet name' is 'Private subnet'. A note says 'You can add more subnets after AWS creates the VPC.' Below this, there's a section for 'Specify the details of your NAT gateway (NAT gateway rates apply.)' with an 'Elastic IP Allocation ID' field and a link to 'Use a NAT instance instead'. There's also a 'Service endpoints' section with an 'Add Endpoint' button. At the bottom, there are options for 'Enable DNS hostnames' (Yes selected) and 'Hardware tenancy' (Default selected). The footer includes links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'.

In the next section, we can choose to launch an NAT gateway, or an NAT instance in our public subnet. For this demo, let's launch an NAT instance by clicking on Use an NAT instance instead:

Specify the details of your NAT instance (Instance rates apply). [Use a NAT gateway instead](#)

Instance type:\*  [Change](#)

Key pair name:  [Change](#)

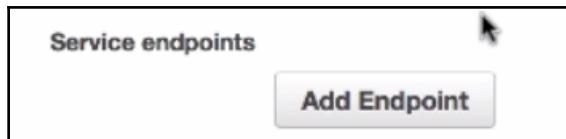
Service endpoints [Add Endpoint](#)

Enable DNS hostnames:\*  Yes  No

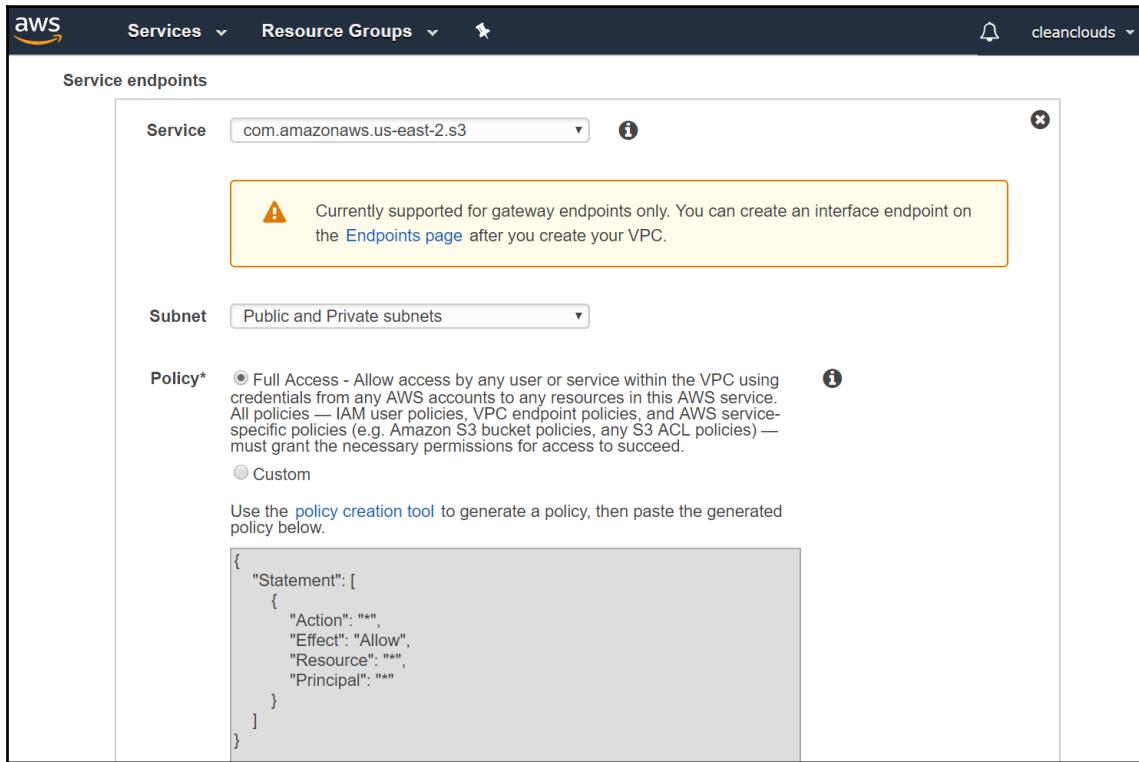
Hardware tenancy:\*  [Change](#)

[Cancel and Exit](#) [Back](#) [Create VPC](#)

Pick the instance type you would like to launch. Remember, the larger the instance, the higher the network bandwidth. Then, enter the private key pair associated with your account. The next section allows us to add a VPC endpoint for S3:



We could leave the default settings after clicking on **Add Endpoint**:



Let's also leave **Enable DNS hostname** on:



This means our instances with private IP addresses will have a DNS hostname as well. Leave **Hardware tenancy as Default**, and then click on **Create VPC**. After a few minutes, your VPC will have been created:



Click on **OK** to see your new VPC:

The screenshot shows the AWS VPC Dashboard. On the left sidebar, under 'Your VPCs', 'TrainingDemo' is listed. The main pane displays the 'Summary' tab for the 'TrainingDemo' VPC. Key details shown include:

VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP options set	Route table
vpc-0af777887a1118bef	available	10.0.0.0/16		dopt-6a495902	rtb-0ae25cce140bb2269

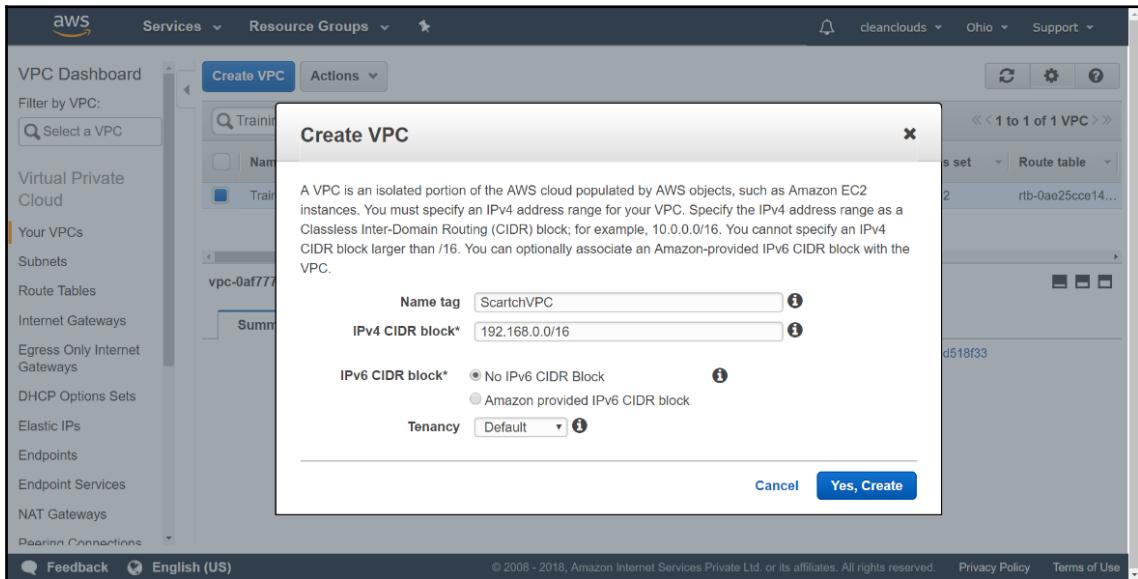
Network ACL: acl-06d5bcc42ed518f33

Tenancy: Default  
DNS resolution: yes  
DNS hostnames: yes

Other tabs visible in the summary pane include CIDR Blocks, Flow Logs, and Tags.

However, to get the most control, you can create a VPC from scratch, without the VPC Wizard.

Click on **Create VPC**. In the popup, give your VPC a name, and enter an IP address block using CIDR notation. /16 is a typical size, and this gives your VPC 65536 private IP addresses it can use:



You can leave the **Tenancy** option as **Default**, but just so you know what this is, the other option, **Dedicated** tenancy, makes it so that any instance you launch in this VPC will be launched in a host server on which you are the only customer. In other words, with the **Default** tenancy, your EC2 instances get created on host machines that are shared by other AWS customers. But with **Dedicated** tenancy, only your dedicated instances will run on that host machine; no other customers will share it. This is sometimes required for security compliance, or strict software licenses that are not cloud-friendly. Of course, additional hourly charges apply.

Click **Yes, Create**, and you should see your new VPC appear:

The screenshot shows the AWS VPC Dashboard. On the left sidebar, under 'Your VPCs', 'ScartchVPC' is selected. The main area displays a table of VPCs with columns: Name, VPC ID, State, IPv4 CIDR, IPv6 CIDR, DHCP options set, and Route table. The 'ScartchVPC' row is highlighted. Below the table, the VPC details for 'ScartchVPC' are shown, including its VPC ID, state, CIDRs, DHCP options set, and route table. The 'Summary' tab is selected.

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP options set	Route table
vpc-48023d20	available	172.31.0.0/16			dopt-6a495902	rtb-b46ca8df
ScartchVPC	available	192.168.0.0/16			dopt-6a495902	rtb-065ac615f7...
TrainingDemo	available	10.0.0.0/16			dopt-6a495902	rtb-0ae25cce14...

**ScartchVPC**

**Summary**    CIDR Blocks    Flow Logs    Tags

VPC ID: vpc-00999c48787d5e681 | ScartchVPC  
State: available  
IPv4 CIDR: 192.168.0.0/16  
IPv6 CIDR:  
DHCP options set: dopt-6a495902  
Route table: rtb-065ac615f7def78fc

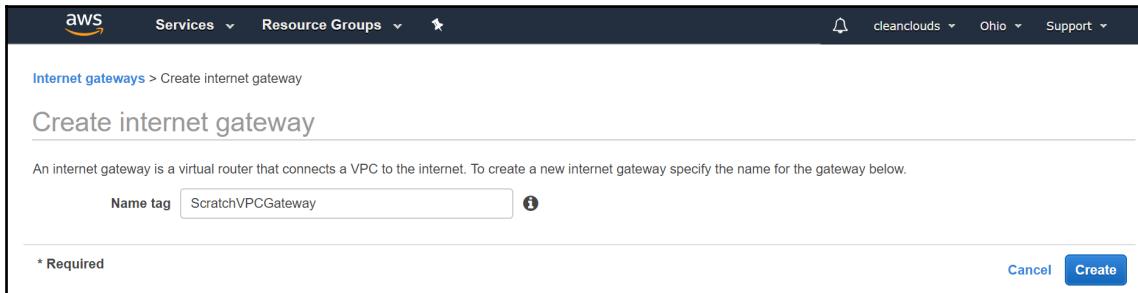
Network ACL: acl-080850d72e13cd7dd  
Tenancy: Default  
DNS resolution: yes  
DNS hostnames: no

---

Feedback    English (US)    © 2008 - 2018, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.    Privacy Policy    Terms of Use

If you want your VPC to have internet access, you need to attach an internet gateway. In the navigation, click **Internet Gateways**, and then **Create Internet Gateway**.

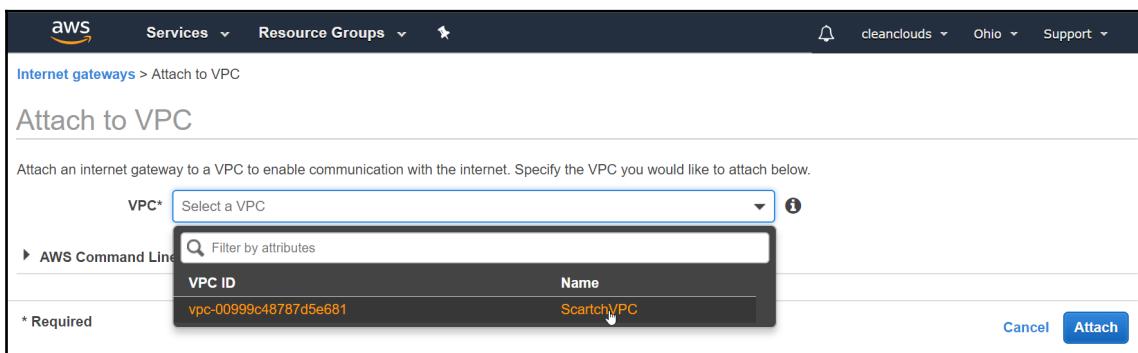
Give it a name, and then click **Yes, Create**:



Click on Close.

The next step is to select the internet gateway, and then click **Actions | Attach to VPC**.

Click the VPC you just created, and then click **Attach**:



Verify that the newly created Internet Gateway is attached to the VPC we created.

Now, let's create some subnets. In the navigation, click **Subnets**, and then **Create Subnet**.

Give it a name. We're going to make this a public subnet, so let's call it `Public1`.

Click the VPC we just created, and select any **Availability Zone**. Enter a **CIDR block**, but remember this must be a subset of the IP address range you gave your VPC. So, let's make this a `/24`, which will give it 251 available addresses:

Subnets > Create subnet

Create subnet

Specify your subnet's IP address block in CIDR format; for example, 10.0.0.0/24. IPv4 block sizes must be between a /16 netmask and /28 netmask, and can be the same size as your VPC. An IPv6 CIDR block must be a /64 CIDR block.

VPC CIDRs	CIDR	Status	Status Reason
	192.168.0.0/16	associated	

Name tag  ⓘ

VPC\*  ⓘ

Availability Zone  ⓘ

IPv4 CIDR block\*  ⓘ

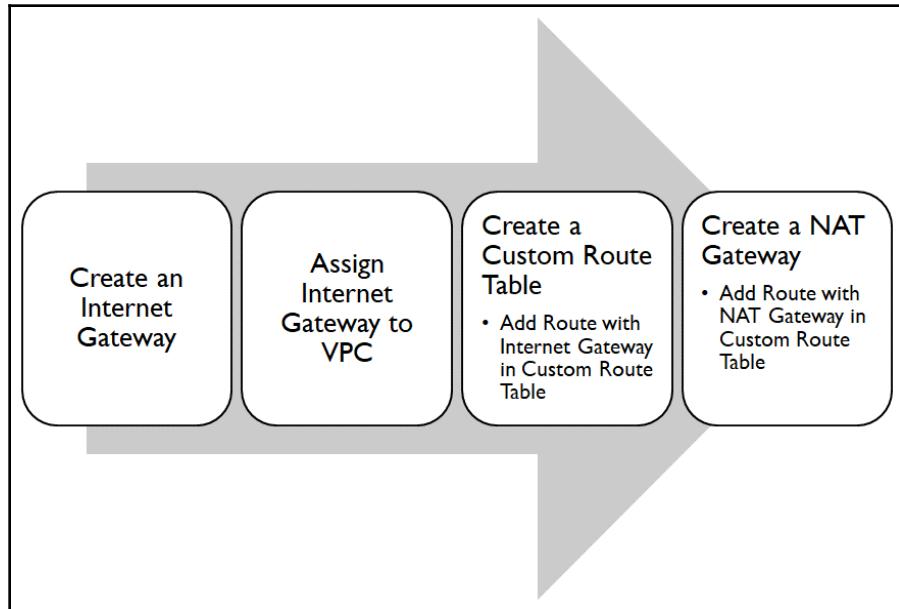
\* Required

Cancel **Create**

Feedback English (US) © 2008 - 2018, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Remember that five addresses are taken by AWS.

We will configure the following things in the next section to achieve internet access for our instances in the public subnet:

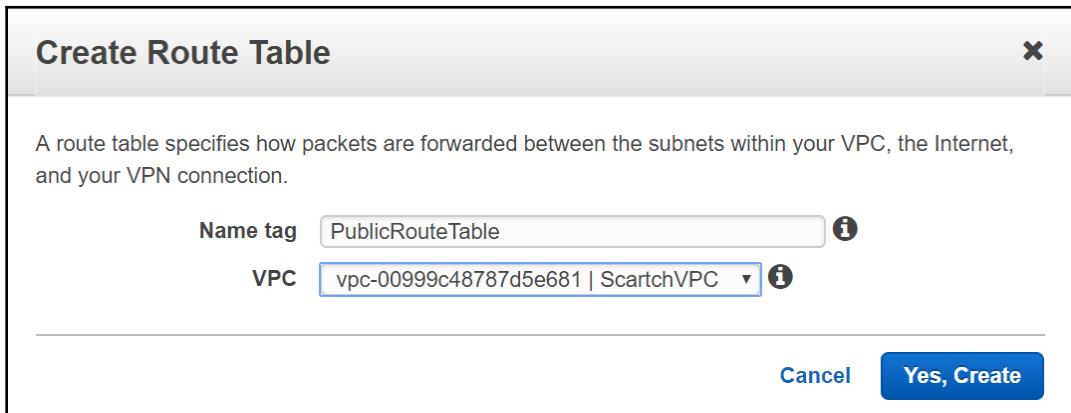


How can we know whether a subnet is Public or Private?  
If an Internet Gateway is assigned to the subnet, then it is Public.

And since we want this to be a public subnet, we have to give our subnet a route to the internet, through the internet gateway.

In the navigation, click **Route Tables** and then **Create Route Table**.

Give it a name, such as `PublicRouteTable`, and select your VPC:



Click **Yes, Create**. Select a new route table, and then on the **Routes** tab, click **Edit** and **Add another route**:

The screenshot shows the AWS VPC Dashboard. On the left sidebar, under 'Route Tables', the 'PublicRouteTable' is selected. In the main area, the 'Routes' tab is active, showing a table with one rule: 'Destination: 192.168.0.0/16, Target: local, Status: Active, Propagated: No'. Above the table, there are buttons for 'Edit' and 'View: All rules'. Other tabs include 'Summary', 'Subnet Associations', 'Route Propagation', and 'Tags'.

For the **Destination**, type `0.0.0.0/0`, which is the way we specify the entire internet. Click in the **Target** box, and then click the internet gateway we just created and click **Save**:

The screenshot shows the AWS Route Table configuration screen for a PublicRouteTable named "rtb-054a47c242ed87df5". The "Routes" tab is selected. A table lists a single route rule:

Destination	Target	Status	Propagated	Remove
192.168.0.0/16	local	Active	No	
0.0.0.0/0	igw-05eb8e92087173726	No		X

Buttons at the bottom include "Add another route" and "Save".

Now that we've created the route table, let's associate it with our subnet. Click the **Subnet Associations** tab, and then click **Edit**:

The screenshot shows the AWS Route Table Subnet Associations screen for the same PublicRouteTable. The "Subnet Associations" tab is selected. An "Edit" button is visible. A message states: "You do not have any subnet associations. The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:"

Select our public subnet, and click **Save**:

The screenshot shows the AWS Route Table configuration interface for a specific route table named 'rtb-054a47c242ed87df5'. The 'Subnet Associations' tab is active. A public subnet, 'subnet-00c64b67d84399200 | Public1', is listed with its IPv4 CIDR range '192.168.0.0/24'. The 'Associate' checkbox is checked. The 'Save' button is highlighted in blue.

Here, we are talking about accessing the internet from an instance launched in a private subnet.

We can achieve this using NAT Devices. There are two ways to achieve this in AWS. One is by creating an NAT instance and another is by creating an NAT Gateway.



Add NAT Device in the Public Subnet. Why? Only then it will be able to access the internet, right?

We will use an NAT Gateway here to demonstrate the configuration of internet access for instances available in the Private subnet.

Now, let's put an NAT gateway in our new subnet. In the navigation menu, click on **NAT Gateways**, and then **Create an NAT Gateway**.

Click in the **Subnet** box, and then select our public subnet. Click on **Create New EIP**, so our gateway will have an elastic IP, and can access the internet:

NAT Gateways > Create NAT Gateway

## Create NAT Gateway

Create a NAT gateway and assign it an Elastic IP address. [Learn more.](#)

Subnet\*  C ⓘ

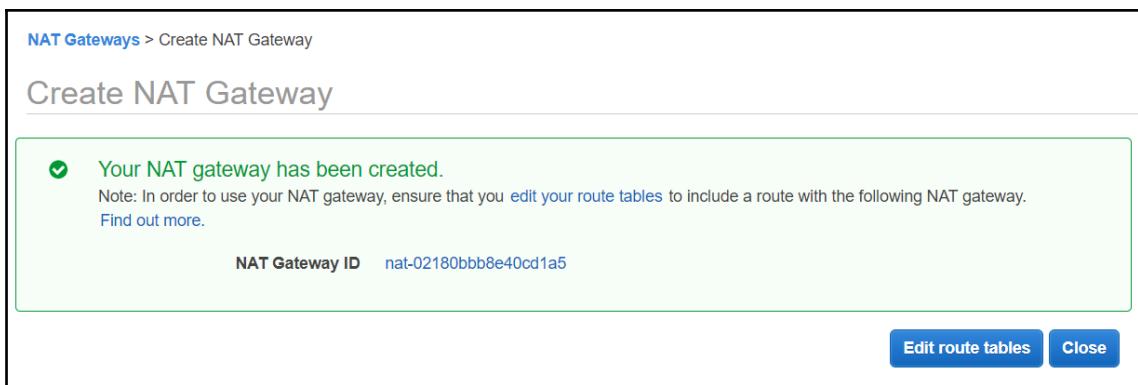
Elastic IP Allocation ID\*  C ⓘ [Create New EIP](#)  ⓘ

New EIP (52.71.84.72) creation successful.

\* Required Cancel Create a NAT Gateway

Finally, click **Create an NAT Gateway**.

You will get the following pop-up window:



Now, let's create a private subnet, and then give it a route through the NAT gateway to access the internet and public AWS services. Back in the navigation, click **Subnets** and then **Create subnet**. Give it a name, let's call it `Private1`, click the **VPC** we just created, and select any **Availability Zone**. Enter an **IPv4 CIDR block** that won't overlap with our public subnet, and click **Yes, Create**:

The screenshot shows the 'Create subnet' wizard. The 'Name tag' field contains 'Private1'. The 'VPC\*' dropdown is set to 'vpc-080d822b0553327a2'. The 'VPC CIDRs' table shows one entry: '192.168.0.0/16' with status 'associated'. The 'Availability Zone' dropdown is set to 'us-east-1a'. The 'IPv4 CIDR block\*' field contains '192.168.1.0/24'. A note at the bottom left says '\* Required'. At the bottom right are 'Cancel' and 'Create' buttons.

Now, let's create a route table for our private subnet. In the navigation menu, click **Route Tables** and then **Create Route Table**. Give it a name, this time `PrivateRouteTable`, and select your **VPC**. Click **Yes, Create**. Select the new route table, then the **Routes** tab, and then click **Edit | Add another route**. For the **Destination**, type `0.0.0.0/0`, and in the **Target** box, click the NAT gateway we just created and click **Save**:

The screenshot shows the 'PrivateRouteTable' route table. The 'Routes' tab is selected. There is one existing rule: '192.168.0.0/16' with 'Target' 'local' and 'Status' 'Active'. A new rule is being added: '0.0.0.0/0' with 'Target' 'nat-02180bbb8e40cd1a5' and 'Status' 'No'. The 'Add another route' button is visible at the bottom left. Navigation tabs include Summary, Routes, Subnet Associations, Route Propagation, and Tags.

Finally, let's associate that route table with our private subnet. So, click on the **Subnet Associations** tab and then **Edit**. Select our private subnet, and then click **Save**:

The screenshot shows the AWS Route Table Subnet Associations interface. The top navigation bar includes tabs for Summary, Routes, Subnet Associations (which is highlighted), Route Propagation, and Tags. Below the tabs are two buttons: Cancel and Save (the latter is highlighted). The main content area has a table with columns: Associate, Subnet, IPv4 CIDR, IPv6 CIDR, and Current Route Table. There are two rows: one for a public subnet (unchecked) and one for a private subnet (checked). The private subnet row shows the subnet ID, name, IPv4 CIDR (192.168.1.0/24), IPv6 CIDR (-), and Current Route Table (Main).

Associate	Subnet	IPv4 CIDR	IPv6 CIDR	Current Route Table
<input type="checkbox"/>	subnet-00c64b67d84399200   Public1	192.168.0.0/24	-	Main
<input checked="" type="checkbox"/>	subnet-0b89925cd6dbace61   Private1	192.168.1.0/24	-	Main

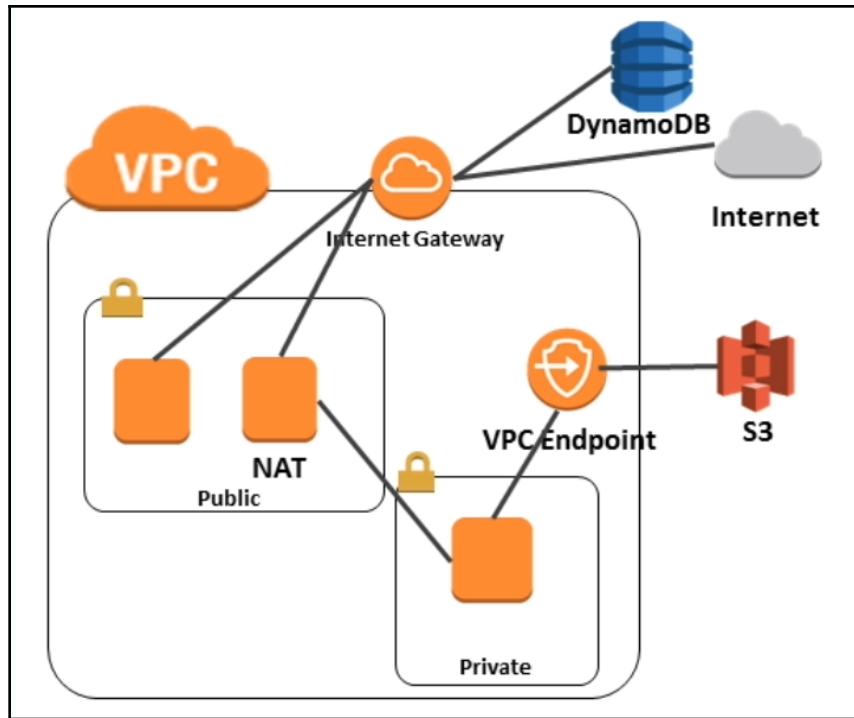
You can keep creating subnets if you like. In a later section, we will show you how to take advantage of multiple availability zones to make your application environment highly available. In the next section, we will discuss several ways to connect to your VPCs, including VPNs and direct connect.

## Connecting to a VPC

In the previous section, we demonstrated methods for creating a VPC. In this section, we're going to discuss what you need to do to securely connect to your VPC. We're going to look at the two types of gateways, the internet gateway and a virtual private gateway. Then, we're going to discuss making a hardware VPN connection from your data center to a virtual private gateway, or using a traditional software VPN. Next, we'll briefly discuss connecting over a private dedicated line using direct connect. Finally, we'll talk about connecting two VPCs together using peering. In our VPC example shown earlier, we created an internet gateway and attached it to our VPC.

## Internet gateway

An internet gateway is a service managed by AWS that connects the internet to your VPC. It is highly available, and scales to meet your traffic requirements:

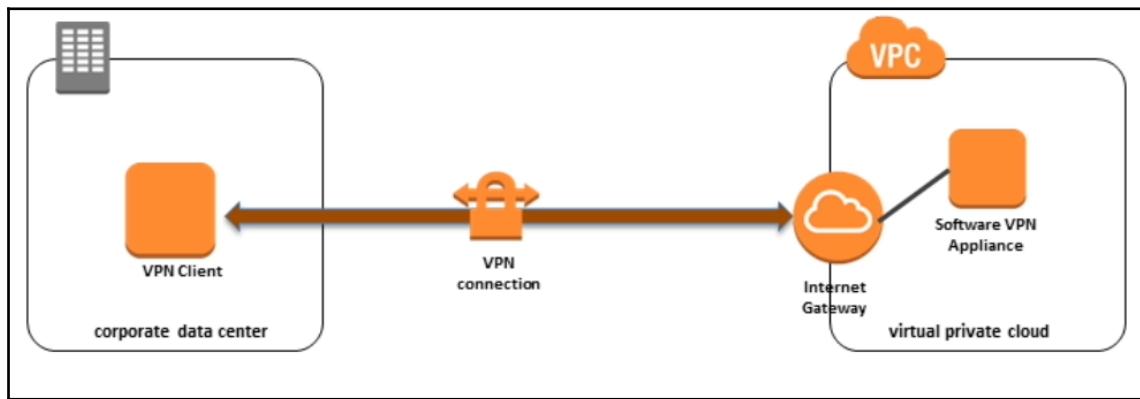


For an instance to be directly reachable from the internet, it needs to have a public or elastic IP, and it needs to be launched in a subnet that has a route to the internet, through the internet gateway. You may be surprised to learn that many AWS managed services, such as DynamoDB, Kinesis, SQS, and SNS, are not reachable from private subnets in your VPC. Only instances in a public subnet, with public or elastic IPs, can reach these services, because communication with them is over the public network, which means it has to pass the internet gateway. Instances in a private subnet can only reach these services by routing the requests through an NAT instance or NAT gateway.

Since NAT instances can sometimes become a bottleneck, AWS has started introducing VPC endpoints for some of their public services. You add a VPC endpoint inside of your VPC, and then you can communicate with the service as if it is inside your VPC. So, private instances can communicate directly with the service without an NAT. The first service to support VPC endpoints is S3, an object storage service that we will cover in a later section.

## Software VPN

We can communicate with our instances from the outside by using their public IP addresses. However, if you wanted a secure way to extend your private network, you could use a software-based virtual private network (VPN) appliance. This gives you a secure encrypted connection to your VPC:

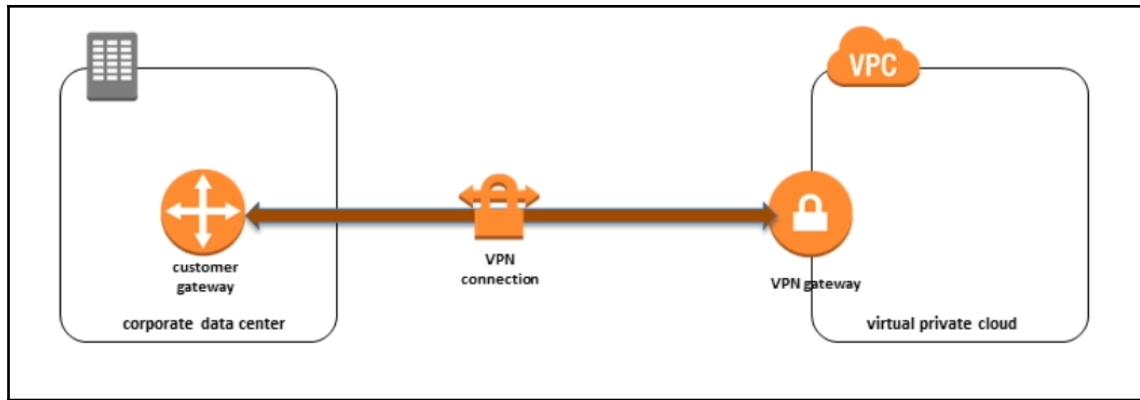


You install the appliance, by launching an AMI from the AWS marketplace, which is provided by third-party vendor such as OpenVPN.

## Virtual gateway

AWS directly supports another type of VPN connection, known as a hardware VPN.

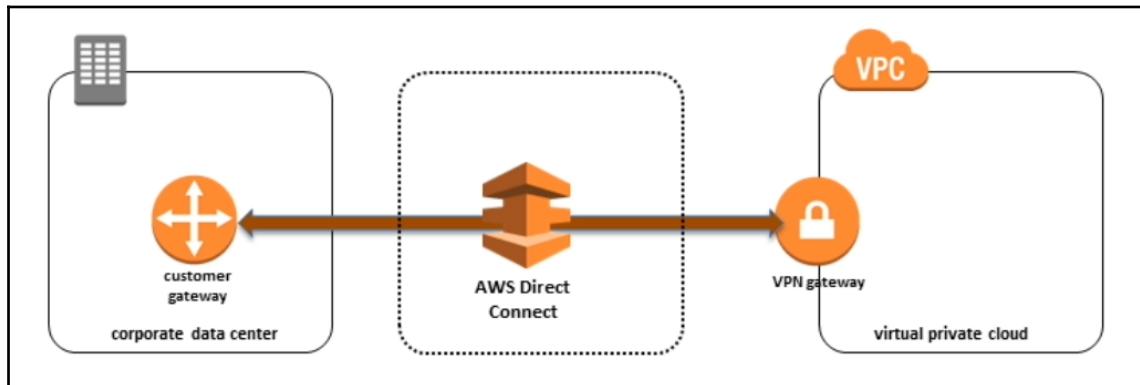
For this, you attach a virtual private gateway to your VPC and install a supported customer gateway in your data center:



Like the software VPN, traffic is encrypted with IPsec tunneling. Both hardware and software VPNs use the public internet for the transport layer.

## Direct connect

A third option, direct connect, does not use the public internet. Instead, you use a dedicated fiber optic connection between your data center and a direct connect facility. You work with a service partner, such as Equinix or AT&T, to make the fiber connection and then make it cross connect to the AWS network in the direct connect location:

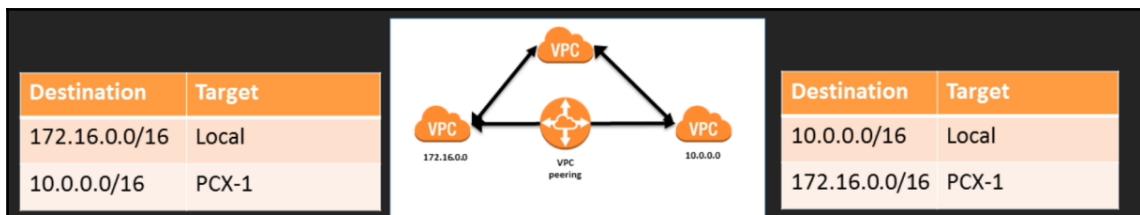


A direct connect location is associated with a particular AWS region.

The benefits of direct connect include speeds available up to 10 GBps, improved performance, and enhanced security and compliance.

## VPC peering

If you wanted to connect two VPCs, AWS offers VPC peering, which allows instances in two VPCs to communicate, as if they're in the same network:



Both VPCs must be in the same AWS region, but they do not have to share the same AWS account. For security purposes, peer VPCs cannot share other connections, such as gateways or other peer connections. If two VPCs are peered with the same VPC, they still cannot communicate with one another, unless you peer them directly. In the next section, we will secure our VPC using network access control lists, and Bastion instances.

## Securing your VPC

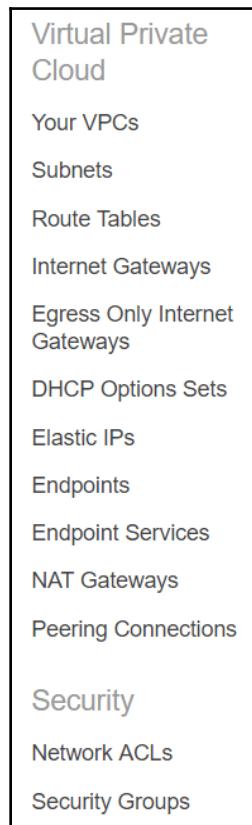
In the previous section, we looked at ways to connect to your VPC, which included gateways, VPN connections, direct connect, and peering. In this section, we're going to add some additional security to your VPC, by adding network access control lists to our subnets. We're also going to talk more about private subnets, and how administrators can still connect to private instances, by using Bastion instances. In an earlier chapter, we talked about security groups, and how these are like firewalls that protect our instances. An additional type of firewall we can use is network access control lists, or just network ACLs.

## NACLs

While security groups surround our instances, network ACLs allow and deny traffic at the subnet boundary, both inbound and outbound.

Since we already have security groups, it may seem that network ACLs are a bit redundant. However, best practice is to back up critical firewall rules, by including them in both security groups and network ACLs. By default, every subnet already has a network ACL, but they're configured with just one rule, allow all traffic. So technically, you could consider adding any other rules to them, to deny traffic as optional. Or you can just rely on your security group rules. While this might be okay for low-security environments, consider what would happen if someone misconfigures a security group. It opens up access to your database from the internet. If you have a deny rule in the network ACL, then no harm would be done, as this would block that traffic.

So let's go to the management console and take a look at some network ACLs. Click on **VPC**, and then **Network ACLs**:



In the list, you will see the ones that are already created for our subnets:

The screenshot shows a table with columns: Name, Network ACL ID, Associated With, Default, and VPC. The rows represent three Network ACLs:

Name	Network ACL ID	Associated With	Default	VPC
aci-0951a487729e4c...	aci-0951a487729e4c...	2 Subnets	Yes	vpc-0fb97ec0046229caa   TrainingDemo
aci-39c0ba41	aci-39c0ba41	6 Subnets	Yes	vpc-534b882b
aci-010144a495aa4c...	aci-010144a495aa4c...	2 Subnets	Yes	vpc-080d822b0553327a2   ScratchVPC

Click on the one for our VPC. Take a look here at the **Inbound Rules** and **Outbound Rules**. You will notice there is an **ALLOW** for all traffic, above a **DENY** for all traffic:

The screenshot shows the Inbound Rules tab for Network ACL acl-0951a487729e4c046. It displays two rules:

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	ALL Traffic	ALL	ALL	0.0.0.0/0	ALLOW
*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY

Network ACL rules are processed in order, according to the rule number. Since the **ALLOW** comes before the **DENY**, the **ALLOW** has precedence. Click on the **Subnet Associations** tab, and you will see that the same network ACL is associated with both our private and public subnet:

The screenshot shows the AWS Network ACL management interface. The top navigation bar includes tabs for Summary, Inbound Rules, Outbound Rules, Subnet Associations (which is highlighted in blue), and Tags. Below the tabs, there's an 'Edit' button. The main content area is titled 'acl-0951a487729e4c046'. It shows two subnets associated with this ACL: 'subnet-0e1d363df0c0ac07a | Public1' (IPv4 CIDR: 10.0.0.0/24) and 'subnet-04388fa895808cbe1 | Private1' (IPv4 CIDR: 10.0.1.0/24). There is also an IPv6 CIDR column which is empty for both subnets.

So let's go ahead and make our private subnet more secure. Click on **Create Network ACL**. Let's give it a name, and select our VPC:

The screenshot shows the 'Create Network ACL' dialog box. It contains a descriptive text about what a Network ACL is, followed by input fields for 'Name tag' (set to 'PrivateNACL') and 'VPC' (set to 'vpc-080d822b0553327a2 | ScratchVPC'). At the bottom are 'Cancel' and 'Yes, Create' buttons. The 'Yes, Create' button is highlighted in blue.

Let's assume that this is in a subnet, we will put in some Oracle databases. So let's add a rule, to allow that traffic from our public subnet, and deny everything else. Click the **Edit** menu in the **Inbound Rules** tab, and then **Add another rule**:

The screenshot shows the 'Inbound Rules' tab selected in the 'acl-098a163c73454d822 | PrivateNACL' interface. A single rule is listed:

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	Oracle (1521)	TCP (6)	1521	192.168.1.0/24	ALLOW

Buttons for 'Cancel' and 'Save' are visible at the top left, and a 'View: All rules' dropdown is at the top center.

Enter a rule number, such as 100, and then select **Oracle (1521)**, for the protocol. For the source, enter the IP address for our public subnet. If you're not sure of the IP, go back to the subnets list, and get the IP address for the public subnet. Then, click **Save**.

Note that this will allow inbound traffic on port 1521, from our public subnet only:

The screenshot shows the 'Inbound Rules' tab selected in the 'acl-098a163c73454d822 | PrivateNACL' interface. Two rules are listed:

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	Oracle (1521)	TCP (6)	1521	192.168.1.0/24	ALLOW
*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY

A blue 'Edit' button is visible at the top left. A 'View: All rules' dropdown is at the top center.

All other traffic will be denied. Unlike security groups, network ACLs are stateless, so we have to configure rules for outbound as well.

For **Outbound Rules**, we need to configure the high ephemeral port range. Click on **Edit** | **Add another rule**, the port range we should use is 1024–65535. For the **Destination**, put the IP address of the public subnet, and click **Save**:

The screenshot shows the 'Outbound Rules' tab of a Network ACL named 'acl-098a163c73454d822'. It displays a single rule (Rule # 100) allowing TCP port 1024-65535 to the destination 192.168.0.0/24. The 'Allow / Deny' field is set to 'ALLOW'. Buttons for 'Cancel' and 'Save' are visible at the top.

Now let's go to the **Subnet Associations** tab, click on **Edit** and add our private subnet:

The screenshot shows the 'Subnet Associations' tab of the same Network ACL. It lists two subnets: 'subnet-00c64b67d84399200 | Public1' and 'subnet-0b89925cd6dbace61 | Private1'. The 'Private1' subnet is selected and associated with the current Network ACL. Buttons for 'Cancel' and 'Save' are visible at the top.

Now this NACL has replaced a less secure one, that was being shared by the public subnet.

## Bastion instances

To follow best security practices, we need to secure our vulnerable instances, such as backend app servers, and databases in private subnets. In a previous section, we mentioned that NAT instances, or an NAT gateway, can be used to give our private instances access to the internet, to download security patches, or to access public AWS services. However, our server administrators will need to be able to connect with our private instances, in order to perform upgrades or security updates. Since there is no direct route to the instances from the internet, we need to launch a Bastion instance, in a public subnet. The administrators can connect to this instance, and from there are able to log in to the private instances. The process is different for connecting to private Linux and Windows instances.

For Linux instances, we launch a Linux instance as a Bastion, and connect to it using SSH, with key forwarding. Once logged in, we can SSH to the private instances. For Windows instances, we launch a Windows instance in a public subnet, and install RD Gateway. With RD Gateway installed, we use our RDP client, and authenticate on the RD Gateway, which then makes an RDP connection to our private instance.

## Highly available architectures

In the previous section, we created a secure environment for our applications, by adding additional security to our VPCs. In this section, we will discuss VPC architectures, that can make the environments for our applications highly available. To make our environments highly available, we are going to deploy instances into multiple availability zones, and load balance the requests with elastic load balancing. Finally, we'll add additional resilience to our environment, by automatically scaling horizontally.

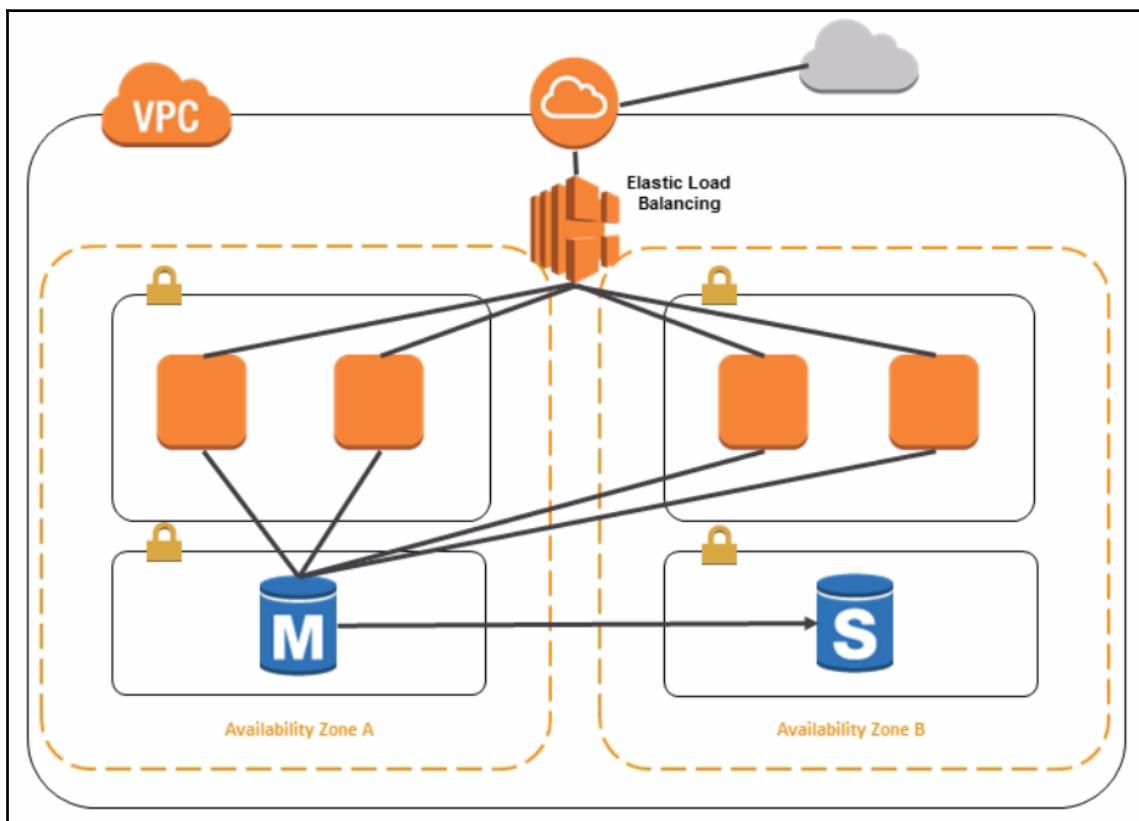
## Availability zones

AWS is built on a global infrastructure, located in more than a dozen regions, including several countries in Europe and Asia, as well as Australia, the United States, Canada, and Brazil.

Each region is divided up into separate physical areas, in what AWS calls availability zones, or just AZs. The data centers are built in small, fault isolated clusters, in these availability zones, which are typically tens of miles apart. This makes a regionwide outage less likely, in the case of a natural disaster. Availability zones within a region, are connected with low-latency private connections. Each region has a minimum of two AZs, but the largest currently has five. Although VPCs must be contained in a single region, they can span multiple availability zones. This allows us to deploy EC2 instances in multiple availability zones, and load balance the traffic.

Thus, if a single availability zone has an outage, the instances in the healthy availability zones, will continue to respond to the requests. Of course, we can only be truly highly available, if we eliminate single points of failure, such as a relational database. Most relational databases allow you to configure a hot standby, or slave, that can take over if the master is unreachable. In AWS, we can make our database highly available, by deploying the slave in a different availability zone from the master.

Here is a highly available VPC architecture, where we have created subnets for our instances, in two different AZs:



Note that a single subnet cannot span availability zones. We use an elastic load balancer, a load balancing service, managed by AWS, to distribute the request to the instances in different AZs.

We have also separated the master and slave databases into AZs.

## Elastic load balancer

An elastic load balancer, or ELB for short, has a number of noteworthy features, including the ability to perform health checks on instances. If an instance fails a health check, it automatically stops routing any traffic to it. Originally, AWS offered a single type of ELB, which is now called a classic load balancer. You can configure it to listen for TCP, SSL, HTTP and HTTPS.

For HTTP and HTTPS, it uses a least outstanding requests algorithm for routing. For other protocols, it uses simple round-robin.

The newer type of ELB is called an application load balancer. This allows you to route requests, based upon the content of the request. For example, a request for different microservices, could go to different backend instances.

In addition to HTTP and HTTPS, it has support for the new faster version of HTTP/2. For both types of load balancers, you can terminate HTTPS at the ELB, by uploading a certificate. Either type of load balancer can be used to make your environment highly available. However, not every software application is ready to be put in a load balanced environment.

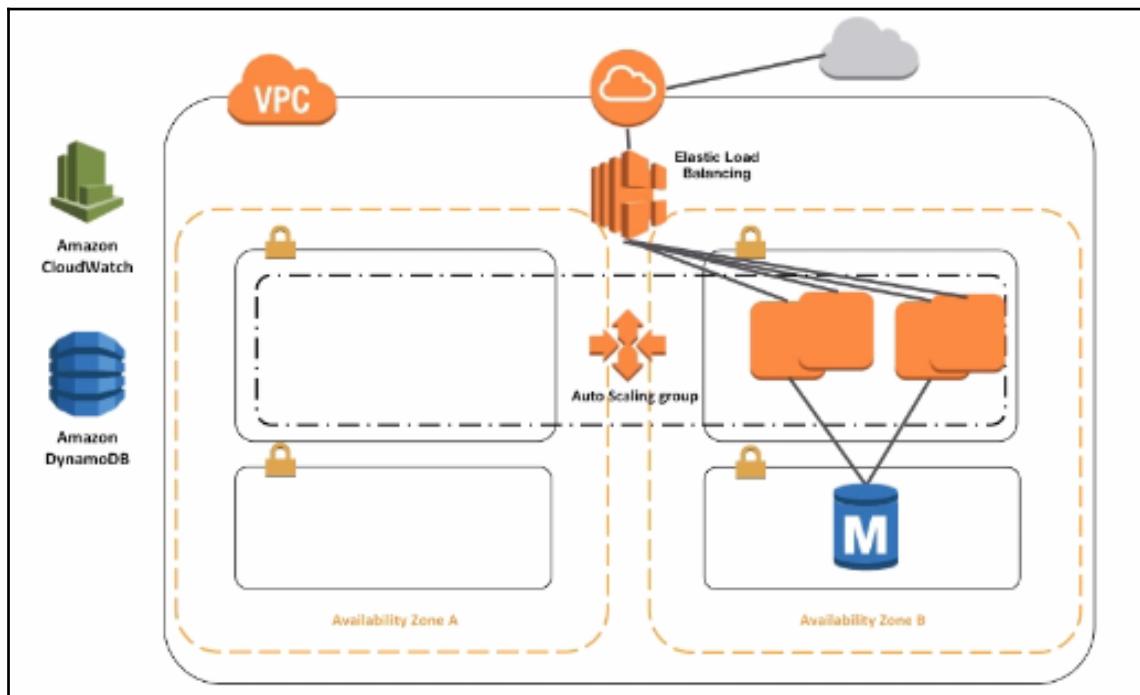
## Load balancing stateful applications

Many applications store some type of session or state data, to know for example, that a user has logged in or added an item to their shopping cart. If they store this data at the local filesystem level, then this data will be kept only on one instance, and the other instances won't have a copy. So, if the ELB routes a subsequent request to a different instance, the application will lose the session state and the user may have to log back in again, or have to add their item back into the shopping cart.

One way to cope with this, is through replication of the state data across all the instances. But this can be prone to failure, of the sinking application. Another option, which is supported by ELBs, is to enable sticky sessions. In this case, the ELB uses a cookie, to keep track of the instance ID that the users request was directed to, and to continue to send other requests to the same instance. A much better option, is to store the state data in a separate storage system, that is accessible by all the instances. Some good choices for this are DynamoDB, a NoSQL database, or ElastiCache, which uses Memcached or Redis.

If we go back to our architecture diagram for a second, you can see that if AWS experiences an AZ outage, the load balancer will send all the requests to only one AZ, and we may not have enough capacity for that AZ to handle the increased request volume. The solution for this, is to automatically launch more instances in the working AZ.

For this, we add two more services for our VPC, auto scaling and CloudWatch:



## Auto scaling

Auto scaling allows us to horizontally scale, by launching and terminating instances in response to the request to load. So if our application gets a lot of traffic during the day, perhaps we will have ten instances getting requests from the ELB. But overnight, we don't have a lot of users, so maybe we'll go down to a minimum of two. We can specify the minimum and maximum number of instances for our auto scaling group. To maintain our high availability, we should probably never specify a minimum of less than two. When you create an auto scaling group, you will also have to define the launch configuration. This is simply all of the attributes of the EC2s that the auto scaling group will launch, including the AMI, instance size and type, security group, and so on. The way that auto scaling knows when it's time to launch or terminate instances, is through the CloudWatch monitoring service CloudWatch. You configure alarms based upon default metrics, such as average CPU utilization, for the instances in the end or average response latency measured at the ELB. You could also use your own custom metrics, such as memory or thread utilization by pushing metrics to CloudWatch from your EC2 instances, using the CloudWatch API, or by streaming log files to CloudWatch, using the CloudWatch logs agent. Auto scaling launches and terminates instances, when notified by CloudWatch that an alarm has been triggered. An example of an alarm, would be average CPU utilization above 70% for three consecutive measurements. You define a scaling policy, that determines what action to take for a particular alarm. So for this high CPU utilization alarm, we can have auto scaling launch two instances, or maybe add 50% more instances. We could scale in, by configuring an alarm for low CPU utilization, such as below 20%, and to find a scaling policy to terminate two instances. Let's go through a real world example, to help you understand the concepts. Here, you can see we've created some new subnets in the VPC. You can see now we have subnets in both a and b availability zones:

The screenshot shows the AWS VPC Dashboard. On the left sidebar, 'Subnets' is selected. The main area displays a table of subnets:

Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4	IPv6 C
Public2	subnet-026f48454da16e44a	available	vpc-00999c48787d5e681	192.168.2.0/24	251	-
Public1	subnet-02258387369d1217d	available	vpc-00999c48787d5e681	192.168.0.0/24	250	-
Private1	subnet-0c85c728450812b17	available	vpc-00999c48787d5e681	192.168.1.0/24	251	-
<b>Private2</b>	<b>subnet-032788b49432e20f3</b>	<b>available</b>	<b>vpc-00999c48787d5e681</b>	<b>192.168.3.0/24</b>	<b>251</b>	-

Details for Subnet Private2:

Description	Subnet ID: subnet-032788b49432e20f3	State: available
VPC	vpc-00999c48787d5e681   SearchVPC	IPv4 CIDR: 192.168.3.0/24
Available IPv4 Addresses	251	IPv6 CIDR: -
Availability Zone	us-east-2b	Route Table: rtb-065ac615f7dcf78fc
Network ACL	acl-080850d72e13cd7dd	Default subnet: No
Auto-assign public IPv4	No	Auto-assign IPv6 address: No

The next thing we need to add to our VPC, is an elastic load balancer. You can just go to **EC2s**, and click **Load Balancers**, and then **Create Load Balancer**.

You will see that you have the option of an **Application Load Balancer**, a **Network Load Balancer**, or a **Classic Load Balancer**:

Select load balancer type

Elastic Load Balancing supports three types of load balancers: Application Load Balancers, Network Load Balancers (new), and Classic Load Balancers. Choose the load balancer type that meets your needs. [Learn more about which load balancer is right for you](#)

<b>Application Load Balancer</b>	<b>Network Load Balancer</b>	<b>Classic Load Balancer</b>
<b>Create</b>	<b>Create</b>	<b>Create</b>
Choose an Application Load Balancer when you need a flexible feature set for your web applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.	Choose a Network Load Balancer when you need ultra-high performance and static IP addresses for your application. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second while maintaining ultra-low latencies.	PREVIOUS GENERATION for HTTP, HTTPS, and TCP
		Choose a Classic Load Balancer when you have an existing application running in the EC2-Classic network.
		<a href="#">Learn more &gt;</a>
<b>Cancel</b>		

For our purposes, we're not going to need to do content based routing, so let's just choose the simpler **Classic Load Balancer**. We need to give it a name, and select our VPC.

We can leave the listener as **HTTP**:

Step 1: Define Load Balancer

Basic Configuration

This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured your load balancer with a standard web server on port 80.

Load Balancer name:	AutoscalingELB		
Create LB Inside:	vpc-080d822b0553327a2 (192.168.0.0/16)   ScratchVPC		
Create an internal load balancer:	<input type="checkbox"/> (what's this?)		
Enable advanced VPC configuration:	<input checked="" type="checkbox"/>		
Listener Configuration:			
Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	80

Add

Select Subnets

You will need to select a Subnet for each Availability Zone where you wish traffic to be routed by your load balancer. If you have instances in only one Availability Zone, please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

VPC vpc-080d822b0553327a2 (192.168.0.0/16) | ScratchVPC

Please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

[Cancel](#) | [Next: Assign Security Groups](#)

And since this is an internet-facing load balancer, we need to add our two public subnets, and then click **Next: Assign Security Groups**.

We already have a security group defined for the ELB, which has HTTP port 80 open to the internet:

Step 2: Assign Security Groups

You have selected the option of having your Elastic Load Balancer inside of a VPC, which allows you to assign security groups to your load balancer. Please select the security groups to assign to this load balancer. This can be changed at any time.

Assign a security group:

Create a new security group  
 Select an existing security group

Filter [VPC security groups](#)

Security Group ID	Name	Description	Actions
sg-0c41dbb45da5c7de0	default	default VPC security group	<a href="#">Copy to new</a>

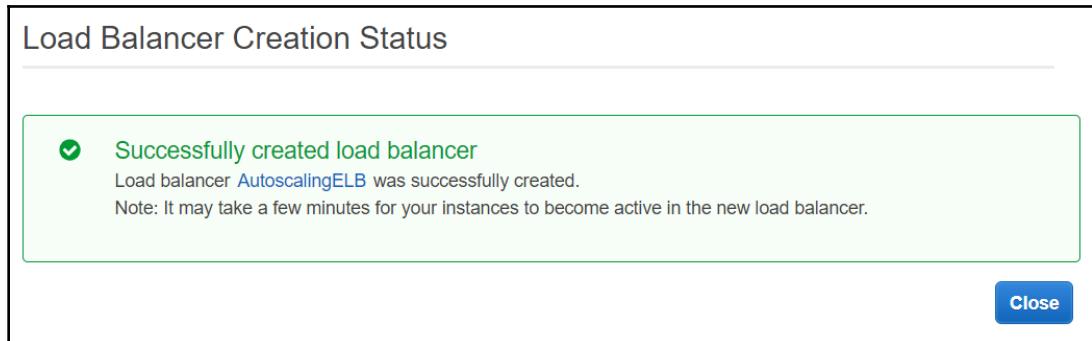
Click **Next: Configure Health Check**, and we'll leave the health check as it is, which is to check the index page of our web server:

The screenshot shows the 'Configure Health Check' step of a wizard. At the top, there are seven tabs: 1. Define Load Balancer, 2. Assign Security Groups, 3. Configure Security Settings, 4. Configure Health Check (which is selected), 5. Add EC2 Instances, 6. Add Tags, and 7. Review. Below the tabs, the heading 'Step 4: Configure Health Check' is displayed. A note states: 'Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer.' Under 'Advanced Details', there are four configuration items: 'Ping Protocol' set to 'HTTP', 'Ping Port' set to '80', 'Ping Path' set to '/index.html', 'Response Timeout' set to '5 seconds', 'Interval' set to '30 seconds', 'Unhealthy threshold' set to '2', and 'Healthy threshold' set to '10'. The entire form is enclosed in a light gray border.

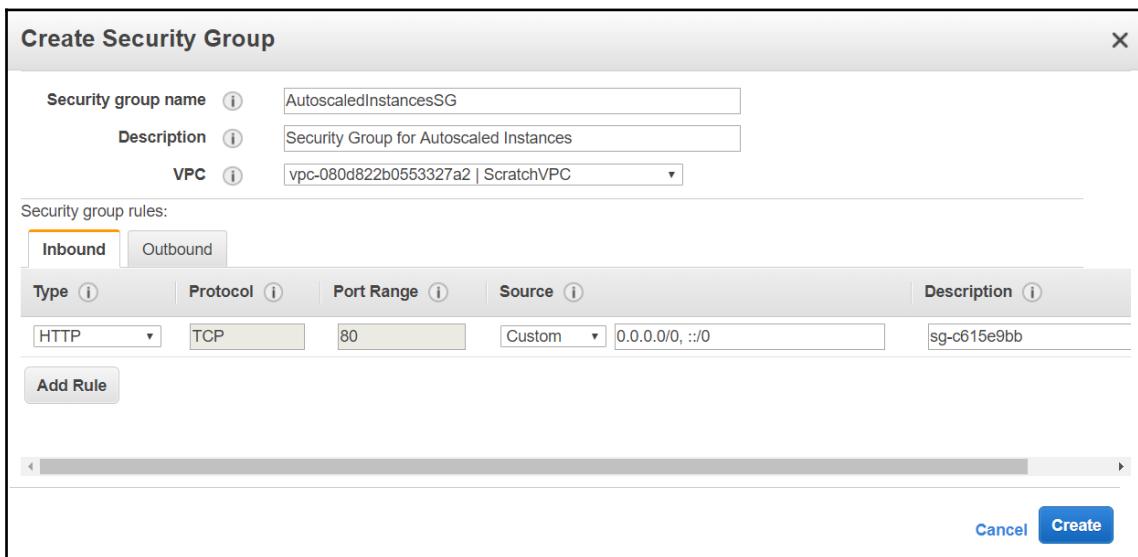
We are not going to add any EC2 instances, because we're going to allow the auto scaling group to do it, so just click **Next: Add Tags**, and then **Review and Create**, and finally **Create**:

The screenshot shows the 'Review' step of the wizard. At the top, there are seven tabs: 1. Define Load Balancer, 2. Assign Security Groups, 3. Configure Security Settings, 4. Configure Health Check, 5. Add EC2 Instances, 6. Add Tags, and 7. Review (which is selected). Below the tabs, the heading 'Step 7: Review' is displayed. The review section is divided into several sections with expandable details: 'Define Load Balancer' (Load Balancer name: AutoscalingELB, Scheme: internet-facing, Port Configuration: 80 (HTTP) forwarding to 80 (HTTP)), 'Configure Health Check' (Ping Target: HTTP:80/index.html, Timeout: 5 seconds, Interval: 30 seconds, Unhealthy threshold: 2, Healthy threshold: 10), 'Add EC2 Instances' (Cross-Zone Load Balancing: Enabled, Connection Draining: Enabled, 300 seconds, Instances: [empty]), 'VPC Information' (VPC: vpc-080d822b0553327a2 (ScratchVPC), Subnets: subnet-0b89925cd6dbace61 (Private1)), and 'Security groups' (Edit security groups). At the bottom right, there are three buttons: 'Cancel', 'Previous', and a blue 'Create' button.

You will see that the load balancer has been successfully created:



Next, let's create a security group for the instances and our auto scaling group. Click on **Create Security Group**, give this a name and description, select our **VPC**, and for the **Inbound** rule, we're going to allow **HTTP**:



However, we're going to restrict it to the security group, for the load balancer. So type in **sg**, and then select **ELB security group**, and then **Create**.

Now let's go and create our auto scaling group.

**Click on Auto Scaling Groups | Create Auto Scaling group | Use an existing launch configuration:**

Create Auto Scaling Group

Complete this wizard to create your Auto Scaling group. First, choose either a launch configuration or a launch template to specify the parameters that your Auto Scaling group uses to launch instances.

**Launch Configuration**  
You can continue to use your launch configurations if they support the Amazon EC2 features you need. [Learn more](#)

**Launch Template New**  
Launch templates can be updated and versioned, and include support for the latest features of Amazon EC2. [Learn more](#) [Create new launch template](#)

**Create a new launch configuration**

**Use an existing launch configuration**

Filter launch configurations... <span style="float: right;">X</span>				
Name	AMI ID	Instance Type	Spot Price	Security Groups
test1	ami-1853ac65	t2 micro		sg-67570011

[Cancel](#) [Next Step](#)

We will then pick one of the AMIs, with just a basic web page:

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

**Step 1: Choose an Amazon Machine Image (AMI)**

AMIs are templates that contain 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.

Cancel and Exit

Quick Start (0)

**My AMIs (1)**

AWS Marketplace (3107)

Community AMIs (0)

**Ownership**

- Owned by me
- Shared with me

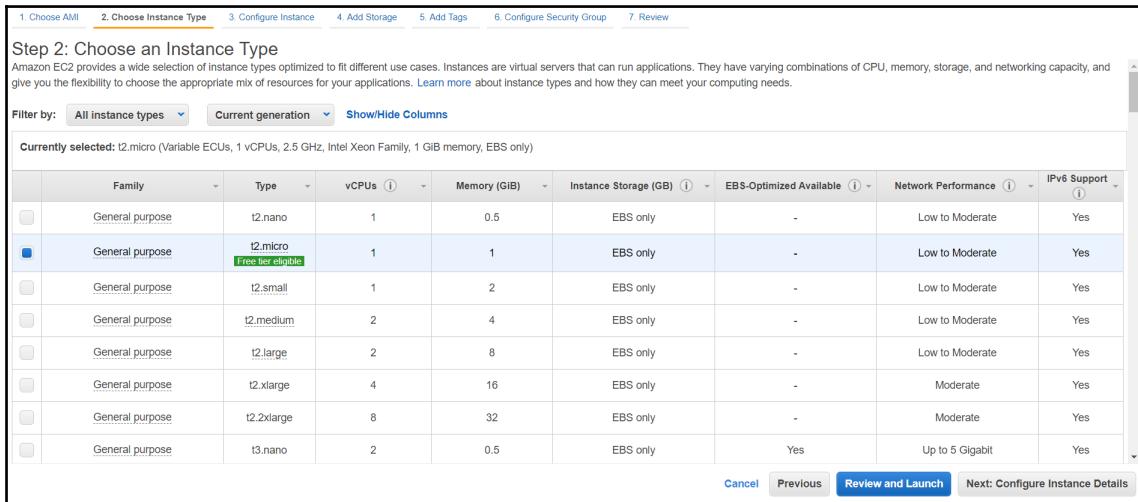
**Architecture**

- 32-bit
- 64-bit

**Root device type**

1 to 1 of 1 AMIs < >				
<b>VPC Image - ami-030e6d48f715d9498</b> Image for Instance Root device type: ebs Virtualization type: hvm Owner: 019859648260 ENA Enabled: Yes	<a href="#">Select</a>			
The following results for "ami-030e6d48f715d9498" were found in other catalogs:				
3107 results in AWS Marketplace AWS Marketplace provides partnered Software that is pre-configured to run on AWS				

We'll leave the default instance type, **t2.micro**, because this is on the free tier:



The screenshot shows the 'Step 2: Choose an Instance Type' page of the AWS CloudFormation 'Create New Stack' wizard. The top navigation bar includes steps 1 through 7. A note at the top states: 'Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.' Below this is a filter section with 'All instance types' selected, 'Current generation' checked, and 'Show/Hide Columns'. A note says 'Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)'. The main table lists various instance types under 'General purpose' family, including t2.nano, t2.micro (selected), t2.small, t2.medium, t2.large, t2.xlarge, t2.2xlarge, and t3.nano. Each row shows details like vCPUs, Memory (GiB), Instance Storage (GB), EBS-Optimized Available, Network Performance, and IPv6 Support. The 't2.micro' row has a green 'Free tier eligible' badge. At the bottom are 'Cancel', 'Previous', 'Review and Launch' (which is blue and bold), and 'Next: Configure Instance Details'.

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
General purpose	<b>t2.micro</b> Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
General purpose	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes

However, in production, you would not want to auto scale these instances, because the CPU is burstable. So in that case, you would choose something like an m3 or an m4. Then, we'll give our launch configuration a name.

We're going to enable **CloudWatch detailed monitoring**, which will allow CloudWatch to record metrics at one minute intervals, instead of the default five minute intervals. Next, click **Next: Add Storage**:

1. Choose AMI   2. Choose Instance Type   3. Configure Instance   4. Add Storage   5. Add Tags   6. Configure Security Group   7. Review

### Step 3: Configure Instance Details

Number of instances	<input type="text" value="1"/>	Launch into Auto Scaling Group
Purchasing option	<input type="checkbox"/> Request Spot Instances	
Network	<input type="text" value="vpc-534b882b (default)"/>	<input type="button" value="Create new VPC"/>
Subnet	<input type="text" value="No preference (default subnet in any Availability Zone)"/>	<input type="button" value="Create new subnet"/>
Auto-assign Public IP	<input type="checkbox"/> Use subnet setting	
Placement group	<input type="checkbox"/> Add instance to placement group.	
IAM role	<input type="text" value="None"/>	<input type="button" value="Create new IAM role"/>
Shutdown behavior	<input type="checkbox"/> Stop	
Enable termination protection	<input type="checkbox"/> Protect against accidental termination	
Monitoring	<input checked="" type="checkbox"/> Enable CloudWatch detailed monitoring Additional charges apply.	
Tenancy	<input type="checkbox"/> Shared - Run a shared hardware instance Additional charges will apply for dedicated tenancy.	
T2/T3 Unlimited	<input type="checkbox"/> Enable	

In **Configure Security Group**, let's choose the security group that we just created:

1. Choose AMI   2. Choose Instance Type   3. Configure Instance   4. Add Storage   5. Add Tags   6. Configure Security Group   7. Review

### Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:  Create a new security group  
 Select an existing security group

Security Group ID	Name	Description	Actions
<input checked="" type="checkbox"/> sg-67570011	AutoScaling-Security-Group-1	AutoScaling-Security-Group-1 (2018-03-22 23:47:21.944+05:30)	<input type="button" value="Copy to nr"/> <input type="button" value="Copy to nr"/>
<input type="checkbox"/> sg-a5afed3	AWS-OpsWorks-AWS-Flow-Ruby-Server	AWS Flow Ruby server - do not change or delete	<input type="button" value="Copy to nr"/> <input type="button" value="Copy to nr"/>
<input type="checkbox"/> sg-0bde8a7d	AWS-OpsWorks-Blank-Server	AWS OpsWorks blank server - do not change or delete	<input type="button" value="Copy to nr"/> <input type="button" value="Copy to nr"/>
<input type="checkbox"/> sg-c0dd8986	AWS-OpsWorks-Custom-Server	AWS OpsWorks custom server - do not change or delete	<input type="button" value="Copy to nr"/> <input type="button" value="Copy to nr"/>
<input type="checkbox"/> sg-bed387c8	AWS-OpsWorks-DB-Master-Server	AWS OpsWorks database master server - do not change or delete	<input type="button" value="Copy to nr"/> <input type="button" value="Copy to nr"/>
<input type="checkbox"/> sg-19d3876f	AWS-OpsWorks-Default-Server	AWS OpsWorks Default server - do not change or delete	<input type="button" value="Copy to nr"/> <input type="button" value="Copy to nr"/>
<input type="checkbox"/> sg-7ac7930c	AWS-OpsWorks-ECS-Cluster	AWS OpsWorks ECS cluster - do not change or delete	<input type="button" value="Copy to nr"/> <input type="button" value="Copy to nr"/>

Inbound rules for sg-67570011 (Selected security groups: sg-67570011)

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	

Click **Review and Launch**, and then **Launch**:

Step 7: Review Instance Launch

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

AMI Details

VPC Image - ami-030e6d48f715d9498  
Image for Instance  
Root Device Type: ebs Virtualization type: hvm

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups

Security Group ID	Name	Description
sg-67570011	AutoScaling-Security-Group-1	AutoScaling-Security-Group-1 (2018-03-22 23:47:21.944+05:30)

All selected security groups inbound rules

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	

Cancel Previous Launch

Finally, let's just choose one of our key pairs, then check the box that we acknowledge we have access to the key, and create the launch configuration:

### Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair

Select a key pair

DemoKeyPair

I acknowledge that I have access to the selected private key file (DemoKeyPair.pem), and that without this file, I won't be able to log into my instance.

Cancel Launch Instances

Now we have finished the launch configuration, let's give the group a name; select our VPC and our two private subnets:

1. Configure Auto Scaling group details    2. Configure scaling policies    3. Configure Notifications    4. Configure Tags    5. Review

Create Auto Scaling Group

Launch Configuration test1

Group name AutoscaleDemoGroup

Group size Start with 1 instances

Network vpc-080d822b0553327a2 (192.168.0.0/16) | ScratchV... [Create new VPC](#)

Subnet

subnet-0b89925cd9dbace61(192.168.1.0/24) | Private1 | us-east-1a  
subnet-00c64b67d84399200(192.168.0.0/24) | Public1 | us-east-1a

[Create new subnet](#)

**⚠ No public IP addresses will be assigned**

None of the instances in this Auto Scaling group will be assigned a public IP address because you have not chosen to launch in your default VPC and subnet.

You can ensure a public IP address is assigned to instances launched with this configuration by selecting only default subnets of your default VPC.

[Learn more](#) about IP addressing in an Amazon VPC.

[Cancel](#) [Next: Configure scaling policies](#)

Now let's scroll down to the **Advanced Details**, click the arrow, and check the **Receive traffic from one or more load balancers** box, and select our load balancer:

**Advanced Details**

**Load Balancing** [i](#)  Receive traffic from one or more load balancers [Learn about Elastic Load Balancing](#)

**Classic Load Balancers** [i](#)  [x](#)

**Target Groups** [i](#)

**Health Check Type** [i](#)  ELB  EC2

**Health Check Grace Period** [i](#)  seconds

**Monitoring** [i](#) Amazon EC2 Detailed Monitoring metrics, which are provided at 1 minute frequency, are not enabled for the launch configuration test1. Instances launched from it will use Basic Monitoring metrics, provided at 5 minute frequency. [Learn more](#)

**Instance Protection** [i](#)

**Service-Linked Role** [i](#)  [C](#) [View Role in IAM](#)

For the **Health Check Type**, select **ELB**, and then click **Next: Configure scaling policies**.

Check the radio button, **Use scaling policies to adjust the capacity of this group**, and let's scale between a minimum of 1 instances, and a maximum of 10:

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

**Create Auto Scaling Group**  
You can optionally add scaling policies if you want to adjust the size (number of instances) of your group automatically. A scaling policy is a set of instructions for making such adjustments in response to an Amazon CloudWatch alarm that you assign to it. In each policy, you can choose to add or remove a specific number of instances or a percentage of the existing group size, or you can set the group to an exact size. When the alarm triggers, it will execute the policy and adjust the size of your group accordingly. [Learn more](#) about scaling policies.

Keep this group at its initial size  
 Use scaling policies to adjust the capacity of this group

Scale between  and  instances. These will be the minimum and maximum size of your group.

**Increase Group Size**

Name:   
Execute policy when: [Edit](#) [Remove](#)  
aws:CloudWatchMetrics:CPUUtilization <= 20 for 3 consecutive periods of 300 seconds  
breaches the alarm threshold: CPUUtilization <= 20 for 3 consecutive periods of 300 seconds  
for the metric dimensions AutoScalingGroupName = AutoscaleDemoGroup

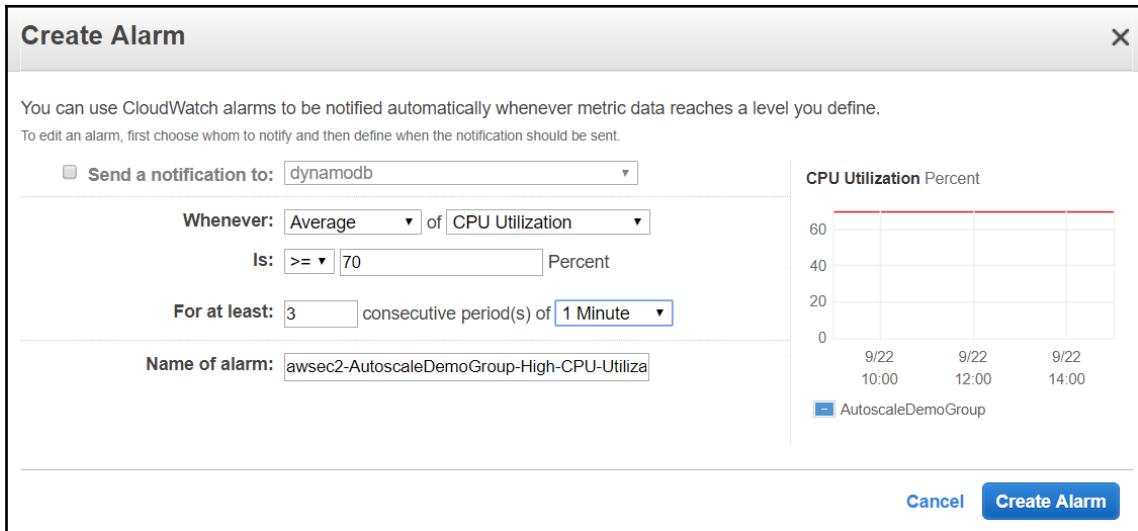
Take the action: [Add](#)  instances [▼](#)  
And then wait:  seconds before allowing another scaling activity

[Create a scaling policy with steps](#) [i](#)

Decrease Group Size

[Cancel](#) [Previous](#) [Review](#) [Next: Configure Notifications](#)

Here, we're going to set the scaling policy and the alarm. So for the first policy, which is going to be increasing the group size, we need to configure an alarm. Click on **Add new alarm**. We can uncheck the **Send a notification to**, although we could send ourselves a notification however, if we like. And then let's go with creating an alarm whenever the average of CPU utilization is greater than or equal to, and let's put 70%, and for at least three consecutive periods, of one minute. We'll just leave the **Name of alarm** as it is, and click **Create Alarm**:



Now we need to do the same thing for decreasing the group size. Create an alarm, let's uncheck the **Send a notification to**, and change the rule to whenever the average of CPU utilization is less than or equal to 20%, for 3 consecutive periods of **5 Minutes**.

Let's just call this, low CPU utilization. Then finally, click **Create Alarm**:

**Create Alarm**

You can use CloudWatch alarms to be notified automatically whenever metric data reaches a level you define. To edit an alarm, first choose whom to notify and then define when the notification should be sent.

Send a notification to: dynamodb

Whenever: Average of CPU Utilization  
Is: <= 20 Percent

For at least: 3 consecutive period(s) of 5 Minutes

Name of alarm: awsec2-AutoScaleDemoGroup-High-CPU-Utiliza

CPU Utilization Percent

9/22 10:00 9/22 12:00 9/22 14:00

AutoscaleDemoGroup

**Create Alarm**

Let's go back to our first group, for the increased group size, and select **Create a simple scaling policy**.

Let's just say that we're going to **Add**, 2 instances, whenever this alarm goes off:

**Increase Group Size**

Name: Increase Group Size

Execute policy when: awsec2-AutoScaleDemoGroup-High-CPU-Utilization Edit Remove  
breaches the alarm threshold: CPUUtilization <= 20 for 3 consecutive periods of 300 seconds  
for the metric dimensions AutoScalingGroupName = AutoscaleDemoGroup

Take the action: Add 2 instances

And then wait: 300 seconds before allowing another scaling activity

Create a scaling policy with steps ⓘ

Let's go down to the same link and, under **Decreased Group Size**, create a simple scaling policy, and let's say we remove two instances, whenever this alarm goes off:

**Decrease Group Size**

Name: Decrease Group Size

Execute policy when: awsec2-AutoScalingDemoGroup-High-CPU-Utilization [Edit](#) [Remove](#)  
breaches the alarm threshold: CPUUtilization <= 20 for 3 consecutive periods of 300 seconds  
for the metric dimensions AutoScalingGroupName = AutoScalingDemoGroup

Take the action: [Remove](#) ▾  instances ▾

And then wait:  seconds before allowing another scaling activity

[Create a scaling policy with steps](#) [i](#)

Finally, let's just click on **Review**, and then **Create Auto Scaling group**, and then **Close**.

We now have our auto scaling group. If we go check our **EC2s**, there should be two more launching.

Now let's go into our **Load balancers**, and after a few minutes, you should be able to click on the **Instances** tab, and see the two instances in service, behind the load balancer:

Load balancer: AutoscalingELB

Description Instances Health Check Listeners Monitoring Tags

Connection Draining: Enabled, 300 seconds ([Edit](#))

[Edit Instances](#)

Instance ID	Name	Availability Zone	Status	Actions
i-f524146c		us-east-1a	InService <a href="#">i</a>	<a href="#">Remove from Load Balancer</a>
i-fbc1b668		us-east-1b	InService <a href="#">i</a>	<a href="#">Remove from Load Balancer</a>

[Edit Availability Zones](#)

## Summary

In this chapter, we discussed classic and VPC EC2 instances, and how to cope with a mixed environment. We also described the default VPC, which for simple public applications and proof of concepts may be all that you need. However, it will require a lot of modifications to make it suitable for most production workloads and provide the required security. We created a simple VPC, first by using the VPC Wizard, and then we created one from scratch. We created the VPC, attached an internet gateway, created private and public subnets and route tables, and launched an NAT gateway in our public subnet. We discussed accessing the internet through virtual private gateways, VPN connections, direct connect, and VPC peering. We talked about how to make your VPCs secure, by using network access control lists, and Bastion instances.

By way of some subsequent steps, I suggest getting some practice with your free AWS account. Create some VPCs and launch applications on EC2 instances in them. Don't forget to implement best security practices and make your application highly available. You may want to go back and review the demonstration chapters in this book, pausing as needed. Good luck using your new AWS skills.

# Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:

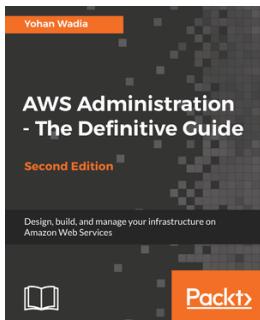


## **Expert AWS Development**

Atul V. Mistry

ISBN: 978-1-78847-758-1

- Learn how to get up and running with AWS Developer Tools.
- Integrate the four major phases in the Release Processes. Source, Build, Test and Production.
- Learn how to integrate Continuous Integration, Continuous Delivery, and Continuous Deployment in AWS.
- Make secure, scalable and fault tolerant applications.
- Understand different architectures and deploy complex architectures within minutes



## **AWS Administration - The Definitive Guide - Second Edition**

Yohan Wadia

ISBN: 978-1-78847-879-3

- Take an in-depth look at what's new with AWS, along with how to effectively manage and automate your EC2 infrastructure with AWS Systems Manager
- Deploy and scale your applications with ease using AWS Elastic Beanstalk and Amazon Elastic File System
- Secure and govern your environments using AWS CloudTrail, AWS Config, and AWS Shield
- Learn the DevOps way using a combination of AWS CodeCommit, AWS CodeDeploy, and AWS CodePipeline
- Run big data analytics and workloads using Amazon EMR and Amazon Redshift
- Learn to back up and safeguard your data using AWS Data Pipeline
- Get started with the Internet of Things using AWS IoT and AWS Greengrass

## **Leave a review - let other readers know what you think**

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!

# Index

## A

accelerated computing-general purpose GPU instances  
about 34  
F1 instance 36  
G3 instance 35  
P2 instance 34  
P3 instance 34  
AMI  
about 52  
AWS marketplace 57, 58  
Community AMIs 55, 56  
My AMIs 59, 61  
Quick Start 53  
availability zones (AZs) 107  
AWS account  
opening 6, 8, 9, 11, 12, 13  
reference 6  
AWS Management Console  
reference 14  
using 14, 15, 16, 18, 19, 20, 21, 22  
AWS marketplace  
reference 58

## B

Bastion instance  
using 135

## C

C4 instance  
about 29  
features 29  
C5 instance  
about 28  
features 28  
classic EC2s 102

## Classless Inter-Domain Routing (CIDR)

about 86  
EC2 IP address 90  
IPv4 87, 88  
reference 88  
valid private IP address ranges, specifying 88, 89  
compute optimized instance  
about 28  
C4 instance 29  
C5 instance 28

## D

D2 instance  
about 38  
features 38  
default VPC  
creating 103, 104, 107

## E

EC2 instance  
accelerated computing-general purpose GPU instances 34  
compute optimized instance 28  
general purpose instance 24  
launching 39, 40, 42  
memory optimized instance 30  
reference 38  
storage optimized instance 36  
types 24  
EC2 IP address  
elastic IP addresses 92  
elastic network interface (ENI) 92  
private IP addresses 90  
public IP addresses 91  
EC2 storage options  
about 43

**Elastic Block Storage (EBS)** 44  
instance storage 43  
**EC2s**  
in VPC 102  
**Elastic Block Storage (EBS)**  
about 44  
general purpose SSD 45  
provisioned IOPS SSD 45  
throughput optimized HDD 45  
**elastic IP addresses** 92  
**elastic network adapter (ENA)** 37

## F

**F1 instance**  
about 36  
features 36

## G

**G3 instance**  
about 35  
features 35  
**general purpose instance**  
about 25  
**M4 instance** 27  
**M5 instance** 26  
**T2 instance** 25  
**T3 instance** 25  
**general purpose SSD** 45

## H

**H1 instance**  
about 36  
features 36  
**hardware virtual machine (HVM)** 57

## I

**I3 instance**  
about 37  
features 37  
**instance storage** 43  
**Internet Assigned Numbers Authority (IANA)** 88  
**internet gateway** 127  
**IPv4** 87, 88

## K

**key pairs**  
about 62  
generating 63, 64, 65, 66

## L

**Linux instances**  
logging in 67, 68, 70, 71, 72, 74, 75, 76, 77

## M

**M4 instance**  
about 27  
advantages 27  
**M5 instance**  
about 26  
advantages 26  
**memory optimized instance**  
about 30  
**R4 instance** 32  
**R5 instance** 31  
**X1 instance** 31  
**X1e instance** 30  
**z1d instance** 33

## N

**NAT instance**  
launching 97, 98  
**network ACLs (NACLs)**  
about 130  
launching 131, 132, 133, 134

## P

**P2 instance**  
about 34  
features 34  
**P3 instance** 34  
**paravirtual virtualization (PV)** 57  
**private IP addresses** 90  
**provisioned IOPS SSD** 45  
**public IP addresses** 91  
**public subnet**  
about 96  
versus private subnet 96  
**PuTTY**

reference 67  
using 67

## R

R4 instance  
about 32  
features 32  
R5 instance  
about 31  
features 31  
route tables 93, 94, 96

## S

security group  
creating 46, 48, 49, 50, 51, 52  
software VPN  
using 128  
storage optimized instance  
about 36  
D2 instance 38  
H1 instance 36  
I3 instance 37  
subnets  
about 93, 94  
NAT instance, launching 97, 99  
public subnet, versus private subnet 96

## T

T2 instance  
about 25  
advantages 25  
T3 instance  
about 25  
features 25  
throughput optimized HDD 45

## V

valid private IP address ranges  
reference 89  
specifying 88  
VPC architectures  
about 136  
auto scaling 140, 141, 143, 144, 145, 146,

148, 151, 152, 153  
availability zones 136, 138  
elastic load balancer (ELB) 138  
load balancing stateful applications 138

VPC connection  
establishing 126  
software VPN, using 128  
via direct connect 129  
via internet gateway 127  
via VPC peering 130  
virtual gateway, using 128

VPC demo  
creating 108

VPC, securing  
about 130  
with Bastion instance 135  
with NACLs 130, 132, 133, 134

VPC  
about 101  
classic EC2s 102  
creating, with Wizard 108, 113, 115, 116, 118, 119, 120, 122, 123, 125, 126  
default VPC, creating 104, 107  
with EC2s 102

## W

Windows instances  
logging in 78, 80, 81, 82, 83, 85

Wizard  
VPC, creating 108, 113, 115, 116, 118, 119, 120, 122, 123, 124, 125, 126

## X

X1 instance  
about 31  
features 31  
X1e instance  
about 30  
features 30

## Z

z1d instance  
about 33  
features 33