

AWS CloudFormation

The Easy Way

Yan Kurniawan

AWS CloudFormation - The Easy Way

Yan Kurniawan

This book is for sale at http://leanpub.com/aws-cloudformation

This version was published on 2018-03-07



This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2016 - 2018 Yan Kurniawan

to my wife - it's a privilege to share my life with you

to my children - my source of joy and inspiration

Contents

Preface . . . . . **i**
What this book covers ii
Who this book is for iii
What you need for this book iii
Conventions iii
Example Code Files iv

Chapter 1: Getting Started with AWS **1**
What is Amazon Web Services (AWS)? 1
Setting Up Your AWS Account 7
AWS Management Console 12
Create Your First EC2 Instance 16
Connect to Your Instance 26
Terminate Your Instance 28

Chapter 2: Getting Started with CloudFormation **30**
What is AWS CloudFormation 30
AWS CloudFormation Concepts 31
How Does AWS CloudFormation Work? 34
Template Anatomy 37
Format Version 38
Description 39
Metadata 39
Parameters 40
Parameter's Properties 40
Type Property 42
AWS-specific Parameter Types 43
Mappings 44
Conditions 45
Resources 46
Outputs 47
Your First Template 49
Updating Stack 53

Canceling a Stack Update 57
Deleting Stack 58

Preface

AWS (Amazon Web Services) CloudFormation is a service that helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and AWS CloudFormation takes care of provisioning and configuring those resources for you. You don't need to individually create and configure AWS resources and figure out what's dependent on what; AWS CloudFormation handles all of that. [1]

When you use AWS CloudFormation, you can reuse your template to set up your resources consistently and repeatedly. Just describe your resources once and then provision the same resources over and over in multiple regions. This book will show you how to write a reusable CloudFormation template for your cloud infrastructure.

The aim of this book is to help you get started with AWS CloudFormation in an easy way. With detailed steps, complete codes, and screen captures from AWS console, I hope you will get better understanding of how things work.

The example project will help you grasp the concepts quickly. You will be able to build cloud infrastructure for a multi-tier WordPress site using CloudFormation in no time. Throughout the book you will get your hands dirty writing CloudFormation template and deploying stacks. By the end of reading this book, I hope that you will master AWS CloudFormation.

[1] http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html

What this book covers

You'll find the following chapters in this book:

Chapter 1, Getting Started with AWS, introduces you to AWS, shows you how to sign up for AWS (Amazon Web Services), set up your account, get familiar with AWS console, create your first Amazon EC2 (Elastic Compute Cloud) instance, and connect to your EC2 instance using SSH.

Chapter 2, Getting Started with CloudFormation, teaches you the basics of CloudFormation, how CloudFormation works, and the template basics.

Chapter 3, The IAM Stack, shows you how to create IAM Roles, Profiles and Policies for the example project using CloudFormation.

Chapter 4, The Network Stack, guides you to build the VPC, Subnets, and Security Groups for the example project using CloudFormation.

Chapter 5, The Application Stack, guides you to create ELB, Launch Config, Auto Scaling Group, and Auto Scaling Policy for the example project.

Chapter 6, Stack Policy, teaches you how to protect stack resources from update actions using a stack policy.

Appendix A, Converting JSON to YAML, shows you how to convert your existing JSON templates into YAML templates.

Appendix B, CLI Commands, shows you how to do stack operations using AWS CLI.

Appendix C, Intrinsic Function, reference guide for CloudFormation Intrinsic Functions.

Appendix D, Resource Attribute, reference guide for CloudFormation Resource Attributes.

Appendix E, Pseudo Parameters, reference guide for CloudFormation Pseudo Parameters.

Appendix F, Helper Scripts, reference guide for CloudFormation Helper Scripts.

Who this book is for

This book is written for cloud engineers, solutions architects, and system administrators. I assume you are comfortable with the basic AWS resources like VPC, EC2, IAM, Security Groups, and Auto Scaling. For those who are not familiar with AWS, chapter 1 will give you an introduction to AWS and EC2 service, but you will need other resources to learn more about AWS services. You can read through the online AWS documentation https://aws.amazon.com/documentation/ or you can read my other book **Ansible for AWS** https://leanpub.com/ansible-for-aws.

What you need for this book

To run the examples in this book, you will need a computer with any operating system, your favourite text editor and an Internet connection. To use the services in Amazon Web Services, you will need to setup an account and register your credit card with Amazon.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follow: "The `Description` section (optional) enables you to include arbitrary comments about your template. "

A block of code is set as follows:

```
1  ---
2  AWSTemplateFormatVersion: "2010-09-09"
3  Description: A sample template
```

Any command-line input is written as follows:

```
$ vi iam.yml
```



This is an information box

Special information appear in a box like this



This is a warning box

Warnings appear in a box like this



Tips and tricks

Tips and tricks appear like this

Example Code Files

You can find the example code files for this book in the AWS CloudFormation - The Easy Way GitHub repository[2].

[2] https://github.com/yankurniawan/aws-cloudformation

Chapter 1: Getting Started with AWS

This chapter will give you introduction to Amazon Web Services (AWS), show you how to set up your AWS account, create your first EC2 instance, and connect to your EC2 instance using SSH. You can skip this chapter if you already have an AWS account and experience with EC2 instance.

What is Amazon Web Services (AWS)?

Amazon Web Services (AWS) provides computing resources and services that you can use to build applications within minutes at pay-as-you-go pricing. For example, you can rent a server on AWS that you can connect to, configure, secure, and run just as you would a physical server. The difference is the virtual server runs on top of a planet-scale network managed by AWS.

You pay for your virtual server only while it runs, with no up-front purchase costs or ongoing maintenance costs. Backed by the AWS network, your virtual server can do things no physical server can, such as automatically scaling into multiple servers when demand for your application increases.

Using AWS to build your Internet application is like purchasing electricity from a power company instead of running your own generator, and it provides many of the same benefits: capacity exactly matches your need, you pay only for what you use, economies of scale result in lower costs, and the service is provided by a vendor experienced in running large-scale networks. In addition, AWS can offer significant cost savings, up to 80%, compared to the equivalent on-premises deployments.

You can run nearly anything on AWS that you would run on physical hardware: websites, applications, databases, mobile apps, email campaigns, distributed data analysis, media storage, and private networks. The services we provide are designed to work together so that you can build complete solutions. There are currently dozens of services, with more being added each year.[3]

Amazon Web Services offers a broad set of global cloud-based products including storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security, compute and enterprise applications. These services help organizations move faster, lower IT costs, and scale.

For a complete list of AWS products offering you can see https://aws.amazon.com/products/.

Amazon has produced a set of short videos to help you understand AWS basics:

- What is Cloud Computing[4]
- What is Amazon Web Services[5]

[3] Getting Started with AWS

[4] http://youtu.be/jOhbTAU4OPI

[5] http://youtu.be/mZ5H8sn_2ZI

In this book we will use the following AWS services:

- **EC2** (Elastic Compute Cloud)
- **VPC** (Virtual Private Cloud)
- **RDS** (Relational Database Service)
- **ELB** (Elastic Load Balancing)
- **Auto Scaling**
- **Route 53**
- **CloudFormation**
- **IAM**

EC2 (Elastic Compute Cloud)

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios.[6]

VPC (Virtual Private Cloud)

Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the Amazon Web Services (AWS) cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways.

You can easily customize the network configuration for your Amazon Virtual Private Cloud. For example, you can create a public-facing subnet for your webservers that has access to the Internet, and place your backend systems such as databases or application servers in a private-facing subnet with no Internet access. You can leverage multiple layers of security, including security groups and network access control lists, to help control access to Amazon EC2 instances in each subnet.

Additionally, you can create a Hardware Virtual Private Network (VPN) connection between your corporate datacenter and your VPC and leverage the AWS cloud as an extension of your corporate datacenter.[7]

RDS (Relational Database Service)

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, freeing you up to focus on your applications and business. Amazon RDS provides you six familiar database engines to choose from, including Amazon Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MySQL and MariaDB.[8]

[6] https://aws.amazon.com/ec2/
[7] https://aws.amazon.com/vpc/
[8] https://aws.amazon.com/rds/

ELB (Elastic Load Balancing)

Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve fault tolerance in your applications, seamlessly providing the required amount of load balancing capacity needed to route application traffic.

Elastic Load Balancing offers two types of load balancers that both feature high availability, automatic scaling, and robust security. These include the Classic Load Balancer that routes traffic based on either application or network level information, and the Application Load Balancer that routes traffic based on advanced application level information that includes the content of the request. The Classic Load Balancer is ideal for simple load balancing of traffic across multiple EC2 instances, while the Application Load Balancer is ideal for applications needing advanced routing capabilities, microservices, and container-based architectures. Application Load Balancer offers ability to route traffic to multiple services or load balance across multiple ports on the same EC2 instance.[9]

Route 53

Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. It is designed to give developers and businesses an extremely reliable and cost effective way to route end users to Internet applications by translating names like www.example.com into the numeric IP addresses like 192.0.2.1 that computers use to connect to each other.

Amazon Route 53 effectively connects user requests to infrastructure running in AWS – such as Amazon EC2 instances, Elastic Load Balancing load balancers, or Amazon S3 buckets – and can also be used to route users to infrastructure outside of AWS. You can use Amazon Route 53 to configure DNS health checks to route traffic to healthy endpoints or to independently monitor the health of your application and its endpoints. Amazon Route 53 Traffic Flow makes it easy for you to manage traffic globally through a variety of routing types, including Latency Based Routing, Geo DNS, and Weighted Round Robin—all of which can be combined with DNS Failover in order to enable a variety of low-latency, fault-tolerant architectures. Using Amazon Route 53 Traffic Flow's simple visual editor, you can easily manage how your end-users are routed to your application's endpoints—whether in a single AWS region or distributed around the globe. Amazon Route 53 also offers Domain Name Registration – you can purchase and manage domain names such as example.com and Amazon Route 53 will automatically configure DNS settings for your domains.[10]

CloudFormation

AWS CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion.

[9] http://aws.amazon.com/elasticloadbalancing

[10] http://aws.amazon.com/route53

You can use AWS CloudFormation's sample templates or create your own templates to describe the AWS resources, and any associated dependencies or runtime parameters, required to run your application. You don't need to figure out the order for provisioning AWS services or the subtleties of making those dependencies work. CloudFormation takes care of this for you. After the AWS resources are deployed, you can modify and update them in a controlled and predictable way, in effect applying version control to your AWS infrastructure the same way you do with your software. You can also visualize your templates as diagrams and edit them using a drag-and-drop interface with the AWS CloudFormation Designer.

You can deploy and update a template and its associated collection of resources (called a stack) by using the AWS Management Console, AWS Command Line Interface, or APIs. CloudFormation is available at no additional charge, and you pay only for the AWS resources needed to run your applications.[11]

IAM

AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources.[12]

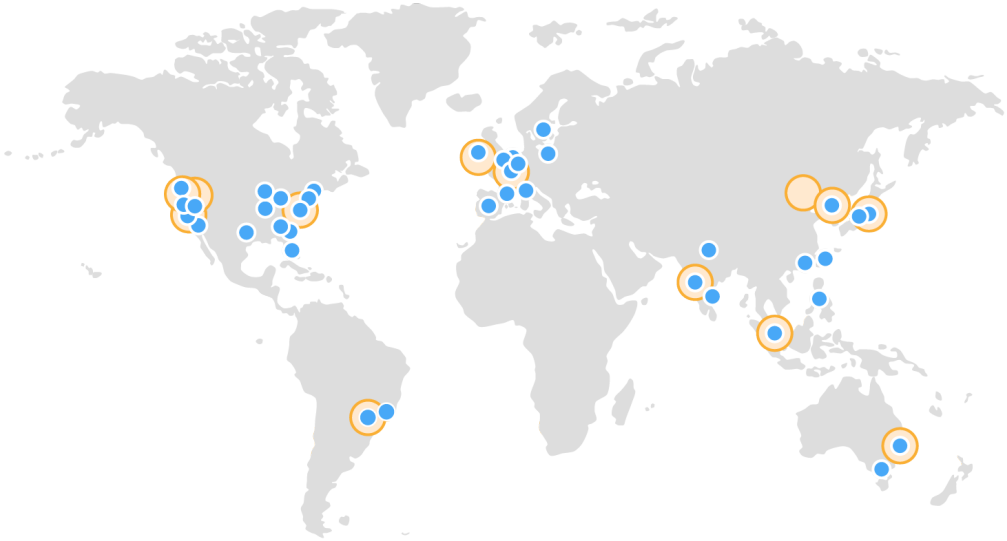
[11] https://aws.amazon.com/cloudformation/

[12] https://aws.amazon.com/iam/



AWS Global Infrastructure

Whether you are a large global company or small start-up, you may have potential customers around the world. With traditional infrastructure, it's hard to deliver great performance to a broadly distributed user base and most companies focus on one geographic region at a time to save costs and time. With Cloud Computing, the game changes - you can easily deploy your application in any or all of the AWS regions around the world. This means you can provide a lower latency and better experience for your customers at minimal cost.[13]



See detailed list of offerings at all AWS locations[14]

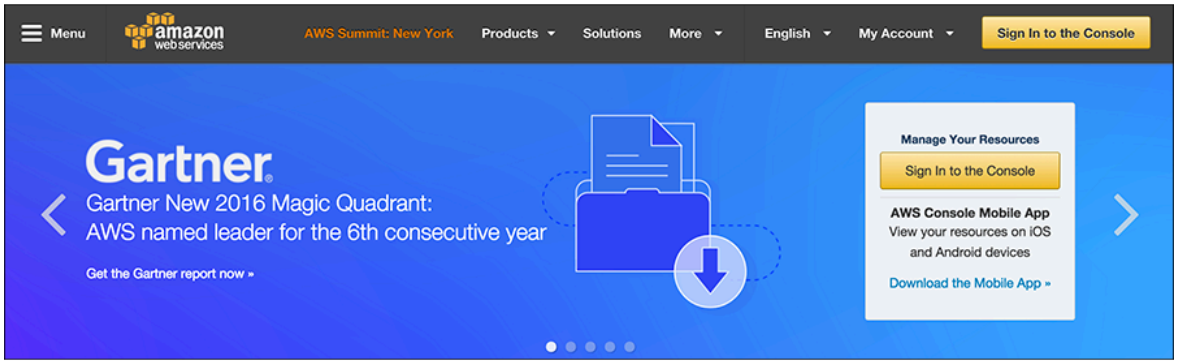
[13] http://aws.amazon.com/what-is-cloud-computing/#global-reach

[14] http://aws.amazon.com/about-aws/globalinfrastructure/regional-product-services

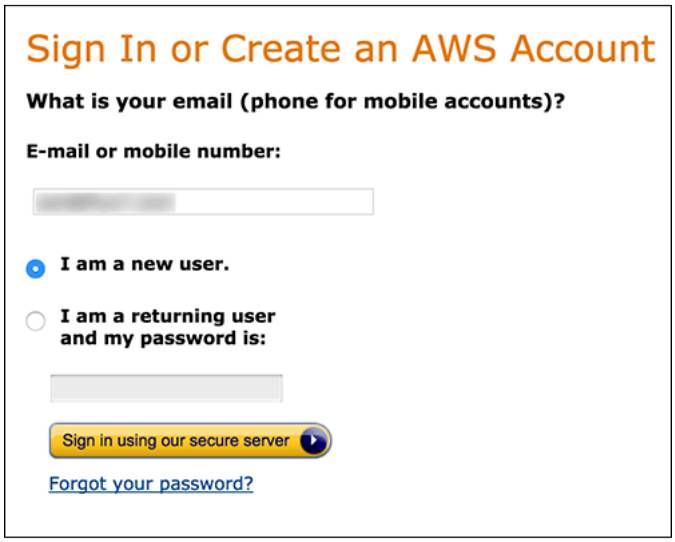
Setting Up Your AWS Account

If you don't already have an Amazon Web Services account, open your web browser on your computer and go to http://aws.amazon.com. Follow the steps below:

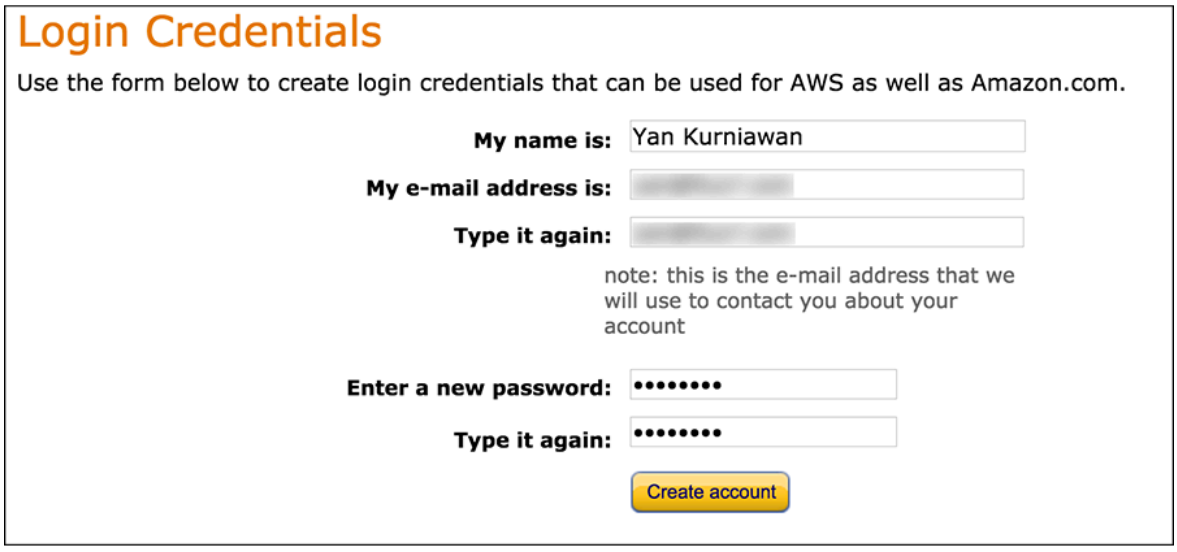
1. Click the **Sign In to the Console** button



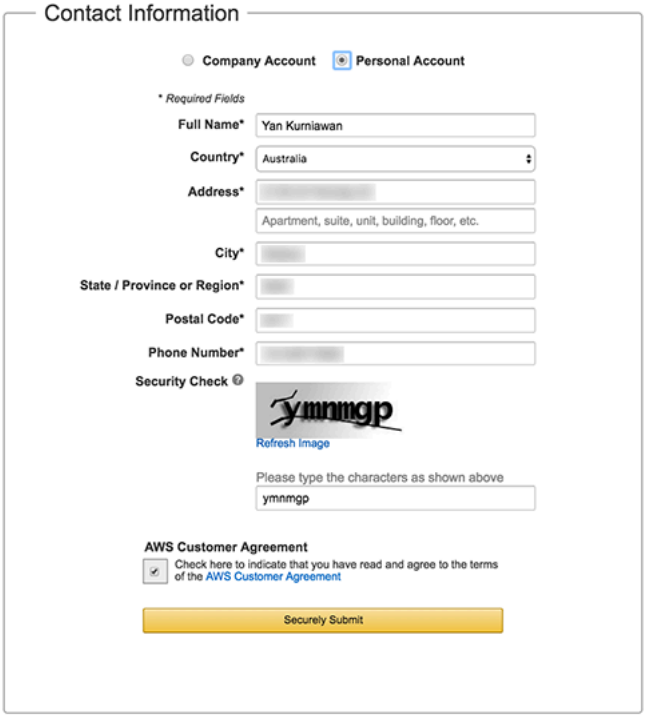
2. On the next screen, select the **I am a new user** radio button, fill in your e-mail address in the given field, and then click the **Sign in using our secure server** button



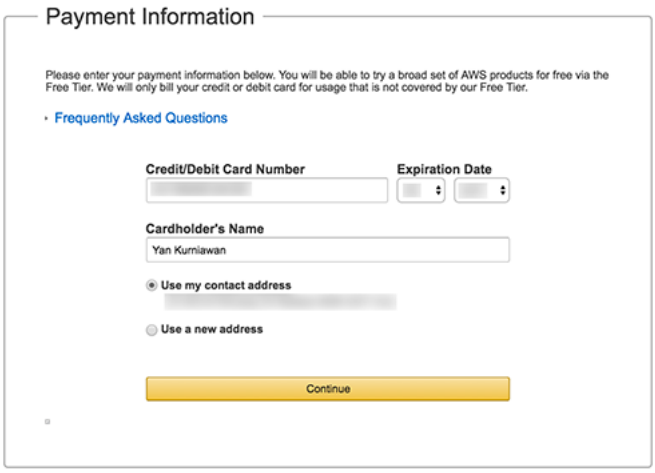
   Technically, if you have Amazon retail account, you can sign in using your Amazon.com account, but it is recommended that you set up a new AWS account.
3. On the next page, enter your name, type your e-mail address again, and enter your password (twice), then click **Create account** button



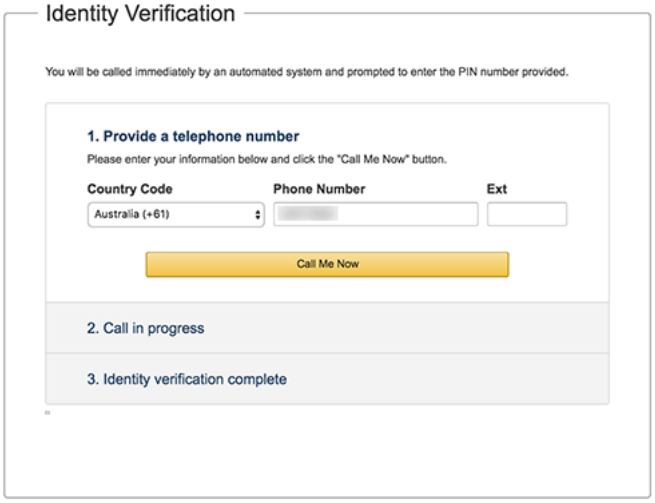
4. On the next screen, select **Personal Account** (or **Company Account** if you want to create an AWS account for your company), enter the required information on the **Contact Information** form, type the **Security Check** characters, confirm your acceptance of the AWS customer agreement, and then click the **Securely Submit** button.



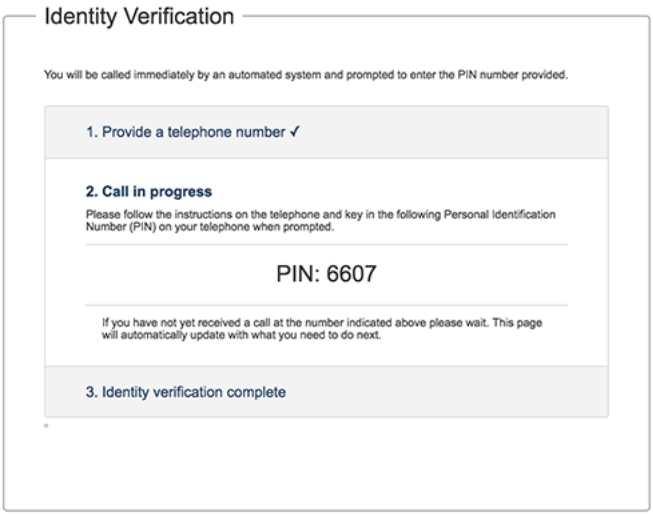
5. The next page asks you for a credit card number and your billing address information. Enter the required information and click **Continue** button.



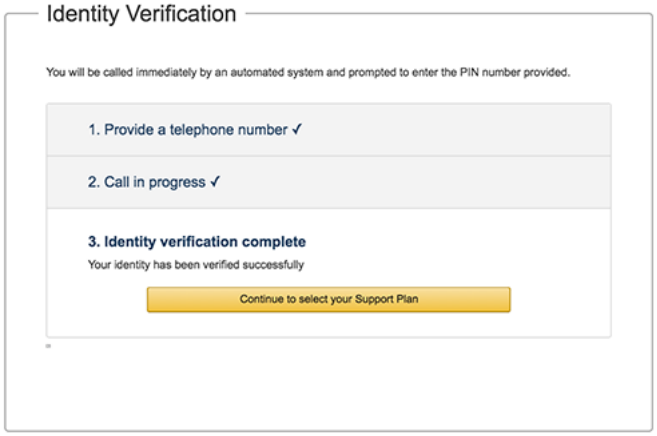
6. On the next page, Amazon wants to confirm your identity. Enter your valid phone or mobile number and click the **Call Me Now** button.



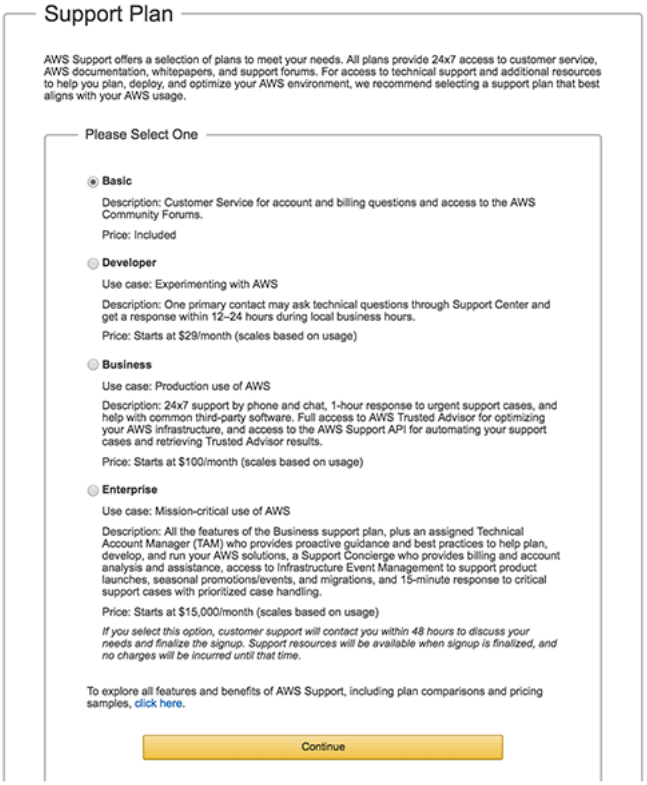
Answer the phone and enter the displayed PIN code on the telephone keypad, or you can say the PIN numbers.



After the identity verification completed successfully, click the **Continue to select your Support Plan** button.



7. On the next page, choose your support plan and click the **Continue** button.



8. Setup is now complete, you'll get an e-mail confirming your account setup.



You have given AWS your credit card information to pay for the resources you use. Be careful about how much AWS resource you use and try to understand the pricing scheme for each service.
EC2 Pricing Scheme[15]
S3 Pricing Scheme[16]



Your initial account sign-up provides free usage tier for a year. For a complete list of services that you can use for free, check out AWS Free Usage Tier[17] page.



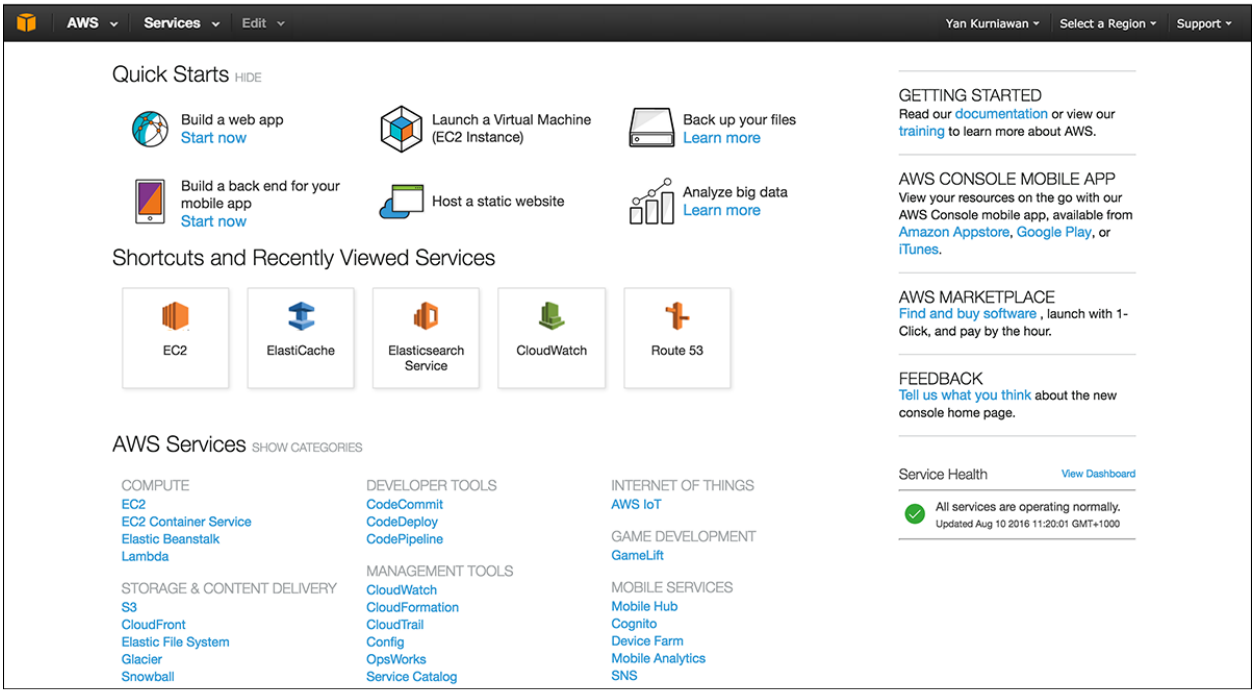
Amazon provides an online calculator[18] to estimate your monthly AWS bill.

[15] http://aws.amazon.com/ec2/pricing
[16] http://aws.amazon.com/s3/pricing
[17] http://aws.amazon.com/free
[18] http://calculator.s3.amazonaws.com/index.html

AWS Management Console

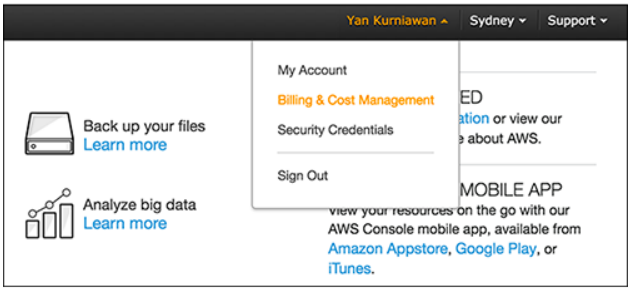
Amazon provides a web-based AWS Management Console. You can access and manage Amazon Web Services resources through a simple and intuitive web-based user interface. The AWS Management Console is a single destination for managing all your AWS resources, from EC2 instances to DynamoDB tables. Use the Console to perform any number of tasks, from deploying new applications to monitoring the health of your application.

The AWS Management Console also enables you to manage all aspects of your AWS account, including accessing your monthly spending by service, managing security credentials, or even setting up new IAM Users.

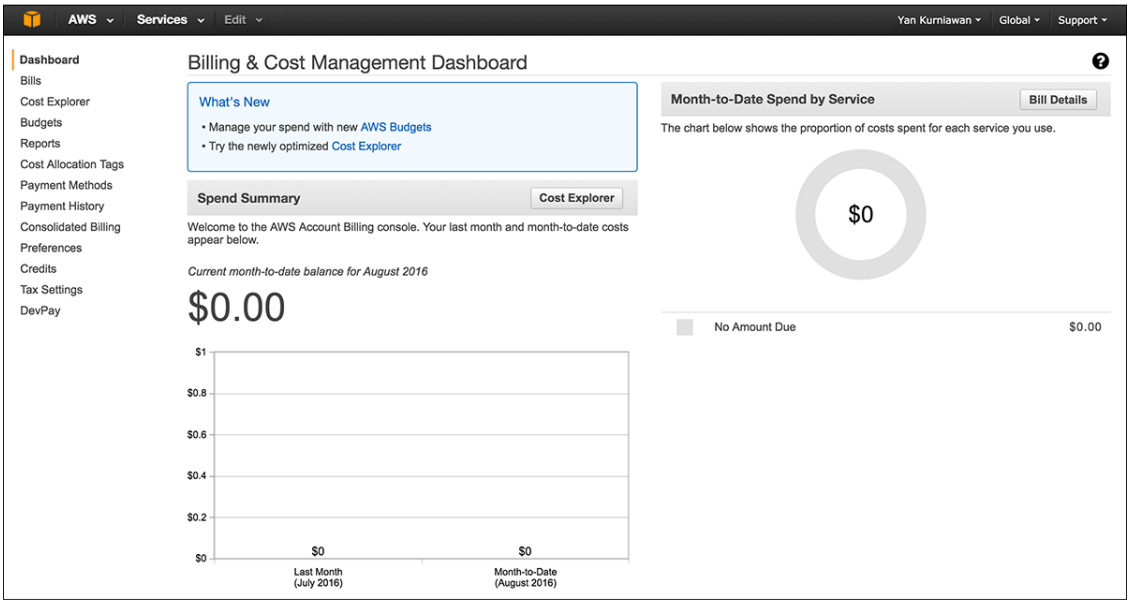


The Console Home

To see your monthly billing, you can choose **Billing & Cost Management** from the pull down menu under your account name on top right of the page.

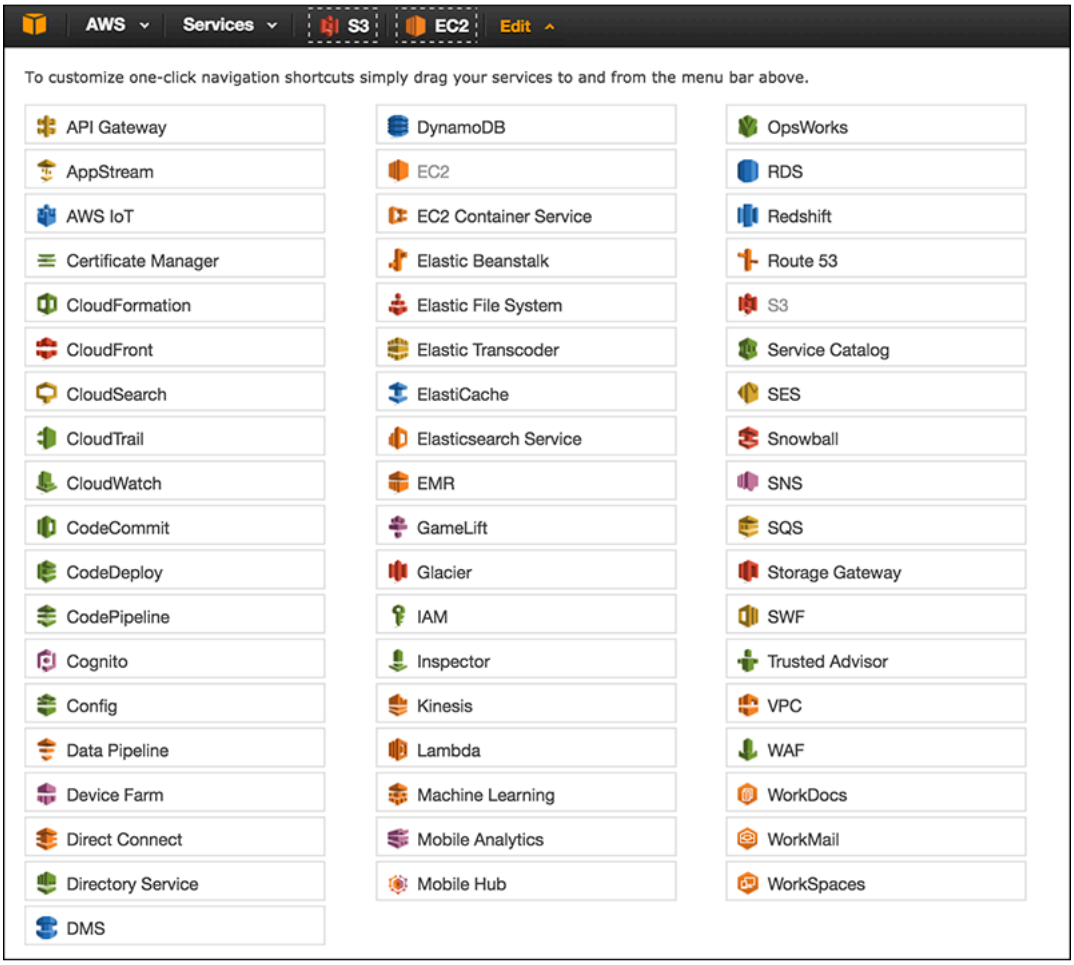


Billing and Cost Management



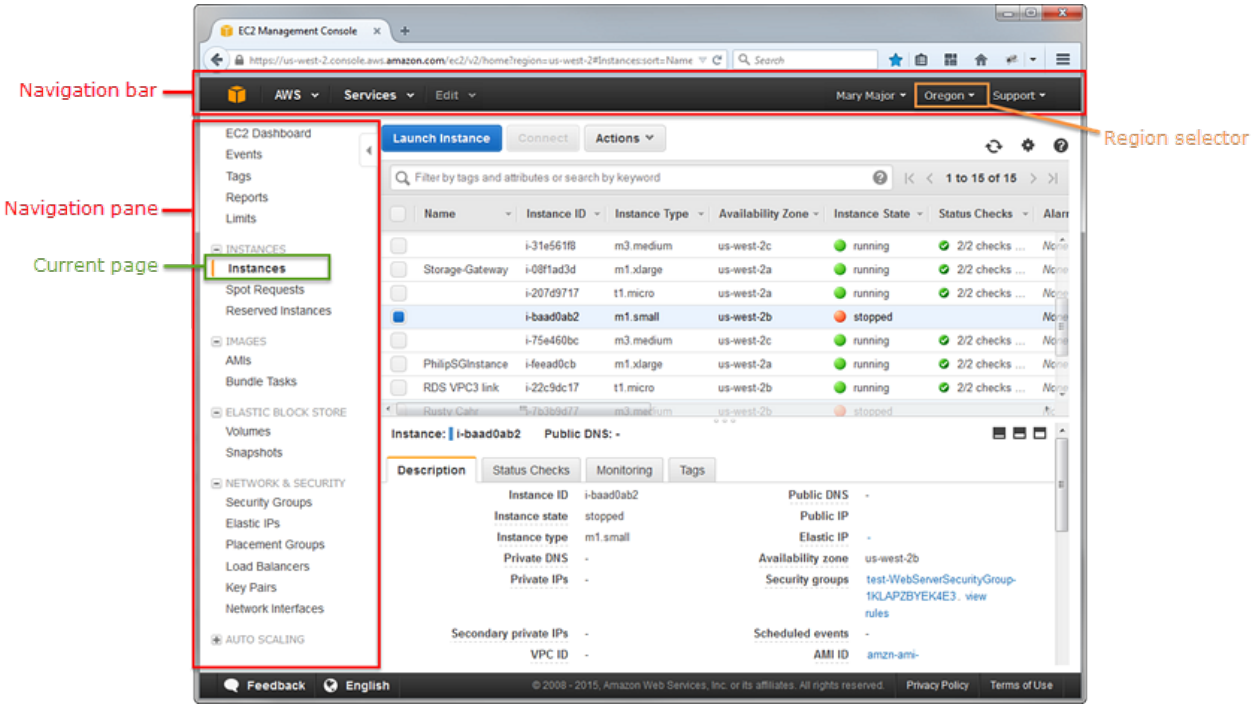
The Billing Dashboard

You can customize one-click navigation on the menu bar. Click on the **Edit** pull down menu and drag your selected service to/from the menu bar.



Customize One-click Navigation

Each service has its own console, which you can access from the AWS Management Console.



EC2 Console

For a complete guide to AWS Management Console, visit AWS Management Console Getting Started Guide[19] page.

[19] http://docs.aws.amazon.com/awsconsolehelpdocs/latest/gsg/getting-started.html

Create Your First EC2 Instance

EC2 - the Elastic Compute Cloud, is the most widely used AWS service. EC2 is the most revolutionary of the AWS services because it has transformed the way of provisioning servers. EC2 provides virtual servers in a matter of minutes with a few clicks on the AWS Management Console.

Let's create our first EC2 instance.

Create a Key Pair

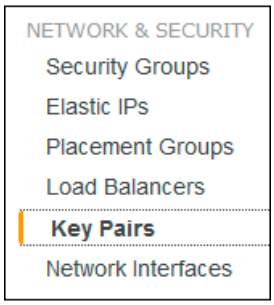
First, we need to create a key pair. AWS uses public-key cryptography to secure the login information for your instance. A Linux instance has no password; you use a key pair to log in to your instance securely. You specify the name of the key pair when you launch your instance, then provide the private key when you log in. Key pairs are specific to a region; for example, if you plan to launch an instance in the Asia Pacific (Sydney) Region, you must create a key pair for the instance in the Asia Pacific (Sydney) Region.

To create a key pair

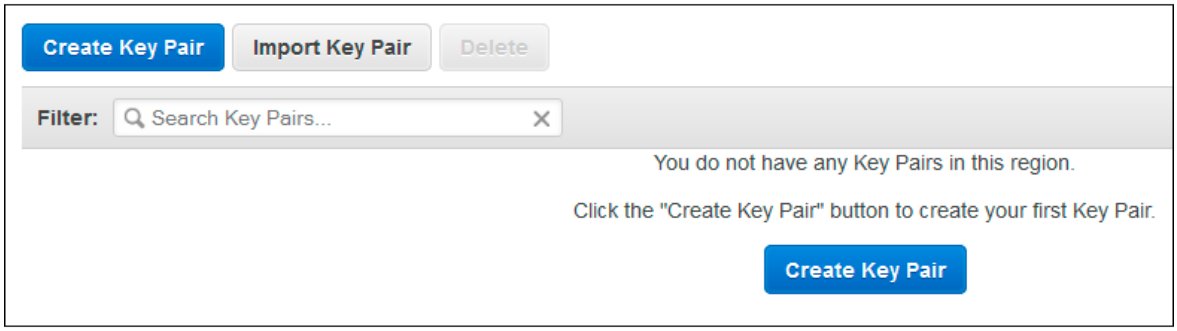
1. Open the Amazon EC2 Dashboard https://console.aws.amazon.com/ec2
2. From the navigation bar, select a region for the key pair.



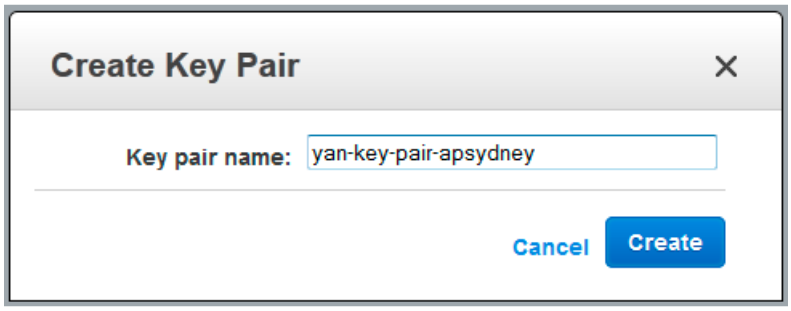
3. Click **Key Pairs** in the navigation pane.



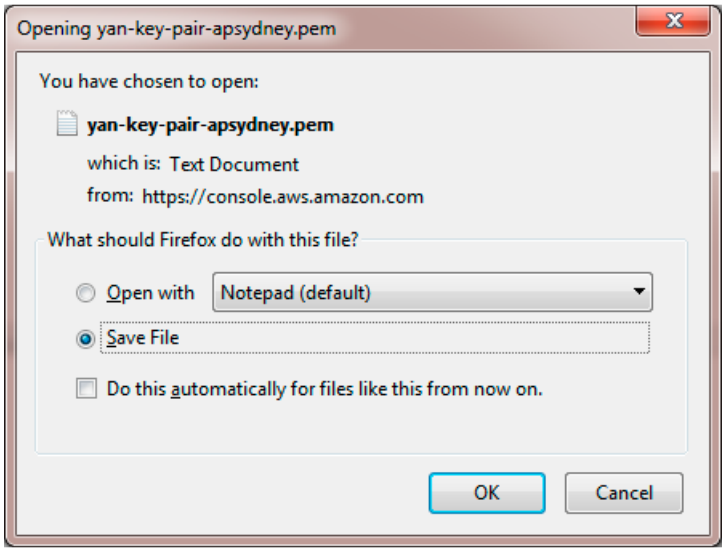
4. Click **Create Key Pair**



5. Enter a name for the new key pair in the **Key pair name** field of the **Create Key Pair** dialog box, and then click **Create**. Choose a name that is easy for you to remember, such as your name, followed by `-key-pair`, plus the region name. For example, `yan-key-pair-apsydney`.



6. If you use Google Chrome as your browser, the private key file is automatically downloaded by your browser. The base file name is the name you specified as the name of your key pair, and the file name extension is `.pem`. If you use Firefox, the browser might ask you to Open/Save the file. Save the private key file in a safe place.
   Important
   This is the only chance for you to save the private key file. You'll need to provide the name of your key pair when you launch an instance and the corresponding private key each time you connect to the instance.



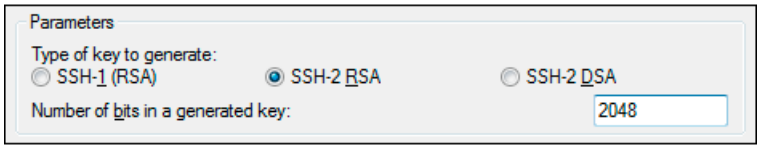
7. If you will use an SSH client on a Mac or Linux computer to connect to your Linux instance, use the `chmod` command to set the permissions of your private key file so that only you can read it.

   ```
   $ chmod 400 yan-key-pair-apsydney.pem
   ```

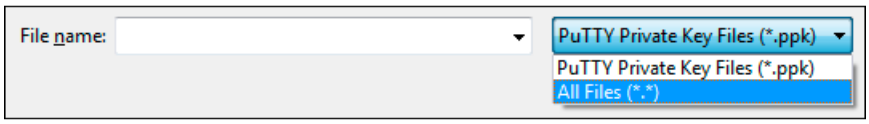
If you'll connect to your Linux instance from a computer running Windows, you can use PuTTY or MindTerm. If you use PuTTY, you'll need to install it and use the following procedure to convert the `.pem` file to a `.ppk` file.

To prepare to connect to a Linux instance from Windows using PuTTY

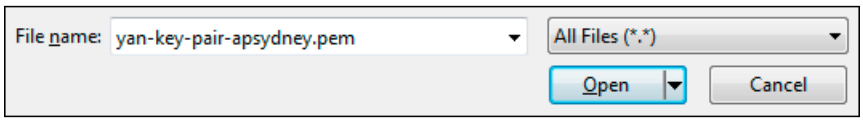
1. Download and install PuTTY from http://www.chiark.greenend.org.uk/~sgtatham/putty. Be sure to install the entire suite (Download the installer file under **A Windows installer for everything except PuTTYtel** section).
2. Start **PuTTYgen** (for example, from the **Start** menu, click **All Programs** > **PuTTY** > **PuTTYgen**).
3. Under **Type of key to generate**, select **SSH-2 RSA**.



4. Click **Load**. By default, PuTTYgen displays only files with the extension `.ppk`. To locate your `.pem` file, select the option to display files of all types.



5. Select the private key file that you created in the previous procedure and click **Open**. Click OK to dismiss the confirmation dialog box.



6. Click **Save private key**. PuTTYgen displays a warning about saving the key without a passphrase. Click **Yes**.
7. Specify the same name for the key that you used for the key pair. PuTTY automatically adds the `.ppk` file extension.

Create a Security Group

Security groups act as a firewall for associated instances, controlling both inbound and outbound traffic at the instance level. You must add rules to a security group that enable you to connect to your instance from your IP address using SSH. You can also add rules that allow inbound and outbound HTTP and HTTPS access from anywhere.

Note that if you plan to launch instances in multiple regions, you'll need to create a security group in each region.



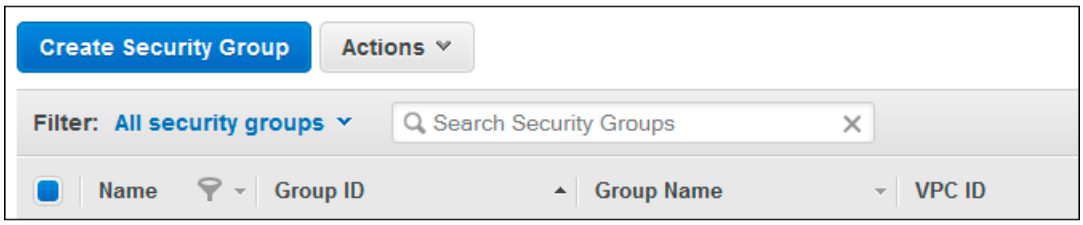
You'll need the public IP address of your local computer, which you can get using a service from Amazon AWS http://checkip.amazonaws.com. If you are connecting through an Internet service provider (ISP) or from behind a firewall without a static IP address, you need to find out the range of IP addresses used by client computers.

To create a security group

1. Open the Amazon EC2 Dashboard https://console.aws.amazon.com/ec2
2. From the navigation bar, select a region for the security group. Security groups are specific to a region; for example, if you plan to launch an instance in the Asia Pacific (Sydney) Region, you must create a security group for the instance in the Asia Pacific (Sydney) Region.
3. Click **Security Groups** in the navigation pane.

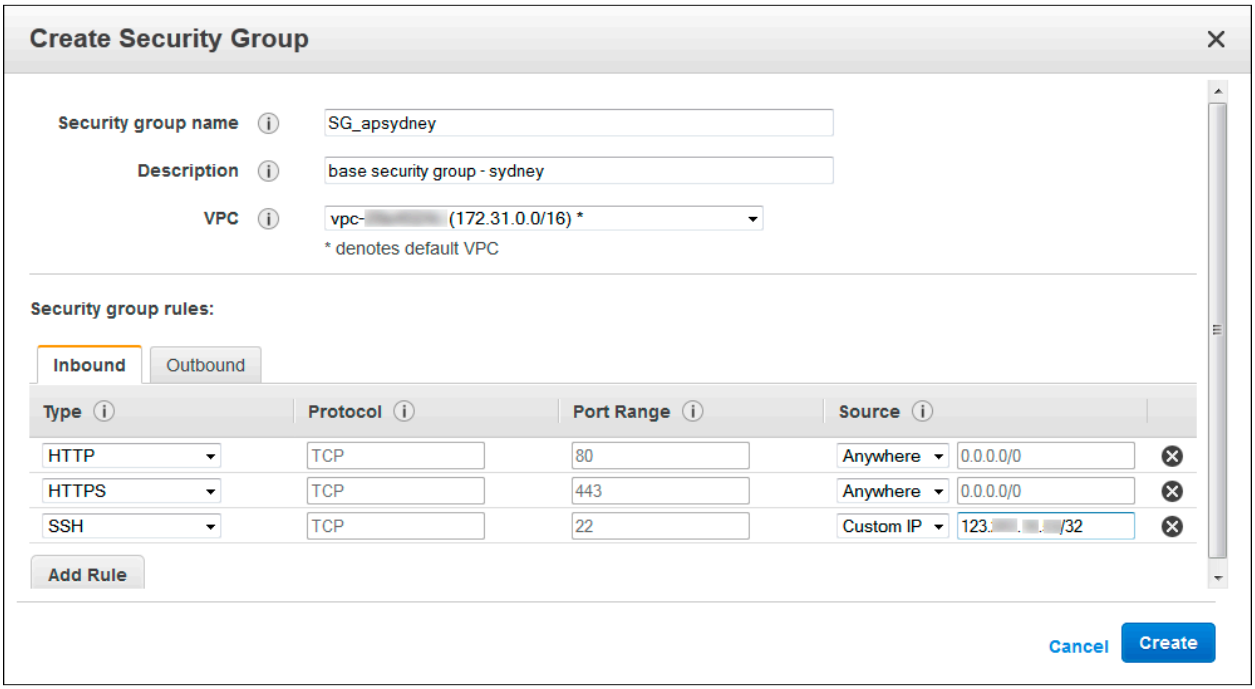


4. Click **Create Security Group**.



5. Enter a name for the new security group and a description. Choose a name that is easy for you to remember, such as SG_ plus the region name. For example, `SG_apsydney`.
   On the **Inbound** tab, create the following rules (click **Add Rule** for each new rule), and click **Create** when you're done:

   - Select **HTTP** from the **Type** list, and make sure that **Source** is set to **Anywhere** (0.0.0.0/0).
   - Select **HTTPS** from the **Type** list, and make sure that **Source** is set to **Anywhere** (0.0.0.0/0).
   - Select **SSH** from the **Type** list. In the **Source** box, ensure **Custom IP** is selected, and specify the public IP address of your computer or network in CIDR notation. To specify an individual IP address in CIDR notation, add the routing prefix /32. For example, if your IP address is 203.0.100.2, specify 203.0.100.2/32. If your company allocates addresses from a range, specify the entire range, such as 203.0.100.0/24.
     Caution
     For security reasons, Amazon doesn't recommend that you allow SSH access from all IP addresses (0.0.0.0/0) to your instance, except for testing purposes and only for a short time.



Create Security Group



You can select **My IP** in the **Source** box to allow traffic from your IP address.

The following procedure is intended to help you launch your first instance quickly and doesn't go through all possible options. For more information about the advanced options see AWS Documentation on Launching an Instance[20].

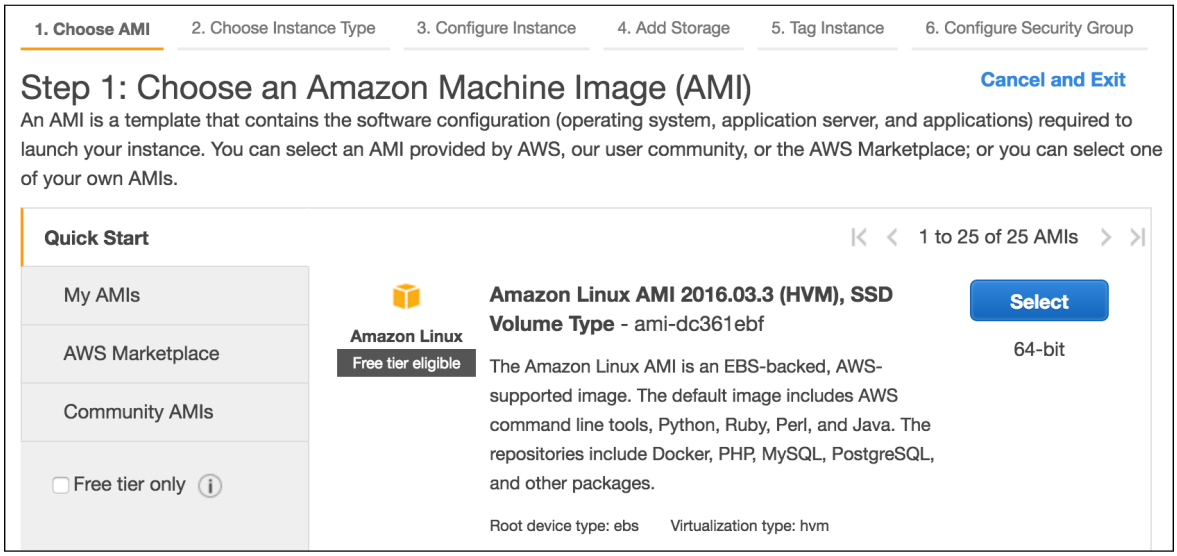
To launch an instance

1. Open the Amazon EC2 Dashboard https://console.aws.amazon.com/ec2.
2. From the console dashboard, click **Launch Instance**.

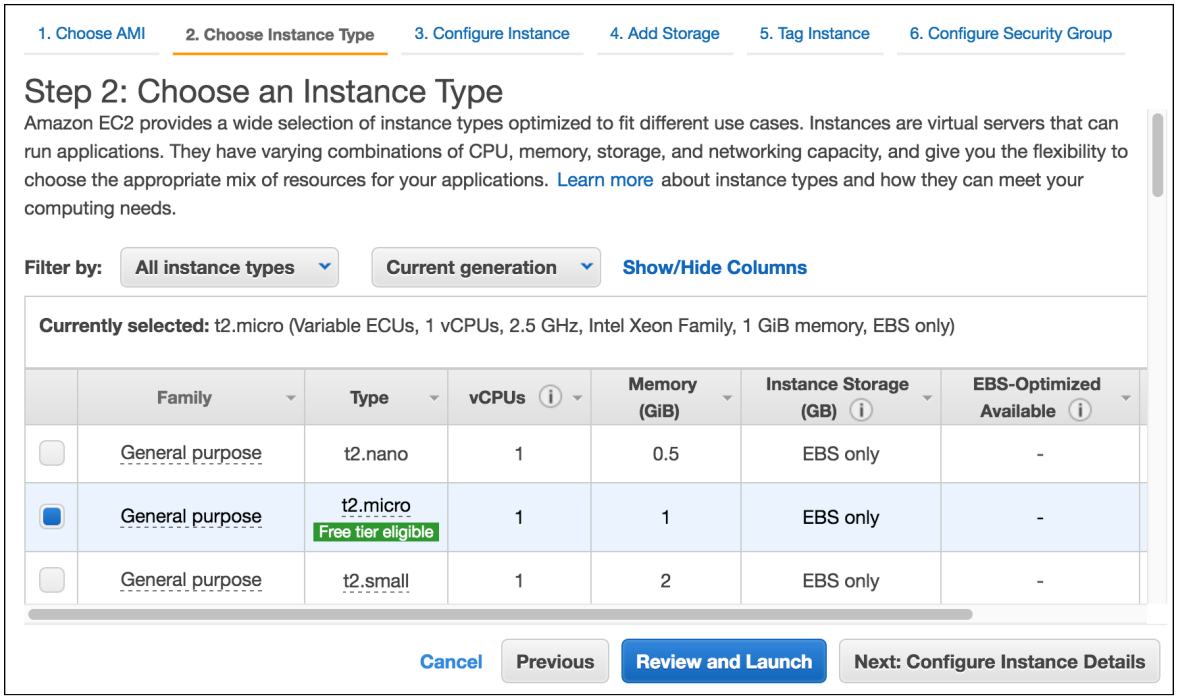


[20] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/launching-instance.html

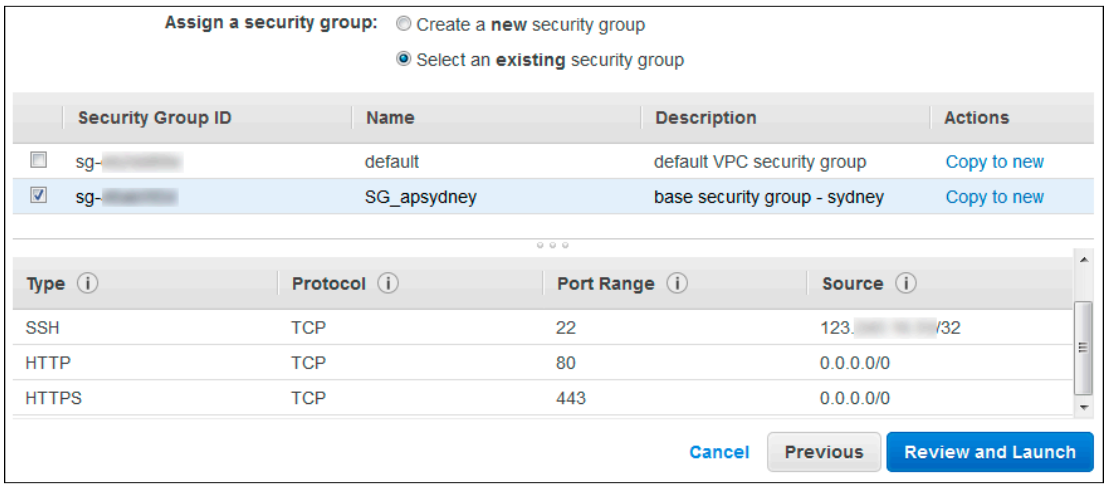
3. The **Choose an Amazon Machine Image (AMI)** page displays a list of basic configurations called Amazon Machine Images (AMIs) that serve as templates for your instance. Select the 64-bit Amazon Linux AMI. Notice that this configuration is marked "Free tier eligible". Click the **Select** button.



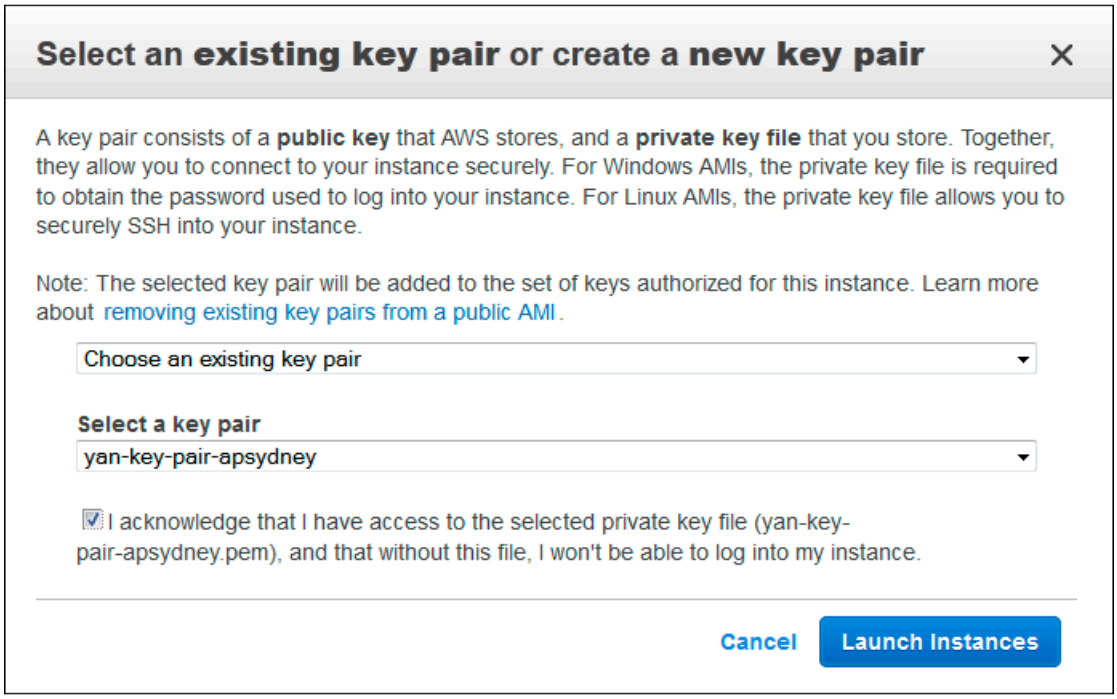
4. On the **Choose an Instance Type** page, you can select the hardware configuration of your instance. The t2.micro (or t1.micro, depends on the AMI virtualization type) instance is selected by default. Click **Review and Launch** to let the wizard complete other configuration settings for you, so you can get started quickly.



5. On the **Review Instance Launch** page, you can review the settings for your instance. Under **Security Groups**, you'll see that the wizard created and selected a security group for you. Instead, select the security group that you created when getting set up using the following steps:
   - Click **Edit security groups**.
   - On the **Configure Security Group** page, ensure the **Select an existing security group** option is selected.
   - Select your security group from the list of existing security groups, and click **Review and Launch**.

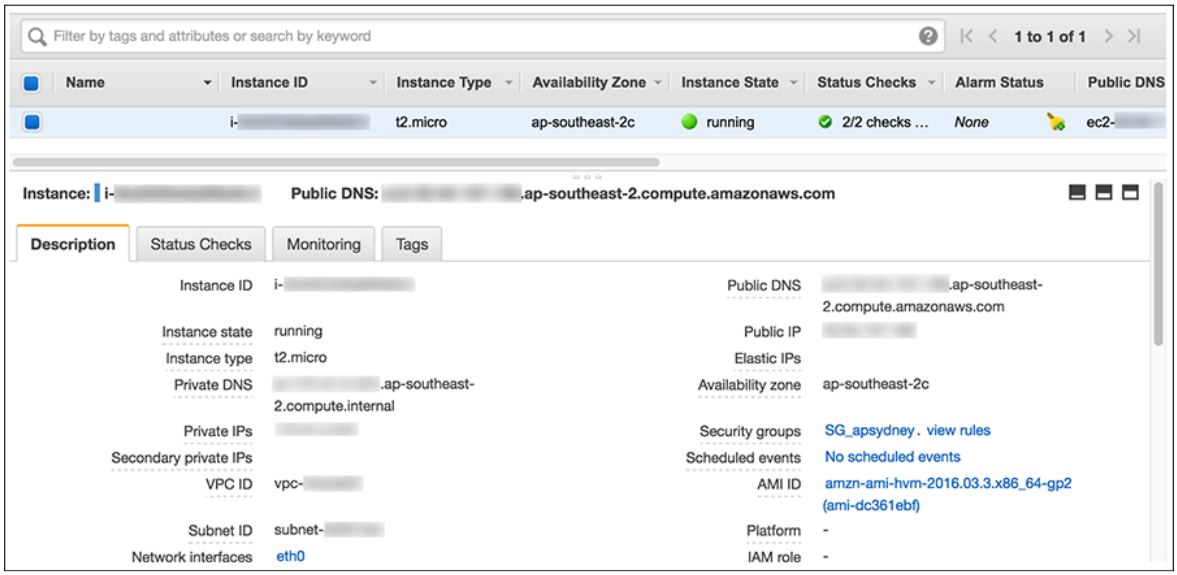


6. On the **Review Instance Launch** page, click **Launch**.
7. In the **Select an existing key pair or create a new key pair** dialog box, select **Choose an existing key pair**, then select the key pair you created when getting set up. Select the acknowledgment check box, and then click **Launch Instances**.



8. A confirmation page lets you know that your instance is launching. Click **View Instances** to close the confirmation page and return to the console.
9. On the **Instances** screen, you can view the status of your instance. It takes a short time for an instance to launch. When you launch an instance, its initial state is `pending`. After the instance starts, its state changes to `running`, and it receives a public DNS name. (If the **Public DNS** column is hidden, click the **Show/Hide** icon and select **Public DNS**).







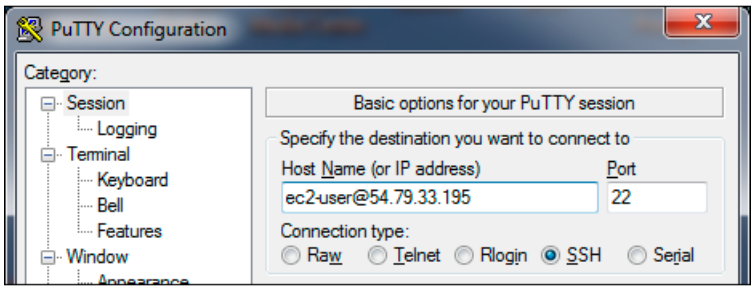
To learn more about Amazon EC2 instance type, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html

Connect to Your Instance

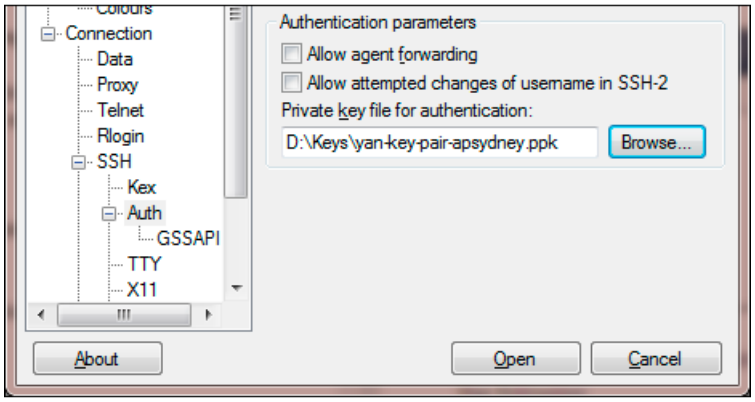
If your computer uses Windows operating system, you will need to install PuTTY to connect to your Linux EC2 instance.

To connect to your Linux instance using PuTTY

1. Start PuTTY (from the **Start** menu, click **All Programs > PuTTY > PuTTY**).
2. In the Category pane, select **Session** and complete the following fields:
   - In the **Host Name (or IP address)** box, enter ec2-user@*public_ip*. You can use either **Public IP** address or **Public DNS** name of your instance.
   - Under **Connection type**, select **SSH**.
   - Ensure that **Port** is 22



3. In the **Category** pane, expand **Connection**, expand **SSH**, and then select **Auth**. Complete the following:
   - Click **Browse**
   - Select the `.ppk` file that you generated for your key pair, and then click **Open**
   - Click **Open** to start the PuTTY session.



4. If this is the first time you have connected to this instance, PuTTY displays a security alert dialog box that asks whether you trust the host you are connecting to. Click **Yes**. A window opens and you are connected to your instance.

Connect from Mac or Linux using an SSH client

If you are using Mac or Linux computer to connect to your instance, your computer most likely includes and SSH client by default. You can check for an SSH client by typing **ssh** at the command line. If your computer doesn't recognize the command, the OpenSSH project provides a free implementation of the full suite of SSH tools. For more information, see http://www.openssh.org.

Open your command shell and run the following command:

`$ ssh -i` */path/key_pair.pem* `ec2-user@`*public_ip*



For Amazon Linux, the default user name is `ec2-user`. For RHEL5, the user name is often `root` but might be `ec2-user`. For Ubuntu, the user name is `ubuntu`. For SUSE Linux, the user name is `root`. Otherwise, check with your AMI provider.

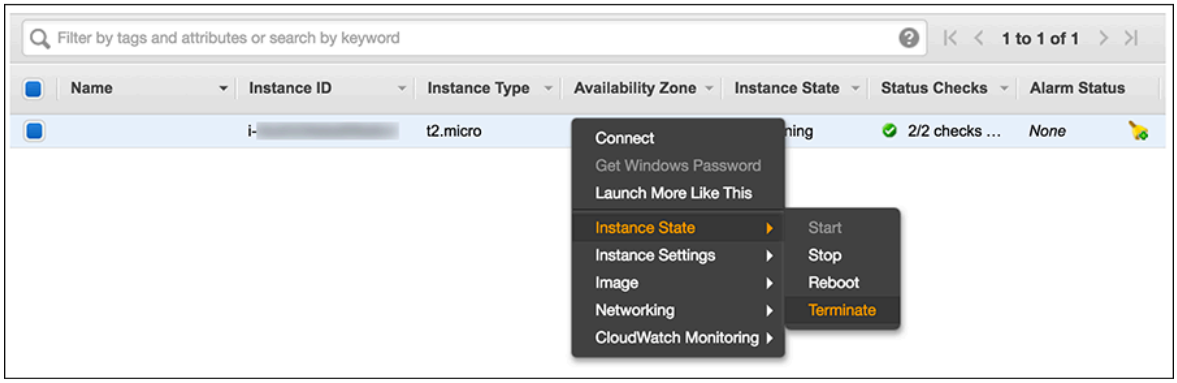
Terminate Your Instance

The purpose of the tutorial in this chapter is to show you how to launch EC2 instance from AWS Management Console, so you can get basic understanding of EC2, key pair, and security groups. After you've finished with the instance that you created for this chapter, you should clean up, terminate the instance.

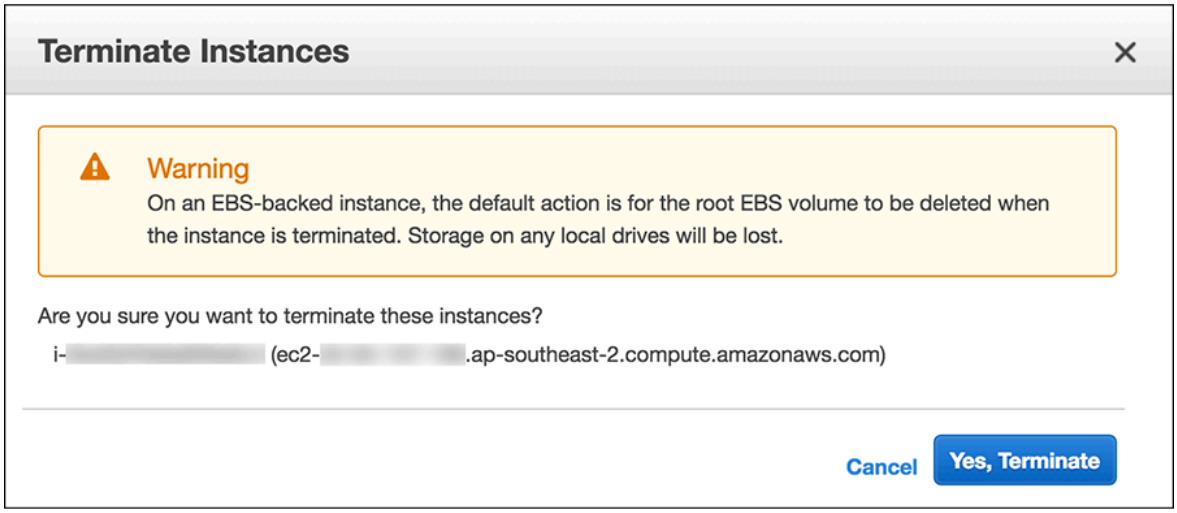
Terminating an instance effectively deletes it because you can't reconnect to an instance after you've terminated it. This differs from stopping the instance; when you stop an instance, it is shut down and you are not billed for hourly usage or data transfer. Also, you can restart a stopped instance at any time.

To terminate the instance

1. Locate your instance in the list of instances on the **Instances** page. If you can't find your instance, verify that you have selected the correct region.
2. Right-click the instance, select **Instance State** and then click **Terminate**.



3. Click **Yes**, **Terminate** when prompted for confirmation.





Most parts of this chapter is based on Amazon AWS Online Documentation. This chapter only covers the basics of AWS and EC2. If you want to learn more about EC2, see Amazon EC2 User Guide[21]. For a complete list of Amazon AWS Documentation, visit AWS Documentation[22]. You can also watch Introduction to AWS videos here: https://aws.amazon.com/training/intro_series/.

[21] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
[22] http://aws.amazon.com/documentation

Chapter 2: Getting Started with CloudFormation

What is AWS CloudFormation

AWS CloudFormation is a service that helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and AWS CloudFormation takes care of provisioning and configuring those resources for you. You don't need to individually create and configure AWS resources and figure out what's dependent on what; AWS CloudFormation handles all of that. [23]

Simplify Infrastructure Management

For a scalable web application that also includes a back-end database, you might use an Auto Scaling group, an Elastic Load Balancing load balancer, and an Amazon Relational Database Service database instance. Normally, you might use each individual service to provision these resources. And after you create the resources, you would have to configure them to work together. All these tasks can add complexity and time before you even get your application up and running.

Instead, you can create or modify an existing AWS CloudFormation template. A template describes all of your resources and their properties. When you use that template to create an AWS CloudFormation stack, AWS CloudFormation provisions the Auto Scaling group, load balancer, and database for you. After the stack has been successfully created, your AWS resources are up and running. You can delete the stack just as easily, which deletes all the resources in the stack. By using AWS CloudFormation, you easily manage a collection of resources as a single unit.

Quickly Replicate Your Infrastructure

If your application requires additional availability, you might replicate it in multiple regions so that if one region becomes unavailable, your users can still use your application in other regions. The challenge in replicating your application is that it also requires you to replicate your resources. Not only do you need to record all the resources that your application requires, but you must also provision and configure those resources in each region.

[23] http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html

When you use AWS CloudFormation, you can reuse your template to set up your resources consistently and repeatedly. Just describe your resources once and then provision the same resources over and over in multiple regions.

Easily Control and Track Changes to Your Infrastructure

In some cases, you might have underlying resources that you want to upgrade incrementally. For example, you might change to a higher performing instance type in your Auto Scaling launch configuration so that you can reduce the maximum number of instances in your Auto Scaling group. If problems occur after you complete the update, you might need to roll back your infrastructure to the original settings. To do this manually, you not only have to remember which resources were changed, you also have to know what the original settings were.

When you provision your infrastructure with AWS CloudFormation, the AWS CloudFormation template describes exactly what resources are provisioned and their settings. Because these templates are text files, you simply track differences in your templates to track changes to your infrastructure, similar to the way developers control revisions to source code. For example, you can use a version control system with your templates so that you know exactly what changes were made, who made them, and when. If at any point you need to reverse changes to your infrastructure, you can use a previous version of your template.

AWS CloudFormation Concepts

When you use AWS CloudFormation, you work with *templates* and *stacks*. You create templates to describe your AWS resources and their properties. Whenever you create a stack, AWS CloudFormation provisions the resources that are described in your template. [24]

Templates

An AWS CloudFormation template is a JSON (JavaScript Object Notation) or YAML (YAML Ain't Markup Language) formatted text file. You can save these files with any extension, such as .json, .yaml, .template, or .txt. AWS CloudFormation uses these templates as blueprints for building your AWS resources. For example, in a template, you can describe an Amazon EC2 instance, such as the instance type, the AMI ID, block device mappings, and its Amazon EC2 key pair name. Whenever you create a stack, you also specify a template that AWS CloudFormation uses to create whatever you described in the template.



Throughout this book we will use YAML format to create our template. It's human friendly, easier to write and read than JSON.

[24] http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-whatis-concepts.html

For example, if you created a stack with the following template, AWS CloudFormation provisions an instance with an `ami-2f726546` AMI ID, `t1.micro` instance type, `testkey` key pair name, and an Amazon EBS volume.

```
---
AWSTemplateFormatVersion: "2010-09-09"
Description: A sample template
  Resources:
    MyEC2Instance:
      Type: "AWS::EC2::Instance"
      Properties:
        ImageId: "ami-2f726546"
        InstanceType: t1.micro
        KeyName: testkey
        BlockDeviceMappings:
          -
            DeviceName: /dev/sdm
            Ebs:
              VolumeType: io1
              Iops: 200
              DeleteOnTermination: false
              VolumeSize: 20
```

You can also specify multiple resources in a single template and configure these resources to work together. For example, you can modify the previous template to include an Elastic IP (EIP) and associate it with the Amazon EC2 instance, as shown in the following example:

```
---

AWSTemplateFormatVersion: "2010-09-09"
Description: A sample template
  Resources:
    MyEC2Instance:
      Type: "AWS::EC2::Instance"
      Properties:
        ImageId: "ami-2f726546"
        InstanceType: t1.micro
        KeyName: testkey
        BlockDeviceMappings:
          -
            DeviceName: /dev/sdm
            Ebs:
```

```
16              VolumeType: io1
17              Iops: 200
18              DeleteOnTermination: false
19              VolumeSize: 20
20    MyEIP:
21      Type: AWS::EC2::EIP
22      Properties:
23        InstanceId: !Ref MyEC2Instance
```

The previous templates are centered around a single Amazon EC2 instance; however, AWS CloudFormation templates have additional capabilities that you can use to build complex sets of resources and reuse those templates in multiple contexts. For example, you can add input parameters whose values are specified when you create an AWS CloudFormation stack. In other words, you can specify a value like the instance type when you create a stack instead of when you create the template, making the template easier to reuse in different situations.

Stacks

When you use AWS CloudFormation, you manage related resources as a single unit called a stack. You create, update, and delete a collection of resources by creating, updating, and deleting stacks. All the resources in a stack are defined by the stack's AWS CloudFormation template. Suppose you created a template that includes an Auto Scaling group, Elastic Load Balancing load balancer, and an Amazon Relational Database Service (Amazon RDS) database instance. To create those resources, you create a stack by submitting the template that you created, and AWS CloudFormation provisions all those resources for you.You can work with stacks by using the AWS CloudFormation console, API, or AWS CLI.

Change Sets

If you need to make changes to the running resources in a stack, you update the stack. Before making changes to your resources, you can generate a change set, which is summary of your proposed changes.

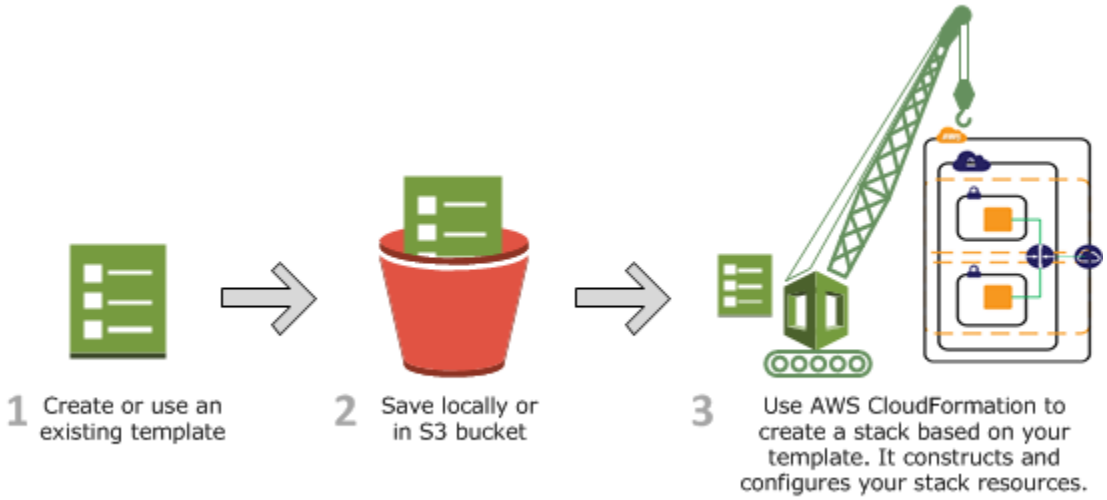
Change sets allow you to see how your changes might impact your running resources, especially for critical resources, before implementing them.

For example, if you change the name of an Amazon RDS database instance, AWS CloudFormation will create a new database and delete the old one. You will lose the data in the old database unless you've already backed it up. If you generate a change set, you will see that your change will cause your database to be replaced, and you will be able to plan accordingly before you update your stack.

How Does AWS CloudFormation Work?

When you create a stack, AWS CloudFormation makes underlying service calls to AWS to provision and configure your resources. Note that AWS CloudFormation can perform only actions that you have permission to do. For example, to create EC2 instances by using AWS CloudFormation, you need permissions to create instances. You'll need similar permissions to terminate instances when you delete stacks with instances. You use AWS Identity and Access Management (IAM) to manage permissions.

The calls that AWS CloudFormation makes are all declared by your template. For example, suppose you have a template that describes an EC2 instance with a `t1.micro` instance type. When you use that template to create a stack, AWS CloudFormation calls the Amazon EC2 create instance API and specifies the instance type as `t1.micro`. The following diagram summarizes the AWS CloudFormation workflow for creating stacks.



CloudFormation Workflow

1. You can design an AWS CloudFormation template (a JSON-formatted or YAML-formatted document) in AWS CloudFormation Designer or write one in a text editor. You can also choose to use a provided template. The template describes the resources you want and their settings.
2. Save the template locally or in an S3 bucket. If you created a template, save it with any file extension like .json, .yaml, .template, or .txt.
3. Create an AWS CloudFormation stack by specifying the location of your template file , such as a path on your local computer or an Amazon S3 URL. If the template contains parameters, you can specify input values when you create the stack. Parameters enable you to pass in values to your template so that you can customize your resources each time you create a stack. You can create stacks by using the AWS CloudFormation console, API, or AWS CLI.

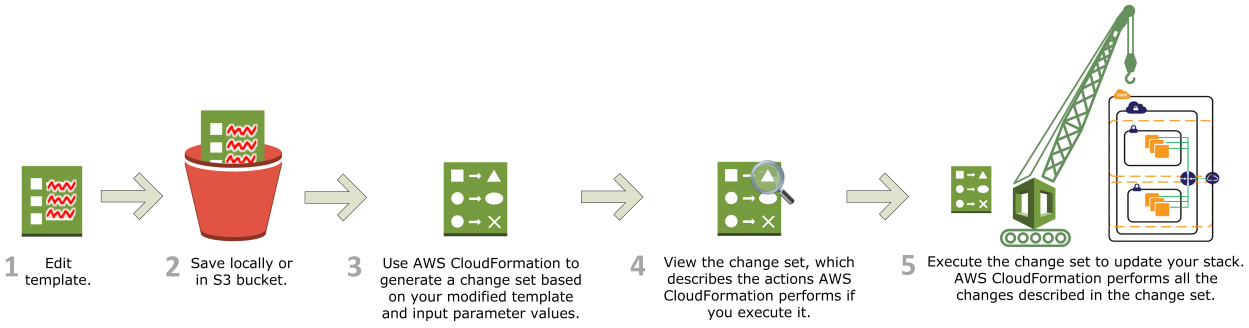


If you specify a template file stored locally, AWS CloudFormation uploads it to an S3 bucket in your AWS account. AWS CloudFormation creates a bucket for each region in which you upload a template file. The buckets are accessible to anyone with Amazon Simple Storage Service (Amazon S3) permissions in your AWS account. If a bucket created by AWS CloudFormation is already present, the template is added to that bucket.
You can use your own bucket and manage its permissions by manually uploading templates to Amazon S3. Then whenever you create or update a stack, specify the Amazon S3 URL of a template file.

AWS CloudFormation provisions and configures resources by making calls to the AWS services that are described in your template. After all the resources have been created, AWS CloudFormation reports that your stack has been created. You can then start using the resources in your stack. If stack creation fails, AWS CloudFormation rolls back your changes by deleting the resources that it created.

Updating a Stack with Change Sets

When you need to update your stack's resources, you can modify the stack's template. You don't need to create a new stack and delete the old one. To update a stack, create a change set by submitting a modified version of the original stack template, different input parameter values, or both. AWS CloudFormation compares the modified template with the original template and generates a change set. The change set lists the proposed changes. After reviewing the changes, you can execute the change set to update your stack or you can create a new change set. The following diagram summarizes the workflow for updating a stack.



Updating a Stack Workflow



Updates can cause interruptions. Depending on the resource and properties that you are updating, an update might interrupt or even replace an existing resource.

1. You can modify an AWS CloudFormation stack template by using AWS CloudFormation Designer or a text editor. For example, if you want to change the instance type for an EC2

instance, you would change the value of the `InstanceType` property in the original stack's template.
2. Save the AWS CloudFormation template locally or in an S3 bucket.
3. Create a change set by specifying the stack that you want to update and the location of the modified template, such as a path on your local computer or an Amazon S3 URL. If the template contains parameters, you can specify values when you create the change set.
4. View the change set to check that AWS CloudFormation will perform the changes that you expect. For example, check whether AWS CloudFormation will replace any critical stack resources. You can create as many change sets as you need until you have included the changes that you want.
5. Execute the change set that you want to apply to your stack. AWS CloudFormation updates your stack by updating only the resources that you modified and signals that your stack has been successfully updated. If the stack updates fails, AWS CloudFormation rolls back changes to restore the stack to the last known working state.



If you specify a template that is stored on your local computer, AWS CloudFormation automatically uploads your template to an S3 bucket in your AWS account.



Change sets don't indicate whether your stack update will be successful. For example, a change set doesn't check if you will surpass an account limit, if you're updating a resource that doesn't support updates, or if you have insufficient permissions to modify a resource, all of which can cause a stack update to fail.

Deleting a Stack

When you delete a stack, you specify the stack to delete, and AWS CloudFormation deletes the stack and all the resources in that stack. You can delete stacks by using the AWS CloudFormation console, API, or AWS CLI.

If you want to delete a stack but want to retain some resources in that stack, you can use a *deletion policy* to retain those resources.

After all the resources have been deleted, AWS CloudFormation signals that your stack has been successfully deleted. If AWS CloudFormation cannot delete a resource, the stack will not be deleted. Any resources that haven't been deleted will remain until you can successfully delete the stack.

Template Anatomy

Templates include several major sections. The `Resources` section is the only required section. Some sections in a template can be in any order. However, as you build your template, it might be helpful to use the logical ordering of the following list, as values in one section might refer to values from a previous section. The list gives a brief overview of each section. [25]

`Format Version` (optional)
: Specifies the AWS CloudFormation template version that the template conforms to. The template format version is not the same as the API or WSDL (Web Service Description Language) version. The template format version can change independently of the API and WSDL versions.

`Description` (optional)
: A text string that describes the template. This section must always follow the template format version section.

`Metadata` (optional)
: Objects that provide additional information about the template.

`Parameters` (optional)
: Specifies values that you can pass in to your template at runtime (when you create or update a stack). You can refer to parameters in the Resources and Outputs sections of the template.

`Mappings` (optional)
: A mapping of keys and associated values that you can use to specify conditional parameter values, similar to a lookup table.You can match a key to a corresponding value by using the `Fn::FindInMap` intrinsic function in the Resources and Outputs section.

`Conditions` (optional)
: Defines conditions that control whether certain resources are created or whether certain resource properties are assigned a value during stack creation or update. For example, you could conditionally create a resource that depends on whether the stack is for a production or test environment.

`Resources` (required)
: Specifies the stack resources and their properties, such as an Amazon Elastic Compute Cloud instance or an Amazon Simple Storage Service bucket. You can refer to resources in the Resources and Outputs sections of the template.

`Outputs` (optional)
: Describes the values that are returned whenever you view your stack's properties. For example, you can declare an output for an Amazon S3 bucket name and then call the `aws cloudformation describe-stacks` AWS CLI command to view the name.

[25] http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-anatomy.html

A YAML-formatted template fragment:

```
---
AWSTemplateFormatVersion: "<version date>"

Description:
  <String>

Metadata:
  <template metadata>

Parameters:
  <set of parameters>

Mappings:
  <set of mappings>

Conditions:
  <set of conditions>

Resources:
  <set of resources>

Outputs:
  <set of outputs>
```

Format Version

The `AWSTemplateFormatVersion` section (optional) identifies the capabilities of the template. The latest template format version is `2010-09-09` and is currently the only valid value.



The template format version is not the same as the API or WSDL version. The template format version can change independently of the API and WSDL versions.

The value for the template format version declaration must be a literal string. You cannot use a parameter or function to specify the template format version. If you don't specify a value, AWS CloudFormation assumes the latest template format version.

The following snippet is an example of a valid template format version declaration:

```
AWSTemplateFormatVersion: "2010-09-09"
```

Description

The `Description` section (optional) enables you to include arbitrary comments about your template. The `Description` must follow the `AWSTemplateFormatVersion` section.

The value for the description declaration must be a literal string that is between 0 and 1024 bytes in length. You cannot use a parameter or function to specify the description. The following snippet is an example of a description declaration:

```
Description: >
  Here are some
  details about
  the template.
```

Metadata

You can use the optional `Metadata` section to include arbitrary JSON/YAML objects that provide details about the template. For example, you can include template implementation details about specific resources, as shown in the following snippet:

```
Metadata:
  Instances:
    Description: "Information about the instances"
  Databases:
    Description: "Information about the databases"
```

Metadata Keys

Some AWS CloudFormation features retrieve settings or configuration information that you define from the `Metadata` section. You define this information in the following AWS CloudFormation-specific metadata keys:

`AWS::CloudFormation::Init`
: Defines configuration tasks for the cfn-init helper script. This script is useful for configuring and installing applications on EC2 instances.

`AWS::CloudFormation::Interface`
: Defines the grouping and ordering of input parameters when they are displayed in the AWS CloudFormation console. By default, the AWS CloudFormation console alphabetically sorts parameters by their logical ID.

`AWS::CloudFormation::Designer`
: Describes how your resources are laid out in AWS CloudFormation Designer (Designer). Designer automatically adds this information when you use it create and update templates.

Parameters

You can use the optional `Parameters` section to pass values into your template when you create a stack. With parameters, you can create templates that are customized each time you create a stack. Each parameter must contain a value when you create a stack. You can specify a default value to make the parameter optional so that you don't need to pass in a value when creating a stack; AWS CloudFormation will use the default value.

You can have a maximum of 60 parameters in an AWS CloudFormation template.

For each parameter, you must declare a logical name, which must be alphanumeric and unique among all logical names within the template. After you declare the parameter's logical name, you can specify the parameter's properties. You must declare parameters as one of following types: `String`, `Number`, `CommaDelimitedList`, or an AWS-specific type. For String, Number, and AWS-specific parameter types, you can define constraints that AWS CloudFormation uses to validate the value of the parameter.

AWS-specific parameter types are AWS values such as Amazon EC2 key pair names and VPC IDs. AWS CloudFormation validates these parameter values against existing values in users' AWS accounts. AWS-specific parameter types are helpful in catching invalid values at the start of creating or updating a stack.



For sensitive parameter values (such as passwords), set the `NoEcho` property to `true`. That way, whenever anyone describes your stack, the parameter value is shown as asterisks (*****).

```
1  Parameters:
2    <ParameterLogicalID>:
3      Type: <DataType>
4      <ParameterProperty>: <value>
```

Parameter's Properties

`AllowedPattern`

A regular expression that represents the patterns you want to allow for `String` types.

Required: No

`AllowedValues`

An array containing the list of values allowed for the parameter.

Required: No

`ConstraintDescription`

A string that explains the constraint when the constraint is violated. For example, without a constraint description, a parameter that has an allowed pattern of `[A-Za-z0-9]+` displays the following error message when the user specifies an invalid value:

```
"Malformed input-Parameter MyParameter must match pattern [A-Za-z0-9]+"
```

By adding a constraint description, such as `must only contain upper- and lowercase letters, and numbers`, you can display a customized error message:

```
"Malformed input-Parameter MyParameter must only contain upper and lower case letters and numbers"
```

Required: No

`Default`

A value of the appropriate type for the template to use if no value is specified when a stack is created. If you define constraints for the parameter, you must specify a value that adheres to those constraints.

Required: No

`Description`

A string of up to 4000 characters that describes the parameter.

Required: No

`MaxLength`

An integer value that determines the largest number of characters you want to allow for `String` types.

Required: No

`MaxValue`

A numeric value that determines the largest numeric value you want to allow for `Number` types.

Required: No

`MinLength`

An integer value that determines the smallest number of characters you want to allow for `String` types.

Required: No

`MinValue`

A numeric value that determines the smallest numeric value you want to allow for `Number` types.

Required: No

`NoEcho`
: Whether to mask the parameter value whenever anyone makes a call that describes the stack. If you set the value to true, the parameter value is masked with asterisks (*****).

  Required: No

`Type` The data type for the parameter (DataType).

  Required:Yes

Type Property

You can specify the following values for the `Type` property:

`String`
: A literal string.

  For example, users could specify `"MyUserName"`.

`Number`
: An integer or float. AWS CloudFormation validates the parameter value as a number; however, when you use the parameter elsewhere in your template (for example, by using the `Ref` intrinsic function), the parameter value becomes a string.

  For example, users could specify `"8888"`.

`List<Number>`
: An array of integers or floats that are separated by commas. AWS CloudFormation validates the parameter value as numbers; however, when you use the parameter elsewhere in your template (for example, by using the `Ref` intrinsic function), the parameter value becomes a list of strings.

  For example, users could specify `"80,20"`, and a `Ref` will result in `["80","20"]`.

`CommaDelimitedList`
: An array of literal strings that are separated by commas. The total number of strings should be one more than the total number of commas. Also, each member string is space trimmed.

  For example, users could specify `"test,dev,prod"`, and a `Ref` will result in `["test","dev","prod"]`.

AWS-specific parameter types
: For AWS-specific parameter types, template users must specify existing AWS values that are in their account.

AWS-specific Parameter Types

AWS CloudFormation supports the following AWS-specific parameter types:

`AWS::EC2::AvailabilityZone::Name`
: An Availability Zone, such as `us-west-2a`.

`AWS::EC2::Image::Id`
: An Amazon EC2 image ID, such as `ami-ff527ecf`. Note that the AWS CloudFormation console won't show a drop-down list of values for this parameter type.

`AWS::EC2::Instance::Id`
: An Amazon EC2 instance ID, such as `i-1e731a32`.

`AWS::EC2::KeyPair::KeyName`
: An Amazon EC2 key pair name.

`AWS::EC2::SecurityGroup::GroupName`
: An EC2-Classic or default VPC security group name, such as `my-sg-abc`.

`AWS::EC2::SecurityGroup::Id`
: A security group ID, such as `sg-a123fd85`.

`AWS::EC2::Subnet::Id`
: A subnet ID, such as `subnet-123a351e`.

`AWS::EC2::Volume::Id`
: An Amazon EBS volume ID, such as `vol-3cdd3f56`.

`AWS::EC2::VPC::Id`
: A VPC ID, such as `vpc-a123baa3`.

`AWS::Route53::HostedZone::Id`
: An Amazon Route 53 hosted zone ID, such as `Z23YXV4OVPL04A`.

`List<AWS::EC2::AvailabilityZone::Name>`
: An array of Availability Zones for a region, such as `us-west-2a, us-west-2b`.

`List<AWS::EC2::Image::Id>`
: An array of Amazon EC2 image IDs, such as `ami-ff527ecf, ami-e7527ed7`. Note that the AWS CloudFormation console won't show a drop-down list of values for this parameter type.

`List<AWS::EC2::Instance::Id>`
: An array of Amazon EC2 instance IDs, such as `i-1e731a32, i-1e731a34`.

List<AWS::EC2::SecurityGroup::GroupName>
An array of EC2-Classic or default VPC security group names, such as `my-sg-abc, my-sg-def`.

List<AWS::EC2::SecurityGroup::Id>
An array of security group IDs, such as `sg-a123fd85, sg-b456fd85`.

List<AWS::EC2::Subnet::Id>
An array of subnet IDs, such as `subnet-123a351e, subnet-456b351e`.

List<AWS::EC2::Volume::Id>
An array of Amazon EBS volume IDs, such as `vol-3cdd3f56, vol-4cdd3f56`.

List<AWS::EC2::VPC::Id>
An array of VPC IDs, such as `vpc-a123baa3, vpc-b456baa3`.

List<AWS::Route53::HostedZone::Id>
An array of Amazon Route 53 hosted zone IDs, such as `Z23YXV4OVPL04A, Z23YXV4OVPL04B`.

AWS CloudFormation validates input values for these types against existing values in a user's account. For example, with the `AWS::EC2::VPC::Id` type, a user must enter an existing VPC ID that is in her account and in the region in which she is creating the stack.

Mappings

The optional `Mappings` section matches a key to a corresponding set of named values. For example, if you want to set values based on a region, you can create a mapping that uses the region name as a key and contains the values you want to specify for each specific region. You use the `Fn::FindInMap` intrinsic function to retrieve values in a map.

You cannot include parameters, pseudo parameters, or intrinsic functions in the `Mappings` section.

The Mappings section consists of the key name `Mappings`. The keys and values in mappings must be literal strings. The following example shows a `Mappings` section containing a single mapping named `Mapping01` (the logical name).

Within a mapping, each map is a key followed by another mapping. The key identifies a map of name-value pairs and must be unique within the mapping. The name can contain only alphanumeric characters (A-Za-z0-9).

```
Mappings:
  Mapping01:
    Key01:
      Name: Value01
    Key02:
      Name: Value02
    Key03:
      Name: Value03
```

Conditions

The optional `Conditions` section includes statements that define when a resource is created or when a property is defined. For example, you can compare whether a value is equal to another value. Based on the result of that condition, you can conditionally create resources. If you have multiple conditions, separate them with commas.

You might use conditions when you want to reuse a template that can create resources in different contexts, such as a test environment versus a production environment. In your template, you can add an `EnvironmentType` input parameter, which accepts either `prod` or `test` as inputs. For the production environment, you might include Amazon EC2 instances with certain capabilities; however, for the test environment, you want to use reduced capabilities to save money. With conditions, you can define which resources are created and how they're configured for each environment type.

Conditions are evaluated based on input parameter values that you specify when you create or update a stack. Within each condition, you can reference another condition, a parameter value, or a mapping. After you define all your conditions, you can associate them with resources and resource properties in the `Resources` and `Outputs` sections of a template.

At stack creation or stack update, AWS CloudFormation evaluates all the conditions in your template before creating any resources. Any resources that are associated with a true condition are created. Any resources that are associated with a false condition are ignored.



During a stack update, you cannot update conditions by themselves.You can update conditions only when you include changes that add, modify, or delete resources.

To conditionally create resources, you must include statements in at least three different sections of a template:

`Parameters` section
: Define the input values that you want to evaluate in your conditions. Conditions will result in true or false based on values from these input parameter.

`Conditions` section
: Define conditions by using the intrinsic condition functions. These conditions determine when AWS CloudFormation creates the associated resources.

`Resources` and `Outputs` sections
: Associate conditions with the resources or outputs that you want to conditionally create. AWS CloudFormation creates entities that are associated with a true condition and ignores entities that are associated with a false condition. Use the `Condition` key and a condition's logical ID to associate it with a resource or output. To conditionally specify a property, use the `Fn::If` function.

```
Conditions:
  <Logical ID>:
    <Intrinsic function>
```

Condition Intrinsic Functions You can use the following intrinsic functions to define conditions:
- `Fn::And`
- `Fn::Equals`
- `Fn::If`
- `Fn::Not`
- `Fn::Or`

Resources

The required `Resources` section declare the AWS resources that you want as part of your stack, such as an Amazon EC2 instance or an Amazon S3 bucket.You must declare each resource separately; however, you can specify multiple resources of the same type. If you declare multiple resources, separate them with commas.

```
Resources:
  <Logical ID>:
    Type: <Resource type>
    Properties:
      <Set of properties>
```

Resource Fields

Logical ID
: The logical ID must be alphanumeric (A-Za-z0-9) and unique within the template. You use the logical name to reference the resource in other parts of the template. For example, if you

want to map an Amazon Elastic Block Store to an Amazon EC2 instance, you reference the logical IDs to associate the block stores with the instance.

In addition to the logical ID, certain resources also have a physical ID, which is the actual assigned name for that resource, such as an Amazon EC2 instance ID or an Amazon S3 bucket name. You use the physical IDs to identify resources outside of AWS CloudFormation templates, but only after the resources have been created. For example, you might give an Amazon EC2 instance resource a logical ID of `MyEC2Instance`; but when AWS CloudFormation creates the instance, AWS CloudFormation automatically generates and assigns a physical ID (such as `i-28f9ba55`) to the instance. You can use this physical ID to identify the instance and view its properties (such as the DNS name) by using the Amazon EC2 console. For resources that support custom names, you can assign your own names (physical IDs) to help you quickly identify resources. For example, you can name an Amazon S3 bucket that stores logs as `MyPerformanceLogs`.

Resource type

The resource type identifies the type of resource that you are declaring. For example, the `AWS::EC2::Instance` declares an Amazon EC2 instance.

Resource properties

Resource properties are additional options that you can specify for a resource. For example, for each Amazon EC2 instance, you must specify an AMI ID for that instance.

Outputs

The optional `Outputs` section declares output values that you can import into other stacks (to create cross-stack references), return in response (to describe stack calls), or view on the AWS CloudFormation console. For example, you can output the S3 bucket name for a stack to make the bucket easier to find.



During a stack update, you cannot update outputs by themselves. You can update outputs only when you include changes that add, modify, or delete resources.

The `Outputs` section consists of the key name `Outputs`, followed by a single colon. Braces enclose all output declarations. If you declare multiple outputs, they are delimited by commas. You can declare a maximum of 60 outputs in an AWS CloudFormation template.

```
Outputs:
  <Logical ID>:
    Description: <Information about the value>
    Value: <Value to return>
    Export:
      Name: <Value to export>
```

Output Fields

Logical ID
: An identifier for this output. The logical ID must be alphanumeric (A-Za-z0-9) and unique within the template.

Description (optional)
: A `String` type up to 4K in length describing the output value.

Value (required)
: The value of the property that is returned by the `aws cloudformation describe-stacks` command. The value of an output can include literals, parameter references, pseudo-parameters, a mapping value, or intrinsic functions.

Export (optional)
: The name of the resource output to be exported for a cross-stack reference.

  The following restrictions apply to cross-stack references:

  - For each AWS account, `Export` names must be unique within a region.
  - You can't create cross-stack references across different regions. You can use the intrinsic function `Fn::ImportValue` only to import values that have been exported within the same region.
  - For outputs, the value of the `Name` property of an Export can't use functions (`Ref` or `GetAtt`) that depend on a resource. Similarly, the `ImportValue` function can't include functions (`Ref` or `GetAtt`) that depend on a resource.
  - You can't delete a stack if another stack references one of its outputs.
  - You can't modify or remove the output value as long as it's referenced by another stack.

You can use intrinsic functions to customize the `Name` value of an export. The following examples use the `Fn::Join` function.

```
Export:
  Name: !Join [ ":", [ !Ref "AWS::StackName", AccountVPC ] ]
```

You can conditionally create an output by associating a condition with it.

Your First Template

Let's create our first template. The following template will create an EC2 instance. We will create the CloudFormation stack and then we will try to modify the template and update the stack.

If you haven't created a key pair, you can go to EC2 console and create a new key pair to be assigned to the EC2 instance we're going to create.

You also need an AMI id. I will use a vanilla Amazon Linux AMI for this example.

```
$ mkdir /home/yan/aws-cloudformation
$ cd /home/yan/aws-cloudformation
$ vi ec2-1.yml
```

```
1  AWSTemplateFormatVersion: 2010-09-09
2  Description: Template for EC2 instance
3  Resources:
4    MyEC2Instance:
5      Type: AWS::EC2::Instance
6      Properties:
7        ImageId: ami-dc361ebf #A comment -- This is an Amazon Linux AMI
8        InstanceType: t2.micro
9        KeyName: yan-key-pair-apsydney #your key pair name
```



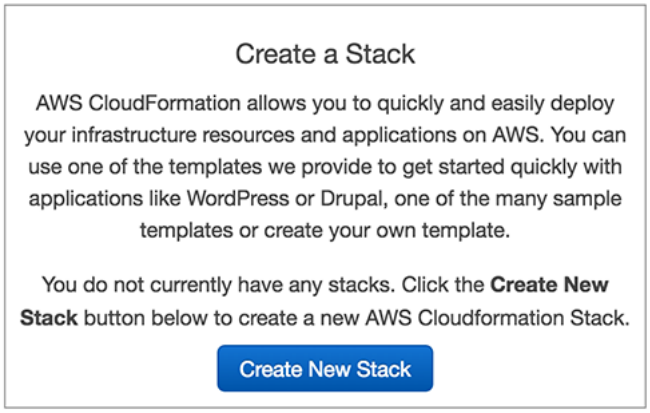
To find out what Resource Properties are available for certain Resource, go to http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html and select the Resource. For example http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html to see available Properties for EC2 Instance resource.

To create a CloudFormation stack:

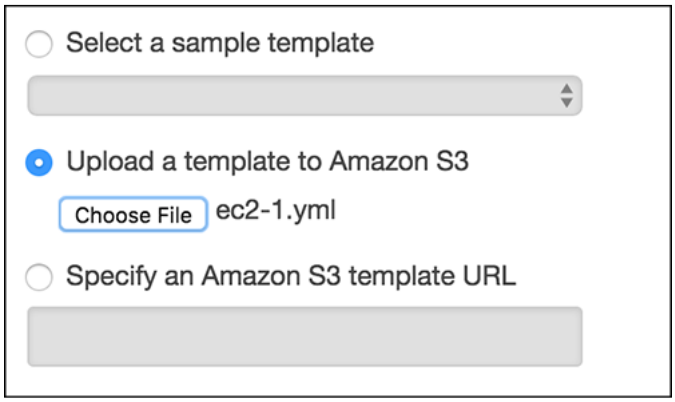
1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation/. If you are not logged in to the AWS Management Console yet, you need to log in before using the AWS CloudFormation console.
2. Create a new stack by using one of the following options:
   - Click **Create Stack**. This is the only option if you have a currently running stack.



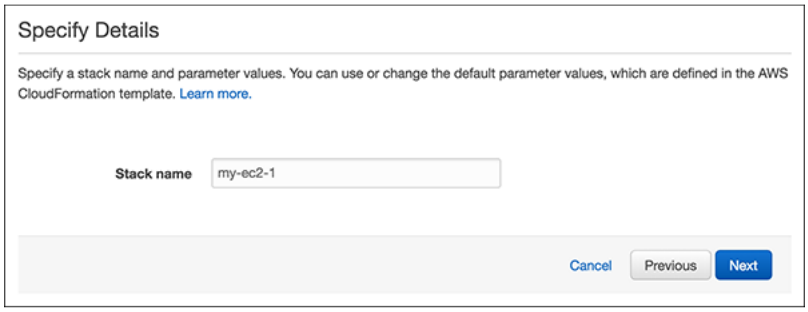
- Click **Create New Stack** in the CloudFormation Stacks main window. This option is visible only if you have no running stacks.



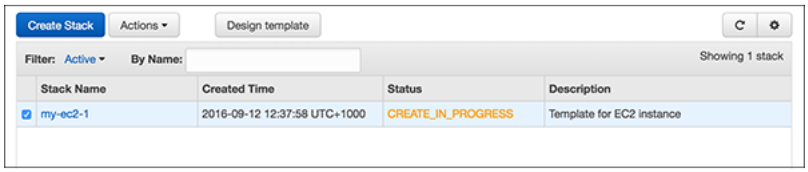
3. On the **Select Template** page, choose **Upload a template to Amazon S3** and select an AWS CloudFormation template on your local computer. Click the **Choose File** button to select the template file that you want to upload. Locate and select the `ec2-1.yml` template.



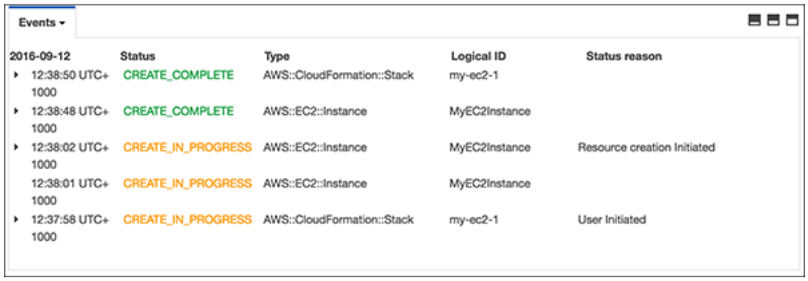
4. Click **Next**
5. On the **Specify Details** page, specify the stack name. The stack name is an identifier that helps you find a particular stack from a list of stacks. A stack name can contain only alphanumeric characters (case sensitive) and hyphens. It must start with an alphabetic character and cannot be longer than 128 characters.



6. Click **Next**
7. On the **Options** page, click **Next**
8. On the **Review** page, click **Create**
9. Your stack appears in the list of AWS CloudFormation stacks, with a status of `CREATE_IN_PROGRESS`



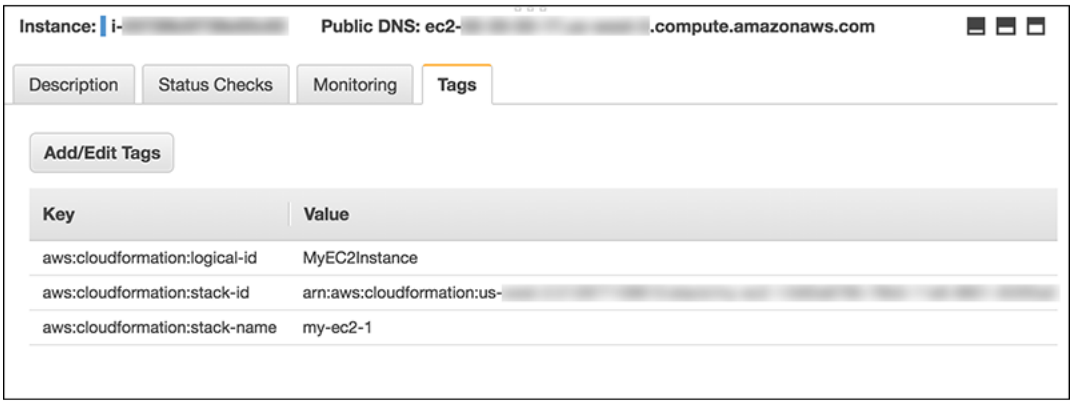
10. While your stack is being created (or afterward), you can use the stack detail pane to view your stack's events, data, or resources. AWS CloudFormation automatically refreshes stack events every minute. By viewing stack creation events, you can understand the sequence of events that lead to your stack's creation (or failure, if you are debugging your stack).





The template creates an EC2 instance in the default VPC, default subnet, and uses the default security group. To find out more about default VPC, default subnet, and default security group, see http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/default-vpc.html.

Go to EC2 console https://console.aws.amazon.com/ec2/ to see the new instance launched. AWS CloudFormation adds tags to the resources it creates. Select the new instance and select the **Tags** tab to see the tags added by CloudFormation.


Instance: i-
Public DNS: ec2-
.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Add/Edit Tags
Key
Value
aws:cloudformation:logical-id
MyEC2Instance
aws:cloudformation:stack-id
arn:aws:cloudformation:us-
aws:cloudformation:stack-name
my-ec2-1

Updating Stack

When you need to make changes to a stack's settings or change its resources, you update the stack instead of deleting it and creating a new stack. For example, if you have a stack with an EC2 instance, you can update the stack to change the instance's AMI ID. When you update a stack, you submit changes, such as new input parameter values or an updated template. AWS CloudFormation compares the changes you submit with the current state of your stack and updates only the changed resources. [26]

When you submit an update, AWS CloudFormation updates resources based on differences between what you submit and the stack's current template. Resources that have not changed run without disruption during the update process. For updated resources, AWS CloudFormation uses one of the following update behaviors:

Update with No Interruption
: AWS CloudFormation updates the resource without disrupting operation of that resource and without changing the resource's physical ID. For example, if you update any property on an `AWS::CloudTrail::Trail` resource, AWS CloudFormation updates the trail without disruption.

Updates with Some Interruption
: AWS CloudFormation updates the resource with some interruption and retains the physical ID. For example, if you update certain properties on an `AWS::EC2::Instance` resource, the instance might have some interruption while AWS CloudFormation and Amazon EC2 reconfigure the instance.

Replacement
: AWS CloudFormation recreates the resource during an update, which also generates a new physical ID. AWS CloudFormation creates the replacement resource first, changes references from other dependent resources to point to the replacement resource, and then deletes the old resource. For example, if you update the Engine property of an `AWS::RDS::DBInstance` resource type, AWS CloudFormation creates a new resource and replaces the current DB instance resource with the new one.

The method AWS CloudFormation uses depends on which property you update for a given resource type.

Depending on the update behaviour, you can decide when to modify resources to reduce the impact of these changes on your application. In particular, you can plan when resources must be replaced during an update. For example, if you update the Port property of an `AWS::RDS::DBInstance` resource type, AWS CloudFormation replaces the DB instance by creating a new DB instance with the updated port setting and deletes the old DB instance. Before the update, you might plan to do

[26] http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks.html

the following to prepare for the database replacement:
- Take a snapshot of the current databases.
- Prepare a strategy for how applications that use that DB instance will handle an interruption while the DB instance is being replaced.
- Ensure that the applications that use that DB instance take into account the updated port setting and any other updates you have made.
- Use the DB snapshot to restore the databases on the new DB instance.

This example is not exhaustive; it's meant to give you an idea of the things to plan for when a resource is replaced during an update.

Let's try to modify our simple template, add some stack parameters and then update the stack.

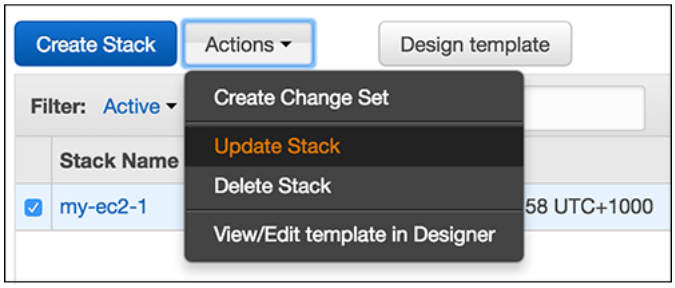
```
$ vi ec2-2.yml
```

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for EC2 instance with parameters
Parameters:
  Subnet:
    Description: The EC2 instance will be launched in this subnet
    Type: AWS::EC2::Subnet::Id
  SecurityGroups:
    Description: Assign Security Groups for the instance
    Type: List<AWS::EC2::SecurityGroup::Id>
  AMIid:
    Description: Image Id
    Type: AWS::EC2::Image::Id
  Keypair:
    Description: Key pair name
    Type: AWS::EC2::KeyPair::KeyName
  InstanceType:
    Type: String
    Default: t2.small
    AllowedValues:
      - t2.micro
      - t2.small
      - t2.medium
    Description: Enter t2.micro, t2.small, or t2.medium. Default is t2.micro.
Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref AMIid
      InstanceType: !Ref InstanceType
```

```
30        KeyName: !Ref Keypair
31        SecurityGroupIds: !Ref SecurityGroups
32        SubnetId: !Ref Subnet
```

To update a CloudFormation stack:

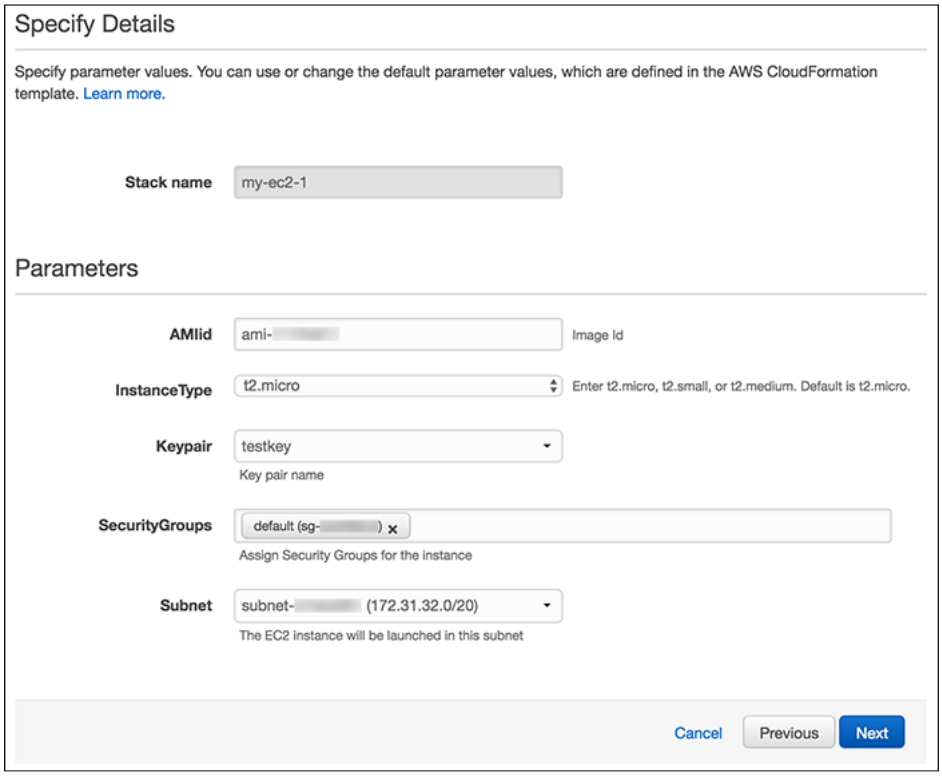
1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Select **Actions - Update Stack**



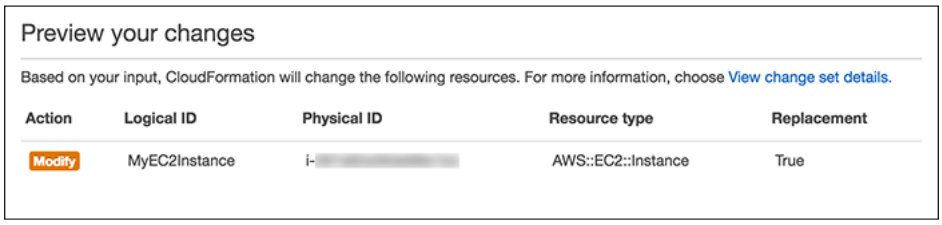
3. On the **Select Template** page, choose **Upload a template to Amazon S3** and select the `ec2-2.yml` AWS CloudFormation template on your local computer. Click the **Choose File** button to select the template file that you want to upload. Locate and select the `ec2-2.yml` template.



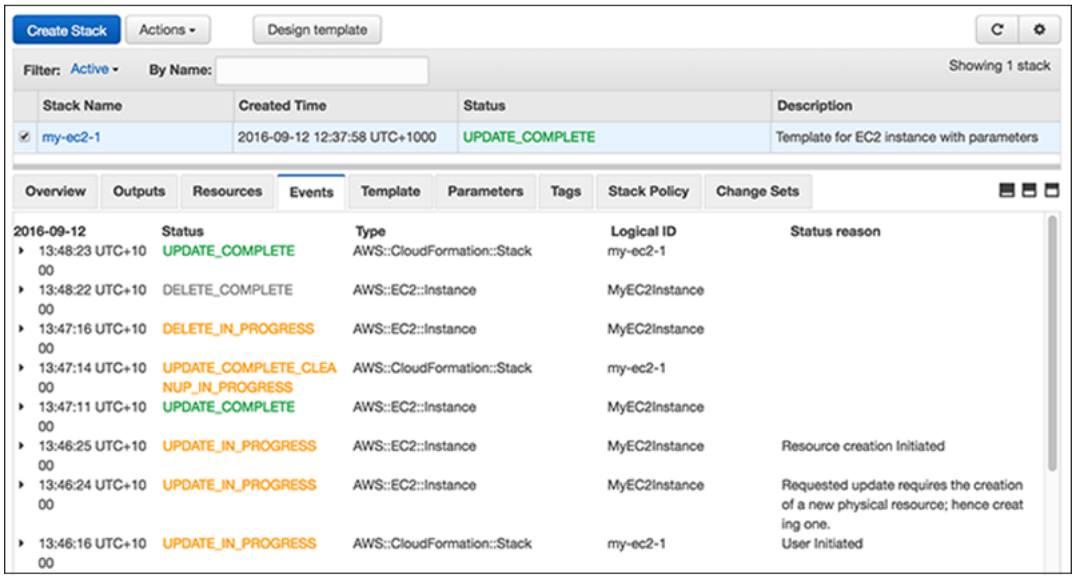
4. Click **Next**
5. On the **Specify Details** page, specify parameters that are defined in the stack template. When you create stacks that contain AWS-specific parameter types, the AWS CloudFormation console provides drop-down lists of valid values for those parameters. Enter the AMI id, select the instance type, key pair, security group, and subnet. Then click **Next**.



6. On the **Options** page, click **Next**
7. In the **Review** section, check that you submitted the correct information, such as the correct parameter values. In the **Preview your changes** section, check that AWS CloudFormation will make all the changes that you expect. For example, you can check that AWS CloudFormation adds, removes, and modifies the resources that you intended to add, remove, or modify. AWS CloudFormation generates this preview by creating a change set for the stack.



   You can see that the stack update will replace the EC2 instance. CloudFormation will create a new EC2 instance with the new parameters, then it will delete existing instance.
8. Click **Update**. Your stack enters the `UPDATE_IN_PROGRESS` state. After it has finished updating, the state is set to `UPDATE_COMPLETE`. If the stack update fails, AWS CloudFormation automatically rolls back changes, and sets the state to `UPDATE_ROLLBACK_COMPLETE`.





You can monitor the progress of a stack update by viewing the stack's events. The console's **Events** tab displays each major step in the creation and update of the stack sorted by the time of each event with latest events on top. The start of the stack update process is marked with an `UPDATE_IN_PROGRESS` event for the stack.

Canceling a Stack Update

After a stack update has begun, you can cancel the stack update if the stack is still in the `UPDATE_IN_PROGRESS` state. After an update has finished, you cannot cancel it. You can, however, update a stack again with any previous settings.

If you cancel a stack update, the stack is rolled back to the stack configuration that existed prior to initiating the stack update.

To cancel a stack update by using the console:

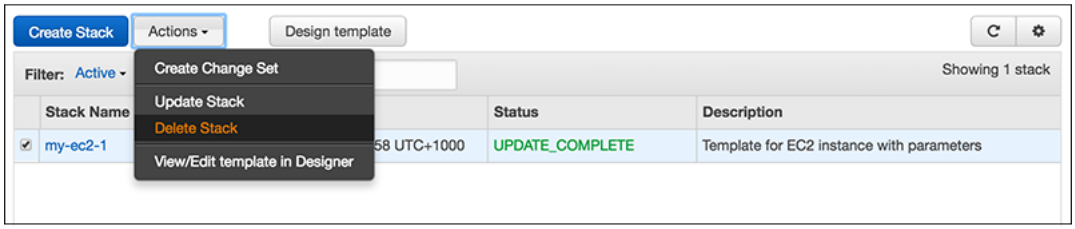
1. From the list of stacks in the AWS CloudFormation console, select the stack that is currently being updated (its state must be `UPDATE_IN_PROGRESS`).
2. Choose **Actions** and then **Cancel Update**.
3. To continue canceling the update, click **Yes, Cancel Update** when prompted. Otherwise, click **Cancel** to resume the update.

The stack proceeds to the `UPDATE_ROLLBACK_IN_PROGRESS` state. After the update cancellation is complete, the stack is set to `UPDATE_ROLLBACK_COMPLETE`.

Deleting Stack

To delete the CloudFormation stack:

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Select **Actions - Delete Stack**



3. Click **Yes, Delete** when prompted for confirmation.

