



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

About the Author	i
Preface	ii
What this book covers	iii
Who this book is for	iv
What you need for this book	iv
Conventions	iv
Example Code Files	v
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

CONTENTS

Updating Stack	53
Canceling a Stack Update	57
Deleting Stack	58
Chapter 3: The IAM Stack	59
Introduction to IAM	59
Getting Started with IAM	62
IAM Best Practices	65
Creating Your First IAM Admin User and Group	67
IAM Stack	68
AWS::IAM::Role	73
AWS::IAM::Policy	76
AWS::IAM::InstanceProfile	78
Chapter 4: The Network Stack	80
Introduction to VPC	80
The Default VPC	82
VPC and Subnet Basics	84
Getting Started with VPCs and Subnets	89
NAT Gateways	97
Security Groups for Your VPC	102
Network Stack	105
Route53 Hosted Zones	128
AWS::EC2::VPC	130
AWS::EC2::Subnet	133
AWS::EC2::RouteTable	135
AWS::EC2::Route	137
AWS::EC2::SubnetRouteTableAssociation	140
AWS::EC2::InternetGateway	142
AWS::EC2::VPCCGatewayAttachment	144
AWS::EC2::NatGateway	146
Chapter 5: The Application Stack	147
Introduction to Auto Scaling	147
How Auto Scaling Works	149
The Architecture	151
Getting Started with Auto Scaling	153
Scaling the Size of Your Auto Scaling Group	160
Load Balance Your Auto Scaling Group	163
Getting Started with ELB	165
Attaching Load Balancer to ASG	170
Scheduled Scaling	174
Dynamic Scaling	176

CONTENTS

Scaling Based on Metrics	181
Application Stack	186
AWS::AutoScaling::LaunchConfiguration	201
AWS::AutoScaling::AutoScalingGroup	209
AWS::AutoScaling::ScalingPolicy	215
AWS::CloudWatch::Alarm	219
AWS::ElasticLoadBalancing::LoadBalancer	223
AWS::RDS::DBSubnetGroup	231
AWS::RDS::DBInstance	233
AWS::EC2::Instance	249
Chapter 6: Stack Policy	259
Example Stack Policy	259
Defining a Stack Policy	260
Setting a Stack Policy	263
Updating Protected Resources	264
More Example Stack Policies	267
Appendix A: Converting JSON to YAML	272
JSON to YAML	272
YAML to JSON	273
Appendix B: CLI Commands	274
Creating a Stack	274
Describing and Listing Your Stacks	275
Viewing Stack Event History	278
Listing Resources	282
Retrieving a Template	282
Validating a Template	283
Uploading Local Artifacts to an S3 Bucket	285
Quickly Deploying Templates with Transforms	286
Deleting a Stack	286
Appendix C: Intrinsic Function	288
Fn::Base64	288
Condition Functions	290
Fn::FindInMap	297
Fn::GetAtt	299
Fn::GetAZs	300
Fn::ImportValue	302
Fn::Join	304
Fn::Select	306
Fn::Sub	308

CONTENTS

Ref	311
Appendix D: Resource Attribute	313
CreationPolicy Attribute	313
DeletionPolicy Attribute	317
DependsOn Attribute	319
Metadata Attribute	324
UpdatePolicy Attribute	325
Appendix E: Pseudo Parameters	332
AWS::AccountId	332
AWS::NotificationARNs	332
AWS::NoValue	333
AWS::Region	333
AWS::StackId	334
AWS::StackName	334
Appendix F: Helper Scripts	335
cfn-init	337
cfn-signal	340
cfn-get-metadata	345
cfn-hup	347

About the Author

Yan Kurniawan is a Cloud Engineer who is working at [Flux7 Labs, Inc.](#)¹. Yan has worked on many AWS projects and he has helped companies from startups to Fortune 100 companies. He is an AWS Certified Solutions Architect Professional and a Puppet Certified Professional.

Based in Sydney, Australia, he enjoys the laid-back lifestyle in one of the world's best cities to live in. He loves taking beautiful cityscape and landscape pictures with his camera.

¹<http://flux7.com>

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.²

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.

²<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 repository³](#).

³<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.⁴

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 Computing⁵](#)
- [What is Amazon Web Services⁶](#)

⁴Getting Started with AWS

⁵<http://youtu.be/jOhbTAU4OPI>

⁶http://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.⁷

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.⁸

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.⁹

⁷<https://aws.amazon.com/ec2/>

⁸<https://aws.amazon.com/vpc/>

⁹<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.¹⁰

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.¹¹

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.

¹⁰<http://aws.amazon.com/elasticloadbalancing>

¹¹<https://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.¹²

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.¹³

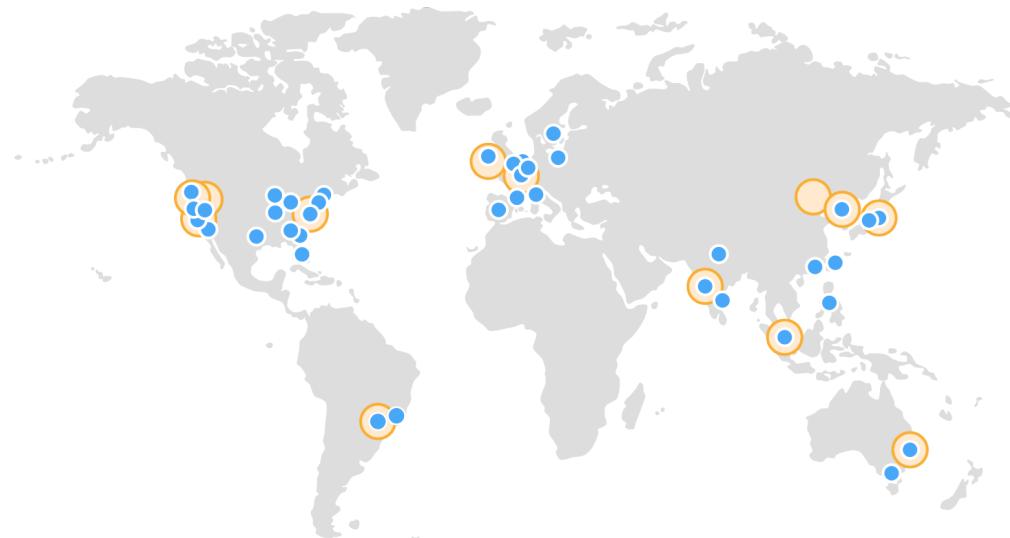
¹²<https://aws.amazon.com/cloudformation/>

¹³<https://aws.amazon.com/documentation/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.¹⁴



See detailed list of offerings at all AWS locations¹⁵

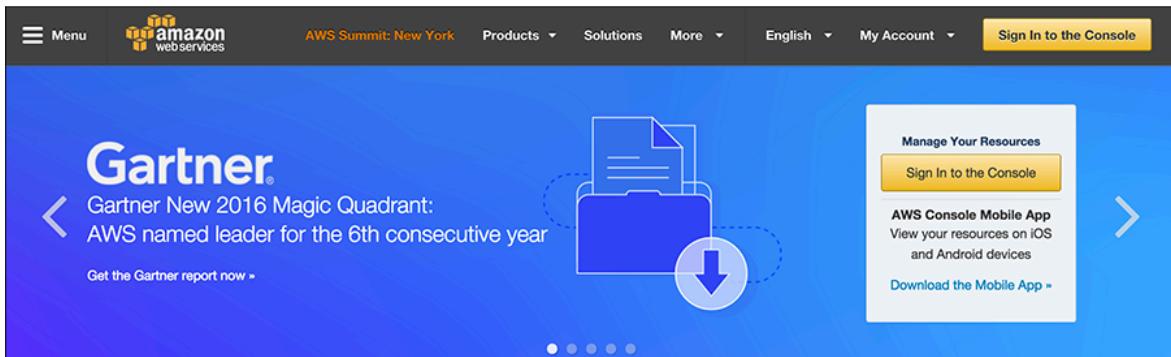
¹⁴<http://aws.amazon.com/what-is-cloud-computing/#global-reach>

¹⁵<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

A screenshot of the 'Sign In or Create an AWS Account' page. The title is 'Sign In or Create an AWS Account' in orange. Below it, a question 'What is your email (phone for mobile accounts)?' is followed by an 'E-mail or mobile number:' input field containing a redacted email address. There are two radio button options: 'I am a new user.' (which is selected) and 'I am a returning user and my password is:' followed by another redacted input field. At the bottom, there's a yellow 'Sign in using our secure server' button with a play icon, and a link 'Forgot your password?'

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

Login Credentials

Use the form below to create login credentials that can be used for AWS as well as Amazon.com.

My name is:	<input type="text" value="Yan Kurniawan"/>
My e-mail address is:	<input type="text"/>
Type it again:	<input type="text"/>
note: this is the e-mail address that we will use to contact you about your account	
Enter a new password:	<input type="password" value="*****"/>
Type it again:	<input type="password" value="*****"/>
Create account	

- 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.

Contact Information

<input checked="" type="radio"/> Company Account	<input type="radio"/> Personal Account
* Required Fields	
Full Name*	<input type="text" value="Yan Kurniawan"/>
Country*	<input type="text" value="Australia"/>
Address*	<input type="text"/> <small>Apartment, suite, unit, building, floor, etc.</small>
City*	<input type="text"/>
State / Province or Region*	<input type="text"/>
Postal Code*	<input type="text"/>
Phone Number*	<input type="text"/>
Security Check 	Refresh Image
Please type the characters as shown above	
<input type="text" value="ymnmgp"/>	
AWS Customer Agreement	
<small><input checked="" type="checkbox"/> Check here to indicate that you have read and agree to the terms of the AWS Customer Agreement</small>	
Securely Submit	

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

Payment Information

Please enter your payment information below. You will be able to try a broad set of AWS products for free via the Free Tier. We will only bill your credit or debit card for usage that is not covered by our Free Tier.

[Frequently Asked Questions](#)

Credit/Debit Card Number Expiration Date

Cardholder's Name Yan Kurniawan

Use my contact address
 Use a new address

Continue

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.

Identity Verification

You will be called immediately by an automated system and prompted to enter the PIN number provided.

1. Provide a telephone number
Please enter your information below and click the "Call Me Now" button.

Country Code Phone Number Ext
Australia (+61)

Call Me Now

2. Call in progress

3. Identity verification complete

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

Identity Verification

You will be called immediately by an automated system and prompted to enter the PIN number provided.

1. Provide a telephone number ✓

2. Call in progress
Please follow the instructions on the telephone and key in the following Personal Identification Number (PIN) on your telephone when prompted.

PIN: 6607

If you have not yet received a call at the number indicated above please wait. This page will automatically update with what you need to do next.

3. Identity verification complete

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

Identity Verification

You will be called immediately by an automated system and prompted to enter the PIN number provided.

1. Provide a telephone number ✓

2. Call in progress ✓

3. Identity verification complete
Your identity has been verified successfully

[Continue to select your Support Plan](#)

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

Support Plan

AWS Support offers a selection of plans to meet your needs. All plans provide 24x7 access to customer service, AWS documentation, whitepapers, and support forums. For access to technical support and additional resources to help you plan, deploy, and optimize your AWS environment, we recommend selecting a support plan that best aligns with your AWS usage.

Please Select One

- Basic**
Description: Customer Service for account and billing questions and access to the AWS Community Forums.
Price: Included
- Developer**
Use case: Experimenting with AWS
Description: One primary contact may ask technical questions through Support Center and get a response within 12–24 hours during local business hours.
Price: Starts at \$29/month (scales based on usage)
- Business**
Use case: Production use of AWS
Description: 24x7 support by phone and chat, 1-hour response to urgent support cases, and help with common third-party software. Full access to AWS Trusted Advisor for optimizing your AWS infrastructure, and access to the AWS Support API for automating your support cases and retrieving Trusted Advisor results.
Price: Starts at \$100/month (scales based on usage)
- Enterprise**
Use case: Mission-critical use of AWS
Description: All the features of the Business support plan, plus an assigned Technical Account Manager (TAM) who provides proactive guidance and best practices to help plan, develop, and run your AWS solutions, a Support Concierge who provides billing and account analysis and assistance, access to Infrastructure Event Management to support product launches, seasonal promotions, events, and migrations, and 15-minute response to critical support cases with prioritized case handling.
Price: Starts at \$15,000/month (scales based on usage)
If you select this option, customer support will contact you within 48 hours to discuss your needs and finalize the signup. Support resources will be available when signup is finalized, and no charges will be incurred until that time.

To explore all features and benefits of AWS Support, including plan comparisons and pricing samples, [click here](#).

Continue

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 Scheme¹⁶](#)

[S3 Pricing Scheme¹⁷](#)



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 Tier¹⁸](#) page.



Amazon provides an [online calculator¹⁹](#) to estimate your monthly AWS bill.

¹⁶<http://aws.amazon.com/ec2/pricing>

¹⁷<http://aws.amazon.com/s3/pricing>

¹⁸<http://aws.amazon.com/free>

¹⁹<http://calculator.s3.amazonaws.com/index.html>

AWS Management Console

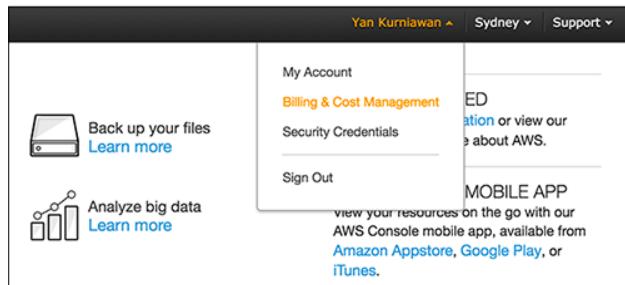
Amazon provides a web-based AWS Management Console. You can access and manage Amazon Web Services resources through a simple and intuitive web-based user interface. The AWS Management Console is a single destination for managing all your AWS resources, from EC2 instances to DynamoDB tables. Use the Console 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.

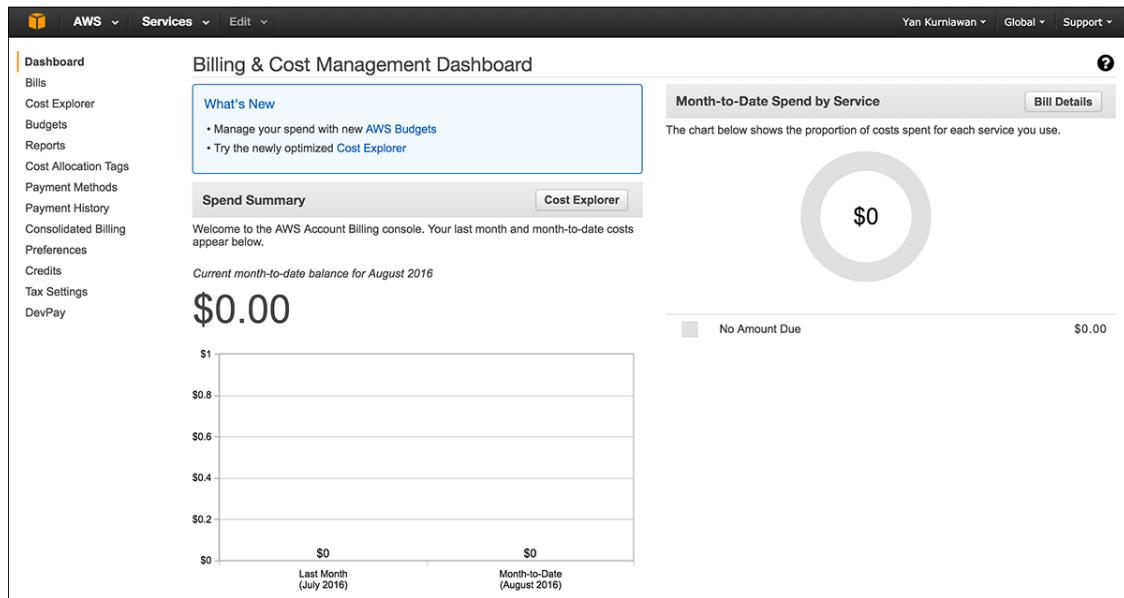
The screenshot shows the AWS Management Console homepage. At the top, there's a navigation bar with 'AWS', 'Services', 'Edit', and user information ('Yan Kurniawan', 'Select a Region', 'Support'). Below the navigation is a 'Quick Starts' section with six cards: 'Build a web app' (Start now), 'Launch a Virtual Machine (EC2 Instance)', 'Back up your files', 'Build a back end for your mobile app' (Start now), 'Host a static website', and 'Analyze big data'. Underneath is a 'Shortcuts and Recently Viewed Services' section with icons for EC2, ElastiCache, Elasticsearch Service, CloudWatch, and Route 53. The main area is titled 'AWS Services' with a 'SHOW CATEGORIES' link. It's organized into several sections: 'COMPUTE' (EC2, EC2 Container Service, Elastic Beanstalk, Lambda), 'STORAGE & CONTENT DELIVERY' (S3, CloudFront, Elastic File System, Glacier, Snowball), 'DEVELOPER TOOLS' (CodeCommit, CodeDeploy, CodePipeline), 'MANAGEMENT TOOLS' (CloudWatch, CloudFormation, CloudTrail, Config, OpsWorks, Service Catalog), 'INTERNET OF THINGS' (AWS IoT), 'GAME DEVELOPMENT' (GameLift), 'MOBILE SERVICES' (Mobile Hub, Cognito, Device Farm, Mobile Analytics, SNS). On the right side, there's a 'GETTING STARTED' section with links to documentation and training, an 'AWS CONSOLE MOBILE APP' section with links to the Appstore, Google Play, and iTunes, an 'AWS MARKETPLACE' section with a link to find and buy software, a 'FEEDBACK' section with a link to tell what you think, and a 'Service Health' section showing 'All services are operating normally' (updated Aug 10 2016 11:20:01 GMT+1000) with a 'View Dashboard' link.

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.

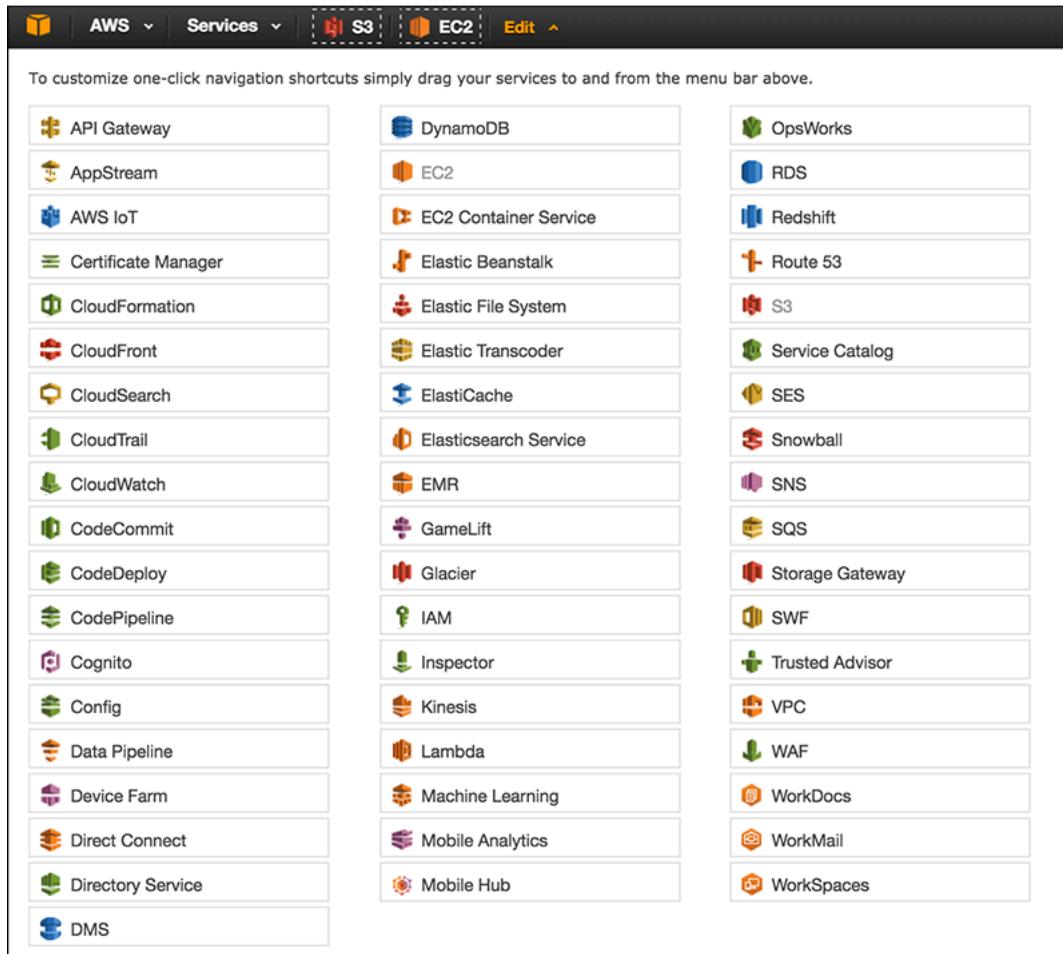


Billing and Cost Management



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.

Navigation bar

Navigation pane

Current page

Region selector

EC2 Console



For a complete guide to AWS Management Console, visit [AWS Management Console Getting Started Guide²⁰](#) page.

²⁰ <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.

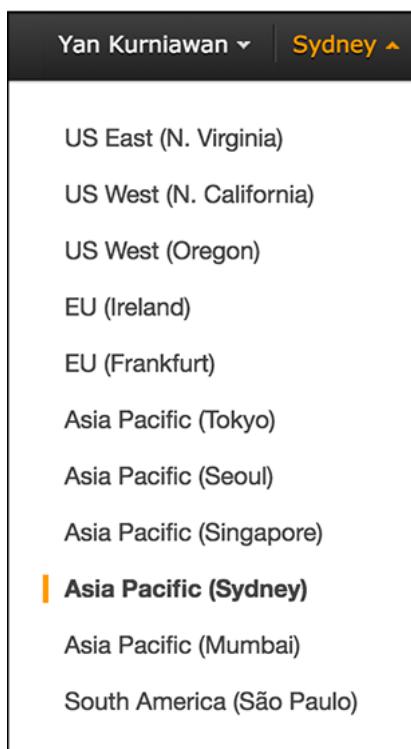
Let's create our first EC2 instance.

Create a Key Pair

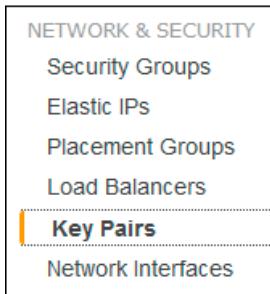
First, we need to create a key pair. AWS uses public-key cryptography to secure the login information for your instance. A Linux instance has no password; you use a key pair to log in to your instance securely. You specify the name of the key pair when you launch your instance, then provide the private 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

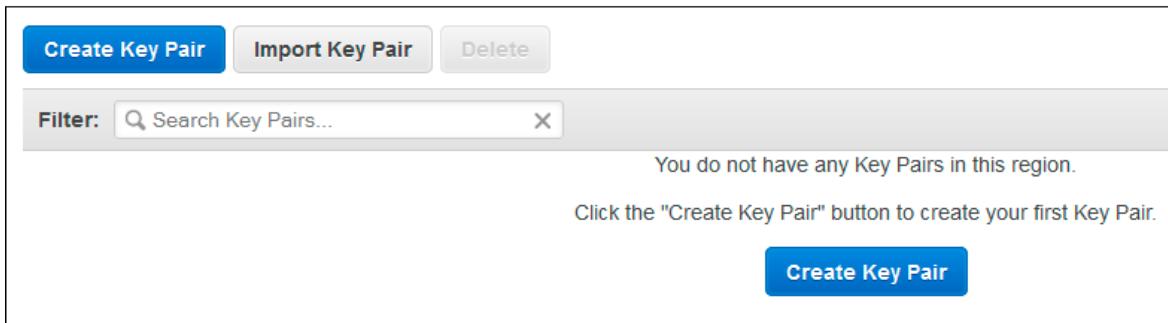
1. Open the Amazon EC2 Dashboard <https://console.aws.amazon.com/ec2>
2. From the navigation bar, select a region for the key pair.



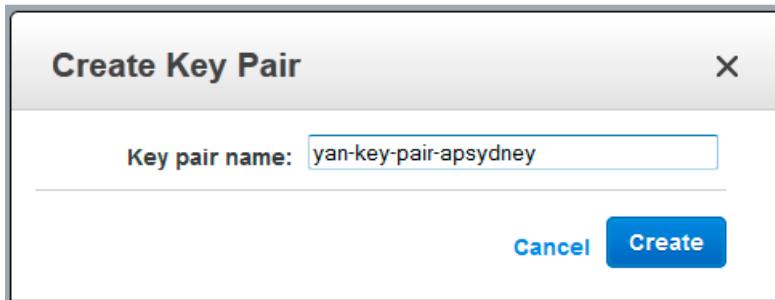
3. Click Key Pairs in the navigation pane.



4. Click Create Key Pair



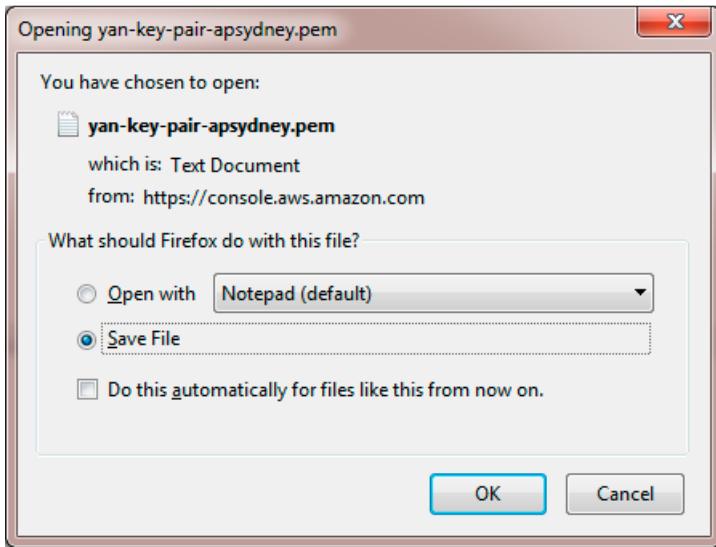
5. Enter a name for the new key pair in the Key pair name field of the Create Key Pair dialog box, and then click Create. Choose a name that is easy for you to remember, such as your name, followed by -key-pair, plus the region name. For example, yan-key-pair-apsydne.



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.



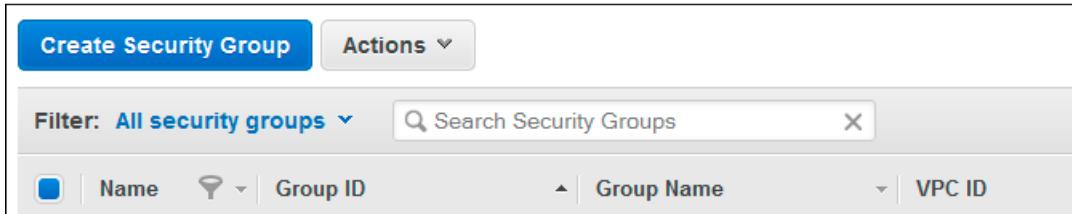
You'll need the public IP address of your local computer, which you can get using a service from Amazon AWS <http://checkip.amazonaws.com>. If you are connecting through an Internet service provider (ISP) or from behind a firewall without a static IP address, you need to find out the range of IP addresses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

Security group name <small>i</small>	SG_apsydney
Description <small>i</small>	base security group - sydney
VPC <small>i</small>	vpc- [REDACTED] (172.31.0.0/16) *

* denotes default VPC

Security group rules:

Type <small>i</small>	Protocol <small>i</small>	Port Range <small>i</small>	Source <small>i</small>
HTTP	TCP	80	Anywhere <small>i</small> 0.0.0.0/0
HTTPS	TCP	443	Anywhere <small>i</small> 0.0.0.0/0
SSH	TCP	22	Custom IP <small>i</small> 123.[REDACTED]/32

Add Rule

Create

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 Instance²¹](#).

To launch an instance

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

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the Asia Pacific (Sydney) region

²¹<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.

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.

Quick Start

My AMIs	Amazon Linux	Amazon Linux AMI 2016.03.3 (HVM), SSD Volume Type - ami-dc361ebf	Select
AWS Marketplace			64-bit
Community AMIs		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.	
<input type="checkbox"/> Free tier only <small>(i)</small>		Root device type: ebs Virtualization type: hvm	

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.

Step 2: Choose an Instance Type

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.

Filter by: All instance types ▾ Current generation ▾ Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs <small>(i)</small>	Memory (GiB)	Instance Storage (GB) <small>(i)</small>	EBS-Optimized Available <small>(i)</small>
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-
<input checked="" type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1	EBS only	-
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-

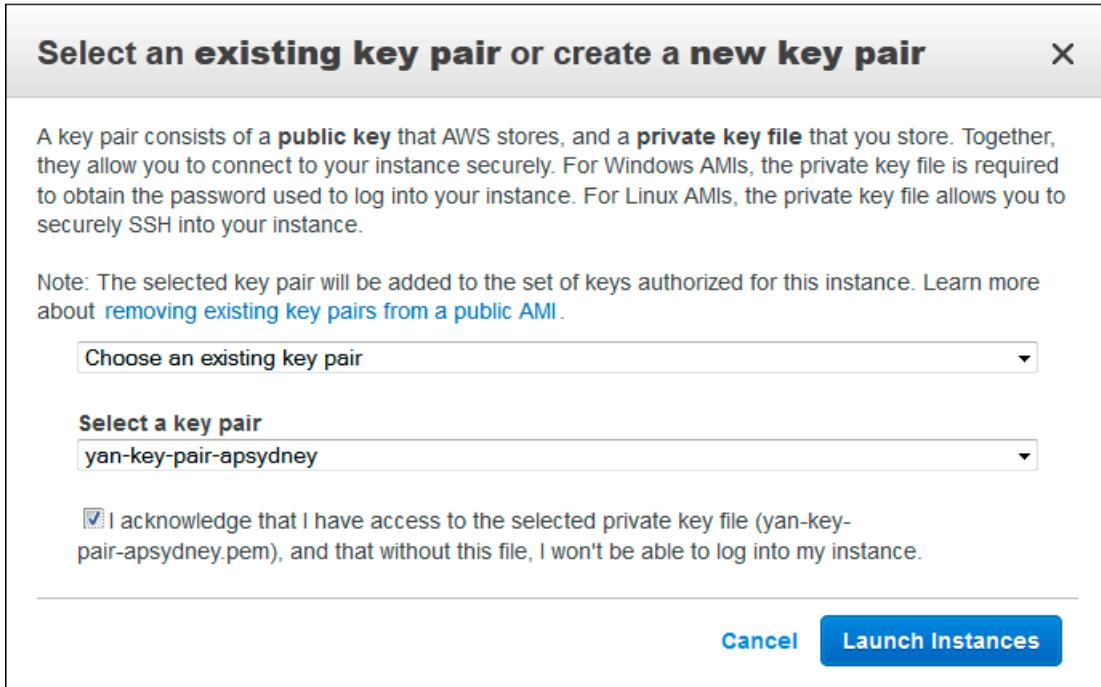
Cancel **Previous** **Review and Launch** **Next: Configure Instance Details**

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**.

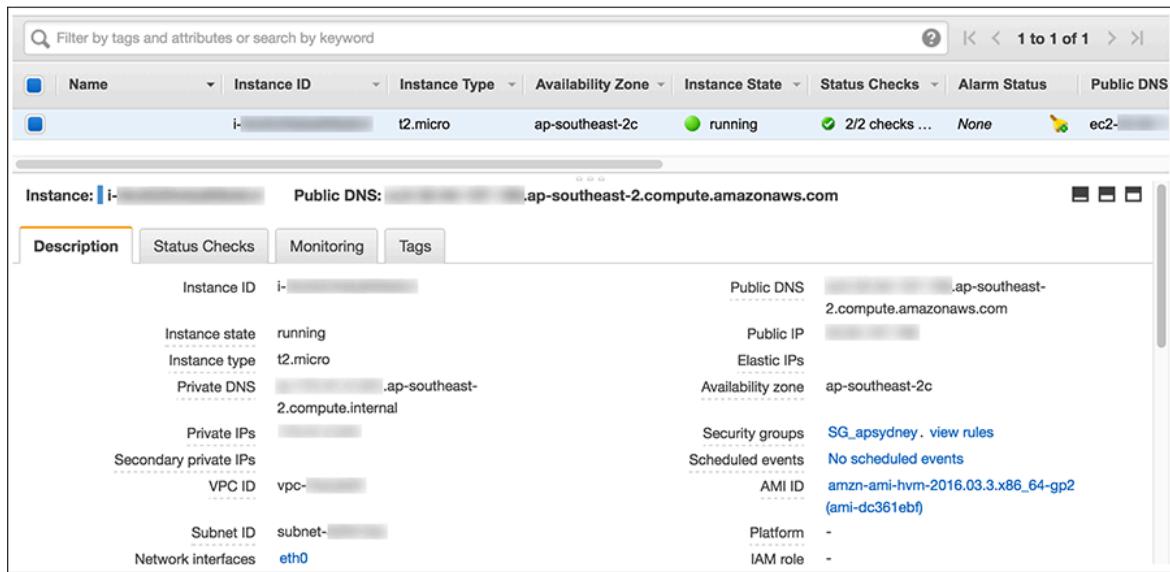
The screenshot shows the 'Assign a security group' section of the AWS CloudFormation 'Review and Launch' page. It includes two radio button options: 'Create a new security group' (unchecked) and 'Select an existing security group' (checked). Below this is a table listing existing security groups. A row for 'SG_apsydney' is selected, indicated by a checked checkbox in the first column. The table has columns for Security Group ID, Name, Description, and Actions (with 'Copy to new' links). At the bottom of the table is a detailed view of the security group's rules, showing three entries: SSH (TCP, port 22, source 123.0.0.1/32), HTTP (TCP, port 80, source 0.0.0.0/0), and HTTPS (TCP, port 443, source 0.0.0.0/0). The bottom right of the screenshot shows 'Cancel', 'Previous', and 'Review and Launch' buttons.

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**).

Filter by tags and attributes or search by keyword							
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
	i-1234567890abcdef	t2.micro	ap-southeast-2c	pending	Initializing	None	



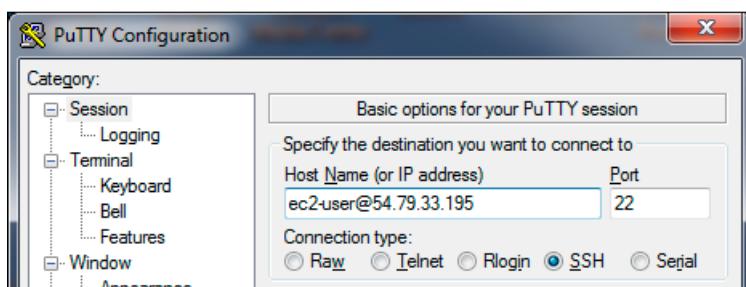
To learn more about Amazon EC2 instance type, see <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html>

Connect to Your Instance

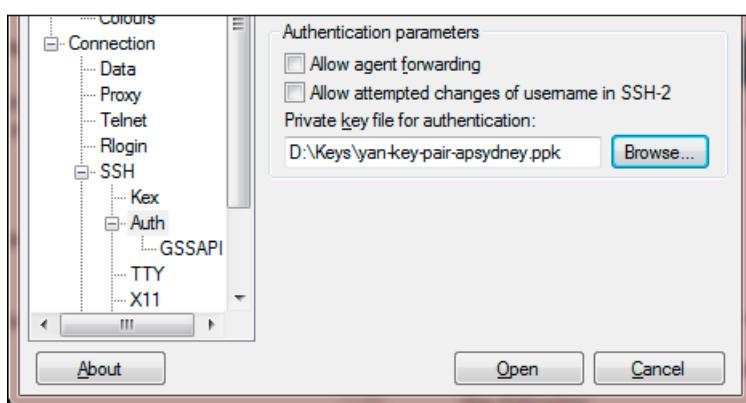
If your computer uses Windows operating system, you will need to install PuTTY to connect to your Linux EC2 instance.

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 an 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.

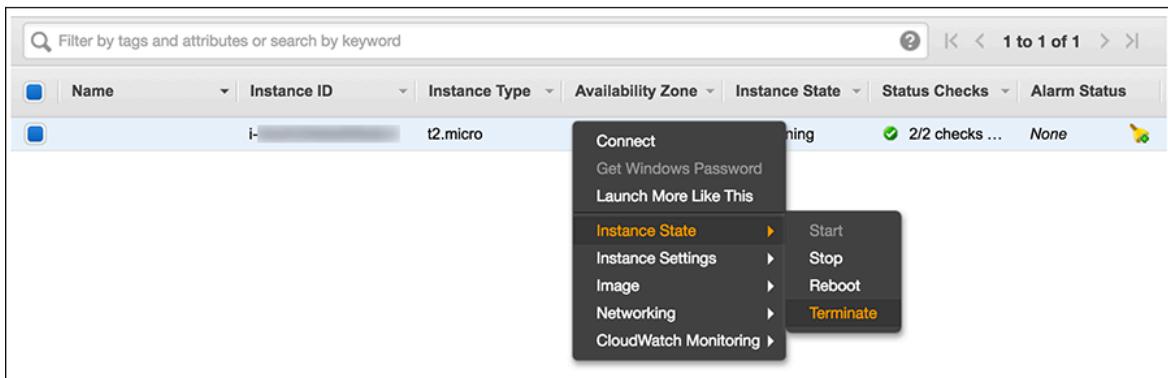
Terminate Your Instance

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.

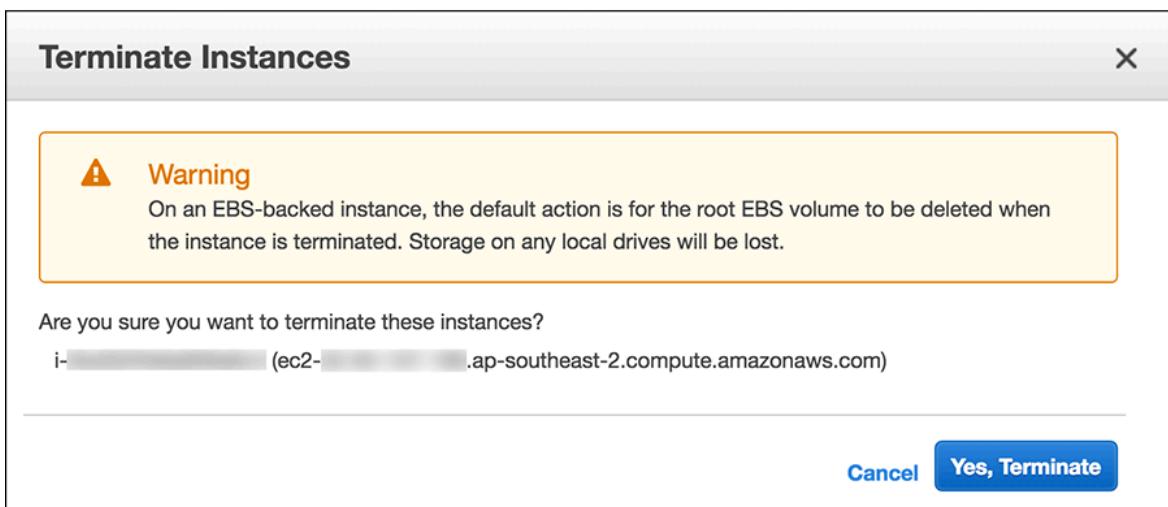
Terminating an instance effectively deletes it because you can't reconnect to an instance after you've terminated it. This differs from stopping the instance; when you stop an instance, it is shut down and you are not billed for hourly usage or data transfer. Also, you can restart a stopped instance 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 Guide²²](#). For a complete list of Amazon AWS Documentation, visit [AWS Documentation²³](#). You can also watch Introduction to AWS videos here: https://aws.amazon.com/training/intro_series/.

²²<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>

²³<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.²⁴

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.

²⁴<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.²⁵

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.

²⁵<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.

```
1 ---  
2 AWSTemplateFormatVersion: "2010-09-09"  
3 Description: A sample template  
4 Resources:  
5   MyEC2Instance:  
6     Type: "AWS::EC2::Instance"  
7     Properties:  
8       ImageId: "ami-2f726546"  
9       InstanceType: t1.micro  
10      KeyName: testkey  
11      BlockDeviceMappings:  
12        -  
13          DeviceName: /dev/sdm  
14          Ebs:  
15            VolumeType: io1  
16            Iops: 200  
17            DeleteOnTermination: false  
18            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:

```
1 ---  
2  
3 AWSTemplateFormatVersion: "2010-09-09"  
4 Description: A sample template  
5 Resources:  
6   MyEC2Instance:  
7     Type: "AWS::EC2::Instance"  
8     Properties:  
9       ImageId: "ami-2f726546"  
10      InstanceType: t1.micro  
11      KeyName: testkey  
12      BlockDeviceMappings:  
13        -  
14          DeviceName: /dev/sdm  
15          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.

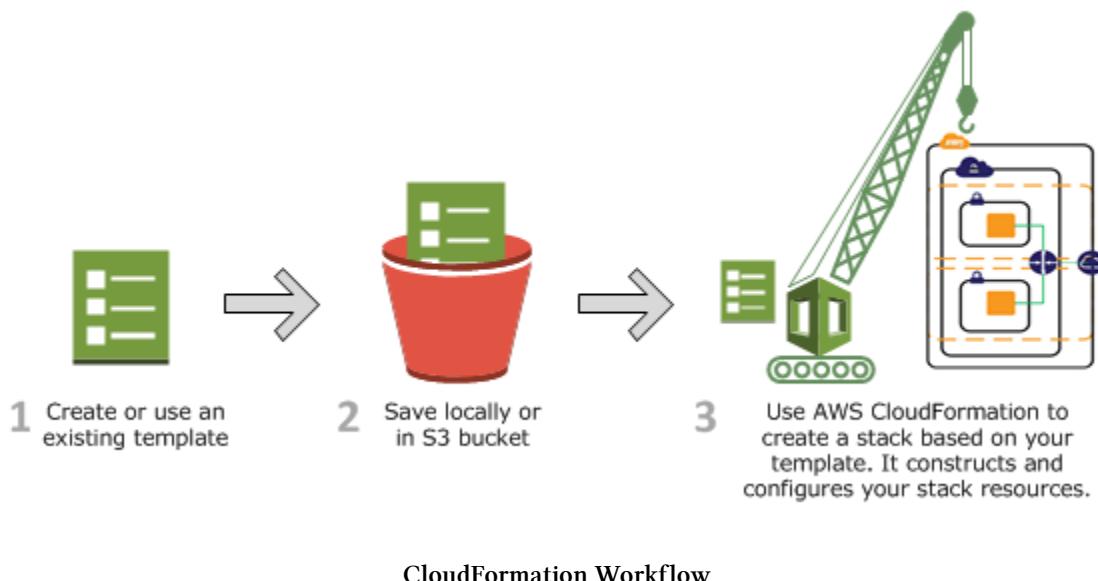
Change sets allow you to see how your changes might impact your running resources, especially for critical resources, before implementing them.

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.



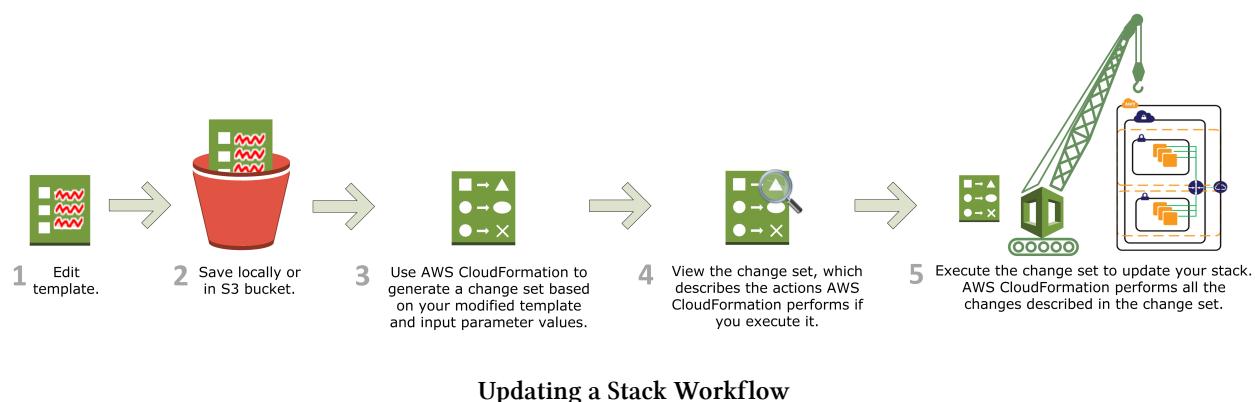
- 1 You can design an AWS CloudFormation template (a JSON-formatted or YAML-formatted document) in AWS CloudFormation Designer or write one in a text editor. You can also choose to use a provided template. The template describes the resources you want and their settings.
- 2 Save the template locally 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.

- i** 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.



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 update 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.²⁶

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.

²⁶<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-anatomy.html>

A YAML-formatted template fragment:

```
1  ---
2 AWSTemplateFormatVersion: "<version date>"
3
4 Description:
5   <String>
6
7 Metadata:
8   <template metadata>
9
10 Parameters:
11   <set of parameters>
12
13 Mappings:
14   <set of mappings>
15
16 Conditions:
17   <set of conditions>
18
19 Resources:
20   <set of resources>
21
22 Outputs:
23   <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:

```
1 Description: >
2   Here are some
3   details about
4   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:

```
1 Metadata:
2   Instances:
3     Description: "Information about the instances"
4   Databases:
5     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 Z23YXV40VPL04A.

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 Z23YXV40VPL04A, Z23YXV40VPL04B.

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).

```
1 Mappings:  
2   Mapping01:  
3     Key01:  
4       Name: Value01  
5     Key02:  
6       Name: Value02  
7     Key03:  
8       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.

```
1 Conditions:  
2   <Logical ID>:  
3     <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.

```
1 Resources:  
2   <Logical ID>:  
3     Type: <Resource type>  
4     Properties:  
5       <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.

```

1 Outputs:
2   <Logical ID>:
3     Description: <Information about the value>
4     Value: <Value to return>
5     Export:
6       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.

```

1 Export:
2   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
```



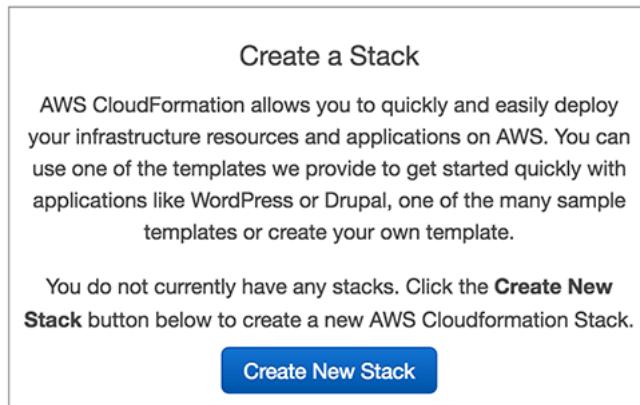
To find out what Resource Properties are available for certain Resource, go to <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html> and select the Resource. For example <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html>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

To create a CloudFormation stack:

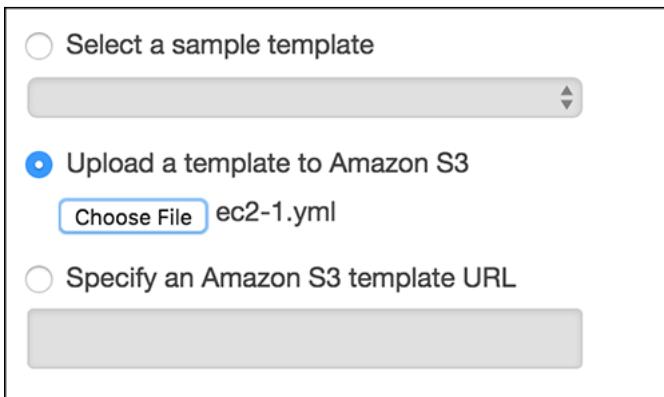
1. Open the CloudFormation console at <https://console.aws.amazon.com/cloudformation/>. If you are not logged in to the AWS Management Console yet, you need to log in before using the AWS CloudFormation console.
2. Create a new stack by using one of the following options:
 - Click **Create Stack**. This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



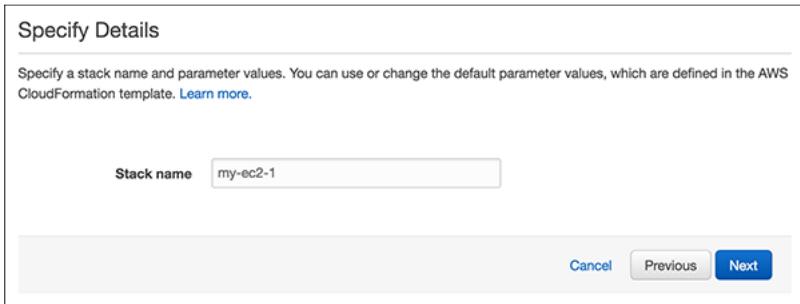
- Click **Create New Stack** in the CloudFormation Stacks main window. This option is visible only if you have no running stacks.



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**

Stack Name				Created Time	Status	Description
<input checked="" type="checkbox"/> my-ec2-1		2016-09-12 12:37:58 UTC+1000	CREATE_IN_PROGRESS		Template for EC2 instance	

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).

Events				
	Status	Type	Logical ID	Status reason
▶ 12:38:50 UTC+ 1000	CREATE_COMPLETE	AWS::CloudFormation::Stack	my-ec2-1	
▶ 12:38:48 UTC+ 1000	CREATE_COMPLETE	AWS::EC2::Instance	MyEC2Instance	
▶ 12:38:02 UTC+ 1000	CREATE_IN_PROGRESS	AWS::EC2::Instance	MyEC2Instance	Resource creation initiated
12:38:01 UTC+ 1000	CREATE_IN_PROGRESS	AWS::EC2::Instance	MyEC2Instance	
▶ 12:37:58 UTC+ 1000	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	my-ec2-1	User Initiated



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.

The screenshot shows the AWS CloudFormation Tags page for an EC2 instance. At the top, it displays the instance ID (i-...) and Public DNS (ec2-...). Below this, there are tabs for Description, Status Checks, Monitoring, and Tags, with Tags being the active tab. A button labeled "Add/Edit Tags" is visible. A table lists three 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.²⁷

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

²⁷<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.

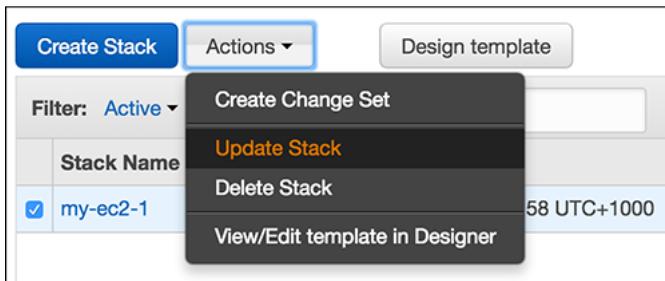
```
$ vi ec2-2.yml
```

```
1 AWSTemplateFormatVersion: 2010-09-09
2 Description: Template for EC2 instance with parameters
3 Parameters:
4   Subnet:
5     Description: The EC2 instance will be launched in this subnet
6     Type: AWS::EC2::Subnet::Id
7   SecurityGroups:
8     Description: Assign Security Groups for the instance
9     Type: List<AWS::EC2::SecurityGroup::Id>
10  AMIid:
11    Description: Image Id
12    Type: AWS::EC2::Image::Id
13  Keypair:
14    Description: Key pair name
15    Type: AWS::EC2::KeyPair::KeyName
16  InstanceType:
17    Type: String
18    Default: t2.small
19    AllowedValues:
20      - t2.micro
21      - t2.small
22      - t2.medium
23    Description: Enter t2.micro, t2.small, or t2.medium. Default is t2.micro.
24 Resources:
25   MyEC2Instance:
26     Type: AWS::EC2::Instance
27     Properties:
28       ImageId: !Ref AMIid
29       InstanceType: !Ref InstanceType
```

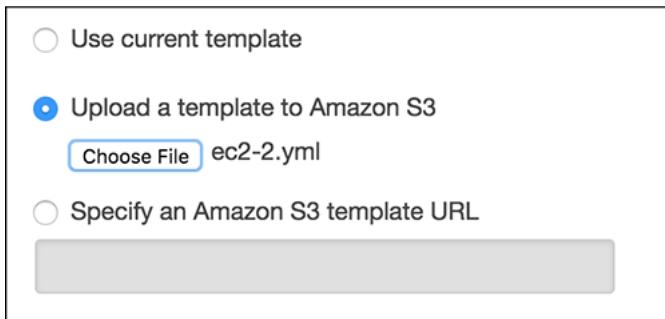
```
30      KeyName: !Ref Keypair  
31      SecurityGroupIds: !Ref SecurityGroups  
32      SubnetId: !Ref Subnet
```

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**.

Specify Details

Specify parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. [Learn more.](#)

Stack name my-ec2-1

Parameters

AMId	ami- [redacted]	Image Id
InstanceType	t2.micro	Enter t2.micro, t2.small, or t2.medium. Default is t2.micro.
Keypair	testkey	Key pair name
SecurityGroups	default (sg- [redacted]) x	Assign Security Groups for the instance
Subnet	subnet- [redacted] (172.31.32.0/20)	The EC2 instance will be launched in this subnet

Cancel **Previous** **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.

Preview your changes

Based on your input, CloudFormation will change the following resources. For more information, choose [View change set details](#).

Action	Logical ID	Physical ID	Resource type	Replacement
Modify	MyEC2Instance	i- [redacted]	AWS::EC2::Instance	True

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**.

Stack Name	Created Time	Status	Description					
my-ec2-1	2016-09-12 12:37:58 UTC+1000	UPDATE_COMPLETE	Template for EC2 instance with parameters					
Overview	Outputs	Resources	Events	Template	Parameters	Tags	Stack Policy	Change Sets
2016-09-12	Status	Type	Logical ID	Status reason				
▶ 13:48:23 UTC+1000	UPDATE_COMPLETE	AWS::CloudFormation::Stack	my-ec2-1					
▶ 13:48:22 UTC+1000	DELETE_COMPLETE	AWS::EC2::Instance	MyEC2Instance					
▶ 13:47:16 UTC+1000	DELETE_IN_PROGRESS	AWS::EC2::Instance	MyEC2Instance					
▶ 13:47:14 UTC+1000	UPDATE_COMPLETE_CLEANUP_IN_PROGRESS	AWS::CloudFormation::Stack	my-ec2-1					
▶ 13:47:11 UTC+1000	UPDATE_COMPLETE	AWS::EC2::Instance	MyEC2Instance					
▶ 13:46:25 UTC+1000	UPDATE_IN_PROGRESS	AWS::EC2::Instance	MyEC2Instance	Resource creation initiated				
▶ 13:46:24 UTC+1000	UPDATE_IN_PROGRESS	AWS::EC2::Instance	MyEC2Instance	Requested update requires the creation of a new physical resource; hence creating one.				
▶ 13:46:16 UTC+1000	UPDATE_IN_PROGRESS	AWS::CloudFormation::Stack	my-ec2-1	User Initiated				



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.

Cancelling 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:

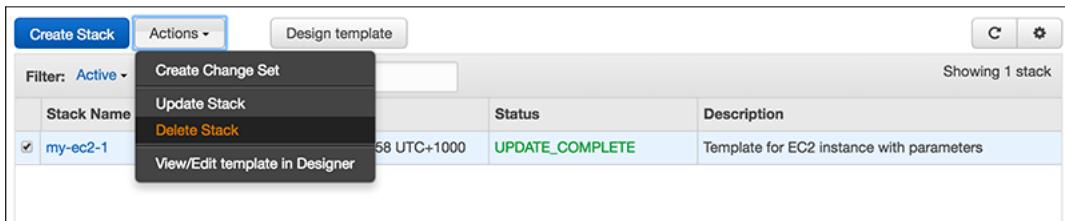
1. From the list of stacks in the AWS CloudFormation console, select the stack that is currently being updated (its state must be UPDATE_IN_PROGRESS).
2. Choose **Actions** and then **Cancel Update**.
3. To continue canceling the update, click **Yes, Cancel Update** when prompted. Otherwise, click **Cancel**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

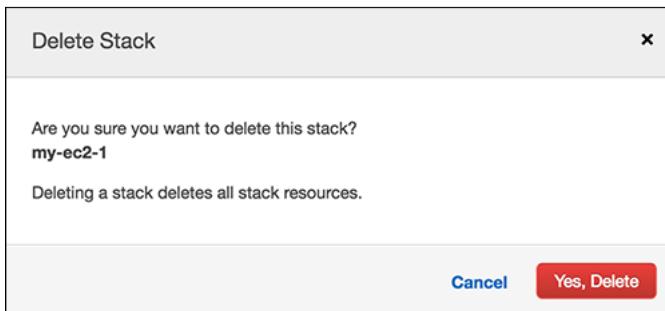
Deleting Stack

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.



Chapter 3: The IAM Stack

Introduction to IAM

AWS Identity and Access Management (IAM) is a web service that enables Amazon Web Services (AWS) customers to manage users and user permissions in AWS. The service is targeted at organizations with multiple users or systems in the cloud that use AWS products such as Amazon EC2, Amazon SimpleDB, and the AWS Management Console. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users can access.²⁸

This short video from Amazon provides a brief introduction to AWS IAM: <https://youtu.be/Ul6FW4UANGc>.

IAM Features

IAM gives you the following features:²⁹

Shared access to your AWS account

You can grant other people permission to administer and use resources in your AWS account without having to share your password or access key.

Granular permissions

You can grant different permissions to different people for different resources. For example, you might allow some users complete access to Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB, Amazon Redshift, and other AWS services. For other users, you can allow read-only access to just some S3 buckets, or permission to administer just some EC2 instances, or to access your billing information but nothing else.

Secure access to AWS resources for applications that run on Amazon EC2

You can use IAM features to securely give applications that run on EC2 instances the credentials that they need in order to access other AWS resources, like S3 buckets and RDS or DynamoDB databases.

Identity federation

You can allow users who already have passwords elsewhere—for example, in your corporate network or with an Internet identity provider—to get temporary access to your AWS account.

²⁸<https://aws.amazon.com/documentation/iam/>

²⁹<http://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html#intro-features>

Identity information for assurance

If you use AWS CloudTrail, you receive log records that include information about those who made requests for resources in your account. That information is based on IAM identities.

PCI DSS Compliance

IAM supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS). For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see <http://aws.amazon.com/compliance/pci-dss-level-1-faqs/>.

Integrated with many AWS services

For a list of AWS services that work with IAM, visit: http://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html

Free to use

AWS Identity and Access Management is a feature of your AWS account offered at no additional charge. You will be charged only for use of other AWS products by your IAM users.

AWS Security Token Service is an included feature of your AWS account offered at no additional charge. You are charged only for the use of other AWS services that are accessed by your AWS STS temporary security credentials.

Accessing IAM

You can work with AWS Identity and Access Management in any of the following ways:

AWS Management Console

The console is a browser-based interface to manage IAM and AWS resources.

AWS Command Line Tools

You can use the AWS command line tools to issue commands at your system's command line to perform IAM and AWS tasks; this can be faster and more convenient than using the console. The command line tools are also useful if you want to build scripts that perform AWS tasks. AWS provides two sets of command line tools: the AWS Command Line Interface (AWS CLI) and the AWS Tools for Windows PowerShell.

AWS SDKs

AWS provides SDKs (software development kits) that consist of libraries and sample code for various programming languages and platforms (Java, Python, Ruby, .NET, iOS, Android, etc.). The SDKs provide a convenient way to create programmatic access to IAM and AWS. For example, the SDKs take care of tasks such as cryptographically signing requests, managing errors, and retrying requests automatically.

IAM HTTPS API

You can access IAM and AWS programmatically by using the IAM HTTPS API, which lets you issue HTTPS requests directly to the service. When you use the HTTPS API, you must include code to digitally sign requests using your credentials.

Getting Started with IAM

When you create an AWS account, you create an account (or “root”) identity, which you use to sign in to AWS. You can sign in to the AWS Management Console using this root identity, the email address and password that you provided when creating the account. This combination of your email address and password is also called your *root account credentials*.

When you use your root account credentials, you have complete, unrestricted access to all resources in your AWS account, including access to your billing information and the ability to change your password. This level of access is necessary when you first set up your account. However, it’s recommended that you don’t use root account credentials for everyday access, and do not share your root account credentials with anyone, because doing so gives them unrestricted access to your account. It is not possible to restrict the permissions that are granted to the root account.³⁰

IAM Users

The “identity” aspect of AWS Identity and Access Management (IAM) helps you with the question “Who is that user?”, often referred to as authentication. Instead of sharing your root account credentials with others, you can create individual IAM users within your account that correspond to users in your organization. IAM users are not separate accounts; they are users within your account. Each user can have its own password for access to the AWS Management Console. You can also create an individual access key for each user so that the user can make programmatic requests to work with resources in your account. An IAM user doesn’t have to represent an actual person; you can create an IAM user in order to generate an access key for an application that runs in your corporate network and needs AWS access.

IAM Policy

By default, users can’t access anything in your account. You grant permissions to a user by creating a policy, which is a document that lists the actions that a user can perform and the resources that the actions can affect. The following example shows a policy.

³⁰http://docs.aws.amazon.com/IAM/latest/UserGuide/introduction_identity-management.html

```
1 {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Action": "s3:*",
7             "Resource": "*"
8         }
9     ]
10 }
```

This policy grants permission to perform all S3 actions (`s3:*`). When you attach the policy to a user, that user then has those S3 permissions. Typically, users in your account have different policies attached to them, policies that represent permissions that the users need in order to work in your AWS account.

Any actions or resources that are not explicitly allowed are denied by default. For example, if this is the only policy attached to a user, the user is not allowed to perform any actions in Amazon EC2, Amazon DynamoDB, or in any other AWS product, because permissions to work with those products are not included in the policy.

IAM Group

You can organize IAM users into IAM groups and attach a policy to a group. In that case, individual users still have their own credentials, but all the users in a group have the permissions that are attached to the group. Users or groups can have multiple policies attached to them that grant different permissions. In that case, the users' permissions are calculated based on the combination of policies. But the basic principle still applies: If the user has not been granted an explicit permission for an action and a resource, the user does not have those permissions.

IAM Role

An IAM role is similar to a user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have any credentials (password or access keys) associated with it. Instead, if a user is assigned to a role, access keys are created dynamically and provided to the user.

You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. For example, you might want to grant users in your AWS account access to resources they don't usually have, or grant users in one AWS account access to resources in another account. Or you might want to allow a mobile app to use AWS resources, but not want to embed AWS keys within the app (where they can be difficult to rotate and where users can potentially

extract them). Sometimes you want to give AWS access to users who already have identities defined outside of AWS, such as in your corporate directory. Or, you might want to grant access to your account to third parties so that they can perform an audit on your resources.

IAM Best Practices

To help secure your AWS resources, follow these recommendations for the AWS Identity and Access Management (IAM) service.³¹

- **Lock away your AWS account (root) access keys**

You use an access key (an access key ID and secret access key) to make programmatic requests to AWS. However, do not use your AWS account (root) access key. The access key for your AWS account gives full access to all your resources for all AWS services, including your billing information. You cannot restrict the permissions associated with your AWS account access key.

- **Create individual IAM users**

Don't use your AWS root account credentials to access AWS, and don't give your credentials to anyone else. Instead, create individual users for anyone who needs access to your AWS account. Create an IAM user for yourself as well, give that user administrative privileges, and use that IAM user for all your work. For more information see: http://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_create-admin-group.html.

- **Use groups to assign permissions to IAM users**

Instead of defining permissions for individual IAM users, it's usually more convenient to create groups that relate to job functions (administrators, developers, accounting, etc.), define the relevant permissions for each group, and then assign IAM users to those groups. All the users in an IAM group inherit the permissions assigned to the group. That way, you can make changes for everyone in a group in just one place. As people move around in your company, you can simply change what IAM group their IAM user belongs to.

- **Grant least privilege**

When you create IAM policies, follow the standard security advice of granting least privilege—that is, granting only the permissions required to perform a task. Determine what users need to do and then craft policies for them that let the users perform only those tasks. It's more secure to start with a minimum set of permissions and grant additional permissions as necessary, rather than starting with permissions that are too lenient and then trying to tighten them later.

- **Configure a strong password policy for your users**

If you allow users to change their own passwords, require that they create strong passwords and that they rotate their passwords periodically. On the **Account Settings** page of the IAM console, you can create a password policy for your account. You can use the password policy to define password requirements, such as minimum length, whether it requires non-alphabetic characters, how frequently it must be rotated, and so on. For more information, see http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_passwords_account-policy.html

- **Enable MFA for privileged users**

For extra security, enable multifactor authentication (MFA) for privileged IAM users (users

³¹<http://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html>

who are allowed access to sensitive resources or APIs). With MFA, users have a device that generates a unique authentication code (a one-time password, or OTP) and users must provide both their normal credentials (like their user name and password) and the OTP. The MFA device can either be a special piece of hardware, or it can be a virtual device (for example, it can run in an app on a smartphone). For more information, see http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa.html.

- **Use roles for applications that run on Amazon EC2 instances**

Applications that run on an Amazon EC2 instance need credentials in order to access other AWS services. To provide credentials to the application in a secure way, use IAM roles. A role is an entity that has its own set of permissions, but that isn't a user or group. Roles also don't have their own permanent set of credentials the way IAM users do. In the case of Amazon EC2, IAM dynamically provides temporary credentials to the EC2 instance, and these credentials are automatically rotated for you.

- **Delegate by using roles instead of by sharing credentials**

You might need to allow users from another AWS account to access resources in your AWS account. If so, don't share security credentials, such as access keys, between accounts. Instead, use IAM roles. You can define a role that specifies what permissions the IAM users in the other account are allowed, and from which AWS accounts the IAM users are allowed to assume the role.

- **Rotate credentials regularly**

Change your own passwords and access keys regularly, and make sure that all IAM users in your account do as well. That way, if a password or access key is compromised without your knowledge, you limit how long the credentials can be used to access your resources. You can apply a password policy to your account to require all your IAM users to rotate their passwords, and you can choose how often they must do so.

- **Remove unnecessary credentials**

Remove IAM user credentials (that is, passwords and access keys) that are not needed. For example, an IAM user that is used for an application does not need a password (passwords are necessary only to sign in to AWS websites). Similarly, if a user does not and will never use access keys, there's no reason for the user to have them.

- **Use policy conditions for extra security**

To the extent that it's practical, define the conditions under which your IAM policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from, or you can specify that a request is allowed only within a specified date range or time range. You can also set conditions that require the use of SSL or MFA (multifactor authentication). For example, you can require that a user has authenticated with an MFA device in order to be allowed to terminate an Amazon EC2 instance.

- **Monitor activity in your AWS account**

You can use logging features in AWS to determine the actions users have taken in your account and the resources that were used. The log files show the time and date of actions, the source IP for an action, which actions failed due to inadequate permissions, and more.

Creating Your First IAM Admin User and Group

This procedure describes how to create an IAM group named **Administrators**, grant the group full permissions for all AWS services, and then create an administrative IAM user for yourself by adding the user to the Administrators group.

To create the Administrators group:

1. Sign in to the Identity and Access Management (IAM) console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Groups**, and then choose **Create New Group**.
3. In the **Group Name** box, type **Administrators**, and then choose **Next Step**.
4. In the list of policies, select the check box next to the **AdministratorAccess** policy. You can use the **Filter** drop-down menu and **Filter** box to filter the list or to search for a specific policy.
5. Choose **Next Step**, review the details of your request, and then choose **Create Group**.

Your new group is listed under **Group Name**.

To create an IAM user for yourself, add the user to the Administrators group, and create a password for the user:

1. In the navigation pane, choose **Users**, and then choose **Create New Users**.
2. In box 1., type a user name.
3. Clear the check box next to **Generate an access key for each user**, and then choose **Create**.
4. In the list of users, choose the name (not the check box) of the user you just created. You can use the **Search** box to search for the user name.
5. Choose the **Groups** tab at the bottom of the **User Summary** page (not the **Groups** link in the left-hand navigation bar), and then choose **Add User to Groups**.
6. Select the check box next to the **Administrators** group. Then choose **Add to Groups**. This returns you to the **Summary** page for the user you just created.
7. Still on the new user's **Summary** page, choose the **Security Credentials** tab. Under **Sign-In Credentials**, choose **Manage Password**.
8. Choose **Assign a custom password**. Then type a password in the **Password** and **Confirm Password** boxes. When you are finished, choose **Apply**.

You created an IAM group named **Administrators**, granted full permissions for all AWS services to the group, created an IAM user for yourself, and added the user to the group. You can use the same process to create more groups and users, and to give your users access to your AWS account resources.

To learn about using policies to restrict users' permissions to specific AWS resources, go to http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_examples.html.

IAM Stack

Organize Your Stacks By Lifecycle and Ownership

When you're designing your CloudFormation templates, it's recommended to organize your stacks by lifecycle and ownership. Use the lifecycle and ownership of your AWS resources to help you decide what resources should go in each stack. Normally, you might put all your resources in one stack, but as your stack grows in scale and broadens in scope, managing a single stack can be cumbersome and time consuming. By grouping resources with common lifecycles and ownership, owners can make changes to their set of resources by using their own process and schedule without affecting other resources.³²

For example, imagine a team of developers and engineers who own a website that is hosted on autoscaling instances behind a load balancer. Because the website has its own lifecycle and is maintained by the website team, you can create a stack for the website and its resources. Now imagine that the website also uses back-end databases, where the databases are in a separate stack that are owned and maintained by database administrators. Whenever the website team or database team needs to update their resources, they can do so without affecting each other's stack. If all resources were in a single stack, coordinating and communicating updates can be difficult.

For additional guidance about organizing your stacks, you can use two common frameworks: a multi-layered architecture and service-oriented architecture (SOA).

A layered architecture organizes stacks into multiple horizontal layers that build on top of one another, where each layer has a dependency on the layer directly below it. You can have one or more stacks in each layer, but within each layer, your stacks should have AWS resources with similar lifecycles and ownership.

With a service-oriented architecture, you can organize big business problems into manageable parts. Each of these parts is a service that has a clearly defined purpose and represents a self-contained unit of functionality. You can map these services to a stack, where each stack has its own lifecycle and owners. All of these services (stacks) can be wired together so that they can interact with one another.

For our example project, we will create 3 stacks: IAM stack, Network stack, and Application stack.

IAM stack will contain all the IAM resources we need for our project, including IAM instance profiles and service roles. Network stack will contain all the network resources like VPC, subnets, security groups, NAT gateways, and Route53 hosted zone. Application stack will contain all the resources for our application, like Auto Scaling Groups, Elastic Load Balancers, and RDS database instances.

³²<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/best-practices.html#organizingstacks>

Use Cross-Stack References to Export Shared Resources

When you're organizing your AWS resources based on lifecycle and ownership, you might want to build a stack that uses resources that are in another stack. You can hard-code values or use input parameters to pass resource names and IDs. However, these methods can make templates difficult to reuse or can increase the overhead to get a stack running. Instead, use cross-stack references to export resources from a stack so that other stacks can use them. Stacks can use the exported resources by calling them using the `Fn::ImportValue` function.

For example, you might have a network stack that includes a VPC, a security group, and a subnet. You want all public web applications to use these resources. By exporting the resources, you allow all stacks with public web applications to use them.

To create a cross-stack reference, use the `Export` output field to flag the value of a resource output for export. Then, use the `Fn::ImportValue` intrinsic function to import the value.

The following restrictions apply to cross-stack references:

- Export names must be unique within an account and 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.
- Cross-stack references can't use a `Ref` or `GetAtt` that depends on another 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.

Instance Profile

Applications that run on an EC2 instance must include AWS credentials in their AWS API requests. You could have your developers store AWS credentials directly within the EC2 instance and allow applications in that instance to use those credentials. But developers would then have to manage the credentials and ensure that they securely pass the credentials to each instance and update each EC2 instance when it's time to rotate the credentials. That's a lot of additional work.³³

Instead, you can and should use an IAM role to manage temporary credentials for applications that run on an EC2 instance. When you use a role, you don't have to distribute long-term credentials to an EC2 instance. Instead, the role supplies temporary permissions that applications can use when they make calls to other AWS resources. When you launch an EC2 instance, you specify an IAM role to associate with the instance. Applications that run on the instance can then use the role-supplied temporary credentials to sign API requests.

Using roles to grant permissions to applications that run on EC2 instances requires a bit of extra configuration. An application running on an EC2 instance is abstracted from AWS by the virtualized

³³http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html

operating system. Because of this extra separation, an additional step is needed to assign an AWS role and its associated permissions to an EC2 instance and make them available to its applications. This extra step is the creation of an *instance profile* that is attached to the instance. The instance profile contains the role and can provide the role's credentials to an application that runs on the instance. Those credentials can then be used in the application's API calls to access resources and to limit access to only those resources that the role specifies. Note that only one role can be assigned to an EC2 instance at a time, and all applications on the instance share the same role and permissions.

Using roles in this way has several benefits. Because role credentials are temporary and rotated automatically, you don't have to manage credentials, and you don't have to worry about long-term security risks. In addition, if you use a single role for multiple instances, you can make a change to that one role and the change is propagated automatically to all the instances.



A role is assigned to an EC2 instance when you launch it. It cannot be assigned to an instance that is already running. If you need to add a role to an instance that is already running, you can create an image of the instance, and then launch a new instance from the image with the desired role assigned.

IAM Template

In our IAM template we will define an instance profile to be used by the application server in our example project. The example instance profile will allow the application server to interact with S3 to download and upload files.

Let's create the IAM template:

```
$ vi iam.yml

1 AWSTemplateFormatVersion: 2010-09-09
2 Description: IAM Stack
3
4 Resources:
5   AppRole:
6     Type: AWS::IAM::Role
7     Properties:
8       AssumeRolePolicyDocument:
9         Version: 2012-10-17
10        Statement:
11          - Effect: Allow
12            Principal:
13              Service:
14                - ec2.amazonaws.com
15            Action:
```

```
16      - sts:AssumeRole
17      Path: /
18  RolePolicies:
19    Type: AWS::IAM::Policy
20    Properties:
21      PolicyName: S3Access
22      PolicyDocument:
23        Version: 2012-10-17
24        Statement:
25          - Effect: Allow
26            Action:
27              - s3:*
28            Resource: '*'
29        Roles:
30          - Ref: AppRole
31  AppInstanceProfile:
32    Type: AWS::IAM::InstanceProfile
33    Properties:
34      Path: /
35      Roles:
36        - Ref: AppRole
37 Outputs:
38   AppInstanceProfile:
39     Description: Application EC2 Instance Profile
40     Value: !Ref AppInstanceProfile
41     Export:
42       Name: !Sub ${AWS::StackName}-AppInstanceProfile
```

Next, create the CloudFormation stack:

1. Open the CloudFormation console at <https://console.aws.amazon.com/cloudformation/>.
2. Click **Create Stack**.
3. On the **Select Template** page, choose **Upload a template to Amazon S3** and select the `iam.yaml` template.
4. Click **Next**.
5. On the **Specify Details** page, specify the stack name, for example `test-iam`.
6. Click **Next**.
7. On the **Options** page, click **Next**.
8. On the **Review** page, select the **I acknowledge that AWS CloudFormation might create IAM resources** box, then click **Create**.
9. Your IAM stack appears in the list of AWS CloudFormation stacks.

The IAM stack creates 3 AWS resources: IAM Role (`AWS::IAM::Role`), IAM Policy (`AWS::IAM::Policy`), and IAM Instance Profile (`AWS::IAM::InstanceProfile`). It also creates an output `AppInstanceProfile`, which contains the IAM Instance Profile name, and export this output so it can be referenced by other stacks. If the stack name is `test-iam` the export Name will be `test-iam-AppInstanceProfile`.

You can open the IAM console at <https://console.aws.amazon.com/iam/> to see the IAM Role. Click **Roles** in the navigation pane and select the role. On the **Permissions** tab, under **Inline Policies** section you can click **Show Policy** to review the IAM Policy.

AWS::IAM::Role

Creates an AWS Identity and Access Management (IAM) role. Use an IAM role to enable applications running on an EC2 instance to securely access your AWS resources.

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-iam-role.html>

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::IAM::Role"
2 Properties:
3   AssumeRolePolicyDocument:
4     <JSON object>
5   ManagedPolicyArns:
6     - <String>
7   Path: <String>
8   Policies:
9     - <Policies>
10  RoleName: <String>
```

Properties

AssumeRolePolicyDocument

The trust policy that is associated with this role.

Required: Yes

Type: A JSON policy document

Update requires: No interruption

Note: You can associate only one assume role policy with a role.

ManagedPolicyArns

One or more managed policy ARNs to attach to this role.

Required: No

Type: List of strings

Update requires: No interruption

Path

The path associated with this role. For information about IAM paths, see http://docs.aws.amazon.com/IAM/latest/UserGuide/Using_Identifiers.html#Identifiers_FriendlyNames.

Required: No

Type: String

Update requires: Replacement

Policies

The policies to associate with this role. The name of each policy for a role, user, or group must be unique.

Required: No

Type: List of IAM Policies

Update requires: No interruption

RoleName

A name for the IAM role. For valid values, see the `RoleName` parameter for the `CreateRole` action in the IAM API Reference http://docs.aws.amazon.com/IAM/latest/APIReference/API_CreateRole.html. If you don't specify a name, AWS CloudFormation generates a unique physical ID and uses that ID for the group name.

Required: No

Type: String

Update requires: Replacement



If you specify a `RoleName`, you cannot do updates that require this resource to be replaced.

You can still do updates that require no or some interruption. If you must replace the resource, specify a new name. If you specify a name, you must specify the `CAPABILITY_NAMED_IAM` value to acknowledge your template's capabilities.



Naming an IAM resource can cause an unrecoverable error if you reuse the same template in multiple regions. To prevent this, you can use `Fn::Join` and `AWS::Region` to create a region-specific name, as in the following example: `!Join ["", [!Ref "AWS::Region", !Ref "MyResourceName"]]`.

Return Values

Ref

When the logical ID of this resource is provided to the `Ref` intrinsic function, `Ref` returns the resource name. For example:

```
Ref: "RootRole"
```

For the `IAM::Role` with the logical ID “RootRole”, `Ref` will return the resource name.

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and sample return values.

`Arn` Returns the Amazon Resource Name (ARN) for the instance profile. For example:

```
"Fn::GetAtt" : ["MyRole", "Arn"]
```

This will return a value such as “arn:aws:iam::1234567890:role/MyRole-AJJHDSKSDF”.

AWS::IAM::Policy

Associates an IAM policy with IAM users, roles, or groups.

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-iam-policy.html>

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::IAM::Policy"
2 Properties:
3   Groups:
4     - <String>
5   PolicyDocument: <JSON object>
6   PolicyName: <String>
7   Roles:
8     - <String>
9   Users:
10    - <String>
```

Properties

Groups

The names of groups to which you want to add the policy.

Required: Conditional. You must specify at least one of the following properties: Groups, Roles, or Users.

Type: List of strings

Update requires: No interruption

PolicyDocument

A policy document that contains permissions to add to the specified users or groups.

Required: Yes

Type: JSON object

Update requires: No interruption

PolicyName

The name of the policy. If you specify multiple policies for an entity, specify unique names. For example, if you specify a list of policies for an IAM role, each policy must have a unique name.

Required: Yes

Type: String

Update requires: No interruption

Roles

The names of AWS::IAM::Roles to attach to this policy.

Required: Conditional. You must specify at least one of the following properties: Groups, Roles, or Users.

Type: List of strings

Update requires: No interruption

Note: If a policy has a Ref to a role and if a resource (such as AWS::ECS::Service) also has a Ref to the same role, add a DependsOn attribute to the resource so that the resource depends on the policy. This dependency ensures that the role's policy is available throughout the resource's lifecycle. For example, when you delete a stack with an AWS::ECS::Service resource, the DependsOn attribute ensures that the AWS::ECS::Service resource can complete its deletion before its role's policy is deleted.

Users

The names of users for whom you want to add the policy.

Required: Conditional. You must specify at least one of the following properties: Groups, Roles, or Users.

Type: List of strings

Update requires: No interruption

Return Values

Ref

When the logical ID of this resource is provided to the Ref intrinsic function, Ref returns the resource name.

AWS::IAM::InstanceProfile

Creates an AWS Identity and Access Management (IAM) Instance Profile that can be used with IAM Roles for EC2 Instances.

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-iam-instanceprofile.html>

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::IAM::InstanceProfile"
2 Properties:
3   Path: String
4   Roles:
5     - IAM Roles
```

Properties

Path The path associated with this IAM instance profile. For information about IAM paths, see http://docs.aws.amazon.com/IAM/latest/UserGuide/Using_Identifiers.html#Identifiers_FriendlyNames.

Required: Yes

Type: String

Update requires: Replacement

Roles

The roles associated with this IAM instance profile.

Required: Yes

Type: List of references to AWS::IAM::Roles. Currently, a maximum of one role can be assigned to an instance profile.

Update requires: No interruption

Return Values

Ref

When the logical ID of this resource is provided to the Ref intrinsic function, Ref returns the resource name. For example:

```
Ref: "MyProfile"
```

For the `IAM::InstanceProfile` with the logical ID “MyProfile”, `Ref` will return the resource name.

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and sample return values.

`Arn`

Returns the Amazon Resource Name (ARN) for the instance profile. For example:

```
"Fn::GetAtt" : ["MyProfile", "Arn"]
```

This will return a value such as “arn:aws:iam::1234567890:instance-profile/MyProfile-ASDSDLKJ”.

Chapter 4: The Network Stack

Introduction to VPC

VPCs and Subnets

A *virtual private cloud* (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS cloud. You can launch your AWS resources, such as Amazon EC2 instances, into your VPC. You can configure your VPC; you can select its IP address range, create subnets, and configure route tables, network gateways, and security settings.³⁴

A *subnet* is a range of IP addresses in your VPC. You can launch AWS resources into a subnet that you select. Use a public subnet for resources that must be connected to the Internet, and a private subnet for resources that won't be connected to the Internet.

To protect the AWS resources in each subnet, you can use multiple layers of security, including security groups and network access control lists (ACL).

Supported Platforms

The original release of Amazon EC2 supported a single, flat network that's shared with other customers called the *EC2-Classic* platform. Older AWS accounts still support this platform, and can launch instances into either EC2-Classic or a VPC. Accounts created after 2013-12-04 support EC2-VPC only. For more information, see <http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/default-vpc.html#detecting-platform>.

By launching your instances into a VPC instead of EC2-Classic, you gain the ability to:

- Assign static private IP addresses to your instances that persist across starts and stops
- Assign multiple IP addresses to your instances
- Define network interfaces, and attach one or more network interfaces to your instances
- Change security group membership for your instances while they're running
- Control the outbound traffic from your instances (egress filtering) in addition to controlling the inbound traffic to them (ingress filtering)
- Add an additional layer of access control to your instances in the form of network access control lists (ACL)
- Run your instances on single-tenant hardware

³⁴http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Introduction.html

Default and Nondefault VPCs

If your account supports the EC2-VPC platform only, it comes with a *default* VPC that has a *default* subnet in each Availability Zone. A default VPC has the benefits of the advanced features provided by EC2-VPC, and is ready for you to use. If you have a default VPC and don't specify a subnet when you launch an instance, the instance is launched into your default VPC. You can launch instances into your default VPC without needing to know anything about Amazon VPC.

Regardless of which platforms your account supports, you can create your own VPC, and configure it as you need. This is known as a *nondefault* VPC. Subnets that you create in your *nondefault* VPC and additional subnets that you create in your default VPC are called *nondefault* subnets.

Accessing the Internet

You control how the instances that you launch into a VPC access resources outside the VPC.

Your default VPC includes an Internet gateway, and each default subnet is a public subnet. Each instance that you launch into a default subnet has a private IP address and a public IP address. These instances can communicate with the Internet through the Internet gateway. An Internet gateway enables your instances to connect to the Internet through the Amazon EC2 network edge.

By default, each instance that you launch into a nondefault subnet has a private IP address, but no public IP address, unless you specifically assign one at launch, or you modify the subnet's public IP address attribute. These instances can communicate with each other, but can't access the Internet.

You can enable Internet access for an instance launched into a nondefault subnet by attaching an Internet gateway to its VPC (if its VPC is not a default VPC) and associating an Elastic IP address with the instance.

Alternatively, to allow an instance in your VPC to initiate outbound connections to the Internet but prevent unsolicited inbound connections from the Internet, you can use a network address translation (NAT) device. NAT maps multiple private IP addresses to a single public IP address. A NAT device has an Elastic IP address and is connected to the Internet through an Internet gateway. You can connect an instance in a private subnet to the Internet through the NAT device, which routes traffic from the instance to the Internet gateway, and routes any responses to the instance.

Accessing a Corporate or Home Network

You can optionally connect your VPC to your own corporate data center using an IPsec hardware VPN connection, making the AWS cloud an extension of your data center.

A VPN connection consists of a virtual private gateway attached to your VPC and a customer gateway located in your data center. A virtual private gateway is the VPN concentrator on the Amazon side of the VPN connection. A customer gateway is a physical device or software appliance on your side of the VPN connection.

The Default VPC

If you created your AWS account after 2013-12-04, it supports only EC2-VPC. In this case, you'll have a default VPC in each AWS region. A default VPC is ready for you to use — you can immediately start launching instances into your default VPC without having to perform any additional configuration steps. A default VPC combines the benefits of the advanced networking features provided by the EC2-VPC platform with the ease of use of the EC2-Classic platform.

Amazon creates a default VPC with the following set up:

- a default subnet in each Availability Zone
- an Internet gateway connected to your default VPC
- a main route table for your default VPC with a rule that sends all traffic destined for the Internet to the Internet gateway
- a default security group associated with your default VPC
- a default network access control list (ACL) associated with your VPC
- a default DHCP options set for your AWS account associated with your default VPC



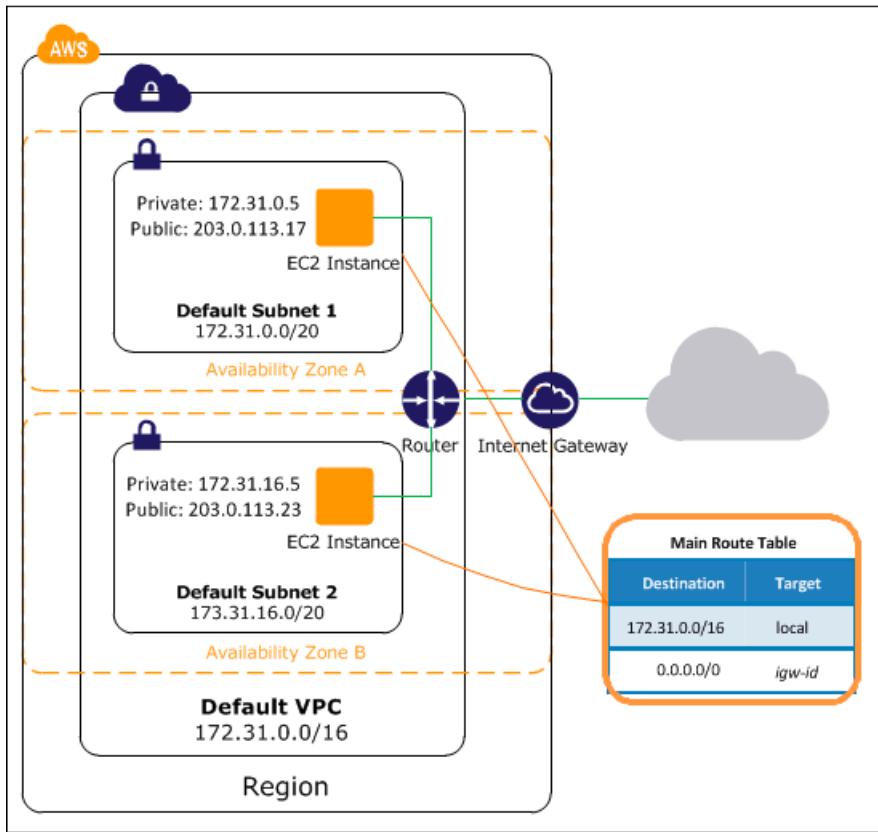
Amazon has multiple locations world-wide. These locations are composed of regions and Availability Zones. Each *region* is a separate geographic area. Each region has multiple, isolated locations known as *Availability Zones*. Amazon provides you the ability to place resources, such as instances, and data in multiple locations. Resources aren't replicated across regions unless you do so specifically.



A *network access control list (ACL)* is an optional layer of security that acts as a firewall for controlling traffic in and out of a subnet. You might set up network ACLs with rules similar to your security groups in order to add an additional layer of security to your VPC.

The following figure illustrates the key components that Amazon set up for a default VPC:³⁵

³⁵<http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/default-vpc.html>



The CIDR block for a default VPC is always `172.31.0.0/16`, which provides up to 65,536 private IP addresses. A default subnet has a /20 subnet mask, which provides up to 4,096 addresses per subnet. Some addresses are reserved for Amazon's use.

By default, a default subnet is connected to the Internet through the Internet gateway. Instances that you launch into a default subnet receive both a private IP address and a public IP address.

VPC and Subnet Basics

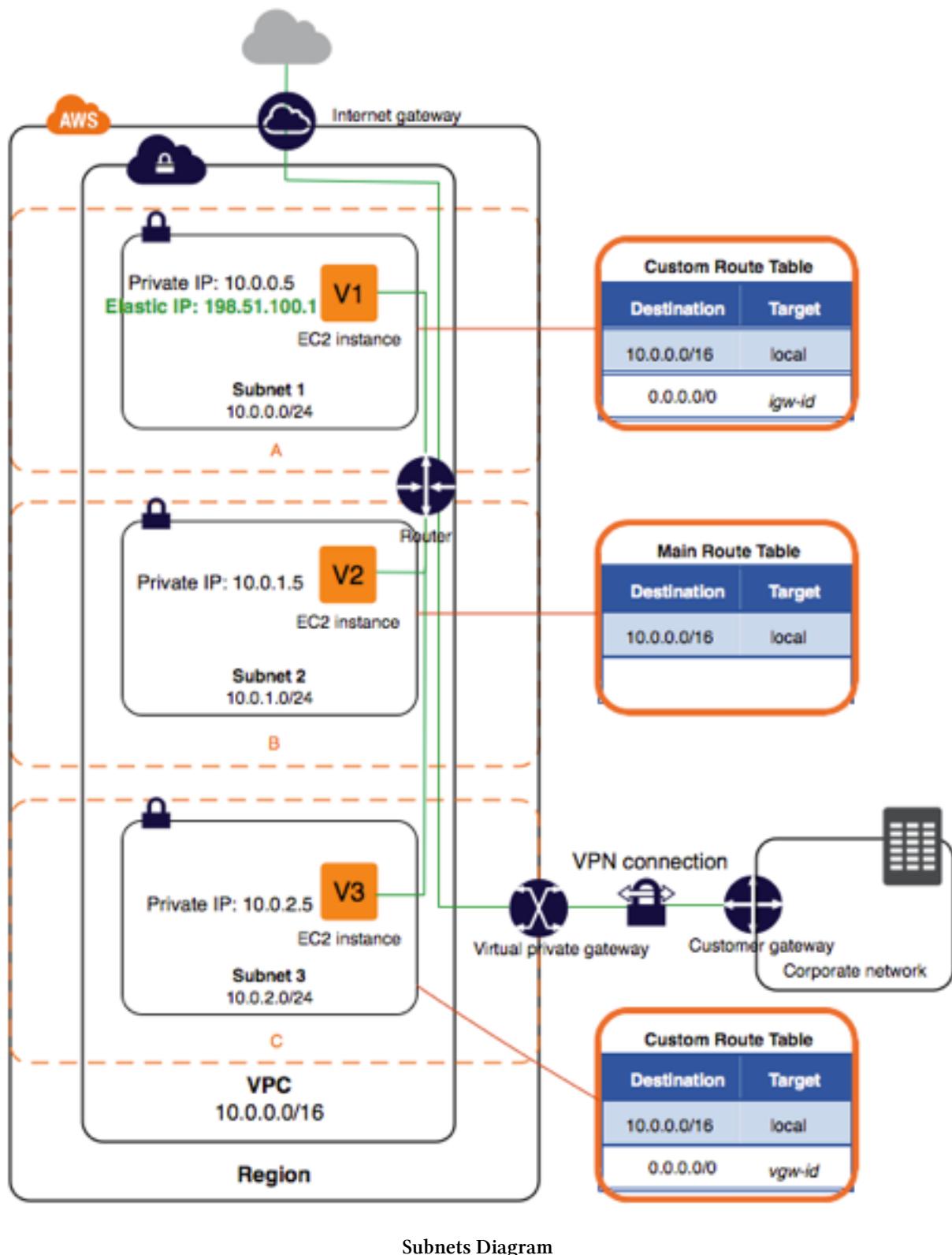
A virtual private cloud (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS cloud. You can launch your AWS resources, such as Amazon EC2 instances, into your VPC.³⁶

When you create a VPC, you specify the range of IP addresses for the VPC in the form of a Classless Inter-Domain Routing (CIDR) block; for example, 10.0.0.0/16. For more information about CIDR notation and what “/16” means, see <https://tools.ietf.org/html/rfc4632>.

When you create a VPC, it spans all of the Availability Zones in the region. After creating a VPC, you can add one or more subnets in each Availability Zone. When you create a subnet, you specify the CIDR block for the subnet, which is a subset of the VPC CIDR block. Each subnet must reside entirely within one Availability Zone and cannot span zones. Availability Zones are distinct locations that are engineered to be isolated from failures in other Availability Zones. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location. AWS assigns a unique ID to each subnet.

The following diagram shows a VPC that has been configured with subnets in multiple Availability Zones. In this scenario, an Internet gateway enables communication over the Internet, and a virtual private network (VPN) connection enables communication with your corporate network.

³⁶http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Subnets.html



Subnets Diagram

If a subnet's traffic is routed to an Internet gateway, the subnet is known as a public subnet. In this diagram, subnet 1 is a public subnet. If you want your instance in a public subnet to communicate with the Internet, it must have a public IP address or an Elastic IP address. For more information about public IP addresses, see <http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-ip-addressing.html#vpc-public-ip-addresses>.



Regardless of the type of subnet, the internal IP address range of the subnet is always private
- Amazon does not announce the address block to the Internet. For more information, see <http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-ip-addressing.html>.

You have a limit on the number of VPCs and subnets you can create in your account. For more information, see http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Appendix_Limits.html.

VPC and Subnet Sizing

You can assign a single CIDR block to a VPC. The allowed block size is between a /28 netmask and /16 netmask. In other words, the VPC can contain from 16 to 65,536 IP addresses.

When you create a VPC, it's recommended that you specify a CIDR block from the private (non-publicly routable) IP address ranges as specified in RFC 1918 <http://www.faqs.org/rfcs/rfc1918.html>:

- 10.0.0.0 - 10.255.255.255 (10/8 prefix)
- 172.16.0.0 - 172.31.255.255 (172.16/12 prefix)
- 192.168.0.0 - 192.168.255.255 (192.168/16 prefix)

You can't change the size of a VPC after you create it. If your VPC is too small to meet your needs, create a new, larger VPC, and then migrate your instances to the new VPC. To do this, create AMIs from your running instances, and then launch replacement instances in your new, larger VPC. You can then terminate your old instances, and delete your smaller VPC.

The CIDR block of a subnet can be the same as the CIDR block for the VPC (for a single subnet in the VPC), or a subset (for multiple subnets). The allowed block size is between a /28 netmask and /16 netmask. If you create more than one subnet in a VPC, the CIDR blocks of the subnets cannot overlap.

For example, if you create a VPC with CIDR block 10.0.0.0/24, it supports 256 IP addresses. You can break this CIDR block into two subnets, each supporting 128 IP addresses. One subnet uses CIDR block 10.0.0.0/25 (for addresses 10.0.0.0 - 10.0.0.127) and the other uses CIDR block 10.0.0.128/25 (for addresses 10.0.0.128 - 10.0.0.255).

There are many tools available to help you calculate subnet CIDR blocks; for example, see <http://www.subnet-calculator.com/cidr.php>. Also, your network engineering group can help you determine the CIDR blocks to specify for your subnets.

The first four IP addresses and the last IP address in each subnet CIDR block are not available for you to use, and cannot be assigned to an instance. For example, in a subnet with CIDR block `10.0.0.0/24`, the following five IP addresses are reserved:

- `10.0.0.0`: Network address.
- `10.0.0.1`: Reserved by AWS for the VPC router.
- `10.0.0.2`: Reserved by AWS. The IP address of the DNS server is always the base of the VPC network range plus two; however, AWS also reserves the base of each subnet range plus two. For more information, see http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_DHCP_Options.html#AmazonDNS.
- `10.0.0.3`: Reserved by AWS for future use.
- `10.0.0.255`: Network broadcast address. AWS does not support broadcast in a VPC, therefore the address is reserved.

Subnet Routing

By design, each subnet must be associated with a route table, which specifies the allowed routes for outbound traffic leaving the subnet. Every subnet that you create is automatically associated with the main route table for the VPC. You can change the association, and you can change the contents of the main route table. For more information, see http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Route_Tables.html.

In the previous diagram, the route table associated with subnet 1 routes all traffic (`0.0.0.0/0`) to an Internet gateway (for example, `igw-1a2b3c4d`). Because instance V1 has an Elastic IP address, it can be reached from the Internet.



The Elastic IP address or public IP address that's associated with your instance is accessed through the Internet gateway of your VPC. Traffic that goes through a VPN connection between your instance and another network traverses a virtual private gateway, not the Internet gateway, and therefore does not access the Elastic IP address or public IP address.

The instance V2 can't reach the Internet, but can reach other instances in the VPC. You can allow an instance in your VPC to initiate outbound connections to the Internet but prevent unsolicited inbound connections from the Internet using a network address translation (NAT) gateway or instance. Because you can allocate a limited number of Elastic IP addresses, it's recommended that you use a NAT device if you have more instances that require a static public IP address. For more information, see <http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-nat.html>.

The route table associated with subnet 3 routes all traffic (`0.0.0.0/0`) to a virtual private gateway (for example, `vgw-1a2b3c4d`). Instance V3 can reach computers in the corporate network over the VPN connection.

Subnet Security

AWS provides two features that you can use to increase security in your VPC: *security groups* and *network ACLs*. Security groups control inbound and outbound traffic for your instances, and network ACLs control inbound and outbound traffic for your subnets. In most cases, security groups can meet your needs; however, you can also use network ACLs if you want an additional layer of security for your VPC. For more information, see http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Security.html.

By design, each subnet must be associated with a network ACL. Every subnet that you create is automatically associated with the VPC's default network ACL. You can change the association, and you can change the contents of the default network ACL. For more information, see http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_ACLs.html.

You can create a flow log on your VPC or subnet to capture the traffic that flows to and from the network interfaces in your VPC or subnet. You can also create a flow log on an individual network interface. Flow logs are published to CloudWatch Logs. For more information, see <http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/flow-logs.html>.

Connections with Your Local Network and Other VPCs

You can optionally set up a connection between your VPC and your corporate or home network. If you have an IP address prefix in your VPC that overlaps with one of your networks' prefixes, any traffic to the network's prefix is dropped. For example, let's say that you have the following:

- A VPC with CIDR block `10.0.0.0/16`
- A subnet in that VPC with CIDR block `10.0.1.0/24`
- Instances running in that subnet with IP addresses `10.0.1.4` and `10.0.1.5`
- On-premises host networks using CIDR blocks `10.0.37.0/24` and `10.1.38.0/24`

When those instances in the VPC try to talk to hosts in the `10.0.37.0/24` address space, the traffic is dropped because `10.0.37.0/24` is part of the larger prefix assigned to the VPC (`10.0.0.0/16`). The instances can talk to hosts in the `10.1.38.0/24` space because that block isn't part of `10.0.0.0/16`.

You can also create a VPC peering connection between your VPCs, or with a VPC in another AWS account. A VPC peering connection enables you to route traffic between the VPCs using private IP addresses; however, you cannot create a VPC peering connection between VPCs that have overlapping CIDR blocks. For more information, see <http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-peering.html>.

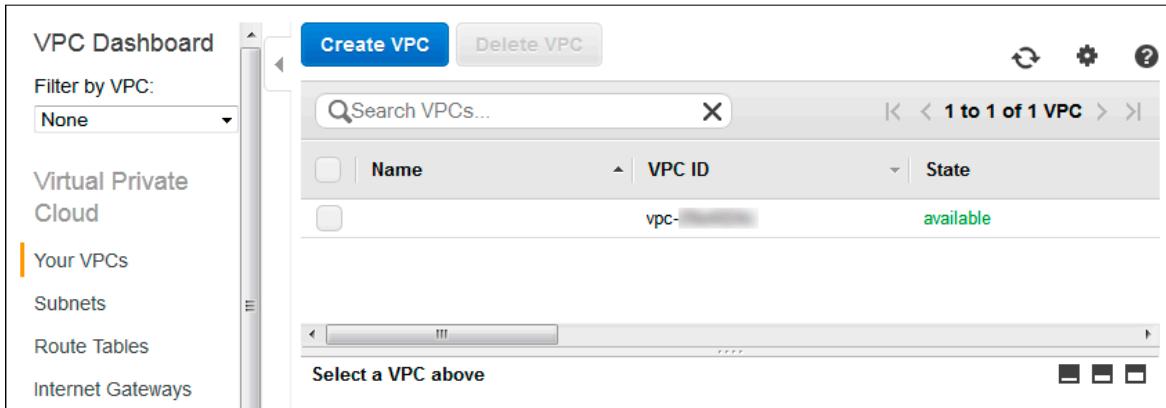
It's recommended that you create a VPC with a CIDR range large enough for expected future growth, but not one that overlaps with current or expected future subnets anywhere in your corporate or home network, or that overlaps with current or future VPCs.

Getting Started with VPCs and Subnets

You can create a VPC and subnets using the Amazon VPC console. The following procedures are for manually creating a VPC, subnets, internet gateway, and route tables.

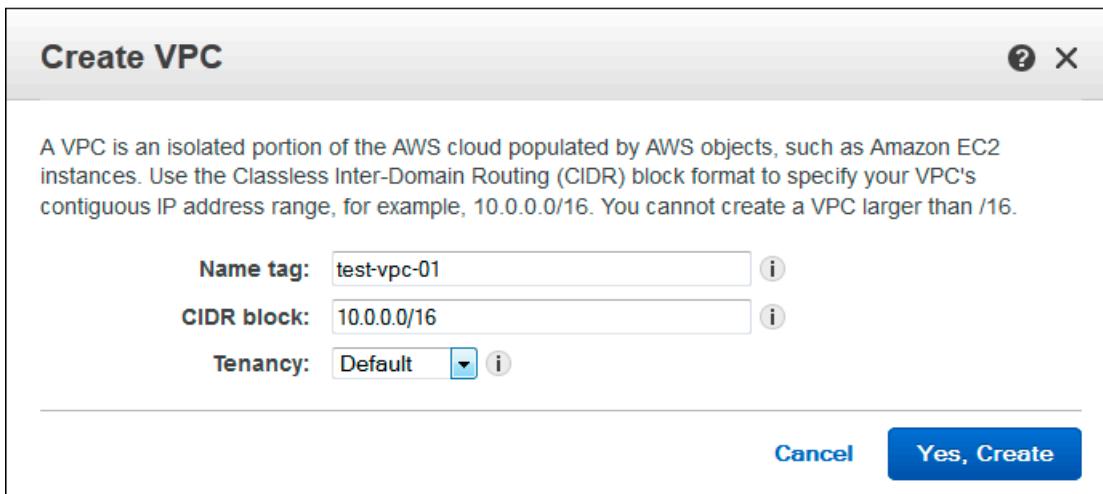
To create a VPC:

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, click Your VPCs.



The screenshot shows the VPC Dashboard. On the left, there's a sidebar with options: Virtual Private Cloud, Your VPCs (which is selected and highlighted in orange), Subnets, Route Tables, and Internet Gateways. The main area has a 'Create VPC' button at the top. Below it is a search bar labeled 'Search VPCs...'. A table lists one VPC entry: 'Name' is 'vpc-' and 'State' is 'available'. At the bottom, there's a note 'Select a VPC above' and some icons.

3. Click Create VPC.
4. Enter the desired Name tag and CIDR block for the VPC and select Tenancy: Default.



The screenshot shows the 'Create VPC' dialog. It includes a descriptive text about VPCs, input fields for 'Name tag' (set to 'test-vpc-01'), 'CIDR block' (set to '10.0.0.0/16'), and 'Tenancy' (set to 'Default'). At the bottom, there are 'Cancel' and 'Yes, Create' buttons.

5. Click Yes, Create button.



Each VPC has a related instance tenancy attribute. You can't change the instance tenancy of a VPC after you create it. A dedicated VPC tenancy attribute means that all instances launched into the VPC are Dedicated Instances, regardless of the value of the tenancy attribute for the instance. If you set this to default, then the tenancy attribute will follow the tenancy attribute setting on each instance. A default tenancy instance runs on shared hardware. A dedicated tenancy instance runs on single-tenant hardware.

Dedicated Instances are Amazon EC2 instances that run in a virtual private cloud (VPC) on hardware that's dedicated to a single customer. Your Dedicated Instances are physically isolated at the host hardware level from your instances that aren't Dedicated Instances and from instances that belong to other AWS accounts. To see the pricing for dedicated instances go to <http://aws.amazon.com/ec2/purchasing-options/dedicated-instances/>.

To create an Internet gateway and attach it to a VPC:

1. In the VPC Dashboard navigation pane, click **Internet Gateways**.

	Name	ID	State
<input type="checkbox"/>	igw-XXXXXX	igw-XXXXXX	attached

2. Click **Create Internet Gateway**.
3. Enter the desired Name tag for this Internet gateway.

Create Internet Gateway

An Internet gateway is a virtual router that connects a VPC to the Internet.

Name tag: test-igw-01

Cancel Yes, Create

4. Click **Yes, Create** button.

5. Select the new Internet gateway and click Attach to VPC

Name	ID	State
test-igw-01	igw-[REDACTED]	detached
	igw-[REDACTED]	attached

6. Select the desired VPC and click Yes, Attach.

Attach to VPC

Attach an Internet gateway to a VPC to enable communication with the Internet.

VPC: vpc-[REDACTED] (10.0.0.0/16) | test-vpc-01

Cancel Yes, Attach

To create subnets on the VPC:

1. In the VPC Dashboard navigation pane, click Subnets.

Name	Subnet ID	State
subnet-[REDACTED]	[REDACTED]	available
subnet-[REDACTED]	[REDACTED]	available

2. Click Create Subnet

3. Enter the desired Name tag and CIDR block for the public subnet, select VPC and Availability Zone.

Create Subnet ? X

Use the CIDR format to specify your subnet's IP address block (e.g., 10.0.0.0/24). Note that block sizes must be between a /16 netmask and /28 netmask. Also, note that a subnet can be the same size as your VPC.

Name tag:	test-subnet-public	i
VPC:	vpc- (10.0.0.0/16) test-vpc-01	i
Availability Zone:	ap-southeast-2a	i
CIDR block:	10.0.0.0/24	i

Cancel Yes, Create

4. Click Yes, Create.
5. Click Create Subnet.

Enter the desired Name tag and CIDR block for the private subnet, select VPC and Availability Zone.

Create Subnet ? X

Use the CIDR format to specify your subnet's IP address block (e.g., 10.0.0.0/24). Note that block sizes must be between a /16 netmask and /28 netmask. Also, note that a subnet can be the same size as your VPC.

Name tag:	test-subnet-private	i
VPC:	vpc- (10.0.0.0/16) test-vpc-01	i
Availability Zone:	ap-southeast-2a	i
CIDR block:	10.0.1.0/24	i

Cancel Yes, Create

6. Click Yes, Create.

Route Tables

The following are the basic things that you need to know about route tables:³⁷

- Your VPC has an implicit router.
- Your VPC automatically comes with a main route table that you can modify.
- You can create additional custom route tables for your VPC.
- Each subnet must be associated with a route table, which controls the routing for the subnet. If you don't explicitly associate a subnet with a particular route table, the subnet uses the main route table.
- You can replace the main route table with a custom table that you've created (so that this table is the default table each new subnet is associated with).
- Each route in a table specifies a destination CIDR and a target.

Main Route Table

When you create a VPC, it automatically has a main route table. To see the main route tables for your VPC, in the VPC Dashboard navigation pane, click **Route Tables**.

The screenshot shows the AWS VPC Dashboard. On the left, there's a navigation pane with options like Virtual Private Cloud, Your VPCs, Subnets, Route Tables (which is selected and highlighted in orange), Internet Gateways, DHCP Options Sets, Elastic IPs, and Peering Connections. The main area is titled 'Route Tables' and shows a search bar with 'rtb-' and a result table. The table has columns for Name, Route Table ID, Associated With, and Main. One row is shown with 'rtb-' as the name, 'rtb-' as the ID, '0 Subnets' as the count, and 'Yes' as the main status. Below the table is a summary section with tabs for Summary, Routes (which is selected and highlighted in orange), Subnet Associations, Route Propagation, and Tags. The 'Routes' tab shows a single entry: Destination 10.0.0.0/16, Target local, Status Active, and Propagated No. There are also 'Edit' and 'Delete' buttons at the top of this section.

The Main Route Table

Initially, the main route table (and every route table in a VPC) contains only a single route: a local route that enables communication within the VPC. You can't modify the local route in a route table. Whenever you launch an instance in the VPC, the local route automatically covers that instance; you don't need to add the new instance to a route table.

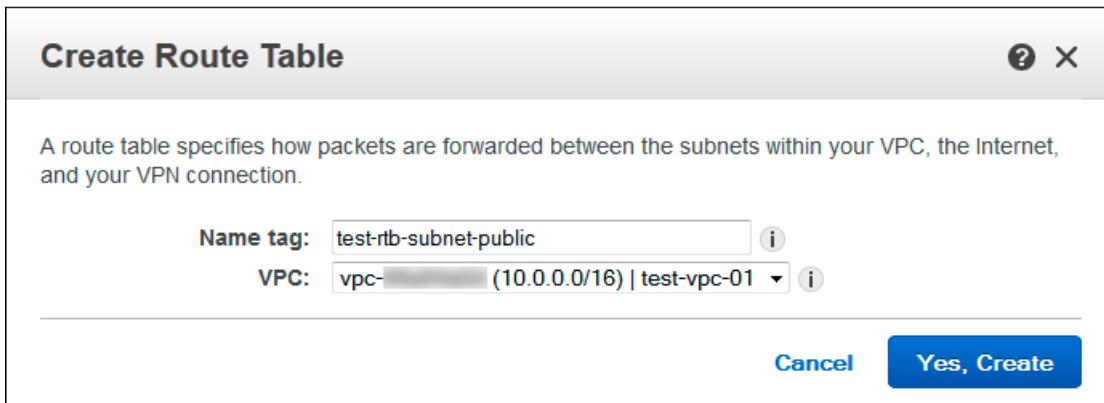
³⁷http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Route_Tables.html

If you don't explicitly associate a subnet with a route table, the subnet is implicitly associated with the main route table. However, you can still explicitly associate a subnet with the main route table. You might do that if you change which table is the main route table. The console shows the number of subnets associated with each table. Only explicit associations are included in that number.

Custom Route Tables

To create a custom route table for the public VPC:

1. In the VPC Dashboard navigation pane, click **Route Tables**.
2. Click **Create Route Table**.
3. Enter the desired **Name tag** and select **VPC**.



4. Click **Yes, Create**.

To edit the custom route table and associate it to the public subnet:

1. In the VPC Dashboard navigation pane, click **Route Tables**. Select the custom route table we created for public subnet and select the **Routes** tab.

Name	Route Table ID	Associated With	Main
test-rtb-subnet-public	rtb-[REDACTED]	0 Subnets	No
	rtb-[REDACTED]	0 Subnets	Yes
	rtb-[REDACTED]	0 Subnets	Yes

rtb-[REDACTED] | test-rtb-subnet-public

Summary Routes Subnet Associations Route Propagation Tags

Edit

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No

2. Click Edit button.
3. Enter $0.0.0.0/0$ CIDR block in the Destination field and select the Internet gateway from the Target list.

Summary Routes Subnet Associations Route Propagation Tags

Cancel Save

Destination	Target	Status	Propagated	Remove
10.0.0.0/16	local	Active	No	
0.0.0.0/0	igw-[REDACTED]		No	✖

Add another route

4. Click Save.
5. Select Subnet Associations tab.

rtb- | test-rtb-subnet-public

Summary Routes Subnet Associations Route Propagation

Edit

Subnet	CIDR
You do not have any subnet associations.	
The following subnets have not been associated with any route tables and are therefore using the main table routes:	
Subnet	CIDR
subnet- (10.0.0.0/24) test-subnet-public	10.0.0.0/24
subnet- (10.0.1.0/24) test-subnet-private	10.0.1.0/24

6. Click Edit.
7. Select the Associate check box for public subnet and click Save.

rtb- | test-rtb-subnet-public

Summary Routes Subnet Associations Route Propagation Tags

Cancel Save

Associate	Subnet	CIDR	Current Route Table
<input checked="" type="checkbox"/>	subnet- (10.0.0.0/24) test-subnet-public	10.0.0.0/24	Main
<input type="checkbox"/>	subnet- (10.0.1.0/24) test-subnet-private	10.0.1.0/24	Main

NAT Gateways

You can use a network address translation (NAT) gateway to enable instances in a private subnet to connect to the Internet or other AWS services, but prevent the Internet from initiating a connection with those instances.

You are charged for creating and using a NAT gateway in your account. NAT gateway hourly usage and data processing rates apply. Amazon EC2 charges for data transfer also apply. For more information, see <http://aws.amazon.com/vpc/pricing/>.

NAT Gateway Basics

To create a NAT gateway, you must specify the public subnet in which the NAT gateway will reside. You must also specify an Elastic IP address to associate with the NAT gateway when you create it. After you've created a NAT gateway, you must update the route table associated with one or more of your private subnets to point Internet-bound traffic to the NAT gateway. This enables instances in your private subnets to communicate with the Internet.³⁸

Each NAT gateway is created in a specific Availability Zone and implemented with redundancy in that zone. You have a limit on the number of NAT gateways you can create in an Availability Zone. For more information, see http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Appendix_Limits.html.



If you have resources in multiple Availability Zones and they share one NAT gateway, in the event that the NAT gateway's Availability Zone is down, resources in the other Availability Zones lose Internet access. To create an Availability Zone-independent architecture, create a NAT gateway in each Availability Zone and configure your routing to ensure that resources use the NAT gateway in the same Availability Zone.

If you no longer need a NAT gateway, you can delete it. Deleting a NAT gateway disassociates its Elastic IP address, but does not release the address from your account.

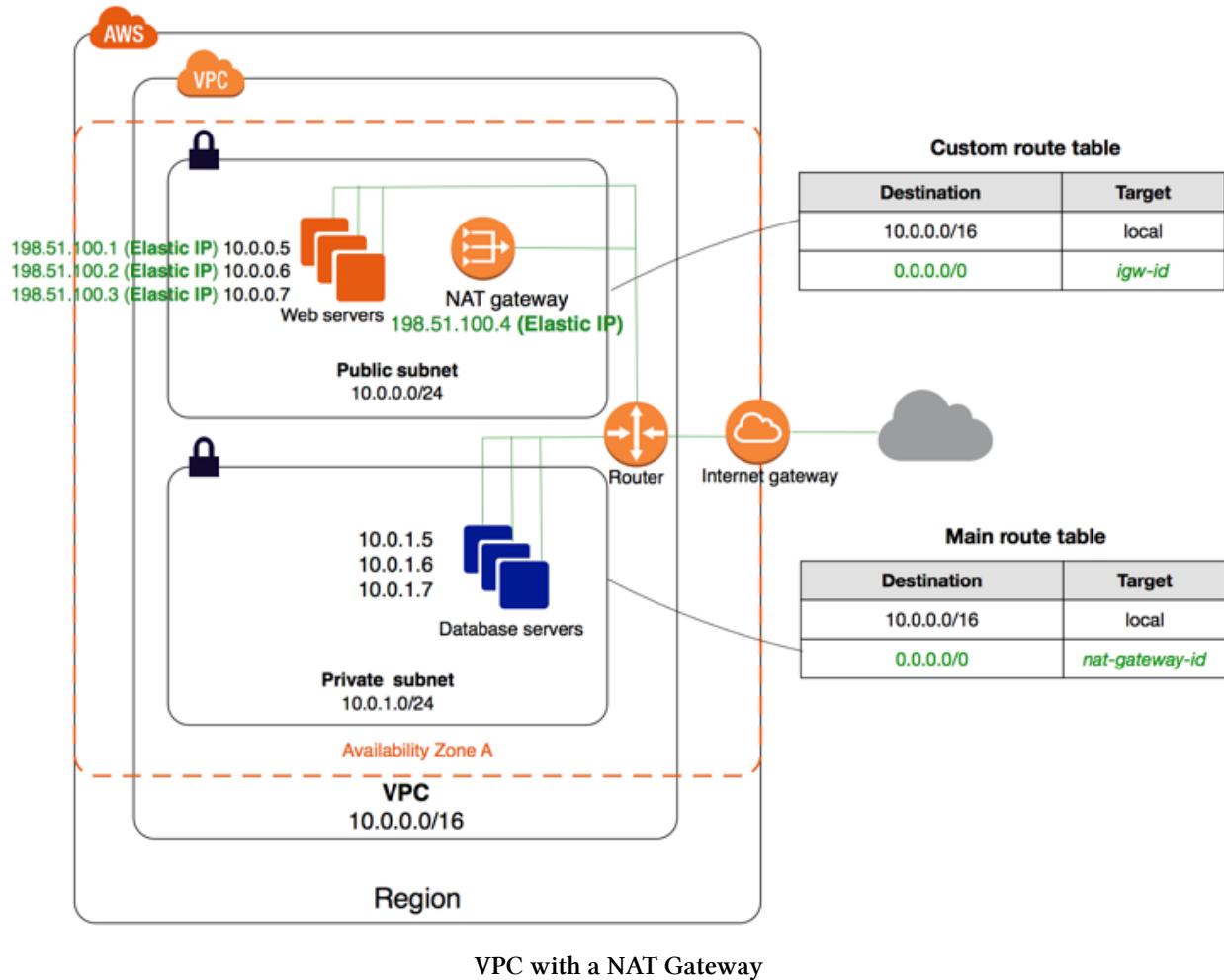
A NAT gateway has the following characteristics:

- A NAT gateway supports bursts of up to 10 Gbps of bandwidth. If you require more than 10 Gbps bursts, you can distribute the workload by splitting your resources into multiple subnets, and creating a NAT gateway in each subnet.
- You can associate exactly one Elastic IP address with a NAT gateway. You cannot disassociate an Elastic IP address from a NAT gateway after it's created. If you need to use a different Elastic IP address for your NAT gateway, you must create a new NAT gateway with the required address, update your route tables, and then delete the existing NAT gateway if it's no longer required.

³⁸<http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-nat-gateway.html>

- A NAT gateway supports the following protocols: TCP, UDP, and ICMP.
- You cannot associate a security group with a NAT gateway. You can use security groups for your instances in the private subnets to control the traffic to and from those instances.
- You can use a network ACL to control the traffic to and from the subnet in which the NAT gateway is located. The network ACL applies to the NAT gateway's traffic. A NAT gateway uses ports 1024 - 65535.
- When a NAT gateway is created, it receives an elastic network interface that's automatically assigned a private IP address from the IP address range of your subnet. You can view the NAT gateway's network interface in the Amazon EC2 console. You cannot modify the attributes of this network interface.
- A NAT gateway cannot be accessed by a ClassicLink connection associated with your VPC.

The following diagram illustrates the architecture of a VPC with a NAT gateway. The main route table sends Internet traffic from the instances in the private subnet to the NAT gateway. The NAT gateway sends the traffic to the Internet gateway using the NAT gateway's Elastic IP address as the source IP address.



Using a NAT Gateway with VPC Endpoints, VPN, AWS Direct Connect, or VPC Peering

A NAT gateway cannot send traffic over VPC endpoints, VPN connections, AWS Direct Connect, or VPC peering connections. If your instances in the private subnet need to access resources over a VPC endpoint, a VPN connection, or AWS Direct Connect, use the private subnet's route table to route the traffic directly to these devices.

For example, your private subnet's route table has the following routes: Internet-bound traffic (0.0.0.0/0) is routed to a NAT gateway, Amazon S3 traffic (p1-xxxxxxxx; a specific IP address range for Amazon S3) is routed to a VPC endpoint, and 10.25.0.0/16 traffic is routed to a VPC peering connection. The p1-xxxxxxxx and 10.25.0.0/16 IP address ranges are more specific than 0.0.0.0/0; when your instances send traffic to Amazon S3 or the peered VPC, the traffic is sent to the VPC endpoint or the VPC peering connection. When your instances send traffic to the Internet (other than the Amazon S3 IP addresses), the traffic is sent to the NAT gateway.

You cannot route traffic to a NAT gateway through a VPC peering connection, a VPN connection, or AWS Direct Connect. A NAT gateway cannot be used by resources on the other side of these connections.

Creating a NAT Gateway

To create a NAT gateway, you must specify a subnet and an Elastic IP address. Ensure that the Elastic IP address is currently not associated with an instance or a network interface. If you are migrating from a NAT instance to a NAT gateway and you want to reuse the NAT instance's Elastic IP address, you must first disassociate the address from your NAT instance.

To create a NAT gateway

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **NAT Gateways**, **Create NAT Gateway**.
3. In the dialog box, specify the subnet in which to create the NAT gateway, and select an Elastic IP address to associate with the NAT gateway. When you're done, choose **Create a NAT Gateway**.
4. The NAT gateway displays in the console. After a few moments, its status changes to **Available**, after which it's ready for you to use.

If the NAT gateway goes to a status of **Failed**, there was an error during creation.

Updating Your Route Table

After you've created your NAT gateway, you must update your route tables for your private subnets to point Internet traffic to the NAT gateway. AWS uses the most specific route that matches the traffic to determine how to route the traffic (longest prefix match).

To create a route for a NAT gateway

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Route Tables**.
3. Select the route table associated with your private subnet and choose **Routes**, **Edit**.
4. Choose **Add another route**. For **Destination**, enter `0.0.0.0/0`. For **Target**, select the ID of your NAT gateway.

If you're migrating from using a NAT instance, you can replace the current route that points to the NAT instance with a route to the NAT gateway.

5. Choose **Save**.

To ensure that your NAT gateway can access the Internet, the route table associated with the subnet in which your NAT gateway resides must include a route that points Internet traffic to an Internet gateway. If you delete a NAT gateway, the NAT gateway routes remain in a blackhole status until you delete or update the routes.

Deleting a NAT Gateway

You can delete a NAT gateway using the Amazon VPC console. After you've deleted a NAT gateway, its entry remains visible in the Amazon VPC console for a short while (usually an hour), after which it's automatically removed. You cannot remove this entry yourself.

To delete a NAT gateway

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose NAT Gateways.
3. Select the NAT gateway, and then choose Delete NAT Gateway.
4. In the confirmation dialog box, choose Delete NAT Gateway.

Security Groups for Your VPC

A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in a VPC, you can assign the instance to up to five security groups. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC could be assigned to a different set of security groups. If you don't specify a particular group at launch time, the instance is automatically assigned to the default security group for the VPC.³⁹

For each security group, you add rules that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic. This section describes the basics things you need to know about security groups for your VPC and their rules.

You might set up network ACLs with rules similar to your security groups in order to add an additional layer of security to your VPC.

Security Group Basics

The following are the basic characteristics of security groups for your VPC:

- You can create up to 500 security groups per VPC. You can add up to 50 inbound or outbound rules to each security group, and associate up to 5 security groups per network interface.
- You can specify allow rules, but not deny rules.
- You can specify separate rules for inbound and outbound traffic.
- By default, no inbound traffic is allowed until you add inbound rules to the security group.
- By default, an outbound rule allows all outbound traffic. You can remove the rule and add outbound rules that allow specific outbound traffic only.
- Security groups are stateful - responses to allowed inbound traffic are allowed to flow outbound regardless of outbound rules, and vice versa.
- Instances associated with a security group can't talk to each other unless you add rules allowing it (exception: the default security group has these rules by default).
- Security groups are associated with network interfaces. After you launch an instance, you can change the security groups associated with the instance, which changes the security groups associated with the primary network interface (eth0). You can also change the security groups associated with any other network interface.

Default Security Group for Your VPC

Your VPC automatically comes with a default security group. Each EC2 instance that you launch in your VPC is automatically associated with the default security group if you don't specify a different security group when you launch the instance.

³⁹http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html

The default security group inbound rule allows inbound traffic from instances assigned to the same security group and the default security group outbound rule allows all outbound traffic.

You can change the rules for the default security group.

You can't delete a default security group. If you try to delete the default security group, you'll get the following error: Client.CannotDelete: the specified group: "sg-51530134" name: "default" cannot be deleted by a user.

Security Group Rules

You can add or remove rules for a security group (also referred to as *authorizing* or *revoking* inbound or outbound access). A rule applies either to inbound traffic (ingress) or outbound traffic (egress). You can grant access to a specific CIDR range, or to another security group in your VPC or in a peer VPC (requires a VPC peering connection).

The following are the basic parts of a security group rule:

- (Inbound rules only) The source of the traffic (CIDR range or security group) and the destination port or port range.
- (Outbound rules only) The destination for the traffic (CIDR range or security group) and the destination port or port range.
- Any protocol that has a standard protocol number. If you specify ICMP as the protocol, you can specify any or all of the ICMP types and codes.



If your instance (host A) initiates traffic to host B and uses a protocol other than TCP, UDP, or ICMP, your instance's firewall only tracks the IP address and protocol number for the purpose of allowing response traffic from host B. If host B initiates traffic to your instance in a separate request within 600 seconds of the original request or response, your instance accepts it regardless of inbound security group rules, because it's regarded as response traffic. You can control this by modifying your security group's outbound rules to permit only certain types of outbound traffic. Alternatively, you can use a network ACL for your subnet — network ACLs are stateless and therefore do not automatically allow response traffic.

When you specify a security group as the source for a rule, this allows instances associated with the source security group to access instances in the security group. (Note that this does not add rules from the source security group to this security group.)

Some systems for setting up firewalls let you filter on source ports. Security groups let you filter only on destination ports.

When you add or remove rules, they are automatically applied to all instances associated with the security group.

Stale Security Group Rules

If your VPC has a VPC peering connection with another VPC, a security group rule can reference another security group in the peer VPC. This allows instances associated with the referenced security group to communicate with instances associated with the referencing security group.

If the owner of the peer VPC deletes the referenced security group, or if you or the owner of the peer VPC deletes the VPC peering connection, the security group rule is marked as `stale`. You can delete stale security group rules as you would any other security group rule.

Network Stack

The network stack contains all of the network resources that will be used by our project: VPC, subnets, internet gateway, NAT gateways, route tables, and security groups. We will have a separate application stack for the web application resources. To ensure that the web applications use the security group and subnet from the network stack, we'll create a cross-stack reference that allows the application stack to reference resource outputs from the network stack. With a cross-stack reference, owners of the web application stacks don't need to create or maintain networking rules or assets.



If your organization splits up security and network functions, you can put the security groups resources in a separate CloudFormation stack owned by the security team.

Highly Available Network

Amazon operates state-of-the-art, highly-available data centers in multiple locations world-wide. These locations are composed of regions and Availability Zones. Each region is a separate geographic area. Each region has multiple, isolated locations known as Availability Zones. Each Availability Zone is isolated, but the Availability Zones in a region are connected through low-latency links.

An Availability Zone is represented by a region code followed by a letter identifier; for example, us-east-1a. To ensure that resources are distributed across the Availability Zones for a region, Amazon independently maps Availability Zones to identifiers for each account. For example, your Availability Zone us-east-1a might not be the same location as us-east-1a for another account. There's no way for you to coordinate Availability Zones between accounts.

You can achieve High Availability by deploying your application across multiple Availability Zones. Redundant instances for each tier (e.g. web, application, and database) of an application could be placed in distinct Availability Zones thereby creating a multi-site solution. The desired goal is to have an independent copy of each application stack in two or more Availability Zones.

To achieve even more fault tolerance with less manual intervention, you can use *Elastic Load Balancing*. You get improved fault tolerance by placing your compute instances behind an Elastic Load Balancer, as it can automatically balance traffic across multiple instances and multiple Availability Zones and ensure that only healthy Amazon EC2 instances receive traffic. You can set up an Elastic Load Balancer to balance incoming application traffic across Amazon EC2 instances in a single Availability Zone or multiple Availability Zones. Elastic Load Balancing can detect the health of Amazon EC2 instances. When it detects unhealthy Amazon EC2 instances, it no longer routes traffic to those unhealthy instances. Instead, it spreads the load across the remaining healthy instances. If all of your Amazon EC2 instances in a particular Availability Zone are unhealthy, but you have set up instances in multiple Availability Zones, Elastic Load Balancing will route traffic to your healthy Amazon EC2 instances in those other zones. It will resume load balancing to the original Amazon EC2 instances when they have been restored to a healthy state.

This multi-site solution is highly available, and by design will cope with individual component or even Availability Zone failures.

Base Network Template

The following network template defines basic infrastructure for a highly available AWS network design. This template includes a VPC, 2 public subnets (multiple Availability Zones), 2 private subnets (multiple Availability Zones), a public route table, 2 private route tables, an internet gateway, and 2 NAT gateways (multiple Availability Zones).



If you choose to create a NAT gateway in your VPC, you are charged for each “NAT Gateway-hour” that your NAT gateway is provisioned and available. Data processing charges apply for each Gigabyte processed through the NAT gateway. Each partial NAT Gateway-hour consumed is billed as a full hour. You also incur standard AWS data transfer charges for all data transferred via the NAT gateway. For more information on NAT Gateway pricing, see <https://aws.amazon.com/vpc/pricing/>.

```
$ vi base-network.yaml
```

```
1 AWSTemplateFormatVersion: 2010-09-09
2 Description: Network Stack
3 Parameters:
4   VpcCidr:
5     Description: CIDR block for VPC
6     Type: String
7     AllowedPattern: ((\d{1,3})\.)\{3\}\d{1,3}/\d{1,2}
8     ConstraintDescription: CIDR Block (eg 10.0.0.0/16)
9     Default: 10.0.0.0/16
10  PublicSubnet0Cidr:
11    Description: CIDR block for PublicSubnet0
12    Type: String
13    AllowedPattern: ((\d{1,3})\.)\{3\}\d{1,3}/\d{1,2}
14    ConstraintDescription: CIDR Block (eg 10.0.0.0/16)
15    Default: 10.0.1.0/24
16  PublicSubnet1Cidr:
17    Description: CIDR block for PublicSubnet1
18    Type: String
19    AllowedPattern: ((\d{1,3})\.)\{3\}\d{1,3}/\d{1,2}
20    ConstraintDescription: CIDR Block (eg 10.0.0.0/16)
21    Default: 10.0.2.0/24
22  PrivateSubnet0Cidr:
23    Description: CIDR block for PrivateSubnet0
```

```
24     Type: String
25     AllowedPattern: ((\d{1,3})\.){3}\d{1,3}/\d{1,2}
26     ConstraintDescription: CIDR Block (eg 10.0.0.0/16)
27     Default: 10.0.3.0/24
28     PrivateSubnet1Cidr:
29         Description: CIDR block for PrivateSubnet1
30         Type: String
31         AllowedPattern: ((\d{1,3})\.){3}\d{1,3}/\d{1,2}
32         ConstraintDescription: CIDR Block (eg 10.0.0.0/16)
33         Default: 10.0.4.0/24
34     Resources:
35     Vpc:
36         Type: AWS::EC2::VPC
37         Properties:
38             CidrBlock:
39                 Ref: VpcCidr
40             EnableDnsHostnames: true
41             EnableDnsSupport: true
42             Tags:
43                 - Key: Name
44                 Value: !Sub ${AWS::StackName}-VPC
45
46     PublicSubnet0:
47         Type: AWS::EC2::Subnet
48         Properties:
49             AvailabilityZone: !Select [0, !GetAZs '']
50             MapPublicIpOnLaunch: true
51             CidrBlock:
52                 Ref: PublicSubnet0Cidr
53             Tags:
54                 - Key: Name
55                 Value: !Sub ${AWS::StackName}-PublicSubnet0
56             VpcId:
57                 Ref: Vpc
58
59     PublicSubnet1:
60         Type: AWS::EC2::Subnet
61         Properties:
62             AvailabilityZone: !Select [1, !GetAZs '']
63             MapPublicIpOnLaunch: true
64             CidrBlock:
65                 Ref: PublicSubnet1Cidr
```

```
66      Tags:
67      - Key: Name
68          Value: !Sub ${AWS::StackName}-PublicSubnet1
69      VpcId:
70          Ref: Vpc
71
72  PrivateSubnet0:
73      Type: AWS::EC2::Subnet
74      Properties:
75          AvailabilityZone: !Select [0, !GetAZs '']
76          CidrBlock:
77              Ref: PrivateSubnet0Cidr
78          Tags:
79          - Key: Name
80              Value: !Sub ${AWS::StackName}-PrivateSubnet0
81      VpcId:
82          Ref: Vpc
83
84  PrivateSubnet1:
85      Type: AWS::EC2::Subnet
86      Properties:
87          AvailabilityZone: !Select [1, !GetAZs '']
88          CidrBlock:
89              Ref: PrivateSubnet1Cidr
90          Tags:
91          - Key: Name
92              Value: !Sub ${AWS::StackName}-PrivateSubnet1
93      VpcId:
94          Ref: Vpc
95
96  PublicRT:
97      Type: AWS::EC2::RouteTable
98      Properties:
99          Tags:
100         - Key: Name
101             Value: !Sub ${AWS::StackName}-PublicRT
102         VpcId:
103             Ref: Vpc
104
105 PrivateRT0:
106     Type: AWS::EC2::RouteTable
107     Properties:
```

```
108     Tags:
109         - Key: Name
110             Value: !Sub ${AWS::StackName}-PrivateRT0
111     VpcId:
112         Ref: Vpc
113
114     PrivateRT1:
115         Type: AWS::EC2::RouteTable
116         Properties:
117             Tags:
118                 - Key: Name
119                     Value: !Sub ${AWS::StackName}-PrivateRT1
120             VpcId:
121                 Ref: Vpc
122
123     PublicSubnet0Association:
124         Type: AWS::EC2::SubnetRouteTableAssociation
125         Properties:
126             RouteTableId:
127                 Ref: PublicRT
128             SubnetId:
129                 Ref: PublicSubnet0
130
131     PublicSubnet1Association:
132         Type: AWS::EC2::SubnetRouteTableAssociation
133         Properties:
134             RouteTableId:
135                 Ref: PublicRT
136             SubnetId:
137                 Ref: PublicSubnet1
138
139     PrivateSubnet0Association:
140         Type: AWS::EC2::SubnetRouteTableAssociation
141         Properties:
142             RouteTableId:
143                 Ref: PrivateRT0
144             SubnetId:
145                 Ref: PrivateSubnet0
146
147     PrivateSubnet1Association:
148         Type: AWS::EC2::SubnetRouteTableAssociation
149         Properties:
```

```
150     RouteTableId:  
151         Ref: PrivateRT1  
152     SubnetId:  
153         Ref: PrivateSubnet1  
154  
155     InternetGateway:  
156         Type: AWS::EC2::InternetGateway  
157         Properties:  
158             Tags:  
159                 - Key: Name  
160                 Value: !Sub ${AWS::StackName}-InternetGateway  
161  
162     AttachIgw:  
163         Type: AWS::EC2::VPCGatewayAttachment  
164         Properties:  
165             InternetGatewayId:  
166                 Ref: InternetGateway  
167             VpcId:  
168                 Ref: Vpc  
169  
170     PublicRTIgwRoute:  
171         Type: AWS::EC2::Route  
172         DependsOn: AttachIgw  
173         Properties:  
174             DestinationCidrBlock: 0.0.0.0/0  
175             GatewayId:  
176                 Ref: InternetGateway  
177             RouteTableId:  
178                 Ref: PublicRT  
179  
180     NatGateway0EIP:  
181         Type: AWS::EC2::EIP  
182         Properties:  
183             Domain: vpc  
184  
185     NatGateway1EIP:  
186         Type: AWS::EC2::EIP  
187         Properties:  
188             Domain: vpc  
189  
190     NatGateway0:  
191         Type: AWS::EC2::NatGateway
```

```
192 Properties:  
193     AllocationId:  
194         Fn::GetAtt:  
195             - NatGateway0EIP  
196             - AllocationId  
197     SubnetId:  
198         Ref: PublicSubnet0  
199  
200 NatGateway1:  
201     Type: AWS::EC2::NatGateway  
202     Properties:  
203         AllocationId:  
204             Fn::GetAtt:  
205                 - NatGateway1EIP  
206                 - AllocationId  
207         SubnetId:  
208             Ref: PublicSubnet1  
209  
210 NatGateway0PrivateRT0:  
211     Type: AWS::EC2::Route  
212     Properties:  
213         DestinationCidrBlock: 0.0.0.0/0  
214         NatGatewayId:  
215             Ref: NatGateway0  
216         RouteTableId:  
217             Ref: PrivateRT0  
218  
219 NatGateway1PrivateRT1:  
220     Type: AWS::EC2::Route  
221     Properties:  
222         DestinationCidrBlock: 0.0.0.0/0  
223         NatGatewayId:  
224             Ref: NatGateway1  
225         RouteTableId:  
226             Ref: PrivateRT1  
227  
228 Outputs:  
229     VpcId:  
230         Value:  
231             Ref: Vpc  
232         Export:  
233             Name: !Sub ${AWS::StackName}-VpcId
```

```
234     VpcCidr:  
235         Value:  
236             Ref: VpcCidr  
237         Export:  
238             Name: !Sub ${AWS::StackName}-VpcCidr  
239     PrivateSubnet0:  
240         Value:  
241             Ref: PrivateSubnet0  
242         Export:  
243             Name: !Sub ${AWS::StackName}-PrivateSubnet0  
244     PrivateSubnet0Cidr:  
245         Value:  
246             Ref: PrivateSubnet0Cidr  
247         Export:  
248             Name: !Sub ${AWS::StackName}-PrivateSubnet0Cidr  
249     PrivateSubnet1:  
250         Value:  
251             Ref: PrivateSubnet1  
252         Export:  
253             Name: !Sub ${AWS::StackName}-PrivateSubnet1  
254     PrivateSubnet1Cidr:  
255         Value:  
256             Ref: PrivateSubnet1Cidr  
257         Export:  
258             Name: !Sub ${AWS::StackName}-PrivateSubnet1Cidr  
259     PublicSubnet0:  
260         Value:  
261             Ref: PublicSubnet0  
262         Export:  
263             Name: !Sub ${AWS::StackName}-PublicSubnet0  
264     PublicSubnet0Cidr:  
265         Value:  
266             Ref: PublicSubnet0Cidr  
267         Export:  
268             Name: !Sub ${AWS::StackName}-PublicSubnet0Cidr  
269     PublicSubnet1:  
270         Value:  
271             Ref: PublicSubnet1  
272         Export:  
273             Name: !Sub ${AWS::StackName}-PublicSubnet1  
274     PublicSubnet1Cidr:  
275         Value:
```

```
276      Ref: PublicSubnet1Cidr
277      Export:
278      Name: !Sub ${AWS::StackName}-PublicSubnet1Cidr
```

Network Template with Security Groups

In our example project we will deploy a multi-tier WordPress site with a web server *Auto Scaling Group* behind an *Elastic Load Balancer* and Aurora RDS for the database tier. For each tier we will set up a security group to control inbound traffic.

The following network template defines basic infrastructure and includes the security groups for all tiers. I added a security group for an OpenVPN server. An OpenVPN server will be used to allow connection to servers in private subnets. For AllowedIP parameter, you can enter your network IP address or CIDR block which will be allowed to SSH to the OpenVPN server.

```
$ vi network-with-sg.yml
```

```
1 AWSTemplateFormatVersion: 2010-09-09
2 Description: Network Stack with Security Groups
3 Parameters:
4   VpcCidr:
5     Description: CIDR block for VPC
6     Type: String
7     AllowedPattern: ((\d{1,3})\.)\{3\}\d{1,3}/\d{1,2}
8     ConstraintDescription: CIDR Block (eg 10.0.0.0/16)
9     Default: 10.0.0.0/16
10  PublicSubnet0Cidr:
11    Description: CIDR block for PublicSubnet0
12    Type: String
13    AllowedPattern: ((\d{1,3})\.)\{3\}\d{1,3}/\d{1,2}
14    ConstraintDescription: CIDR Block (eg 10.0.0.0/16)
15    Default: 10.0.1.0/24
16  PublicSubnet1Cidr:
17    Description: CIDR block for PublicSubnet1
18    Type: String
19    AllowedPattern: ((\d{1,3})\.)\{3\}\d{1,3}/\d{1,2}
20    ConstraintDescription: CIDR Block (eg 10.0.0.0/16)
21    Default: 10.0.2.0/24
22  PrivateSubnet0Cidr:
23    Description: CIDR block for PrivateSubnet0
24    Type: String
25    AllowedPattern: ((\d{1,3})\.)\{3\}\d{1,3}/\d{1,2}
26    ConstraintDescription: CIDR Block (eg 10.0.0.0/16)
27    Default: 10.0.3.0/24
28  PrivateSubnet1Cidr:
29    Description: CIDR block for PrivateSubnet1
30    Type: String
31    AllowedPattern: ((\d{1,3})\.)\{3\}\d{1,3}/\d{1,2}
```

```
32     ConstraintDescription: CIDR Block (eg 10.0.0.0/16)
33     Default: 10.0.4.0/24
34     AllowedIP:
35         Description: IP address or CIDR block allowed to SSH into OpenVPN server
36         Type: String
37         AllowedPattern: ((\d{1,3})\.)\{3\}\d{1,3}/\d{1,2}
38         ConstraintDescription: IP address (eg 123.123.123.123/32) or CIDR Block (eg \
39 10.0.0.0/16)
40         Default: 0.0.0.0/0
41
42 Resources:
43 Vpc:
44     Type: AWS::EC2::VPC
45     Properties:
46         CidrBlock:
47             Ref: VpcCidr
48         EnableDnsHostnames: true
49         EnableDnsSupport: true
50         Tags:
51             - Key: Name
52                 Value: !Sub ${AWS::StackName}-VPC
53
54 PublicSubnet0:
55     Type: AWS::EC2::Subnet
56     Properties:
57         AvailabilityZone: !Select [0, !GetAZs '']
58         MapPublicIpOnLaunch: true
59         CidrBlock:
60             Ref: PublicSubnet0Cidr
61         Tags:
62             - Key: Name
63                 Value: !Sub ${AWS::StackName}-PublicSubnet0
64     VpcId:
65         Ref: Vpc
66
67 PublicSubnet1:
68     Type: AWS::EC2::Subnet
69     Properties:
70         AvailabilityZone: !Select [1, !GetAZs '']
71         MapPublicIpOnLaunch: true
72         CidrBlock:
73             Ref: PublicSubnet1Cidr
```

```
74     Tags:
75     - Key: Name
76         Value: !Sub ${AWS::StackName}-PublicSubnet1
77     VpcId:
78         Ref: Vpc
79
80     PrivateSubnet0:
81         Type: AWS::EC2::Subnet
82         Properties:
83             AvailabilityZone: !Select [0, !GetAZs '']
84             CidrBlock:
85                 Ref: PrivateSubnet0Cidr
86             Tags:
87             - Key: Name
88                 Value: !Sub ${AWS::StackName}-PrivateSubnet0
89             VpcId:
90                 Ref: Vpc
91
92     PrivateSubnet1:
93         Type: AWS::EC2::Subnet
94         Properties:
95             AvailabilityZone: !Select [1, !GetAZs '']
96             CidrBlock:
97                 Ref: PrivateSubnet1Cidr
98             Tags:
99             - Key: Name
100                Value: !Sub ${AWS::StackName}-PrivateSubnet1
101            VpcId:
102                Ref: Vpc
103
104    PublicRT:
105        Type: AWS::EC2::RouteTable
106        Properties:
107            Tags:
108            - Key: Name
109                Value: !Sub ${AWS::StackName}-PublicRT
110            VpcId:
111                Ref: Vpc
112
113    PrivateRT0:
114        Type: AWS::EC2::RouteTable
115        Properties:
```

```
116     Tags:
117         - Key: Name
118             Value: !Sub ${AWS::StackName}-PrivateRT0
119     VpcId:
120         Ref: Vpc
121
122     PrivateRT1:
123         Type: AWS::EC2::RouteTable
124         Properties:
125             Tags:
126                 - Key: Name
127                     Value: !Sub ${AWS::StackName}-PrivateRT1
128             VpcId:
129                 Ref: Vpc
130
131     PublicSubnet0Association:
132         Type: AWS::EC2::SubnetRouteTableAssociation
133         Properties:
134             RouteTableId:
135                 Ref: PublicRT
136             SubnetId:
137                 Ref: PublicSubnet0
138
139     PublicSubnet1Association:
140         Type: AWS::EC2::SubnetRouteTableAssociation
141         Properties:
142             RouteTableId:
143                 Ref: PublicRT
144             SubnetId:
145                 Ref: PublicSubnet1
146
147     PrivateSubnet0Association:
148         Type: AWS::EC2::SubnetRouteTableAssociation
149         Properties:
150             RouteTableId:
151                 Ref: PrivateRT0
152             SubnetId:
153                 Ref: PrivateSubnet0
154
155     PrivateSubnet1Association:
156         Type: AWS::EC2::SubnetRouteTableAssociation
157         Properties:
```

```
158     RouteTableId:  
159         Ref: PrivateRT1  
160     SubnetId:  
161         Ref: PrivateSubnet1  
162  
163     InternetGateway:  
164         Type: AWS::EC2::InternetGateway  
165         Properties:  
166             Tags:  
167                 - Key: Name  
168                     Value: !Sub ${AWS::StackName}-InternetGateway  
169  
170     AttachIgw:  
171         Type: AWS::EC2::VPCGatewayAttachment  
172         Properties:  
173             InternetGatewayId:  
174                 Ref: InternetGateway  
175             VpcId:  
176                 Ref: Vpc  
177  
178     PublicRTIgwRoute:  
179         Type: AWS::EC2::Route  
180         DependsOn: AttachIgw  
181         Properties:  
182             DestinationCidrBlock: 0.0.0.0/0  
183             GatewayId:  
184                 Ref: InternetGateway  
185             RouteTableId:  
186                 Ref: PublicRT  
187  
188     NatGateway0EIP:  
189         Type: AWS::EC2::EIP  
190         Properties:  
191             Domain: vpc  
192  
193     NatGateway1EIP:  
194         Type: AWS::EC2::EIP  
195         Properties:  
196             Domain: vpc  
197  
198     NatGateway0:  
199         Type: AWS::EC2::NatGateway
```

```
200    Properties:
201        AllocationId:
202            Fn::GetAtt:
203                - NatGateway0EIP
204                - AllocationId
205        SubnetId:
206            Ref: PublicSubnet0
207
208    NatGateway1:
209        Type: AWS::EC2::NatGateway
210        Properties:
211            AllocationId:
212                Fn::GetAtt:
213                    - NatGateway1EIP
214                    - AllocationId
215            SubnetId:
216                Ref: PublicSubnet1
217
218    NatGateway0PrivateRT0:
219        Type: AWS::EC2::Route
220        Properties:
221            DestinationCidrBlock: 0.0.0.0/0
222            NatGatewayId:
223                Ref: NatGateway0
224            RouteTableId:
225                Ref: PrivateRT0
226
227    NatGateway1PrivateRT1:
228        Type: AWS::EC2::Route
229        Properties:
230            DestinationCidrBlock: 0.0.0.0/0
231            NatGatewayId:
232                Ref: NatGateway1
233            RouteTableId:
234                Ref: PrivateRT1
235
236    AppSecurityGroup:
237        Type: AWS::EC2::SecurityGroup
238        Properties:
239            GroupDescription: Security Group for application tier
240            SecurityGroupIngress:
241                - FromPort: 80
```

```
242     ToPort: 80
243     IpProtocol: tcp
244     SourceSecurityGroupId: !Ref AppELBSecurityGroup
245     VpcId:
246     Ref: Vpc
247
248 AppELBSecurityGroup:
249     Type: AWS::EC2::SecurityGroup
250     Properties:
251         GroupDescription: Security Group for application ELB
252         SecurityGroupIngress:
253             - FromPort: 80
254                 ToPort: 80
255                 IpProtocol: tcp
256                 CidrIp: 0.0.0.0/0
257             - FromPort: 443
258                 ToPort: 443
259                 IpProtocol: tcp
260                 CidrIp: 0.0.0.0/0
261         VpcId:
262             Ref: Vpc
263
264 DBSecurityGroup:
265     Type: AWS::EC2::SecurityGroup
266     Properties:
267         GroupDescription: Security Group for MySQL database server
268         SecurityGroupIngress:
269             SecurityGroupIngress:
270                 - FromPort: 3306
271                     ToPort: 3306
272                     IpProtocol: tcp
273                     SourceSecurityGroupId: !Ref AppSecurityGroup
274         VpcId:
275             Ref: Vpc
276
277 OpenVPNSecurityGroup:
278     Type: AWS::EC2::SecurityGroup
279     Properties:
280         GroupDescription: Security Group for OpenVPN server
281         SecurityGroupIngress:
282             - CidrIp: !Ref AllowedIP
283                 FromPort: 22
```

```
284     ToPort: 22
285     IpProtocol: tcp
286     - CidrIp: 0.0.0.0/0
287     FromPort: 943
288     ToPort: 943
289     IpProtocol: tcp
290     - CidrIp: 0.0.0.0/0
291     FromPort: 443
292     ToPort: 443
293     IpProtocol: tcp
294     - CidrIp: 0.0.0.0/0
295     FromPort: 1194
296     ToPort: 1194
297     IpProtocol: tcp
298     VpcId:
299         Ref: Vpc
300
301 Outputs:
302     VpcId:
303         Value:
304             Ref: Vpc
305         Export:
306             Name: !Sub ${AWS::StackName}-VpcId
307     VpcCidr:
308         Value:
309             Ref: VpcCidr
310         Export:
311             Name: !Sub ${AWS::StackName}-VpcCidr
312     PrivateSubnet0:
313         Value:
314             Ref: PrivateSubnet0
315         Export:
316             Name: !Sub ${AWS::StackName}-PrivateSubnet0
317     PrivateSubnet0Cidr:
318         Value:
319             Ref: PrivateSubnet0Cidr
320         Export:
321             Name: !Sub ${AWS::StackName}-PrivateSubnet0Cidr
322     PrivateSubnet1:
323         Value:
324             Ref: PrivateSubnet1
325         Export:
```

```
326      Name: !Sub ${AWS::StackName}-PrivateSubnet1
327  PrivateSubnet1Cidr:
328    Value:
329      Ref: PrivateSubnet1Cidr
330  Export:
331    Name: !Sub ${AWS::StackName}-PrivateSubnet1Cidr
332  PublicSubnet0:
333    Value:
334      Ref: PublicSubnet0
335  Export:
336    Name: !Sub ${AWS::StackName}-PublicSubnet0
337  PublicSubnet0Cidr:
338    Value:
339      Ref: PublicSubnet0Cidr
340  Export:
341    Name: !Sub ${AWS::StackName}-PublicSubnet0Cidr
342  PublicSubnet1:
343    Value:
344      Ref: PublicSubnet1
345  Export:
346    Name: !Sub ${AWS::StackName}-PublicSubnet1
347  PublicSubnet1Cidr:
348    Value:
349      Ref: PublicSubnet1Cidr
350  Export:
351    Name: !Sub ${AWS::StackName}-PublicSubnet1Cidr
352  AppSecurityGroup:
353    Description: The security group ID for App tier
354  Export:
355    Name: !Sub ${AWS::StackName}-AppSecurityGroup
356  Value:
357    Fn::GetAtt:
358      - AppSecurityGroup
359      - GroupId
360  AppELBSecurityGroup:
361    Description: The security group ID for App ELB
362  Export:
363    Name: !Sub ${AWS::StackName}-AppELBSecurityGroup
364  Value:
365    Fn::GetAtt:
366      - AppELBSecurityGroup
367      - GroupId
```

```
368 DBSecurityGroup:  
369     Description: The security group ID for MySQL DB  
370     Export:  
371         Name: !Sub ${AWS::StackName}-DBSecurityGroup  
372     Value:  
373         Fn::GetAtt:  
374             - DBSecurityGroup  
375             - GroupId  
376 OpenVPNSecurityGroup:  
377     Description: The security group ID for OpenVPN server  
378     Export:  
379         Name: !Sub ${AWS::StackName}-OpenVPNSecurityGroup  
380     Value:  
381         Fn::GetAtt:  
382             - OpenVPNSecurityGroup  
383             - GroupId
```

Separate Security Groups Template

If your organization splits up security and network functions, you can put the security groups resources in a separate CloudFormation stack owned by the security team.

The following template defines the security groups for all tiers and export the values to be used by the application stack. After you created the network stack using `base-network.yml` template, you can create a new stack using this following template to create all the security groups needed for our example project. For `NetworkStack` parameter you must enter the network stack name (the name of the CloudFormation stack you created using `base-network.yml` template).

```
$ vi sg.yml
```

```
1 AWSTemplateFormatVersion: 2010-09-09
2 Description: Stack for Security Groups
3 Parameters:
4   NetworkStack:
5     Description: Network stack name
6     Type: String
7   AllowedIP:
8     Description: IP address or CIDR block allowed to SSH into OpenVPN server
9     Type: String
10    AllowedPattern: ((\d{1,3})\.)\{3\}\d{1,3}/\d{1,2}
11    ConstraintDescription: IP address (eg 123.123.123.123/32) or CIDR Block (eg \
12      10.0.0.0/16)
13    Default: 0.0.0.0/0
14
15 Resources:
16   AppSecurityGroup:
17     Type: AWS::EC2::SecurityGroup
18     Properties:
19       GroupDescription: Security Group for application tier
20       SecurityGroupIngress:
21         - FromPort: '80'
22           ToPort: '80'
23           IpProtocol: 'tcp'
24           SourceSecurityGroupId: !Ref AppELBSecurityGroup
25       VpcId:
26         Fn::ImportValue:
27           Fn::Sub: ${NetworkStack}-VpcId
28
29   AppELBSecurityGroup:
30     Type: AWS::EC2::SecurityGroup
31     Properties:
```

```
32     GroupDescription: Security Group for application ELB
33     SecurityGroupIngress:
34       - FromPort: '80'
35         ToPort: '80'
36         IpProtocol: 'tcp'
37         CidrIp: 0.0.0.0/0
38       - FromPort: '443'
39         ToPort: '443'
40         IpProtocol: 'tcp'
41         CidrIp: 0.0.0.0/0
42     VpcId:
43       Fn::ImportValue:
44       Fn::Sub: ${NetworkStack}-VpcId
45
46 DBSecurityGroup:
47   Type: AWS::EC2::SecurityGroup
48   Properties:
49     GroupDescription: Security Group for MySQL database server
50     SecurityGroupIngress:
51       SecurityGroupIngress:
52         - FromPort: '3306'
53           ToPort: '3306'
54           IpProtocol: 'tcp'
55           SourceSecurityGroupId: !Ref AppSecurityGroup
56   VpcId:
57     Fn::ImportValue:
58     Fn::Sub: ${NetworkStack}-VpcId
59
60 OpenVPNSecurityGroup:
61   Type: AWS::EC2::SecurityGroup
62   Properties:
63     GroupDescription: Security Group for OpenVPN server
64     SecurityGroupIngress:
65       - CidrIp: !Ref AllowedIP
66         FromPort: 22
67         ToPort: 22
68         IpProtocol: tcp
69       - CidrIp: '0.0.0.0/0'
70         FromPort: 943
71         ToPort: 943
72         IpProtocol: tcp
73       - CidrIp: '0.0.0.0/0'
```

```
74      FromPort: 443
75      ToPort: 443
76      IpProtocol: tcp
77      - CidrIp: '0.0.0.0/0'
78      FromPort: 1194
79      ToPort: 1194
80      IpProtocol: tcp
81  VpcId:
82      Fn::ImportValue:
83      Fn::Sub: ${NetworkStack}-VpcId
84
85 Outputs:
86  AppSecurityGroup:
87      Description: The security group ID for App tier
88  Export:
89      Name: !Sub ${AWS::StackName}-AppSecurityGroup
90  Value:
91      Fn::GetAtt:
92      - AppSecurityGroup
93      - GroupId
94  AppELBSecurityGroup:
95      Description: The security group ID for App ELB
96  Export:
97      Name: !Sub ${AWS::StackName}-AppELBSecurityGroup
98  Value:
99      Fn::GetAtt:
100     - AppELBSecurityGroup
101     - GroupId
102 DBSecurityGroup:
103     Description: The security group ID for MySQL DB
104  Export:
105     Name: !Sub ${AWS::StackName}-DBSecurityGroup
106  Value:
107     Fn::GetAtt:
108     - DBSecurityGroup
109     - GroupId
110 OpenVPNSecurityGroup:
111     Description: The security group ID for OpenVPN server
112  Export:
113     Name: !Sub ${AWS::StackName}-OpenVPNSecurityGroup
114  Value:
115     Fn::GetAtt:
```

- 116 - OpenVPNSecurityGroup
117 - GroupId

For our example project we will use `network-with-sg.yml` template where we put security groups resources inside the network template.

To create the CloudFormation stack:

1. Open the CloudFormation console at <https://console.aws.amazon.com/cloudformation/>.
2. Click **Create Stack**.
3. On the **Select Template** page, choose **Upload a template to Amazon S3** and select the `network-with-sg.yml` template.
4. Click **Next**
5. On the **Specify Details** page, specify the stack name, for example `test-network`. You can change the parameter values in the **Parameters** section.

The screenshot shows the 'Specify Details' step of the CloudFormation stack creation wizard. The stack name is set to 'test-network'. The 'Parameters' section contains the following configuration:

Parameter	Value	Description
AllowedIP	0.0.0.0/0	IP address or CIDR block allowed to SSH into OpenVPN server
PrivateSubnet0Cidr	10.0.3.0/24	CIDR block for PrivateSubnet0
PrivateSubnet1Cidr	10.0.4.0/24	CIDR block for PrivateSubnet1
PublicSubnet0Cidr	10.0.1.0/24	CIDR block for PublicSubnet0
PublicSubnet1Cidr	10.0.2.0/24	CIDR block for PublicSubnet1
VpcCidr	10.0.0.0/16	CIDR block for VPC

At the bottom right of the form are three buttons: 'Cancel', 'Previous', and a blue 'Next' button.

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

Route53 Hosted Zones

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 is fully compliant with IPv6 as well.

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.⁴⁰

A *public hosted zone* is a container that holds information about how you want to route traffic on the Internet for a domain, such as example.com, and its subdomains (apex.example.com, acme.example.com). After you create a public hosted zone, you create resource record sets that determine how the Domain Name System (DNS) responds to queries for your domain and subdomains. For example, if you have one or more email addresses associated with your domain (john@example.com), you’ll create an MX record in your hosted zone so that email is sent to the email server for your domain.⁴¹

A *private hosted zone* is a container for resource record sets for a domain that you host in one or more Amazon virtual private clouds (VPCs). You create a hosted zone for a domain (such as example.com), and then you create resource record sets to tell Amazon Route 53 how you want traffic to be routed for that domain within and among your VPCs. When you create a private hosted zone, you must associate a VPC with the hosted zone, and the VPC that you specify must have been created by using the same account that you’re using to create the hosted zone. After you create the hosted zone, you can associate additional VPCs with it, including VPCs that you created by using a different AWS account.⁴²

To use private hosted zones, you must set the following Amazon VPC settings to true:

- enableDnsHostnames

⁴⁰<https://aws.amazon.com/route53/>

⁴¹<http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/AboutHZWorkingWith.html>

⁴²<http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/hosted-zones-private.html>

- enableDnsSupport

You can use Amazon Route 53 to configure split-view DNS, also known as split-horizon DNS. If you want to maintain internal and external versions of the same website or application (for example, for testing changes before you make them public), you can configure public and private hosted zones to return different internal and external IP addresses for the same domain name. Just create a public hosted zone and a private hosted zone that have the same domain name, and create the same subdomains in both hosted zones.

AWS::EC2::VPC

Creates a Virtual Private Cloud (VPC) with the CIDR block that you specify. To name a VPC resource, use the Tags property and specify a value for the Name key.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::EC2::VPC"
2 Properties:
3   CidrBlock: <String>
4   EnableDnsSupport: <Boolean>
5   EnableDnsHostnames: <Boolean>
6   InstanceTenancy: <String>
7   Tags:
8     - <Resource Tag>
```

Properties

CidrBlock

The CIDR block you want the VPC to cover. For example: "10.0.0.0/16".

Required: Yes

Type: String

Update requires: Replacement

EnableDnsSupport

Specifies whether DNS resolution is supported for the VPC. If this attribute is true, the Amazon DNS server resolves DNS hostnames for your instances to their corresponding IP addresses; otherwise, it does not. By default the value is set to true.

Required: No

Type: Boolean

Update requires: No interruption

EnableDnsHostnames

Specifies whether the instances launched in the VPC get DNS hostnames. If this attribute is true, instances in the VPC get DNS hostnames; otherwise, they do not. You can only set EnableDnsHostnames to true if you also set the EnableDnsSupport attribute to true. By default, the value is set to false.

Required: No

Type: Boolean

Update requires: No interruption

InstanceTenancy

The allowed tenancy of instances launched into the VPC.

- "default": Instances can be launched with any tenancy.
- "dedicated": Any instance launched into the VPC automatically has dedicated tenancy, unless you launch it with the default tenancy.

Required: No

Type: String

Valid values: "default" or "dedicated"

Update requires: Replacement

Tags

An arbitrary set of tags (key–value pairs) for this VPC. To name a VPC resource, specify a value for the Name key.

Required: No

Type: AWS CloudFormation Resource Tags <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-resource-tags.html>

Update requires: No interruption.

Return Values

Ref

When the logical ID of this resource is provided to the Ref intrinsic function, Ref returns the resource ID, such as vpc-18ac277d.

Fn::GetAtt

Fn::GetAtt returns a value for a specified attribute of this type. This section lists the available attributes and sample return values.

You can obtain the following default resource IDs, which AWS creates whenever you create a VPC.

CidrBlock

The set of IP addresses for the VPC. For example, 10.0.0.0/16.

DefaultNetworkAcl

The default network ACL ID that is associated with the VPC. For example, acl-814dafe3.

DefaultSecurityGroup

The default security group ID that is associated with the VPC. For example, sg-b178e0d3.

Example

```
1 AWSTemplateFormatVersion: '2010-09-09'
2 Resources:
3   myVPC:
4     Type: AWS::EC2::VPC
5     Properties:
6       CidrBlock: 10.0.0.0/16
7       EnableDnsSupport: 'false'
8       EnableDnsHostnames: 'false'
9       InstanceTenancy: dedicated
10      Tags:
11        - Key: foo
12          Value: bar
```

AWS::EC2::Subnet

Creates a subnet in an existing VPC.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::EC2::Subnet"
2 Properties:
3   AvailabilityZone: <String>
4   CidrBlock: <String>
5   MapPublicIpOnLaunch: <Boolean>
6   Tags:
7     <Resource Tag>
8   VpcId: <String>
```

Properties

AvailabilityZone

The availability zone in which you want the subnet. Default: AWS selects a zone for you (recommended).

Required: No

Type: String

Update requires: Replacement

Note: If you update this property, you must also update the CidrBlock property.

CidrBlock

The CIDR block that you want the subnet to cover (for example, "10.0.0.0/24").

Required: Yes

Type: String

Update requires: Replacement

Note: If you update this property, you must also update the AvailabilityZone property.

MapPublicIpOnLaunch

Indicates whether instances that are launched in this subnet receive a public IP address. By default, the value is `false`.

Required: No

Type: Boolean

Update requires: No interruption.

Tags An arbitrary set of tags (key–value pairs) for this subnet.

Required: No

Type: AWS CloudFormation Resource Tags

Update requires: No interruption.

VpcId

A Ref structure that contains the ID of the VPC on which you want to create the subnet. The VPC ID is provided as the value of the "Ref" property, as: { "Ref": "VPCID" }.

Required: Yes

Type: Ref ID

Update requires: Replacement

Note: If you update this property, you must also update the CidrBlock property.

Return Values

You can pass the logical ID of the resource to an intrinsic function to get a value back from the resource. The value that is returned depends on the function used.

Ref

When the logical ID of this resource is provided to the Ref intrinsic function, Ref returns the resource ID, such as subnet-e19f0178.

AWS::EC2::RouteTable

Creates a new route table within a VPC. After you create a new route table, you can add routes and associate the table with a subnet.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::EC2::RouteTable"
2 Properties:
3   VpcId: <String>
4   Tags:
5     - <Resource Tag>
```

Properties

VpcId

The ID of the VPC where the route table will be created.

Example: vpc-11ad4878

Required: Yes

Type: String

Update requires: Replacement

Tags

An arbitrary set of tags (key–value pairs) for this route table.

Required: No

Type: AWS CloudFormation Resource Tags

Update requires: No interruption.

Return Values

Ref

When you specify an AWS::EC2::RouteTable type as an argument to the Ref function, AWS CloudFormation returns the route table ID, such as rtb-12a34567.

Examples

The following example snippet uses the VPC ID from a VPC named myVPC that was declared elsewhere in the same template.

```
1 AWSTemplateFormatVersion: '2010-09-09'
2 Resources:
3   myRouteTable:
4     Type: AWS::EC2::RouteTable
5     Properties:
6       VpcId:
7         Ref: myVPC
8       Tags:
9         - Key: foo
10        Value: bar
```

AWS::EC2::Route

Creates a new route in a route table within a VPC. The route's target can be either a gateway attached to the VPC or a NAT instance in the VPC.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::EC2::Route"
2 Properties:
3   DestinationCidrBlock: <String>
4   GatewayId: <String>
5   InstanceId: <String>
6   NatGatewayId: <String>
7   NetworkInterfaceId: <String>
8   RouteTableId: <String>
9   VpcPeeringConnectionId: <String>
```

Properties

DestinationCidrBlock

The CIDR address block used for the destination match. For example, `0.0.0.0/0`. Routing decisions are based on the most specific match.

Required: Yes

Type: String

Update requires: Replacement

GatewayId

The ID of an Internet gateway or virtual private gateway that is attached to your VPC. For example: `igw-eaad4883`.

For route entries that specify a gateway, you must specify a dependency on the gateway attachment resource.

Required: Conditional. You must specify only one of the following properties: `GatewayId`, `InstanceId`, `NatGatewayId`, `NetworkInterfaceId`, or `VpcPeeringConnectionId`.

Type: String

Update requires: No interruption

InstanceId

The ID of a NAT instance in your VPC. For example, i-1a2b3c4d.

Required: Conditional. You must specify only one of the following properties: GatewayId, InstanceId, NatGatewayId, NetworkInterfaceId, or VpcPeeringConnectionId.

Type: String

Update requires: No interruption

NatGatewayId

The ID of a NAT gateway. For example, nat-0a12bc456789de0fg.

Required: Conditional. You must specify only one of the following properties: GatewayId, InstanceId, NatGatewayId, NetworkInterfaceId, or VpcPeeringConnectionId.

Type: String

Update requires: No interruption

NetworkInterfaceId

Allows the routing of network interface IDs.

Required: Conditional. You must specify only one of the following properties: GatewayId, InstanceId, NatGatewayId, NetworkInterfaceId, or VpcPeeringConnectionId.

Type: String

Update requires: No interruption

RouteTableId

The ID of the route table where the route will be added.

Required: Yes

Type: String

Update requires: Replacement

VpcPeeringConnectionId

The ID of a VPC peering connection.

Required: Conditional. You must specify only one of the following properties: GatewayId, InstanceId, NatGatewayId, NetworkInterfaceId, or VpcPeeringConnectionId.

Type: String

Update requires: No interruption

Return Values

Ref

When the logical ID of this resource is provided to the Ref intrinsic function, Ref returns the resource name.

Examples

The following example creates a route that is added to a gateway.

```
1 AWSTemplateFormatVersion: '2010-09-09'
2 Resources:
3     myRoute:
4         Type: AWS::EC2::Route
5         DependsOn: GatewayToInternet
6         Properties:
7             RouteTableId:
8                 Ref: myRouteTable
9             DestinationCidrBlock: 0.0.0.0/0
10            GatewayId:
11                Ref: myInternetGateway
```

AWS::EC2::SubnetRouteTableAssociation

Associates a subnet with a route table.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::EC2::SubnetRouteTableAssociation"
2 Properties:
3   RouteTableId: <String>
4   SubnetId: <String>
```

Properties

RouteTableId

The ID of the route table. This is commonly written as a reference to a route table declared elsewhere in the template. For example:

```
"RouteTableId" : { "Ref" : "myRouteTable" }
```

Required: Yes

Type: String

Update requires: No interruption. However, the physical ID changes when the route table ID is changed.

SubnetId

The ID of the subnet. This is commonly written as a reference to a subnet declared elsewhere in the template. For example:

```
"SubnetId" : { "Ref" : "mySubnet" }
```

Required: Yes

Type: String

Update requires: Replacement

Return Value

When the logical ID of this resource is provided to the `Ref` intrinsic function, `Ref` returns the resource name. For example:

```
{ "Ref": "MyRTA" }
```

For the subnet route table association with the logical ID “MyRTA”, `Ref` will return the AWS resource name.

Example

```
1 AWSTemplateFormatVersion: '2010-09-09'
2 Resources:
3   mySubnetRouteTableAssociation:
4     Type: AWS::EC2::SubnetRouteTableAssociation
5     Properties:
6       SubnetId:
7         Ref: mySubnet
8       RouteTableId:
9         Ref: myRouteTable
```

AWS::EC2::InternetGateway

Creates a new Internet gateway in your AWS account. After creating the Internet gateway, you then attach it to a VPC.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::EC2::InternetGateway"
2 Properties:
3   Tags:
4     - <Resource Tag>
```

Properties

Tags An arbitrary set of tags (key–value pairs) for this resource.

Required: No

Type: AWS CloudFormation Resource Tags

Update requires: No interruption.

Return Values

Ref

When the logical ID of this resource is provided to the `Ref` intrinsic function, `Ref` returns the resource name.

Example

```
1 AWSTemplateFormatVersion: '2010-09-09'
2 Resources:
3   myInternetGateway:
4     Type: "AWS::EC2::InternetGateway"
5     Properties:
6       Tags:
7         - Key: foo
8           Value: bar
```

AWS::EC2::VPCGatewayAttachment

Attaches a gateway to a VPC.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::EC2::VPCGatewayAttachment"
2 Properties:
3   InternetGatewayId: <String>
4   VpcId: <String>
5   VpnGatewayId: <String>
```

Properties

InternetGatewayId

The ID of the Internet gateway.

Required: Conditional You must specify either InternetGatewayId or VpnGatewayId, but not both.

Type: String

Update requires: No interruption

VpcId

The ID of the VPC to associate with this gateway.

Required: Yes

Type: String

Update requires: No interruption

VpnGatewayId

The ID of the virtual private network (VPN) gateway to attach to the VPC.

Required: Conditional You must specify either InternetGatewayId or VpnGatewayId, but not both.

Type: String

Update requires: No interruption

Return Values

Ref

When the logical ID of this resource is provided to the `Ref` intrinsic function, `Ref` returns the resource name.

Examples

To attach both an Internet gateway and a VPN gateway to a VPC, you must specify two separate `AWS::EC2::VP CGatewayAttachment` resources:

```
1 AttachGateway:  
2   Type: AWS::EC2::VP CGatewayAttachment  
3   Properties:  
4     VpcId:  
5       Ref: VPC  
6     InternetGatewayId:  
7       Ref: myInternetGateway  
8 AttachVpnGateway:  
9   Type: AWS::EC2::VP CGatewayAttachment  
10  Properties:  
11    VpcId:  
12      Ref: VPC  
13    VpnGatewayId:  
14      Ref: myVPNGateway
```

AWS::EC2::NatGateway

The AWS::EC2::NatGateway resource creates a network address translation (NAT) gateway in the specified public subnet. Use a NAT gateway to allow instances in a private subnet to connect to the Internet or to other AWS services, but prevent the Internet from initiating a connection with those instances.



If you add a default route (AWS::EC2::Route resource) that points to a NAT gateway, specify the NAT gateway's ID for the route's NatGatewayId property.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::EC2::NatGateway"
2 Properties:
3   AllocationId: <String>
4   SubnetId: <String>
```

Properties

AllocationId

The allocation ID of an Elastic IP address to associate with the NAT gateway. If the Elastic IP address is associated with another resource, you must first disassociate it.

Required: Yes

Type: String

Update requires: Replacement

SubnetId

The public subnet in which to create the NAT gateway.

Required: Yes

Type: String

Update requires: Replacement

Return Value

Ref

When you pass the logical ID of an AWS::EC2::NatGateway resource to the intrinsic Ref function, the function returns the ID of the NAT gateway, such as nat-0a12bc456789de0fg.

Chapter 5: The Application Stack

Introduction to Auto Scaling

Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called *Auto Scaling groups* (ASG). You can specify the minimum number of instances in each Auto Scaling group, and Auto Scaling ensures that your group never goes below this size. You can specify the maximum number of instances in each Auto Scaling group, and Auto Scaling ensures that your group never goes above this size. If you specify the desired capacity, either when you create the group or at any time thereafter, Auto Scaling ensures that your group has this many instances. If you specify scaling policies, then Auto Scaling can launch or terminate instances as demand on your application increases or decreases.⁴³

Benefits of Auto Scaling

Adding Auto Scaling to your application architecture is one way to maximize the benefits of the AWS cloud. When you use Auto Scaling, your applications gain the following benefits:⁴⁴

- Better fault tolerance. Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it. You can also configure Auto Scaling to use multiple Availability Zones. If one Availability Zone becomes unavailable, Auto Scaling can launch instances in another one to compensate.
- Better availability. Auto Scaling can help you ensure that your application always has the right amount of capacity to handle the current traffic demands.
- Better cost management. Auto Scaling can dynamically increase and decrease capacity as needed. Because you pay for the EC2 instances you use, you save money by launching instances when they are actually needed and terminating them when they aren't needed.

Auto Scaling Components

Key components of Auto Scaling:

⁴³<http://docs.aws.amazon.com/autoscaling/latest/userguide/WhatIsAutoScaling.html>

⁴⁴<http://docs.aws.amazon.com/autoscaling/latest/userguide/auto-scaling-benefits.html>

- **Groups**

When you use Auto Scaling, your EC2 instances are grouped into Auto Scaling groups for the purposes of instance scaling and management. You create Auto Scaling groups by defining the minimum, maximum, and, optionally, the desired number of running EC2 instances the group must have at any point in time.

- **Launch Configurations**

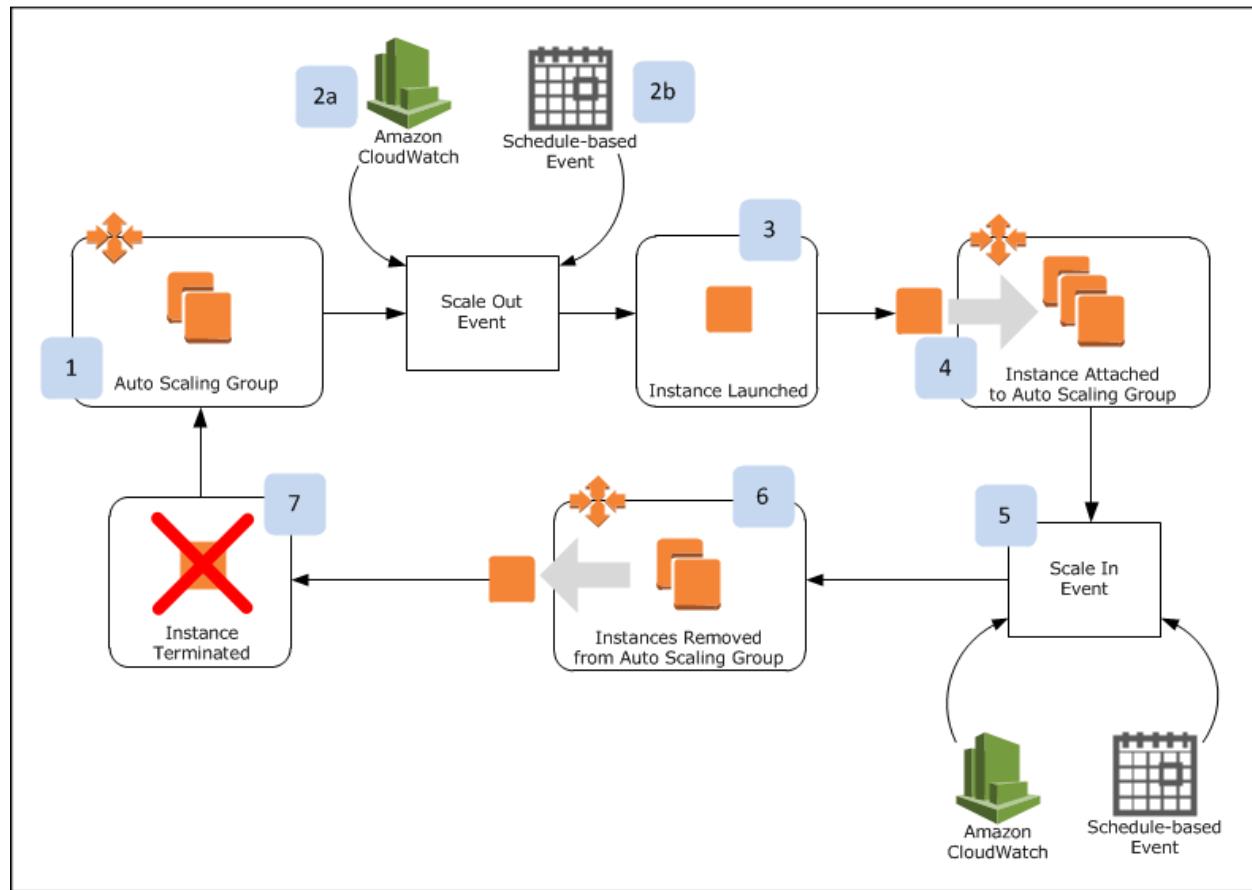
Your Auto Scaling group uses a launch configuration to launch EC2 instances. You create the launch configuration by providing information about the image you want Auto Scaling to use to launch EC2 instances. The information can be the image ID, instance type, key pairs, security groups, and block device mapping.

- **Scaling Plans**

In addition to creating a launch configuration and an Auto Scaling group, you must also create a scaling plan for your Auto Scaling group. A scaling plan tells Auto Scaling when and how to scale. You can create a scaling plan based on the occurrence of specified conditions (dynamic scaling) or you can create a plan based on a specific schedule.

How Auto Scaling Works

Here's how Auto Scaling works in a typical AWS environment:⁴⁵



Autoscaling Lifecycle

1. We'll start with an Auto Scaling group set to have a desired capacity of two instances.
2. A scale out event occurs. This is an event that instructs auto scaling to launch an additional instance. A scale out event could be something like an CloudWatch alarm (item 2a in the diagram), or a schedule-based scaling policy (item 2b in the diagram) that launches instances at a specific day and time.
3. Auto Scaling launches and configures the instance.
4. When the instance is fully configured, Auto Scaling attaches it to the Auto Scaling group.
5. Eventually, a corresponding scale in event occurs. A scale in event is like a scale out event, except that these types of events instruct Auto Scaling to terminate one or more instances.
6. Auto Scaling removes the instances from the group and marks it for termination.

⁴⁵<http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/how-as-works.html>

7. The instance terminates.

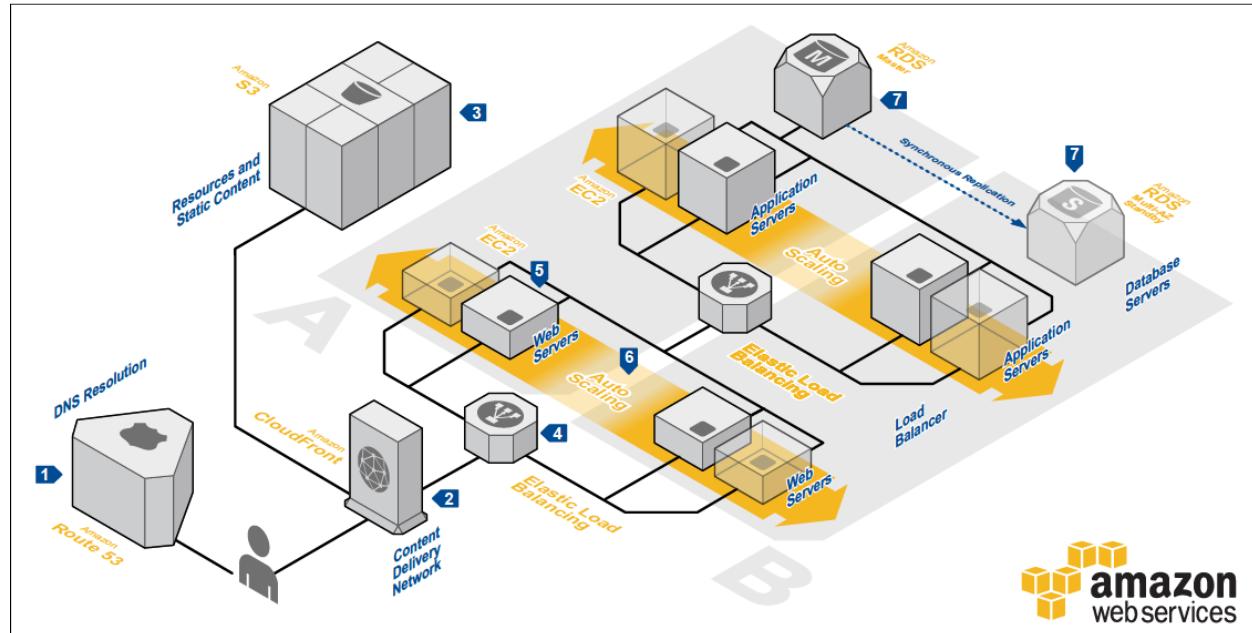
Availability Zones and Regions

Auto Scaling lets you take advantage of the safety and reliability of geographic redundancy by spanning Auto Scaling groups across multiple Availability Zones within a region. When one Availability Zone becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected Availability Zone. When the unhealthy Availability Zone returns to a healthy state, Auto Scaling automatically redistributes the application instances evenly across all of the designated Availability Zones.

An Auto Scaling group can contain EC2 instances that come from one or more EC2 Availability Zones within the same region. However, Auto Scaling group cannot span multiple regions.

The Architecture

To get an idea of how to deploy a highly available and scalable web hosting, you can take a look at the AWS Web Application Hosting reference architecture here: http://media.amazonwebservices.com/architecturecenter/AWS_ac_ra_web_01.pdf



Reference Architecture

System Overview

1. The user's DNS requests are served by **Amazon Route 53**, a highly available Domain Name System (DNS) service. Network traffic is routed to infrastructure running in Amazon Web Services.
2. Static, streaming, and dynamic content is delivered by **Amazon CloudFront**, a global network of edge locations. Requests are automatically routed to the nearest edge location, so content is delivered with the best possible performance.
3. Resources and static content used by the web application are stored on **Amazon Simple Storage Service (S3)**, a highly durable storage infrastructure designed for mission-critical and primary data storage.
4. HTTP requests are first handled by **Elastic Load Balancing**, which automatically distributes incoming application traffic among multiple **Amazon Elastic Compute Cloud (EC2)** instances across Availability Zones (AZs). It enables even greater fault tolerance in your applications, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic.

5. Web servers and application servers are deployed on Amazon EC2 instances. Most organisations will select an **Amazon Machine Image (AMI)** and then customise it to their needs. This custom AMI will then become the starting point for future web development.
6. Web servers and application servers are deployed in an **Auto Scaling** group. Auto scaling automatically adjusts your capacity up or down according to conditions you define. With Auto Scaling you can ensure that the number of **Amazon EC2** instances you're using increases seamlessly during demand spikes to maintain performance and decreases automatically during demand to minimise costs.
7. To provide high availability, the relational database that contains application's data is hosted redundantly on a multi-AZ (multiple Availability Zones - zones A and B here) deployment of **Amazon Relational Database Service (amazon RDS)**.



You can find other AWS architecture references in the **AWS Architecture Center** page:
<https://aws.amazon.com/architecture/>.

The AWS Architecture Center is designed to provide you with the necessary guidance and application architecture best practices to build highly scalable and reliable applications in the AWS cloud. These resources will help you understand the AWS platform, its services and features, and will provide architectural guidance for design and implementation of systems that run on the AWS infrastructure.

Getting Started with Auto Scaling

To help you get a better understanding on how Auto Scaling works, let's try to configure Auto Scaling from AWS console.⁴⁶

Step 1: Create a Launch Configuration

A *launch configuration* is a template that an Auto Scaling group uses to launch EC2 instances. When you create a launch configuration, you specify information for the instances such as the ID of the Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping. If you've launched an EC2 instance before, you specified the same information in order to launch the instance.

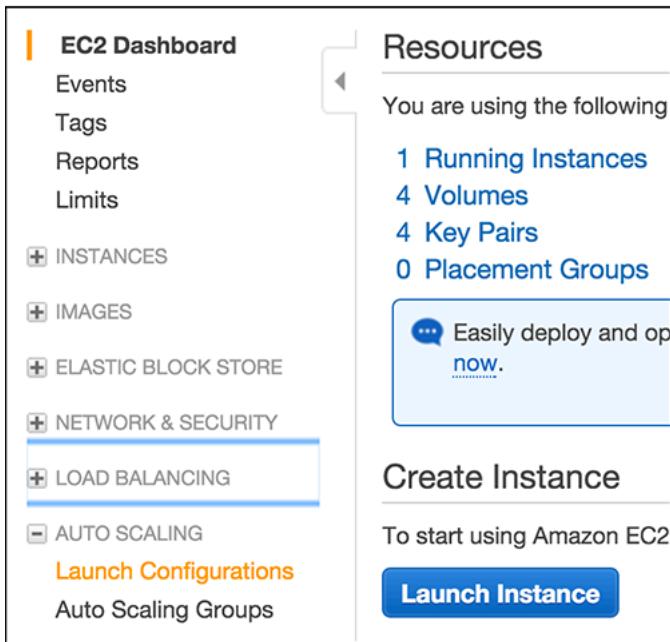
When you create an Auto Scaling group, you must specify a launch configuration. You can specify your launch configuration with multiple Auto Scaling groups. However, you can only specify one launch configuration for an Auto Scaling group at a time, and you can't modify a launch configuration after you've created it. Therefore, if you want to change the launch configuration for your Auto Scaling group, you must create a launch configuration and then update your Auto Scaling group with the new launch configuration. When you change the launch configuration for your Auto Scaling group, any new instances are launched using the new configuration parameters, but existing instances are not affected.⁴⁷

To create a launch configuration:

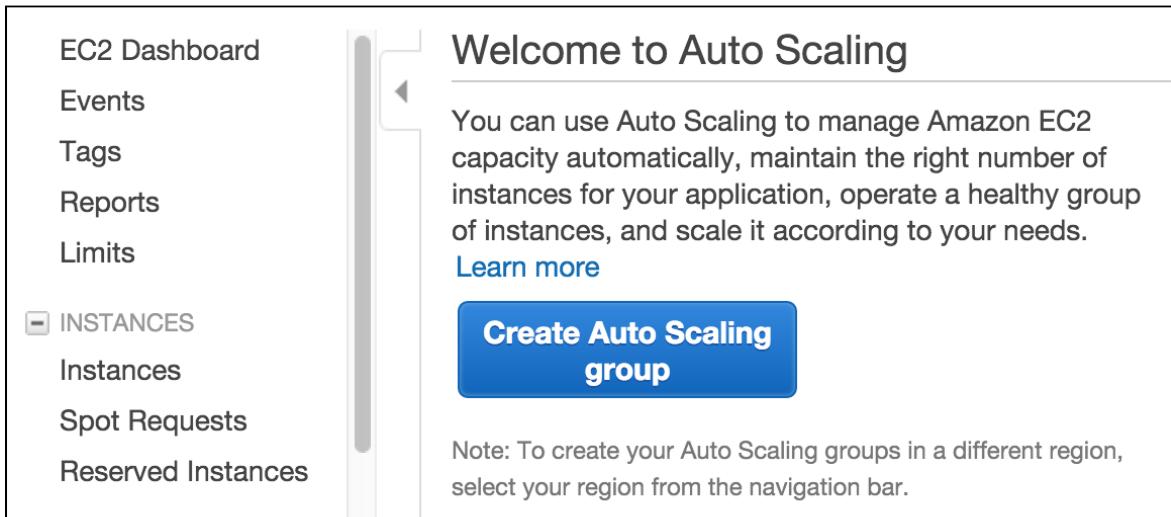
1. Open the Amazon EC2 console.
2. In the navigation pane, under **Auto Scaling**, click **Launch Configurations**.

⁴⁶<http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/GettingStartedTutorial.html>

⁴⁷<http://docs.aws.amazon.com/autoscaling/latest/userguide/LaunchConfiguration.html>



3. Select a region. The Auto Scaling resources that you create are tied to the region you specify and are not replicated across regions.
4. On the **Welcome to Auto Scaling** page, click **Create Auto Scaling group**.



5. On the **Create Auto Scaling Group** page, click **Create launch configuration**.
6. The **Choose AMI** page displays a list of basic configurations, called Amazon Machine Images (AMIs), that serve as templates for your instance. Select **AWS Marketplace** from the left pane. In the **Search AWS Marketplace Products** field, enter “wordpress” and hit Enter. Select **WordPress powered by Bitnami** AMI. Accept the software terms to use this AMI.

The screenshot shows the 'Create Launch Configuration' wizard at step 1. The search bar contains 'wordpress'. A result for 'WordPress powered by Bitnami' is displayed, showing a 4.7 rating, a price of '\$0.00/hr for software + AWS usage fees', and a description of it being a pre-configured image for running WordPress on Amazon EC2. A 'Select' button is visible.

- On the Choose Instance Type page, select a hardware configuration for your instance. Select t2.micro instance type. Click Next: Configure details.

The screenshot shows the 'Choose Instance Type' page. The t2.micro instance type is selected. The table lists various instance types with their details. Buttons for 'Cancel', 'Previous', and 'Next: Configure details' are at the bottom.

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate
<input checked="" type="checkbox"/>	General purpose	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate

- On the Configure Details page, do the following:
 - In the Name field, enter a name of your launch configuration (for example, `test-wordpress-lc`).
 - In the IAM Role field, select the `AppInstanceProfile` name. This is the instance profile we created using the IAM template from chapter 3.
 - Under Advanced Details, select an IP address type. Because we will launch instances in private subnets, select `Do not assign a public IP address to any instances`.
 - Click Skip to review.

Create Launch Configuration

Name	<input type="text" value="test-wordpress-lc"/>
Purchasing option	<input type="checkbox"/> Request Spot Instances
IAM role	<input type="text" value="test-iam-AppInstanceProfile-"/>
Monitoring	<input type="checkbox"/> Enable CloudWatch detailed monitoring Learn more

- On the Review page, click **Edit security groups**, select **Select an existing security group**, choose the AppSecurityGroup security group and then click **Review**.

Create Launch Configuration

Assign a security group:			
<input type="radio"/> Create a new security group <input checked="" type="radio"/> Select an existing security group			
Security	Name	VPC ID	Description
<input type="checkbox"/>	sg-[REDACTED] default	vpc-[REDACTED]	default VPC security group
<input type="checkbox"/>	sg-[REDACTED] default	vpc-[REDACTED]	default VPC security group
<input type="checkbox"/>	sg-[REDACTED] mysg	vpc-[REDACTED]	This security group was generated by AWS M...
<input type="checkbox"/>	sg-[REDACTED] SG_apsydne	vpc-[REDACTED]	base - security group - Sydney
<input type="checkbox"/>	sg-[REDACTED] test-network-AppELBSecurityGroup-[REDACTED]	vpc-[REDACTED]	Security Group for application ELB
<input checked="" type="checkbox"/>	sg-[REDACTED] test-network-AppSecurityGroup-[REDACTED]	vpc-[REDACTED]	Security Group for application tier

- On the Review page, click **Create launch configuration**.
- In the **Select an existing key pair or create a new key pair** dialog box, select one of the listed options. Note that you won't connect to your instance as part of this tutorial. Therefore, you can select **Proceed without a key pair** unless you intend to connect to your instance.
- Click **Create launch configuration** to create your launch configuration.

Step 2: Create an Auto Scaling Group

An Auto Scaling group contains a collection of EC2 instances that share similar characteristics and are treated as a logical grouping for the purposes of instance scaling and management. For example, if a single application operates across multiple instances, you might want to increase the number of instances in that group to improve the performance of the application, or decrease the number of instances to reduce costs when demand is low. You can use the Auto Scaling group to scale the number of instances automatically based on criteria that you specify, or maintain a fixed number of instances even if an instance becomes unhealthy. This automatic scaling and maintaining the number of instances in an Auto Scaling group is the core value of the Auto Scaling service.⁴⁸

⁴⁸<http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/AutoScalingGroup.html>

To create an Auto Scaling group:

1. On the **Configure Auto Scaling group details** page, do the following:
 - In **Group name**, enter a name for your Auto Scaling group (for example, **test-wordpress-asg**)
 - Leave **Group size** set to the default value of 1 instance for this tutorial.
 - Select the **test-network-VPC** VPC in **Network** and select **test-network-PrivateSubnet0** subnet and **test-network-PrivateSubnet1** subnet from **Subnet**.
 - Click **Next: Configure scaling policies**.

The screenshot shows the 'Create Auto Scaling Group' wizard. Step 1: Configure Auto Scaling group details. The 'Launch Configuration' dropdown is set to 'test-wordpress-lc'. The 'Group name' input field contains 'test-wordpress-asg'. The 'Group size' dropdown is set to 'Start with 1 instances'. Under 'Network', the 'vpc-' dropdown is set to '(10.0.0.0/16) | test-network-VPC'. The 'Subnet' dropdown shows two subnets: 'subnet- (10.0.3.0/24) | test-network-PrivateSubnet0 | ap-southeast-2a' and 'subnet- (10.0.4.0/24) | test-network-PrivateSubnet1 | ap-southeast-2b'. A note at the bottom states: 'No instances in this Auto Scaling group will be assigned a public IP address.' A 'Cancel and Exit' button is in the top right corner.

2. In the **Configure scaling policies** page, select **Keep this group at its initial size** for this tutorial and click **Review**.
3. On the **Review** page, click **Create Auto Scaling group**.
4. On the **Auto Scaling group creation status** page, click **Close**.

Step 3: Verify Your Auto Scaling Group

Now that you have created your Auto Scaling group, you are ready to verify that the group has launched your EC2 instance.

To verify that your Auto Scaling group has launched your EC2 instance:

1. On the **Auto Scaling Groups** page, select the Auto Scaling group that you just created.
2. The **Details** tab provides information about the Auto Scaling group.

The screenshot shows the AWS Auto Scaling Groups page. At the top, there is a filter bar with a search input 'Filter Auto Scaling groups...'. Below it, a table lists one Auto Scaling Group:

Name	Launch Configuration	Instances	Desired	Min	Max	Availability Zones
test-wordpress-asg	test-wordpress-lc	0	1	1	1	ap-southeast-2b, ap-so

Below the table, the details for the 'test-wordpress-asg' group are shown. The 'Details' tab is selected. The configuration includes:

- Launch Configuration:** test-wordpress-lc
- Target Groups:**
 - Desired:** 1
 - Min:** 1
 - Max:** 1
 - Health Check Type:** EC2
 - Health Check Grace Period:** 300
- Availability Zone(s):** ap-southeast-2b, ap-southeast-2a
- Subnet(s):** subnet-XXXXXX, subnet-YZZZZZ
- Default Cooldown:** 300
- Placement Group:**
- Suspended Processes:**

3. Select the **Scaling History** tab. The **Status** column contains the current status of your instance. When your instance is launching, the status column shows **In progress**. The status changes to **Successful** after the instance is launched. You can also click the refresh button to see the current status of your instance. If you see **Failed** status and this description: “Launching a new EC2 instance. Status Reason: In order to use this AWS Marketplace product you need to accept terms and subscribe.” go to the link provided and accept the software terms.

The screenshot shows the 'Activity History' tab for the 'test-wordpress-asg' group. The history items are listed as follows:

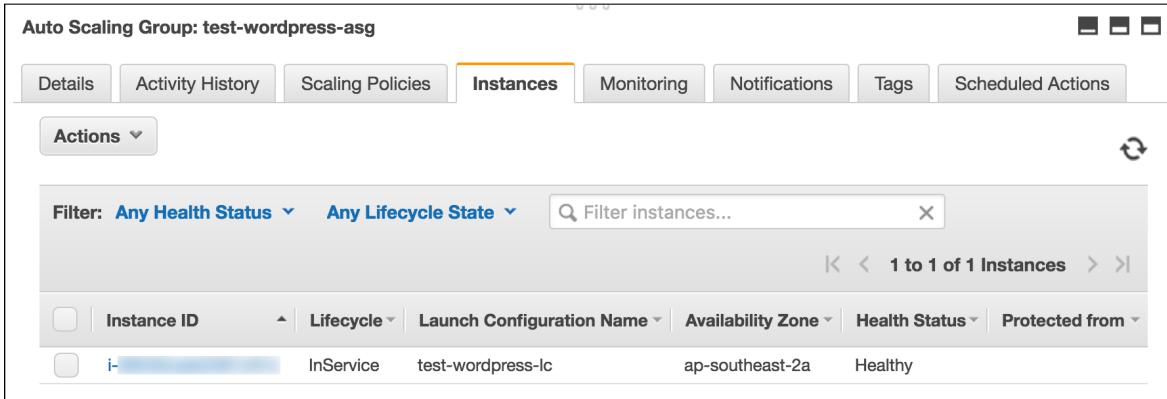
Status	Description	Start Time	End Time
Successful	Launching a new EC2 instance: i-XXXXXX	13:16:05 UTC+10	13:16:37 UTC

For the successful event, the details are:

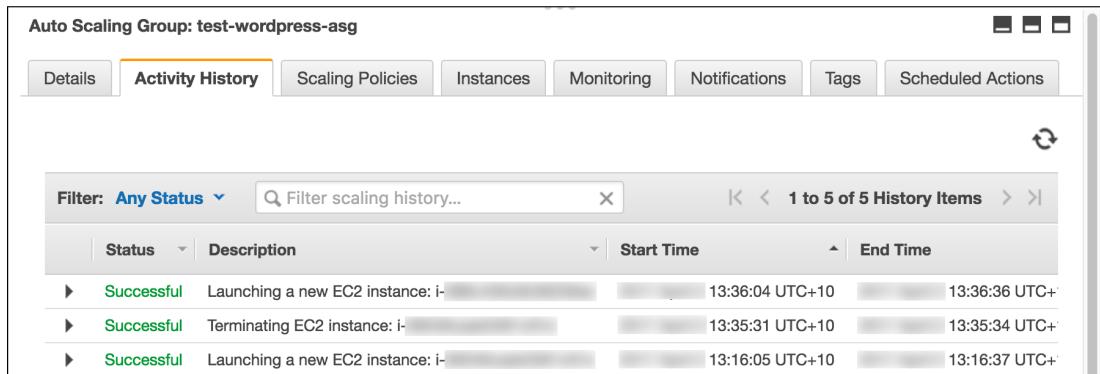
Description: Launching a new EC2 instance: i-XXXXXX

Cause: At 2016-03-16T03:16:03Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.

4. Select the **Instances** tab. The **Lifecycle** column contains the state of your newly launched instance. You can see that your Auto Scaling group has launched your EC2 instance, and it is in the **InService** lifecycle state. The **Health Status** column shows the result of the EC2 instance health check on your instance.



5. If you want, you can try the following experiment to learn more about Auto Scaling. The minimum size for your Auto Scaling group is 1 instance. Therefore, if you terminate the running instance, Auto Scaling must launch a new instance to replace it.
 - On the Instances tab, click the ID of the instance. This takes you to the Instances page and selects the instance.
 - Click Actions, select Instance State, and then click Terminate. When prompted for confirmation, click Yes, Terminate.
 - In the navigation pane, select Auto Scaling Groups and then select the Scaling History tab. The default cooldown for the Auto Scaling group is 300 seconds (5 minutes), so it takes about 5 minutes until you see the scaling activity. When the scaling activity starts, you'll see an entry for the termination of the first instance and an entry for the launch of a new instance. The Instances tab shows the new instance only.



- In the navigation pane, select Instances. This page shows both the terminated instance and the running instance.

Scaling the Size of Your Auto Scaling Group

Scaling is the ability to increase or decrease the compute capacity of your application. Scaling starts with an event, or scaling action, which instructs Auto Scaling to either launch or terminate EC2 instances.

Auto Scaling provides a number of ways to adjust scaling to best meet the needs of your applications. As a result, it's important that you have a good understanding of your application. You should keep the following considerations in mind:⁴⁹

- What role do you want Auto Scaling to play in your application's architecture? It's common to think about Auto Scaling as a way to increase and decrease capacity, but Auto Scaling is also useful for when you want to maintain a steady number of servers.
- What cost constraints are important to you? Because Auto Scaling uses EC2 instances, you only pay for the resources you use. Knowing your cost constraints can help you decide when to scale your applications, and by how much.
- What metrics are important to your application? CloudWatch supports a number of different metrics that you can use with your Auto Scaling group. You can view available EC2 metrics here: <http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/ec2-metricscollected.html>. For example, you can use CPUUtilization metric in your scaling policy to increase/decrease Auto Scaling group size based on average CPU usage of the EC2 instances.

Some scaling implementations you can use, based on the needs of your applications:

- **Maintaining a fixed number of EC2 instances in your Auto Scaling group**⁵⁰

After you have created your launch configuration and Auto Scaling group, the Auto Scaling group starts by launching the minimum number of EC2 instances (or the desired capacity, if specified). If there are no other scaling conditions attached to the Auto Scaling group, the Auto Scaling group maintains this number of running instances at all times.

To maintain the same number of instances, Auto Scaling performs a periodic health check on running instances within an Auto Scaling group. When it finds that an instance is unhealthy, it terminates that instance and launches a new one.

All instances in your Auto Scaling group start in the healthy state. Instances are assumed to be healthy unless Auto Scaling receives notification that they are unhealthy. This notification can come from one or more of the following sources: Amazon EC2, Elastic Load Balancing, or your customized health check.

- **Manual Scaling**⁵¹

At any time, you can manually change the size of an existing Auto Scaling group. Auto Scaling manages the process of launching or terminating instances to maintain the updated group size.

⁴⁹http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/scaling_plan.html

⁵⁰<http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/as-maintain-instance-levels.html>

⁵¹<http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/as-manual-scaling.html>

- **Scheduled Scaling** ⁵²

Scaling based on a schedule allows you to scale your application in response to predictable load changes. For example, every week the traffic to your web application starts to increase on Wednesday, remains high on Thursday, and starts to decrease on Friday. You can plan your scaling activities based on the predictable traffic patterns of your web application.

To configure your Auto Scaling group to scale based on a schedule, you need to create scheduled actions. A scheduled action tells Auto Scaling to perform a scaling action at certain time in future. To create a scheduled scaling action, you specify the start time at which you want the scaling action to take effect, and you specify the new minimum, maximum, and desired size you want for that group at that time. At the specified time, Auto Scaling updates the group to set the new values for minimum, maximum, and desired sizes, as specified by your scaling action.

- **Dynamic Scaling** ⁵³

Scale dynamically in response to changes in the demand for your application. You must specify when and how to scale.

When you use Auto Scaling to scale dynamically, you must define how you want to scale in response to changing demand. For example, say you have a web application that currently runs on two instances. You want to launch two additional instances when the load on the current instances rises to 70 percent, and then you want to terminate those additional instances when the load falls to 40 percent. You can configure your Auto Scaling group to scale automatically based on these conditions. We will try this later in this chapter.

Scaling Manually using the Console

To change the size of your Auto Scaling group manually: ⁵⁴

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. On the **Auto Scaling Groups** page, select your Auto Scaling group from the list.
4. The bottom pane displays the details of your Auto Scaling group. Select the **Details** tab and then click **Edit**.
5. In **Desired**, increase the desired capacity by one. For example, if the current value is 1, enter 2. The desired capacity must be less than or equal to the maximum size of the group. Increase **Max** to 5. When you are finished, click **Save**.

⁵²http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/schedule_time.html

⁵³<http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/as-scale-based-on-demand.html>

⁵⁴<http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/as-manual-scaling.html>

Name	Launch Configuration	Instances	Desired	Min	Max	Availability Zones	Default Cooldown
test-wordpress-...	test-wordpress-lc	1	1	1	1	ap-southeast-2b, ap-southeast-2a	300

Launch Configuration: test-wordpress-lc

Load Balancers: [Empty]

Target Groups: [Empty]

Desired: 2

Min: 1

Max: 5

Health Check Type: EC2

Availability Zone(s): ap-southeast-2b, ap-southeast-2a

Subnet(s):

- subnet-... (10.0.3.0/24) | test-network-PrivateSubnet0 | ap-southeast-2a
- subnet-... (10.0.4.0/24) | test-network-PrivateSubnet1 | ap-southeast-2b

Default Cooldown: 300

Placement Group: [Empty]

- To verify that the size of your Auto Scaling group has changed, go to **Instances** tab. You can see that your Auto Scaling group has launched 1 new instance.

Instance ID	Lifecycle	Launch Configuration Name	Availability Zone	Health Status	Protected from
i-...	InService	test-wordpress-lc	ap-southeast-2a	Healthy	[Empty]
i-...	Pending	test-wordpress-lc	ap-southeast-2b	Healthy	[Empty]



If you're planning to make an ASG idle, you can set **Min**, **Desired**, and **Max** to **0** to terminate all instances within the ASG. You can increase these values later to launch new instance(s).

Load Balance Your Auto Scaling Group

When you use Auto Scaling, you can automatically increase the number of EC2 instances you're using when the user demand goes up, and you can decrease the number of EC2 instances when demand goes down. As Auto Scaling dynamically adds and removes EC2 instances, you need to ensure that the traffic coming to your application is distributed across all of your running EC2 instances. AWS provides the Elastic Load Balancing service to distribute the incoming traffic (called the load) automatically among all the EC2 instances that you are running. Elastic Load Balancing manages incoming requests by optimally routing traffic so that no one instance is overwhelmed. Using Elastic Load Balancing with your auto-scaled application makes it easy to route traffic across a dynamically changing fleet of EC2 instances.⁵⁵

What Is Elastic Load Balancing

Elastic Load Balancing automatically distributes incoming traffic across multiple EC2 instances. You create a load balancer and register instances with the load balancer in one or more Availability Zones. The load balancer serves as a single point of contact for clients. This enables you to increase the availability of your application. You can add and remove EC2 instances from your load balancer as your needs change, without disrupting the overall flow of information. If an EC2 instance fails, Elastic Load Balancing automatically reroutes the traffic to the remaining running EC2 instances. If a failed EC2 instance is restored, Elastic Load Balancing restores the traffic to that instance. Elastic Load Balancing can also serve as the first line of defense against attacks on your network. You can offload the work of encryption and decryption to your load balancer so that your EC2 instances can focus on their main work.⁵⁶

If you enable Auto Scaling with Elastic Load Balancing, instances that are launched by Auto Scaling are automatically registered with the load balancer, and instances that are terminated by Auto Scaling are automatically de-registered from the load balancer.

Features of Elastic Load Balancing

Elastic Load Balancing provides the following features:

- You can use the operating systems and instance types supported by Amazon EC2. You can configure your EC2 instances to accept traffic only from your load balancer.
- You can configure the load balancer to accept traffic using the following protocols: HTTP, HTTPS (secure HTTP), TCP, and SSL (secure TCP).

⁵⁵http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/US_SetUpASLBApp.html

⁵⁶<http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elastic-load-balancing.html>

- You can configure your load balancer to distribute requests to EC2 instances in multiple Availability Zones, minimizing the risk of overloading one single instance. If an entire Availability Zone goes offline, the load balancer routes traffic to instances in other Availability Zones.
- There is no limit on the number of connections that your load balancer can attempt to make with your EC2 instances. The number of connections scales with the number of concurrent requests that the load balancer receives.
- You can configure the health checks that Elastic Load Balancing uses to monitor the health of the EC2 instances registered with the load balancer so that it can send requests only to the healthy instances.
- You can use end-to-end traffic encryption on those networks that use secure (HTTPS/SSL) connections.
- [EC2-VPC] You can create an *Internet-facing* load balancer, which takes requests from clients over the Internet and routes them to your EC2 instances, or an *internal-facing* load balancer, which takes requests from clients in your VPC and routes them to EC2 instances in your private subnets. Load balancers in EC2-Classic are always Internet-facing.
- [EC2-Classic] Load balancers for EC2-Classic support both IPv4 and IPv6 addresses. Load balancers for a VPC do not support IPv6 addresses.
- You can monitor your load balancer using CloudWatch metrics, access logs, and AWS CloudTrail.
- You can associate your Internet-facing load balancer with your domain name. Because the load balancer receives all requests from clients, you don't need to create and manage public domain names for the EC2 instances to which the load balancer routes traffic. You can point the instance's domain records at the load balancer instead and scale as needed (either adding or removing capacity) without having to update the records with each scaling activity.

Getting Started with ELB

In this tutorial we will add an Elastic Load Balancer and attach it to our WordPress AutoScaling Group from AWS console.⁵⁷

Step 1: Define Load Balancer

First, provide some basic configuration information for your load balancer, such as a name, a network, and a listener.

A *listener* is a process that checks for connection requests. It is configured with a protocol and a port for front-end (client to load balancer) connections and a protocol, and protocol and a port for back-end (load balancer to back-end instance) connections. In this tutorial, you configure a listener that accepts HTTP requests on port 80 and sends them to the back-end instances on port 80 using HTTP.

To define your load balancer:

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. From the navigation bar, select a region for your load balancers. Be sure to select the same region that you selected for your EC2 instances.
3. In the navigation pane, under LOAD BALANCING, click Load Balancers.
4. Click Create Load Balancer.
5. Select Classic Load Balancer and click Continue.
6. In Load Balancer name, enter a name for your load balancer.
The name of your load balancer must be unique within your set of load balancers for the region, can have a maximum of 32 characters, and can contain only alphanumeric characters and hyphens.
7. From Create LB inside, select test-network-VPC VPC.
8. Leave the default listener configuration.

⁵⁷<http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-getting-started.html>

1. Define Load Balancer 2. Assign Security Groups 3. Configure Security Settings 4. Configure Health Check 5. Add EC2 Instances

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:	test-wordpress-lb		
Create LB Inside:	vpc- (10.0.0.0/16) test-network-VPC		
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			

- Under **Select Subnets**, select `test-network-PublicSubnet0` and `test-network-PublicSubnet1`. The available subnets for the VPC for your load balancer are displayed under **Available Subnets**. Click the icon in the **Actions** column for each subnet to attach. These subnets are moved under **Selected Subnets**.

VPC vpc- (10.0.0.0/16) | test-network-VPC

Available subnets				
Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
	ap-southeast-2a	subnet-	10.0.3.0/24	test-network-PrivateSubnet0
	ap-southeast-2b	subnet-	10.0.4.0/24	test-network-PrivateSubnet1

Selected subnets				
Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
	ap-southeast-2a	subnet-	10.0.1.0/24	test-network-PublicSubnet0
	ap-southeast-2b	subnet-	10.0.2.0/24	test-network-PublicSubnet1

Cancel **Next: Assign Security Groups**

- Click **Next: Assign Security Groups**.

Step 2: Assign Security Groups to Your Load Balancer

To assign security group to your load balancer:

1. On the Assign Security Groups page, select **Select an existing security group**.
2. Select **test-network-AppELBSecurityGroup** Security Group from the list.
3. Click **Next: Configure Security Settings**.

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	Name	Description	Actions
<input type="checkbox"/>	sg-[REDACTED] default	default VPC security group	Copy to new
<input checked="" type="checkbox"/>	sg-[REDACTED] test-network-AppELBSecurityGroup-[REDACTED]	Security Group for application ELB	Copy to new
<input type="checkbox"/>	sg-[REDACTED] test-network-AppSecurityGroup-[REDACTED]	Security Group for application tier	Copy to new
<input type="checkbox"/>	sg-[REDACTED] test-network-DBSecurityGroup-[REDACTED]	Security Group for MySQL database server	Copy to new
<input type="checkbox"/>	sg-[REDACTED] test-network-OpenVPNSecurityGroup-[REDACTED]	Security Group for OpenVPN server	Copy to new

Step 3: Configure Security Settings

For this tutorial, you can click **Next: Configure Health Check** to continue to the next step. For more information about creating a HTTPS load balancer and using additional security features, see <http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-https-load-balancers.html>

Step 4: Configure Health Checks for Your EC2 Instances

Elastic Load Balancing automatically checks the health of the EC2 instances for your load balancer. If Elastic Load Balancing finds an unhealthy instance, it stops sending traffic to the instance and reroutes traffic to healthy instances. In this step, you customize the health checks for your load balancer.

To configure health checks for your instances:

1. On the **Configure Health Check** page, do the following:
 - Leave **Ping Protocol** set to its default value, **HTTP**.
 - Leave **Ping Port** set to its default value, **80**.
 - In the **Ping Path** field, replace the default value with **/**.

- Leave the other fields set to their default values.

Step 4: Configure Health Check
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. Customize the health check to meet your specific needs.

Ping Protocol	HTTP
Ping Port	80
Ping Path	/
Advanced Details	
Response Timeout	5 seconds
Interval	30 seconds
Unhealthy threshold	2
Healthy threshold	10

2. Click Next: Add EC2 Instances.

Step 5: Register EC2 Instances with Your Load Balancer

We will associate the ELB with our Auto Scaling Group later, so we can skip adding instance in this step.

Step 6: Tag Your Load Balancer (Optional)

You can tag your load balancer, or continue to the next step. Note that you can tag your load balancer later on.

To add tags to your load balancer:

1. On the **Add Tags** page, specify a key and a value for the tag.
2. To add another tag, click **Create Tag** and specify a key and a value for the tag.
3. After you are finished adding tags, click **Review and Create**.

Step 7: Create and Verify Your Load Balancer

Before you create the load balancer, review the settings that you selected. After creating the load balancer, we will attach it to our ASG and you can verify that it's sending traffic to your EC2 instances.

To finish creating your load balancer:

1. On the **Review** page, check your settings. If you need to make changes, click the corresponding link to edit the settings.
2. Click **Create** to create your load balancer.
3. After you are notified that your load balancer was created, click **Close**.

4. You can see your newly created Load Balancer in the EC2 console.

The screenshot shows the AWS EC2 Load Balancers console. At the top, there is a search bar and navigation buttons for filtering, sorting, and viewing details. A table lists one load balancer entry:

Name	DNS name	State	VPC ID
test-wordpress-lb	test-wordpress-lb-[REDACTED].ap-southeast-2.elb.amazonaws.com	[REDACTED]	vpc-[REDACTED]

Below the table, the load balancer details are shown:

Load balancer: test-wordpress-lb

Description | Instances | Health Check | Listeners | Monitoring | Tags

Basic Configuration

Name: test-wordpress-lb	Creation time: [REDACTED] at 3:16:04 PM UTC+10
* DNS name: test-wordpress-lb-[REDACTED].ap-southeast-2.elb.amazonaws.com (A Record)	Hosted zone: [REDACTED]
Scheme: internet-facing	Status: 0 of 0 instances in service
Availability Zones: subnet-[REDACTED] - ap-southeast-2b, subnet-[REDACTED] - ap-southeast-2a	VPC: vpc-[REDACTED]

Attaching Load Balancer to ASG

Auto Scaling integrates with Elastic Load Balancing to enable you to attach one or more load balancers to an existing Auto Scaling group. After you attach the load balancer, it automatically registers the instances in the group and distributes incoming traffic across the instances.

When you attach a load balancer, it enters the **Adding** state while registering the instances in the group. After all instances in the group are registered with the load balancer, it enters the **Added** state. After at least one registered instance passes the health check, it enters the **InService** state.

When you detach a load balancer, it enters the **Removing** state while deregistering the instances in the group. If *connection draining* is enabled, Elastic Load Balancing waits for in-flight requests to complete before deregistering the instances. Note that the instances remain running after they are deregistered.⁵⁸

To attach a load balancer to a group:

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. Select your group.
4. In the bottom pane, on the **Details** tab, click **Edit**.
5. In **Load Balancers**, select the load balancer.

The screenshot shows the 'Edit' configuration for the Auto Scaling Group 'test-wordpress-asg'. The 'Launch Configuration' is set to 'test-wordpress-lc'. Under 'Load Balancers', 'test-wordpress-lb' is selected. Under 'Target Groups', there is a single entry. On the right, 'Desired' and 'Min' values are both set to 1. The 'Availability Zone(s)' section shows 'ap-southeast-2b, ap-southeast-2a'. Below that, the 'Subnet(s)' section displays two subnets: 'subnet-network-PrivateSubnet0 | test-network-PrivateSubnet0 | ap-southeast-2a' and 'subnet-network-PrivateSubnet1 | test-network-PrivateSubnet1 | ap-southeast-2b'. A 'Save' button is visible in the top right corner.

6. Click **Save**.

⁵⁸<http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/attach-load-balancer-asg.html>

Verify Your Load Balancer

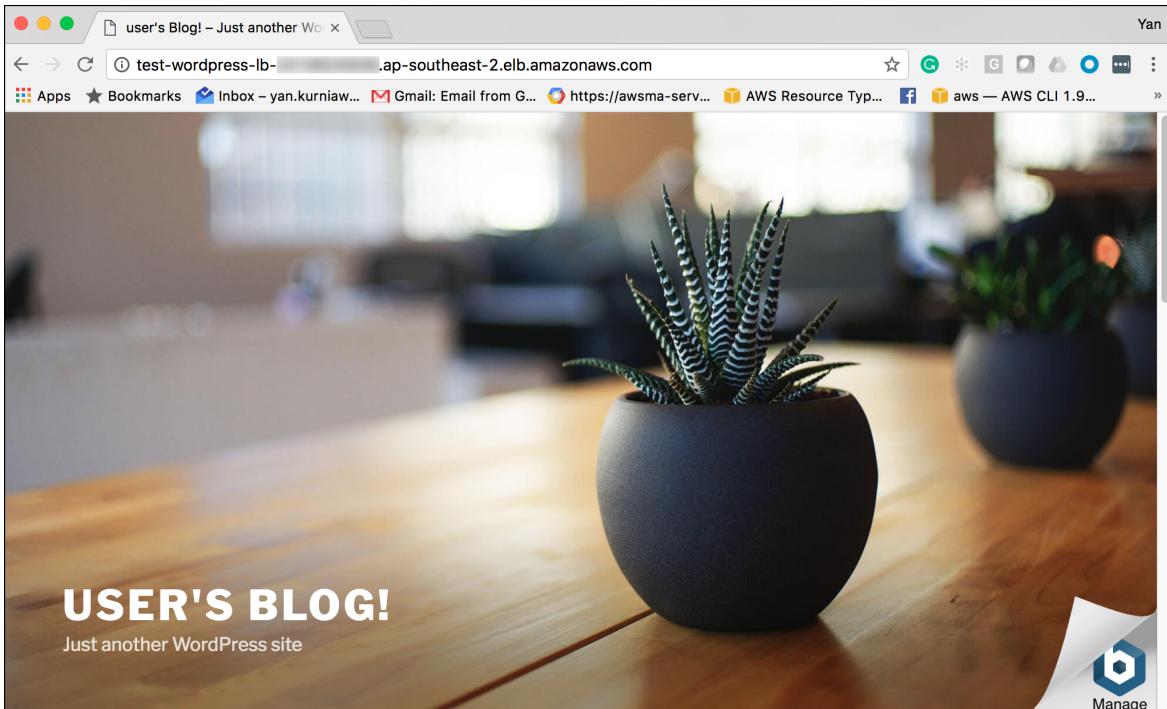
To verify your load balancer:

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under LOAD BALANCING, click Load Balancers.
3. Select your new load balancer.
4. In the bottom pane, on the Description tab, check the Status row. If it indicates that some of your instances are not in service, its probably because they are still in the registration process.

The screenshot shows the AWS Load Balancers console. At the top, there is a search bar and a filter section with columns for Name, DNS name, State, and VPC ID. A single load balancer named "test-wordpress-lb" is listed. Below the table, the "Description" tab is selected, showing the basic configuration of the load balancer. The configuration includes:

Name	test-wordpress-lb	Creation time	at 3:37:10 PM UTC+10
* DNS name	test-wordpress-lb-*.ap-southeast-2.elb.amazonaws.com (A Record)	Hosted zone	[REDACTED]
Scheme	internet-facing	Status	1 of 1 instances in service
Availability Zones	subnet-[REDACTED] - ap-southeast-2b, subnet-[REDACTED] - ap-southeast-2a	VPC	vpc-[REDACTED]

5. After you've verified that at least one of your EC2 instances is InService, you can test your load balancer. Copy the string from the DNS Name field and paste it into the address field of an Internet-connected web browser. If your load balancer is working, you'll see the default page of your WordPress server.



When you no longer need the load balancer, use the following procedure to detach it from your Auto Scaling group.

To detach a load balancer from a group:

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. Select your group.
4. In the bottom pane, on the **Details** tab, click **Edit**.
5. In **Load Balancers**, remove the load balancer.
6. Click **Save**.

Scheduled Scaling

Scaling based on a schedule allows you to scale your application in response to predictable load changes. For example, every week the traffic to your web application starts to increase on Wednesday, remains high on Thursday, and starts to decrease on Friday. You can plan your scaling activities based on the predictable traffic patterns of your web application.

To configure your Auto Scaling group to scale based on a schedule, you need to create scheduled actions. A scheduled action tells Auto Scaling to perform a scaling action at certain time in future. To create a scheduled scaling action, you specify the start time at which you want the scaling action to take effect, and you specify the new minimum, maximum, and desired size you want for that group at that time. At the specified time, Auto Scaling updates the group to set the new values for minimum, maximum, and desired sizes, as specified by your scaling action.

You can create scheduled actions for scaling one time only or for scaling on a recurring schedule.⁵⁹

Considerations for Scheduled Actions

When you create a scheduled action, keep the following in mind.

- Auto Scaling guarantees the order of execution for scheduled actions within the same group, but not for scheduled actions across groups.
- A scheduled action generally executes within seconds. However, the action may be delayed for up to two minutes from the scheduled start time. Because Auto Scaling executes actions within an Auto Scaling group in the order they are specified, scheduled actions with scheduled start times close to each other may take longer to execute.
- You can create a maximum of 125 scheduled actions per month per Auto Scaling group. This allows scaling four times a day for a 31-day month for each Auto Scaling group.
- A scheduled action must have a unique time value. If you attempt to schedule an activity at a time when another existing activity is already scheduled, the call is rejected with an error message noting the conflict.
- Cooldown periods are not supported.

Create a Scheduled Action Using the Console

Complete the following procedure to create a scheduled action to scale your Auto Scaling group.

To create a scheduled action:

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.

⁵⁹ http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/schedule_time.html

2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. Select your Auto Scaling group from the list.
4. The bottom pane displays the details of your Auto Scaling group. Select the **Scheduled Actions** tab and then click **Create Scheduled Action**.
5. In the **Create Scheduled Action** dialog box, do the following:
 - Specify the size of the group using at least one of **Min**, **Max**, and **Desired Capacity**.
 - Select an option from **Recurrence**. If you select **Once**, Auto Scaling performs the action at the specified time. If you select **Cron**, enter a CRON expression that specifies when Auto Scaling performs the action.
 - Specify the start and end time using **Start Time** and **End Time**.
 - Click **Create**.

Update a Scheduled Action

If your requirements change, you can update a scheduled action.

To update a scheduled action

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. Select your Auto Scaling group from the list.
4. The bottom pane displays the details of your Auto Scaling group. Select the **Scheduled Actions** tab, and then select the scheduled action.
5. Click **Actions** and then select **Edit**.
6. In the **Edit Scheduled Action** dialog box, do the following:
 - Update the size of the group as needed using **Min**, **Max**, or **Desired Capacity**.
 - Update the specified recurrence as needed.
 - Update the start and end time as needed.
 - Click **Save**.

Dynamic Scaling

When you use Auto Scaling to scale dynamically, you must define how you want to scale in response to changing demand. For example, say you have a web application that currently runs on two instances. You want to launch two additional instances when the load on the current instances rises to 70 percent, and then you want to terminate those additional instances when the load falls to 40 percent. You can configure your Auto Scaling group to scale automatically based on these conditions.

An Auto Scaling group uses a combination of alarms and policies to determine when the conditions for scaling are met. An *alarm* is an object that watches over a single metric (for example, the average CPU utilization of the EC2 instances in your Auto Scaling group) over a specified time period. When the value of the metric breaches the threshold that you defined, for the number of time periods that you specified, the alarm performs one or more actions (such as sending messages to Auto Scaling). A *policy* is a set of instructions that tells Auto Scaling how to respond to alarm messages.

To set up dynamic scaling, you must create alarms and scaling policies and associate them with your Auto Scaling group. It's recommended that you create two policies for each scaling change that you want to perform: one policy to scale out and another policy to scale in. After the alarm sends a message to Auto Scaling, Auto Scaling executes the associated policy to scale your group in (by terminating instances) or out (by launching instances). The process is as follows:⁶⁰

1. Amazon CloudWatch monitors the specified metrics for all the instances in the Auto Scaling group.
2. As demand grows or shrinks, the change is reflected in the metrics.
3. When the change in the metrics breaches the threshold of the CloudWatch alarm, the CloudWatch alarm performs an action. Depending on the breach, the action is a message sent to either the scale-in policy or the scale-out policy.
4. After the Auto Scaling policy receives the message, Auto Scaling performs the scaling activity for the Auto Scaling group.
5. This process continues until you delete either the scaling policies or the Auto Scaling group.

Scaling Adjustment Types

When a scaling policy is executed, it changes the current capacity of your Auto Scaling group using the scaling adjustment specified in the policy. A scaling adjustment can't change the capacity of the group above the maximum group size or below the minimum group size.

Auto Scaling supports the following adjustment types:

⁶⁰<http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/as-scale-based-on-demand.html>

- **ChangeInCapacity** — Increase or decrease the current capacity of the group by the specified number of instances. A positive value increases the capacity and a negative adjustment value decreases the capacity.

Example: If the current capacity of the group is 3 instances and the adjustment is 5, then when this policy is performed, Auto Scaling adds 5 instances to the group for a total of 8 instances.

- **ExactCapacity** — Change the current capacity of the group to the specified number of instances. Note that you must specify a positive value with this adjustment type.

Example: If the current capacity of the group is 3 instances and the adjustment is 5, then when this policy is performed, Auto Scaling changes the capacity to 5 instances.

- **PercentChangeInCapacity** — Increment or decrement the current capacity of the group by the specified percentage. A positive value increases the capacity and a negative value decreases the capacity. If the resulting value is not an integer, Auto Scaling rounds it as follows:

- Values greater than 1 are rounded down. For example, 12.7 is rounded to 12.
- Values between 0 and 1 are rounded to 1. For example, .67 is rounded to 1.
- Values between 0 and -1 are rounded to -1. For example, -.58 is rounded to -1.
- Values less than -1 are rounded up. For example, -6.67 is rounded to -6.

Example: If the current capacity is 10 instances and the adjustment is 10 percent, then when this policy is performed, Auto Scaling adds 1 instance to the group for a total of 11 instances.

Scaling Policy Types

When you create a scaling policy, you must specify its policy type. The policy type determines how the scaling action is performed. Auto Scaling supports the following policy types:

- **Simple scaling** — Increase or decrease the current capacity of the group based on a single scaling adjustment.
- **Step scaling** — Increase or decrease the current capacity of the group based on a set of scaling adjustments, known as step adjustments, that vary based on the size of the alarm breach.

Simple Scaling Policies

After a scaling activity is started, the policy must wait for the scaling activity or health check replacement to complete and the cooldown period to expire before it can respond to additional alarms. Cooldown periods help to prevent Auto Scaling from initiating additional scaling activities before the effects of previous activities are visible. You can use the default cooldown period associated with your Auto Scaling group, or you can override the default by specifying a cooldown period for your policy.

Note that Auto Scaling originally supported only this type of scaling policy. If you created your scaling policy before policy types were introduced, your policy is treated as a simple scaling policy.

Step Scaling Policies

After a scaling activity is started, the policy continues to respond to additional alarms, even while a scaling activity or health check replacement is in progress. Therefore, all alarms that are breached are evaluated by Auto Scaling as it receives the alarm messages. If you are creating a policy to scale out, you can specify the estimated warm-up time that it will take for a newly launched instance to be ready to contribute to the aggregated metrics.

Cooldown periods are not supported for step scaling policies. Therefore, you can't specify a cooldown period for these policies and the default cooldown period for the group doesn't apply. It's recommended that you use step scaling policies even if you have a single step adjustment, because Amazon continuously evaluates alarms and do not lock the group during scaling activities or health check replacements.

Step Adjustments

When you create a step scaling policy, you add one or more step adjustments, which enables you to scale based on the size of the alarm breach. Each step adjustment specifies a lower bound for the metric value, an upper bound for the metric value, and the amount by which to scale, based on the scaling adjustment type.

There are a few rules for the step adjustments for your policy:

- The ranges of your step adjustments can't overlap or have a gap.
- At most one step adjustment can have a null lower bound (negative infinity). If one step adjustment has a negative lower bound, then there must be a step adjustment with a null lower bound.
- At most one step adjustment can have a null upper bound (positive infinity). If one step adjustment has a positive upper bound, then there must be a step adjustment with a null upper bound.
- The upper and lower bound can't be null in the same step adjustment.
- If the metric value is above the breach threshold, the lower bound is inclusive and the upper bound is exclusive. If the metric value is below the breach threshold, the lower bound is exclusive and the upper bound is inclusive.

If you are using the API or the CLI, you specify the upper and lower bounds relative to the value of the aggregated metric. If you are using the AWS Management Console, you specify the upper and lower bounds as absolute values.

Auto Scaling applies the aggregation type to the metric data points from all instances and compares the aggregated metric value against the upper and lower bounds defined by the step adjustments to determine which step adjustment to perform. For example, suppose that you have an alarm with a breach threshold of 50 and a scaling adjustment type of PercentChangeInCapacity. You also have scale-out and scale-in policies with the following step adjustments:

Scale-out policy

Lower bound	Upper bound	Adjustment	Metric value
0	10	0	50 <= value < 60
10	20	10	60 <= value < 70
20	null	30	70 <= value < +infinity

Scale-in policy

Lower bound	Upper bound	Adjustment	Metric value
-10	0	0	40 < value <= 50
-20	-10	-10	30 < value <= 40
null	-20	-30	-infinity < value <= 30

Your group has both a current capacity and a desired capacity of 10 instances. The group maintains its current and desired capacity while the aggregated metric value is greater than 40 and less than 60.

If the metric value gets to 60, Auto Scaling increases the desired capacity of the group by 1 instance, to 11 instances, based on the second step adjustment of the scale-out policy (add 10 percent of 10 instances). After the new instance is running and its specified warm-up time has expired, Auto Scaling increases the current capacity of the group to 11 instances. If the metric value rises to 70 even after this increase in capacity, Auto Scaling increases the desired capacity of the group by another 3 instances, to 14 instances, based on the third step adjustment of the scale-out policy (add 30 percent of 11 instances, 3.3 instances, rounded down to 3 instances).

If the metric value gets to 40, Auto Scaling decreases the desired capacity of the group by 1 instance, to 13 instances, based on the second step adjustment of the scale-in policy (remove 10 percent of 14 instances, 1.4 instances, rounded down to 1 instance). If the metric value falls to 30 even after this decrease in capacity, Auto Scaling decreases the desired capacity of the group by another 3 instances, to 10 instances, based on the third step adjustment of the scale-in policy (remove 30 percent of 13 instances, 3.9 instances, rounded down to 3 instances).

Instance Warmup

With step scaling policies, you can specify the number of seconds that it takes for a newly launched instance to warm up. Until its specified warm-up time has expired, an instance is not counted toward the aggregated metrics of the Auto Scaling group.

While scaling out, Auto Scaling does not consider instances that are warming up as part of the current capacity of the group. Therefore, multiple alarm breaches that fall in the range of the same step adjustment result in a single scaling activity. This ensures that Auto Scaling doesn't add more

instances than you need. Using the example in the previous section, suppose that the metric gets to 60, and then it gets to 62 while the new instance is still warming up. The current capacity is still 10 instances, so Auto Scaling should add 1 instance (10 percent of 10 instances), but the desired capacity of the group is already 11 instances, so Auto Scaling does not increase the desired capacity further. However, if the metric gets to 70 while the new instance is still warming up, Auto Scaling should add 3 instances (30 percent of 10 instances), but the desired capacity of the group is already 11, so Auto Scaling adds only 2 instances, for a new desired capacity of 13 instances.

While scaling in, Auto Scaling considers instances that are terminating as part of the current capacity of the group. Therefore, Auto Scaling won't remove more instances from the Auto Scaling group than necessary.

Note that a scale-in activity can't start while a scale-out activity is in progress.

Scaling Based on Metrics

You can create a scaling policy that uses CloudWatch alarms to determine when your Auto Scaling group should scale out or scale in. Each CloudWatch alarm watches a single metric and sends messages to Auto Scaling when the metric breaches a threshold that you specify in your policy. You can use alarms to monitor any of the metrics that the services in AWS that you're using send to CloudWatch, or you can create and monitor your own custom metrics.

When you create a CloudWatch alarm, you can specify an Amazon SNS topic to send an email notification to when the alarm changes state.⁶¹

Create an Auto Scaling Group with Scaling Policies

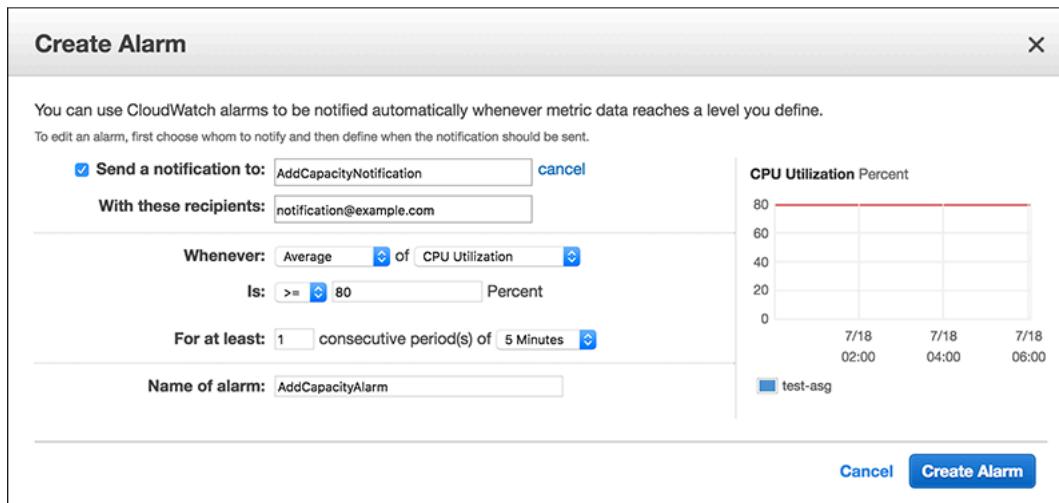
Use the console to create an Auto Scaling group with two scaling policies: a scale out policy that increases the capacity of the group by 30 percent, and a scale in policy that decreases the capacity of the group to two instances.

To create an Auto Scaling group with scaling based on metrics:

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under Auto Scaling, choose **Auto Scaling Groups**.
3. Choose **Create Auto Scaling group**.
4. On the **Create Auto Scaling Group** page, do one of the following:
 - Select **Create an Auto Scaling group from an existing launch configuration**, select an existing launch configuration, and then choose **Next Step**.
 - If you don't have a launch configuration that you'd like to use, choose **Create a new launch configuration** and follow the directions.
5. On the **Configure Auto Scaling group details** page, do the following:
 - For **Group name**, type a name for your Auto Scaling group.
 - For **Group size**, type the desired capacity for your Auto Scaling group.
 - If the launch configuration specifies instances that require a VPC, such as T2 instances, you must select a VPC from **Network**. Otherwise, if your AWS account supports EC2-Classic and the instances don't require a VPC, you can select either **Launch info EC2-Classic** or a VPC.
 - If you selected a VPC in the previous step, select one or more subnets from **Subnet**. If you selected EC2-Classic in the previous step, select one or more Availability Zones from **Availability Zone(s)**.
 - Choose **Next: Configure scaling policies**.
6. On the **Configure scaling policies** page, do the following:
 - Select **Use scaling policies to adjust the capacity of this group**.

⁶¹http://docs.aws.amazon.com/autoscaling/latest/userguide/policy_creating.html

- Specify the minimum and maximum size for your Auto Scaling group using the row that begins with **Scale between**. For example, if your group is already at its maximum size, you need to specify a new maximum in order to scale out.
- Specify your scale out policy under **Increase Group Size**. You can optionally specify a name for the policy, then choose **Add new alarm**.
- On the **Create Alarm** page, choose **create topic**. For **Send a notification to**, type a name for the SNS topic. For **With these recipients**, type one or more email addresses to receive notification. If you want, you can replace the default name for your alarm with a custom name. Next, specify the metric and the criteria for the policy. For example, you can leave the default settings for **Whenever** (Average of CPU Utilization). For **Is**, choose \geq and type 80 percent. For **For at least**, type 1 consecutive period of 5 Minutes. Choose **Create Alarm**.



- For **Take the action**, choose **Add**, type 30 in the next field, and then choose **percent of group**. By default, the lower bound for this step adjustment is the alarm threshold and the upper bound is null (positive infinity). To add another step adjustment, choose **Add step**.
(Optional) Amazon recommends that you use the default to create both scaling policies with steps. If you need to create simple scaling policies, choose **Create a simple scaling policy**.

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

Increase Group Size

Name: Increase Group Size

Execute policy when: AddCapacityAlarm [Edit](#) [Remove](#)
breaches the alarm threshold: CPUUtilization >= 80 for 300 seconds
for the metric dimensions AutoScalingGroupName = test-asg

Take the action: Add percent of group Add step [i](#)

Add instances in increments of at least instance(s)

Instances need: seconds to warm up after each step

[Create a simple scaling policy](#) [i](#)

- Specify your scale in policy under **Decrease Group Size**. You can optionally specify a name for the policy, then choose **Add new alarm**.
- On the **Create Alarm** page, you can select the same notification that you created for the scale out policy or create a new one for the scale in policy. If you want, you can replace the default name for your alarm with a custom name. Keep the default settings for **Whenever** (Average of CPU Utilization). For **Is**, choose \leq and type 40 percent. For **For at least**, type 1 consecutive period of 5 Minutes. Choose **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.

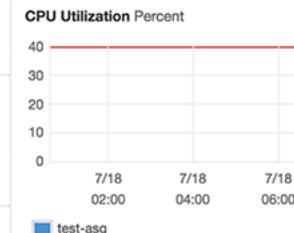
Send a notification to: [cancel](#)

With these recipients:

Whenever: Average of CPU Utilization
Is: \leq Percent

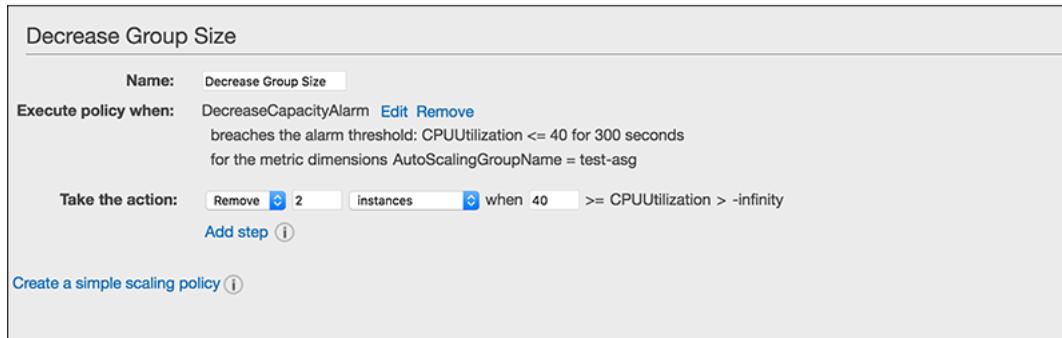
For at least: consecutive period(s) of

Name of alarm:



[Cancel](#) [Create Alarm](#)

- For **Take the action**, choose Remove, type 2 in the next field, and then choose instances. By default, the upper bound for this step adjustment is the alarm threshold and the lower bound is null (negative infinity). To add another step adjustment, choose **Add step**. (Optional) Amazon recommends that you use the default to create both scaling policies with steps. If you need to create simple scaling policies, choose **Create a simple scaling policy**.



- Choose **Review**.
 - On the **Review** page, choose **Create Auto Scaling group**.
7. Use the following steps to verify the scaling policies for your Auto Scaling group.
- The **Auto Scaling Group creation status** page confirms that your Auto Scaling group was successfully created. Choose **View your Auto Scaling Groups**.
 - On the **Auto Scaling Groups** page, select the Auto Scaling group that you just created.
 - On the **Activity History** tab, the **Status** column shows whether your Auto Scaling group has successfully launched instances.
 - On the **Instances** tab, the **Lifecycle** column contains the state of your instances. It takes a short time for an instance to launch. After the instance starts, its lifecycle state changes to **InService**.
 - The **Health Status** column shows the result of the EC2 instance health check on your instance.
 - On the **Scaling Policies** tab, you can see the policies that you created for the Auto Scaling group.

Add a Scaling Policy to an Auto Scaling Group

Use the console to add a scaling policy to an existing Auto Scaling group.

To update an Auto Scaling group with scaling based on metrics:

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select the Auto Scaling group.
4. On the **Scaling Policies** tab, choose **Add policy**.
5. For **Name**, type a name for the policy, and then choose **Create new alarm**.
6. On the **Create Alarm** page, choose **create topic**. For **Send a notification to**, type a name for the SNS topic. For **With these recipients**, type one or more email addresses to receive notification. If you want, you can replace the default name for your alarm with a custom name. Next, specify the metric and the criteria for the alarm, using **Whenever**, **Is**, and **For at least**. Choose **Create Alarm**.

7. Specify the scaling activity for the policy using **Take the action**. By default, the lower bound for this step adjustment is the alarm threshold and the upper bound is null (positive infinity). To add another step adjustment, choose **Add step**.

(Optional) Amazon recommends that you use the default to create both scaling policies with steps. If you need to create simple scaling policies, choose **Create a simple scaling policy**.

8. Choose **Create**.

Application Stack

The application stack contains all of the web application resources that will be used by our WordPress project: Elastic Load Balancers (ELB), AutoScaling Group (ASG), Launch Configuration, Scaling Policies, and Relational Database Service (RDS). These resources will be built on top of the network stack and will use IAM resource created in the IAM stack.



The classic Elastic Load Balancer has been renamed to Classic Load Balancers. In this example project we use Classic Load Balancers. If you need to address more complex load-balancing needs, by managing traffic at the application level, you can use AWS Application Load Balancers (ALB). The Classic Load Balancer operates at Layer 4 of the OSI model. It routes traffic between clients and backend servers based on IP address and TCP port. The Application Load Balancer operates at Layer 7 of the OSI model. It has the ability to inspect application-level content, not just IP and port. ALB supports path based routing and can be integrated with AWS Web Application Firewall.

The following application template defines basic infrastructure for a highly available web application design. This template includes an ELB (multiple Availability Zones), an ASG (multiple Availability Zones), a launch config, scaling policies, CloudWatch alarms, an RDS (Multi AZ), and an OpenVPN Access Server EC2 instance.

```
1 AWSTemplateFormatVersion: 2010-09-09
2 Description: Application Stack
3
4 Parameters:
5   IAMStackName:
6     Type: String
7     Description: IAM CF Stack name
8     Default: test-iam
9   NetworkStackName:
10    Type: String
11    Description: Network CF Stack name
12    Default: test-network
13   WordPressVersion:
14    Type: String
15    Description: WordPress version to be installed
16    Default: 4.9.4
17   WebServerCapacity:
18    Default: 1
19    Description: The initial number of WebServer instances
20    Type: Number
21    MinValue: 1
```

```
22     MaxValue: 5
23     ConstraintDescription: Must be between 1 and 5 EC2 instances
24 WebServerInstanceType:
25     Default: t2.small
26     Description: WebServer EC2 instance type
27     Type: String
28     AllowedValues:
29         - t2.small
30         - m3.medium
31         - m4.large
32     ConstraintDescription: Must be a valid EC2 instance type
33 KeyPair:
34     ConstraintDescription: Must be the name of an existing EC2 KeyPair.
35     Default: yan-key-pair-apsydney
36     Description: Name of an existing EC2 KeyPair to enable SSH access to the ins\
37 tances
38     Type: AWS::EC2::KeyPair::KeyName
39 DBInstanceClass:
40     Default: db.t2.small
41     Description: Database instance class
42     Type: String
43     AllowedValues:
44         - db.t2.small
45         - db.m3.medium
46         - db.m4.large
47     ConstraintDescription: Must be a valid database instance type
48 DBName:
49     Default: wordpressdb
50     Description: The WordPress database name
51     Type: String
52     MinLength: 1
53     MaxLength: 64
54     AllowedPattern: "[a-zA-Z][a-zA-Z0-9]*"
55     ConstraintDescription: Must begin with a letter and contain only alphanumeric\
56 c characters
57 DBUser:
58     NoEcho: true
59     Description: The WordPress database admin account username
60     Type: String
61     MinLength: 1
62     MaxLength: 16
63     AllowedPattern: "[a-zA-Z][a-zA-Z0-9]*"
```

```
64      ConstraintDescription: Must begin with a letter and contain only alphanumeric\
65  c characters
66  DBPassword:
67      NoEcho: true
68      Description: The WordPress database admin account password
69      Type: String
70      MinLength: 8
71      MaxLength: 41
72      AllowedPattern: "[a-zA-Z0-9]*"
73      ConstraintDescription: Must contain only alphanumeric characters
74  DBAllocatedStorage:
75      Default: 5
76      Description: The size of the database (GB)
77      Type: Number
78      MinValue: 5
79      MaxValue: 1024
80      ConstraintDescription: Must be between 5 and 1024 GB
81
82  Mappings:
83  Region2AMI:
84      us-east-1:
85          AmiId: ami-97785bed
86      us-east-2:
87          AmiId: ami-f63b1193
88      us-west-2:
89          AmiId: ami-f2d3638a
90      us-west-1:
91          AmiId: ami-824c4ee2
92      eu-west-1:
93          AmiId: ami-d834aba1
94      eu-west-2:
95          AmiId: ami-403e2524
96      eu-west-3:
97          AmiId: ami-8ee056f3
98      eu-central-1:
99          AmiId: ami-5652ce39
100     ap-northeast-1:
101         AmiId: ami-ceafcba8
102     ap-northeast-2:
103         AmiId: ami-863090e8
104     ap-northeast-3:
105         AmiId: ami-83444afe
```

```
106      ap-southeast-1:  
107          AmiId: ami-68097514  
108      ap-southeast-2:  
109          AmiId: ami-942dd1f6  
110      ap-south-1:  
111          AmiId: ami-531a4c3c  
112      ca-central-1:  
113          AmiId: ami-a954d1cd  
114      sa-east-1:  
115          AmiId: ami-84175ae8  
116      cn-north-1:  
117          AmiId: ami-cb19c4a6  
118      cn-northwest-1:  
119          AmiId: ami-3e60745c  
120  Region2OpenVPNAMI:  
121      us-east-1:  
122          AmiId: ami-f6eed4e0  
123      us-east-2:  
124          AmiId: ami-6d163708  
125      us-west-2:  
126          AmiId: ami-e346559a  
127      us-west-1:  
128          AmiId: ami-091f3069  
129      eu-west-1:  
130          AmiId: ami-238b6a5a  
131      eu-west-2:  
132          AmiId: ami-17c5d373  
133      eu-west-3:  
134          AmiId: ami-64a11619  
135      eu-central-1:  
136          AmiId: ami-17862678  
137      ap-northeast-1:  
138          AmiId: ami-dee1fdb9  
139      ap-northeast-2:  
140          AmiId: ami-c98c52a7  
141      ap-southeast-1:  
142          AmiId: ami-81d75de2  
143      ap-southeast-2:  
144          AmiId: ami-3cd6c45f  
145      ap-south-1:  
146          AmiId: ami-cdd4aaa2  
147      ca-central-1:
```

```
148     AmiId: ami-c6813ea2
149     sa-east-1:
150         AmiId: ami-930673ff
151
152 Resources:
153     WebServerGroup:
154         Type: AWS::AutoScaling::AutoScalingGroup
155         Properties:
156             VPCZoneIdentifier:
157                 - Fn::ImportValue:
158                     !Sub ${NetworkStackName}-PrivateSubnet0
159                 - Fn::ImportValue:
160                     !Sub ${NetworkStackName}-PrivateSubnet1
161             LaunchConfigurationName: !Ref WebServerLC
162             MinSize: 1
163             MaxSize: 5
164             DesiredCapacity: !Ref WebServerCapacity
165             LoadBalancerNames:
166                 - !Ref AppELB
167             Tags:
168                 -
169                     Key: Name
170                     Value: !Sub ${AWS::StackName}-WebServer
171                     PropagateAtLaunch: true
172     UpdatePolicy:
173         AutoScalingRollingUpdate:
174             MinInstancesInService: 1
175             MaxBatchSize: 1
176             PauseTime: PT15M
177             WaitOnResourceSignals: true
178     CreationPolicy:
179         ResourceSignal:
180             Timeout: PT15M
181     WebServerLC:
182         Type: AWS::AutoScaling::LaunchConfiguration
183         Properties:
184             AssociatePublicIpAddress: false
185             ImageId: !FindInMap [ Region2AMI, !Ref "AWS::Region", AmiId ]
186             IamInstanceProfile:
187                 Fn::ImportValue:
188                     !Sub ${IAMStackName}-AppInstanceProfile
189             InstanceType: !Ref WebServerInstanceType
```

```
190      KeyName: !Ref KeyPair
191      SecurityGroups:
192        - Fn::ImportValue:
193          !Sub ${NetworkStackName}-AppSecurityGroup
194      UserData:
195        Fn::Base64: !Sub
196          - |
197            #!/bin/bash
198            exec > >(tee /var/log/userdata.log)
199            exec 2>&1
200            yum install -y httpd php php-mysql
201            wget -O /var/www/html/wordpress-${WordPressVersion}.tar.gz https://w\
202 ordpress.org/wordpress-${WordPressVersion}.tar.gz
203            tar -xzvf /var/www/html/wordpress-${WordPressVersion}.tar.gz -C /var\
204 /www/html/
205            rm -f /var/www/html/wordpress-${WordPressVersion}.tar.gz
206            cp /var/www/html/wordpress/wp-config-sample.php /var/www/html/wordpr\
207 ess/wp-config.php
208            sed -i "s/database_name_here/${DBName}/g" /var/www/html/wordpress/wp\
209 -config.php
210            sed -i "s/username_here/${DBUser}/g" /var/www/html/wordpress/wp-conf\
211 ig.php
212            sed -i "s/password_here/${DBPassword}/g" /var/www/html/wordpress/wp-\\
213 config.php
214            sed -i "s/localhost/${DBAddress}/g" /var/www/html/wordpress/wp-confi\
215 g.php
216            service httpd start
217            chkconfig httpd on
218            /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource W\
219 ebServerGroup --region ${AWS::Region}
220          - {
221            DBAddress: !GetAtt DBInstance.Endpoint.Address
222          }
223      WebServerScaleUpPolicy:
224        Type: AWS::AutoScaling::ScalingPolicy
225        Properties:
226          AdjustmentType: ChangeInCapacity
227          AutoScalingGroupName: !Ref WebServerGroup
228          Cooldown: 60
229          ScalingAdjustment: 1
230      WebServerScaleDownPolicy:
231        Type: AWS::AutoScaling::ScalingPolicy
```

```
232     Properties:  
233         AdjustmentType: ChangeInCapacity  
234         AutoScalingGroupName: !Ref WebServerGroup  
235         Cooldown: 60  
236         ScalingAdjustment: -1  
237     CPUAlarmHigh:  
238         Type: AWS::CloudWatch::Alarm  
239         Properties:  
240             AlarmDescription: Scale-up if CPU > 90% for 10 minutes  
241             MetricName: CPUUtilization  
242             Namespace: AWS/EC2  
243             Statistic: Average  
244             Period: 300  
245             EvaluationPeriods: 2  
246             Threshold: 90  
247             AlarmActions:  
248                 - !Ref WebServerScaleUpPolicy  
249             Dimensions:  
250                 - Name: AutoScalingGroupName  
251                     Value: !Ref WebServerGroup  
252             ComparisonOperator: GreaterThanThreshold  
253     CPUAlarmLow:  
254         Type: AWS::CloudWatch::Alarm  
255         Properties:  
256             AlarmDescription: Scale-down if CPU < 70% for 10 minutes  
257             MetricName: CPUUtilization  
258             Namespace: AWS/EC2  
259             Statistic: Average  
260             Period: 300  
261             EvaluationPeriods: 2  
262             Threshold: 70  
263             AlarmActions:  
264                 - !Ref WebServerScaleDownPolicy  
265             Dimensions:  
266                 - Name: AutoScalingGroupName  
267                     Value: !Ref WebServerGroup  
268             ComparisonOperator: LessThanThreshold  
269  
270     AppELB:  
271         Type: AWS::ElasticLoadBalancing::LoadBalancer  
272         Properties:  
273             Scheme: internet-facing
```

```
274     SecurityGroups:
275         - Fn::ImportValue:
276             !Sub ${NetworkStackName}-AppELBSecurityGroup
277     Subnets:
278         - Fn::ImportValue:
279             !Sub ${NetworkStackName}-PublicSubnet0
280         - Fn::ImportValue:
281             !Sub ${NetworkStackName}-PublicSubnet1
282     CrossZone: true
283     Listeners:
284         -
285             LoadBalancerPort: 80
286             InstancePort: 80
287             Protocol: HTTP
288     HealthCheck:
289         Target: HTTP:80/wordpress/wp-admin/install.php
290         HealthyThreshold: 3
291         UnhealthyThreshold: 5
292         Interval: 30
293         Timeout: 5
294 DBSubnetGroup:
295     Type: AWS::RDS::DBSubnetGroup
296     Properties:
297         DBSubnetGroupDescription: Subnet group for RDS
298         SubnetIds:
299             - Fn::ImportValue:
300                 !Sub ${NetworkStackName}-PrivateSubnet0
301             - Fn::ImportValue:
302                 !Sub ${NetworkStackName}-PrivateSubnet1
303 DBInstance:
304     Type: AWS::RDS::DBInstance
305     Properties:
306         DBName: !Ref DBName
307         Engine: MySQL
308         MultiAZ: true
309         MasterUsername: !Ref DBUser
310         MasterUserPassword: !Ref DBPassword
311         DBInstanceClass: !Ref DBInstanceClass
312         DBSubnetGroupName: !Ref DBSubnetGroup
313         AllocatedStorage: !Ref DBAllocatedStorage
314         VPCSecurityGroups:
315             - Fn::ImportValue:
```

```
316      !Sub ${NetworkStackName}-DBSecurityGroup
317  OpenVPNInstance:
318    Type: AWS::EC2::Instance
319    Properties:
320      ImageId: !FindInMap [ Region2OpenVPNAMI, !Ref "AWS::Region", AmiId ]
321      InstanceType: t2.micro
322      SourceDestCheck: false
323      KeyName: !Ref KeyPair
324      SubnetId:
325        Fn::ImportValue:
326          !Sub ${NetworkStackName}-PublicSubnet0
327        SecurityGroupIds:
328          - Fn::ImportValue:
329            !Sub ${NetworkStackName}-OpenVPNSecurityGroup
330        Tags:
331          -
332            Key: Name
333            Value: !Sub ${AWS::StackName}-OpenVPNServer
334  Outputs:
335    ELBAddress:
336      Value: !GetAtt AppELB.DNSName
```

To create the CloudFormation stack:

1. Open the CloudFormation console at <https://console.aws.amazon.com/cloudformation/>.
2. Click **Create Stack**.
3. On the **Select Template** page, choose **Upload a template to Amazon S3** and select the `app.yaml` template.
4. Click **Next**.
5. On the **Specify Details** page, specify the stack name, for example `test-app`. You can change the parameter values in the **Parameters** section.

Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. [Learn more.](#)

Stack name [Edit](#)

Parameters

DBAllocatedStorage	<input type="text" value="5"/>	The size of the database (GB)
DBInstanceClass	<input type="text" value="db.t2.small"/>	Database instance class
DBName	<input type="text" value="wordpressdb"/>	The WordPress database name
DBPassword	<input type="password" value="*****"/>	The WordPress database admin account password
DBUser	<input type="text" value="*****"/>	The WordPress database admin account username
IAMStackName	<input type="text" value="test-iam"/>	IAM CF Stack name
KeyPair	<input type="text" value="yan-key-pair-apsydney"/>	Name of an existing EC2 KeyPair to enable SSH access to the instances
NetworkStackName	<input type="text" value="test-network"/>	Network CF Stack name
WebServerCapacity	<input type="text" value="1"/>	The initial number of WebServer instances
WebServerInstanceType	<input type="text" value="t2.small"/>	WebServer EC2 Instance type
WordPressVersion	<input type="text" value="4.9.4"/>	WordPress version to be installed

[Cancel](#) [Previous](#) **Next**

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

Once you have deployed the application stack, you should be able to open the WordPress site.

To open the WordPress site:

1. Get the ELB public address from the outputs of the application stack. Open the CloudFormation console at <https://console.aws.amazon.com/cloudformation/>
2. Select the application stack then select the **Outputs** tab. You can see the ELB public address there.
3. In a web browser, enter the public DNS address for your ELB followed by the `wordpress` folder. For example <http://test-app-AppELB-1LHAEL2PA3AGY-248604543.ap-southeast-2.elb.amazonaws.com/wordpress>.
4. You should see the WordPress installation screen.

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title

Username
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password &%znZL4%&x8#!nwN#^ (Hide) **Strong**
Important: You will need this password to log in. Please store it in a secure location.

Your Email
Double-check your email address before continuing.

Search Engine Visibility Discourage search engines from indexing this site
It is up to search engines to honor this request.

Install WordPress

Base Image

In our application template, Amazon Linux AMI from the AWS marketplace is used as the base image for the launch configuration. Packages installation and WordPress configuration are done in the launch configuration's user data. The user data will only run during the first boot cycle when an instance launched.

The user data in the template is just an example of how to do things with CloudFormation. It's not suitable for a production WordPress site. As you can see in the user data there are some `sed` commands that replace database user and password in the WordPress configuration file. Those values will be visible in the EC2 console. Anyone who has permission to view EC2 instances in your AWS account will be able to see the user data and read the password.

It is recommended that you create your own AMI for production, with all the packages installed. For the configuration file and WordPress content, you can put them in S3 and run AWS CLI in the user data to download them from S3. You can also use AWS Key Management Service (KMS) to encrypt the configuration file before uploading it to S3. Then in the user data you will need to decrypt the file after downloading it from S3.

If you use your own AMI as a base image, you can remove the Region2AMI mapping, add a parameter for base AMI ID and use a Ref to parameter in the launch configuration.

OpenVPN

The application stack will create an OpenVPN Access Server. You can connect to the EC2 instances in the private subnets via a VPN connection.

The OpenVPN Access Server AMI used in the template is the free version one, it will allow 2 concurrent users. To support more than 2 concurrent users you can choose to use OpenVPN Access Server AMI with 10, 25, 50, 100, 250, or 500 connected devices. Go to AWS marketplace <https://aws.amazon.com/marketplace> and search for OpenVPN Access Server, then you can update the template AMI ID mapping.

OpenVPN Access Server is a full featured secure network tunneling VPN software solution that integrates OpenVPN server capabilities, enterprise management capabilities, simplified OpenVPN Connect UI, and OpenVPN Client software packages that accommodate Windows, MAC, Linux, Android, and iOS environments. OpenVPN Access Server supports a wide range of configurations, including secure and granular remote access to internal network and/or private cloud network resources and applications with fine-grained access control.⁶²

To configure the OpenVPN Access Server, SSH to the OpenVPN Access Server as openvpnas user:

```
$ ssh -i /.ssh/yan-key-pair-apsydney.pem openvpnas@openvpn-ipaddress
```

The OpenVPN Access Server Setup Wizard runs automatically upon your initial login to the appliance. If you would like to run this wizard again in the future, issue the sudo ovpn-init --ec2 command in the terminal.

```
Please enter 'yes' to indicate your agreement [no]: yes
```

```
Once you provide a few initial configuration settings,  
OpenVPN Access Server can be configured by accessing  
its Admin Web UI using your Web browser.
```

```
Will this be the primary Access Server node?  
(enter 'no' to configure as a backup or standby node)  
> Press ENTER for default [yes]:
```

```
Please specify the network interface and IP address to be  
used by the Admin Web UI:  
(1) all interfaces: 0.0.0.0  
(2) eth0: 10.0.1.241
```

⁶²<https://openvpn.net/index.php/access-server/overview.html>

Please enter the option number from the list above (1-2).

> Press Enter for default [2]: 1

Please specify the port number for the Admin Web UI.

> Press ENTER for default [943]:

Please specify the TCP port number for the OpenVPN Daemon

> Press ENTER for default [443]:

Should client traffic be routed by default through the VPN?

> Press ENTER for default [no]:

Should client DNS traffic be routed by default through the VPN?

> Press ENTER for default [no]: yes

Use local authentication via internal DB?

> Press ENTER for default [yes]:

Private subnets detected: ['10.0.0.0/16']

Should private subnets be accessible to clients by default?

> Press ENTER for EC2 default [yes]:

To initially login to the Admin Web UI, you must use a username and password that successfully authenticates you with the host UNIX system (you can later modify the settings so that RADIUS or LDAP is used for authentication instead).

You can login to the Admin Web UI as "openvpn" or specify a different user account to use for this purpose.

Do you wish to login to the Admin UI as "openvpn"?

> Press ENTER for default [yes]:

> Please specify your OpenVPN-AS license key (or leave blank to specify later):

Initializing OpenVPN...

Adding new user login...

useradd -s /sbin/nologin "openvpn"

Writing as configuration file...

Perform sa init...

```
Wiping any previous userdb...
Creating default profile...
Modifying default profile...
Adding new user to userdb...
Modifying new user as superuser in userdb...
Getting hostname...
Hostname: 13.54.85.77
Preparing web certificates...
Getting web user account...
Adding web group account...
Adding web group...
Adjusting license directory ownership...
Initializing confdb...
Generating init scripts...
Generating PAM config...
Generating init scripts auto command...
Starting openvpnas...
```

NOTE: Your system clock must be correct for OpenVPN Access Server to perform correctly. Please ensure that your time and date are correct on this system.

Initial Configuration Complete!

You can now continue configuring OpenVPN Access Server by directing your Web browser to this URL:

<https://13.54.85.77:943/admin>
Login as "openvpn" with the same password used to authenticate to this UNIX host.

During normal operation, OpenVPN AS can be accessed via these URLs:
Admin UI: <https://13.54.85.77:943/admin>
Client UI: <https://13.54.85.77:943/>

See the Release Notes for this release at:
http://www.openvpn.net/access-server/rn/openvpn_as_2_1_9.html

To set the openvpn user password:

```
$ sudo passwd openvpn
```

You can open the OpenVPN Access Server admin page at https://<server_address>:943/admin.

For more information on how to connect via OpenVPN, go to <https://openvpn.net/index.php/access-server/docs/admin-guides-sp-859543150/howto-connect-client-configuration.html>.

AWS::AutoScaling::LaunchConfiguration

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-launchconfig.html>

The AWS::AutoScaling::LaunchConfiguration type creates an Auto Scaling launch configuration that can be used by an Auto Scaling group to configure Amazon EC2 instances in the Auto Scaling group.



When you update a property of the LaunchConfiguration resource, AWS CloudFormation deletes that resource and creates a new launch configuration with the updated properties and a new name. This update action does not deploy any change across the running Amazon EC2 instances in the auto scaling group. In other words, an update simply replaces the LaunchConfiguration so that when the auto scaling group launches new instances, they will get the updated configuration, but existing instances continue to run with the configuration that they were originally launched with. This works the same way as if you made similar changes manually to an auto scaling group. If you want to update existing instances when you update the LaunchConfiguration resource, you must specify an update policy attribute for the AWS::AutoScaling::AutoScalingGroup resource.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::AutoScaling::LaunchConfiguration"
2 Properties:
3   AssociatePublicIpAddress: <Boolean>
4   BlockDeviceMappings:
5     - <BlockDeviceMapping>
6   ClassicLinkVPCId: <String>
7   ClassicLinkVPCCSecurityGroups:
8     - <String>
9   EbsOptimized: <Boolean>
10  IamInstanceProfile: <String>
11  ImageId: <String>
12  InstanceId: <String>
13  InstanceMonitoring: <Boolean>
14  InstanceType: <String>
15  KernelId: <String>
16  KeyName: <String>
17  PlacementTenancy: <String>
```

```
18 RamDiskId: <String>
19 SecurityGroups:
20   - <SecurityGroup>
21 SpotPrice: <String>
22 UserData: <String>
```

Properties

AssociatePublicIpAddress

For Amazon EC2 instances in a VPC, indicates whether instances in the Auto Scaling group receive public IP addresses. If you specify true, each instance in the Auto Scaling receives a unique public IP address.

Note: If this resource has a public IP address and is also in a VPC that is defined in the same template, you must use the DependsOn attribute to declare a dependency on the VPC-gateway attachment.

Required: No

Type: Boolean

Update requires: Replacement

BlockDeviceMappings

Specifies how block devices are exposed to the instance. You can specify virtual devices and EBS volumes.

Required: No

Type: A list of BlockDeviceMappings.

Update requires: Replacement

ClassicLinkVPCId

The ID of a ClassicLink-enabled VPC to link your EC2-Classic instances to. You can specify this property only for EC2-Classic instances.

Required: No

Type: String

Update requires: Replacement

ClassicLinkVPCTrustedList

The IDs of one or more security groups for the VPC that you specified in the ClassicLinkVPCId property.

Required: Conditional. If you specified the ClassicLinkVPCId property, you must specify this property.

Type: List of strings

Update requires: Replacement

EbsOptimized

Specifies whether the launch configuration is optimized for EBS I/O. This optimization provides dedicated throughput to Amazon EBS and an optimized configuration stack to provide optimal EBS I/O performance.

Additional fees are incurred when using EBS-optimized instances.

Required: No. If this property is not specified, “false” is used.

Type: Boolean

Update requires: Replacement

IamInstanceProfile

Provides the name or the Amazon Resource Name (ARN) of the instance profile associated with the IAM role for the instance. The instance profile contains the IAM role.

Required: No

Type: String (1–1600 chars)

Update requires: Replacement

ImageId

Provides the unique ID of the Amazon Machine Image (AMI) that was assigned during registration.

Required: Yes

Type: String

Update requires: Replacement

InstanceId

The ID of the Amazon EC2 instance you want to use to create the launch configuration. Use this property if you want the launch configuration to use settings from an existing Amazon EC2 instance.

When you use an instance to create a launch configuration, all properties are derived from the instance with the exception of `BlockDeviceMapping` and `AssociatePublicIpAddress`. You can override any properties from the instance by specifying them in the launch configuration.

Required: No

Type: String

Update requires: Replacement

InstanceMonitoring

Indicates whether detailed instance monitoring is enabled for the Auto Scaling group. By default, this property is set to true (enabled).

When detailed monitoring is enabled, Amazon CloudWatch (CloudWatch) generates metrics every minute and your account is charged a fee. When you disable detailed monitoring, CloudWatch generates metrics every 5 minutes.

Required: No

Type: Boolean

Update requires: Replacement

InstanceType

Specifies the instance type of the EC2 instance.

Required: Yes

Type: String

Update requires: Replacement

KernelId

Provides the ID of the kernel associated with the EC2 AMI.

Required: No

Type: String

Update requires: Replacement

KeyName

Provides the name of the EC2 key pair.

Required: No

Type: String

Update requires: Replacement

PlacementTenancy

The tenancy of the instance. An instance with a tenancy of dedicated runs on single-tenant hardware and can only be launched in a VPC. You must set the value of this parameter to dedicated if want to launch dedicated instances in a shared tenancy VPC (a VPC with the instance placement tenancy attribute set to default).

If you specify this property, you must specify at least one subnet in the `VPCZoneIdentifier` property of the `AWS::AutoScaling::AutoScalingGroup` resource.

Required: No

Type: String

Update requires: Replacement

RamDiskId

The ID of the RAM disk to select. Some kernels require additional drivers at launch. Check

the kernel requirements for information about whether you need to specify a RAM disk. To find kernel requirements, refer to the AWS Resource Center and search for the kernel ID.

Required: No

Type: String

Update requires: Replacement

SecurityGroups

A list that contains the EC2 security groups to assign to the Amazon EC2 instances in the Auto Scaling group. The list can contain the name of existing EC2 security groups or references to AWS::EC2::SecurityGroup resources created in the template. If your instances are launched within VPC, specify Amazon VPC security group IDs.

Required: No

Type: A list of EC2 security groups.

Update requires: Replacement

SpotPrice

The spot price for this autoscaling group. If a spot price is set, then the autoscaling group will launch when the current spot price is less than the amount specified in the template.

When you have specified a spot price for an auto scaling group, the group will only launch when the spot price has been met, regardless of the setting in the autoscaling group's DesiredCapacity.

Required: No

Type: String

Update requires: Replacement

Note: When you change your bid price by creating a new launch configuration, running instances will continue to run as long as the bid price for those running instances is higher than the current Spot price.

UserData

The user data available to the launched EC2 instances.

Required: No

Type: String

Update requires: Replacement

Return Value

When the logical ID of this resource is provided to the Ref intrinsic function, Ref returns the resource name.

Template Examples

Example LaunchConfig with block device

This example shows a launch configuration that describes two Amazon Elastic Block Store mappings.

```
1 LaunchConfig:
2   Type: "AWS::AutoScaling::LaunchConfiguration"
3   Properties:
4     KeyName:
5       Ref: "KeyName"
6     ImageId:
7       Fn::FindInMap:
8         - "AWSRegionArch2AMI"
9         - Ref: "AWS::Region"
10        - Fn::FindInMap:
11          - "AWSInstanceType2Arch"
12          - Ref: "InstanceType"
13          - "Arch"
14     UserData:
15       Fn::Base64:
16         Ref: "WebServerPort"
17     SecurityGroups:
18       - Ref: "InstanceSecurityGroup"
19     InstanceType:
20       Ref: "InstanceType"
21     BlockDeviceMappings:
22       - DeviceName: "/dev/sda1"
23       Ebs:
24         VolumeSize: "50"
25         VolumeType: "io1"
26         Iops: 200
27       - DeviceName: "/dev/sdm"
28       Ebs:
29         VolumeSize: "100"
30         DeleteOnTermination: "true"
```

Example LaunchConfig with Spot Price in Autoscaling Group

This example shows a launch configuration that features a spot price in the AutoScaling group. This launch configuration will only be active if the current spot price is less than the amount in the template specification (0.05).

```

1 LaunchConfig:
2   Type: "AWS::AutoScaling::LaunchConfiguration"
3   Properties:
4     KeyName:
5       Ref: "KeyName"
6     ImageId:
7       Fn::FindInMap:
8         - "AWSRegionArch2AMI"
9         - Ref: "AWS::Region"
10        - Fn::FindInMap:
11          - "AWSInstanceType2Arch"
12          - Ref: "InstanceType"
13          - "Arch"
14     SecurityGroups:
15       - Ref: "InstanceSecurityGroup"
16     SpotPrice: "0.05"
17     InstanceType:
18       Ref: "InstanceType"

```

Example LaunchConfig with IAM Instance Profile

Here's a launch configuration using the `IamInstanceProfile` property.

Only the `AWS::AutoScaling::LaunchConfiguration` specification is shown.

```

1 myLCOne:
2   Type: "AWS::AutoScaling::LaunchConfiguration"
3   Properties:
4     ImageId:
5       Fn::FindInMap:
6         - "AWSRegionArch2AMI"
7         - Ref: "AWS::Region"
8         - Fn::FindInMap:
9           - "AWSInstanceType2Arch"
10          - Ref: "InstanceType"
11          - "Arch"
12     InstanceType:
13       Ref: "InstanceType"
14     IamInstanceProfile:
15       Ref: "RootInstanceProfile"

```

Example EBS-optimized volume with specified PIOPS

You can create an AWS CloudFormation stack with auto scaled instances that contain EBS-optimized volumes with a specified PIOPS. This can increase the performance of your EBS-backed instances

as explained in Increasing EBS Performance in the Amazon Elastic Compute Cloud User Guide.



Additional fees are incurred when using EBS-optimized instances.

Because you cannot override PIOPS settings in an auto scaling launch configuration, the AMI in your launch configuration must have been configured with a block device mapping that specifies the desired PIOPS. You can do this by creating your own EC2 AMI with the following characteristics:

- An instance type of `m1.large` or greater. This is required for EBS optimization.
- An EBS-backed AMI with a volume type of “`io1`” and the number of IOPS you want for the Auto Scaling-launched instances.
- The size of the EBS volume must accommodate the IOPS you need. There is a $10 : 1$ ratio between IOPS and Gibibytes (GiB) of storage, so for 100 PIOPS, you need at least 10 GiB storage on the root volume.

Use this AMI in your Auto Scaling launch configuration. For example, an EBS-optimized AMI with PIOPS that has the AMI ID `ami-7430ba44` would be used in your launch configuration like this:

```
1 LaunchConfig:  
2   Type: "AWS::AutoScaling::LaunchConfiguration"  
3   Properties:  
4     KeyName:  
5       Ref: "KeyName"  
6     ImageId: "ami-7430ba44"  
7     UserData:  
8       Fn::Base64:  
9         Ref: "WebServerPort"  
10    SecurityGroups:  
11      - Ref: "InstanceSecurityGroup"  
12    InstanceType: "m1.large"  
13    EbsOptimized: "true"
```

Be sure to set the `InstanceType` to at least `m1.large` and set `EbsOptimized` to true.

When you create a launch configuration such as this one, your launched instances will contain optimized EBS root volumes with the PIOPS that you selected when creating the AMI.

AWS::AutoScaling::AutoScalingGroup

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-group.html>

The AWS::AutoScaling::AutoScalingGroup type creates an Auto Scaling group.

You can add an UpdatePolicy attribute to your Auto Scaling group to control how rolling updates are performed when a change has been made to the Auto Scaling group's launch configuration or subnet group membership.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::AutoScaling::AutoScalingGroup"
2 Properties:
3   AvailabilityZones:
4     - <String>
5   Cooldown: <String>
6   DesiredCapacity: <String>
7   HealthCheckGracePeriod: <Integer>
8   HealthCheckType: <String>
9   InstanceId: <String>
10  LaunchConfigurationName: <String>
11  LoadBalancerNames:
12    - <String>
13  MaxSize: <String>
14  MetricsCollection:
15    - <MetricsCollection>
16  MinSize: <String>
17  NotificationConfigurations:
18    - <NotificationConfigurations>
19  PlacementGroup: <String>
20  Tags:
21    - <Auto Scaling Tag>
22  TargetGroupARNs:
23    - <String>
24  TerminationPolicies:
25    - <String>
26  VPCZoneIdentifier:
27    - <String>
```

Properties

AvailabilityZones

Contains a list of availability zones for the group.

Required: Conditional. If you don't specify the `VPCZoneIdentifier` property, you must specify this property.

Type: List of strings

Update requires: No interruption

Cooldown

The number of seconds after a scaling activity is completed before any further scaling activities can start.

Required: No

Type: String

Update requires: No interruption

DesiredCapacity

Specifies the desired capacity for the Auto Scaling group.

If `SpotPrice` is not set in the `AWS::AutoScaling::LaunchConfiguration` for this Auto Scaling group, then Auto Scaling will begin to bring instances online based on `DesiredCapacity`. CloudFormation will not mark the Auto Scaling group as successful (by setting its status to `CREATE_COMPLETE`) until the desired capacity is reached.

If `SpotPrice` is set, then `DesiredCapacity` will not be used as a criteria for success, since instances will only be started when the spot price has been matched. After the spot price has been matched, however, Auto Scaling uses `DesiredCapacity` as the target capacity for the group.

Required: No

Type: String

Update requires: No interruption

HealthCheckGracePeriod

The length of time in seconds after a new EC2 instance comes into service that Auto Scaling starts checking its health.

Required: No

Type: Integer

Update requires: No interruption

HealthCheckType

The service you want the health status from, Amazon EC2 or Elastic Load Balancer. Valid values are EC2 or ELB.

Required: No

Type: String

Update requires: No interruption

InstanceId

The ID of the Amazon EC2 instance you want to use to create the Auto Scaling group. Use this property if you want to create an Auto Scaling group that uses an existing Amazon EC2 instance instead of a launch configuration.

When you use an Amazon EC2 instance to create an Auto Scaling group, a new launch configuration is first created and then associated with the Auto Scaling group. The new launch configuration derives all its properties from the instance, with the exception of BlockDeviceMapping and AssociatePublicIpAddress.

Required: Conditional. You must specify this property if you don't specify the LaunchConfigurationName property.

Type: String

Update requires: Replacement

LaunchConfigurationName

Specifies the name of the associated AWS::AutoScaling::LaunchConfiguration.

Note: If this resource has a public IP address and is also in a VPC that is defined in the same template, you must use the DependsOn attribute to declare a dependency on the VPC-gateway attachment.

Required: Conditional; you must specify this property if you don't specify the InstanceId property.

Type: String

Update requires: No interruption.

When you update the LaunchConfigurationName, existing Amazon EC2 instances continue to run with the configuration that they were originally launched with. To update existing instances, specify an update policy attribute for this Auto Scaling group.

LoadBalancerNames

A list of Classic load balancers associated with this Auto Scaling group. To specify Application load balancers, use TargetGroupARNs.

Required: No

Type: List of strings

Update requires: No interruption

MaxSize

The maximum size of the Auto Scaling group.

Required: Yes

Type: String

Update requires: No interruption

MetricsCollection

Enables the monitoring of group metrics of an Auto Scaling group.

Required: No

Type: A list of Auto Scaling MetricsCollection

Update requires: No interruption

MinSize

The minimum size of the Auto Scaling group.

Required: Yes

Type: String

Update requires: No interruption

NotificationConfigurations

An embedded property that configures an Auto Scaling group to send notifications when specified events take place.

Required: No

Type: List of Auto Scaling NotificationConfigurations

Update requires: No interruption

PlacementGroup

The name of an existing cluster placement group into which you want to launch your instances. A placement group is a logical grouping of instances within a single Availability Zone. You cannot specify multiple Availability Zones and a placement group.

Required: No

Type: String

Update requires: No interruption

Tags

The tags you want to attach to this resource.

Required: No

Type: List of Auto Scaling Tags

Update requires: No interruption

TargetGroupARNs

A list of Amazon Resource Names (ARN) of target groups to associate with the Auto Scaling group.

Required: No

Type: List of strings

Update requires: No interruption

TerminationPolicies

A policy or a list of policies that are used to select the instances to terminate. The policies are executed in the order that you list them.

Required: No

Type: List of strings

Update requires: No interruption

VPCZoneIdentifier

A list of subnet identifiers of Amazon Virtual Private Cloud (Amazon VPCs).

If you specify the `AvailabilityZones` property, the subnets that you specify for this property must reside in those Availability Zones.

Required: Conditional. If you don't specify the `AvailabilityZones` property, you must specify this property.

Type: List of strings

Update requires: Some interruptions

Note: When you update `VPCZoneIdentifier`, the instances are replaced, but not the Auto Scaling group.

Return Value

When the logical ID of this resource is provided to the `Ref` intrinsic function, `Ref` returns the resource name, such as `mystack-myasgroup-NT5EUXTNTXXD`.

Examples

Auto Scaling Group with an Elastic Load Balancing Load Balancer, Launch Configuration, and Metric Collection

```
1 WebServerGroup:
2   Type: "AWS::AutoScaling::AutoScalingGroup"
3   Properties:
4     AvailabilityZones:
5       Fn::GetAZs: ""
6     LaunchConfigurationName:
7       Ref: "LaunchConfig"
8     MinSize: "2"
9     MaxSize: "2"
10    LoadBalancerNames:
11      - Ref: "ElasticLoadBalancer"
12    MetricsCollection:
13      -
14        Granularity: "1Minute"
15        Metrics:
16          - "GroupMinSize"
17          - "GroupMaxSize"
```

Auto Scaling Group Wait on Signals From New Instances

In the following example, the Auto Scaling group waits for new Amazon EC2 instances to signal the group before Auto Scaling proceeds to update the next batch of instances. In the `UpdatePolicy` attribute, the `WaitOnResourceSignals` flag is set to true. You can use the `cfn-signal` helper script on each instance to signal the Auto Scaling group.

```
1 ASG1:
2   UpdatePolicy:
3     AutoScalingRollingUpdate:
4       MinInstancesInService: "1"
5       MaxBatchSize: "1"
6       PauseTime: "PT12M5S"
7       WaitOnResourceSignals: "true"
8     Type: "AWS::AutoScaling::AutoScalingGroup"
9     Properties:
10    AvailabilityZones:
11      Fn::GetAZs:
12        Ref: "AWS::Region"
13    LaunchConfigurationName:
14      Ref: "ASLC"
15    MaxSize: "3"
16    MinSize: "1"
```

AWS::AutoScaling::ScalingPolicy

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-policy.html>

The AWS::AutoScaling::ScalingPolicy resource adds a scaling policy to an auto scaling group. A scaling policy specifies whether to scale the auto scaling group up or down, and by how much.

You can use a scaling policy together with a CloudWatch alarm. A CloudWatch alarm can automatically initiate actions on your behalf, based on parameters you specify. A scaling policy is one type of action that an alarm can initiate. Note that you can only associate one scaling policy with an alarm.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::AutoScaling::ScalingPolicy"
2 Properties:
3   AdjustmentType: <String>
4   AutoScalingGroupName: <String>
5   Cooldown: <String>
6   EstimatedInstanceWarmup: <Integer>
7   MetricAggregationType: <String>
8   MinAdjustmentMagnitude: <Integer>
9   PolicyType: <String>
10  ScalingAdjustment: <Integer>
11  StepAdjustments:
12    - <StepAdjustments>
```

Properties

AdjustmentType

Specifies whether the ScalingAdjustment is an absolute number or a percentage of the current capacity. Valid values are ChangeInCapacity, ExactCapacity, and PercentChangeInCapacity.

Required: Yes

Type: String

Update requires: No interruption

AutoScalingGroupName

The name or Amazon Resource Name (ARN) of the Auto Scaling Group that you want to attach the policy to.

Required: Yes

Type: String

Update requires: No interruption

Coldown

The amount of time, in seconds, after a scaling activity completes before any further trigger-related scaling activities can start.

Do not specify this property if you are using the StepScaling policy type.

Required: No

Type: String

Update requires: No interruption

EstimatedInstanceWarmup

The estimated time, in seconds, until a newly launched instance can send metrics to CloudWatch. By default, Auto Scaling uses the cooldown period, as specified in the Cooldown property.

Do not specify this property if you are using the SimpleScaling policy type.

Required: No

Type: Integer

Update requires: No interruption

MetricAggregationType

The aggregation type for the CloudWatch metrics. You can specify Minimum, Maximum, or Average. By default, AWS CloudFormation specifies Average.

Do not specify this property if you are using the SimpleScaling policy type.

Required: No

Type: String

Update requires: No interruption

MinAdjustmentMagnitude

For the PercentChangeInCapacity adjustment type, the minimum number of instances to scale. The scaling policy changes the desired capacity of the Auto Scaling group by a minimum of this many instances. This property replaces the MinAdjustmentStep property.

Required: No

Type: Integer

Update requires: No interruption

PolicyType

An Auto Scaling policy type. You can specify SimpleScaling or StepScaling. By default, AWS CloudFormation specifies SimpleScaling.

Required: No

Type: String

Update requires: No interruption

ScalingAdjustment

The number of instances by which to scale. The AdjustmentType property determines if AWS CloudFormation interprets this number as an absolute number (when the ExactCapacity value is specified), increase or decrease capacity by a specified number (when the ChangeInCapacity value is specified), or increase or decrease capacity as a percentage of the existing Auto Scaling group size (when the PercentChangeInCapacity value is specified). A positive value adds to the current capacity and a negative value subtracts from the current capacity. For exact capacity, you must specify a positive value.

Required: Conditional. This property is required if the policy type is SimpleScaling. This property is not supported with any other policy type.

Type: Integer

Update requires: No interruption

StepAdjustments

A set of adjustments that enable you to scale based on the size of the alarm breach.

Required: Conditional. This property is required if the policy type is StepScaling. This property is not supported with any other policy type.

Type: List of Auto Scaling ScalingPolicy StepAdjustments

Update requires: No interruption

Return Value

When you specify an AWS::AutoScaling::ScalingPolicy type as an argument to the Ref function, AWS CloudFormation returns the policy Amazon Resource Name (ARN), such as

```
arn:aws:autoscaling:us-east-1:123456789012:scalingPolicy: \
ab12c4d5-a1b2-a1b2-a1b2-ab12c4d56789:autoScalingGroupName/ \
myStack-AutoScalingGroup-AB12C4D5E6:policyName/ \
myStack-myScalingPolicy-AB12C4D5E6.
```

Examples

Simple policy type

The following example is a simple scaling policy that increases the number instances by one when it is triggered.

```
1 SimpleScaling:  
2   Type: "AWS::AutoScaling::ScalingPolicy"  
3   Properties:  
4     AdjustmentType: "ChangeInCapacity"  
5     PolicyType: "SimpleScaling"  
6     Cooldown: "60"  
7     AutoScalingGroupName:  
8       Ref: "ASG"  
9     ScalingAdjustment: 1
```

Step policy type

The following example is a step scaling policy that increases the number instances by one or two, depending on the size of the alarm breach. For a breach that is less than 50 units than the threshold value, the policy increases the number of instances by one. For a breach that is 50 units or more higher than the threshold, the policy increases the number of instances by two.

```
1 StepScaling:  
2   Type: "AWS::AutoScaling::ScalingPolicy"  
3   Properties:  
4     AdjustmentType: "ChangeInCapacity"  
5     AutoScalingGroupName:  
6       Ref: "ASG"  
7     PolicyType: "StepScaling"  
8     MetricAggregationType: "Average"  
9     EstimatedInstanceWarmup: "60"  
10    StepAdjustments:  
11      -  
12        MetricIntervalLowerBound: "0"  
13        MetricIntervalUpperBound: "50"  
14        ScalingAdjustment: "1"  
15      -  
16        MetricIntervalLowerBound: "50"  
17        ScalingAdjustment: "2"
```

AWS::CloudWatch::Alarm

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-cw-alarm.html>

The AWS::CloudWatch::Alarm type creates a CloudWatch alarm.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::CloudWatch::Alarm"
2 Properties:
3   ActionsEnabled: <Boolean>
4   AlarmActions:
5     - <String>
6   AlarmDescription: <String>
7   AlarmName: <String>
8   ComparisonOperator: <String>
9   Dimensions:
10    - <Metric dimension>
11   EvaluationPeriods: <Integer>
12   InsufficientDataActions:
13    - <String>
14   MetricName: <String>
15   Namespace: <String>
16   OKActions:
17    - <String>
18   Period: <Integer>
19   Statistic: <String>
20   Threshold: <Double>
21   Unit: <String>
```

Properties

ActionsEnabled

Indicates whether or not actions should be executed during any changes to the alarm's state.

Required: No

Type: Boolean

Update requires: No interruption

AlarmActions

The list of actions to execute when this alarm transitions into an ALARM state from any other state. Each action is specified as an Amazon Resource Number (ARN).

Note: For Auto Scaling scaling policies, you can specify only one policy. If you associate more than one policy, CloudWatch executes only the first scaling policy.

Required: No

Type: List of strings

Update requires: No interruption

AlarmDescription

The description for the alarm.

Required: No

Type: String

Update requires: No interruption

AlarmName

A name for the alarm. If you don't specify a name, AWS CloudFormation generates a unique physical ID and uses that ID for the alarm name.

Note: If you specify a name, you cannot perform updates that require replacement of this resource. You can perform updates that require no or some interruption. If you must replace the resource, specify a new name.

Required: No

Type: String

Update requires: Replacement

ComparisonOperator

The arithmetic operation to use when comparing the specified Statistic and Threshold. The specified Statistic value is used as the first operand.

You can specify the following values: `GreaterThanOrEqualToThreshold` | `GreaterThanThreshold` | `LessThanThreshold` | `LessThanOrEqualToThreshold`

Required: Yes

Type: String

Update requires: No interruption

Dimensions

The dimensions for the alarm's associated metric.

Required: No

Type: List of Metric Dimension

Update requires: No interruption

EvaluationPeriods

The number of periods over which data is compared to the specified threshold.

Required: Yes

Type: Integer

Update requires: No interruption

InsufficientDataActions

The list of actions to execute when this alarm transitions into an INSUFFICIENT_DATA state from any other state. Each action is specified as an Amazon Resource Number (ARN). Currently the only action supported is publishing to an Amazon SNS topic or an Amazon Auto Scaling policy.

Required: No

Type: List of strings

Update requires: No interruption

MetricName

The name for the alarm's associated metric.

Required: Yes

Type: String

Update requires: No interruption

Namespace

The namespace for the alarm's associated metric.

Required: Yes

Type: String

Update requires: No interruption

OKActions

The list of actions to execute when this alarm transitions into an OK state from any other state. Each action is specified as an Amazon Resource Number (ARN). Currently the only action supported is publishing to an Amazon SNS topic or an Amazon Auto Scaling policy.

Required: No

Type: List of strings

Update requires: No interruption

Period

The time over which the specified statistic is applied. You must specify a time in seconds that is also a multiple of 60.

Required: Yes

Type: Integer

Update requires: No interruption

Statistic

The statistic to apply to the alarm's associated metric.

You can specify the following values: SampleCount | Average | Sum | Minimum | Maximum

Required: Yes

Type: String

Update requires: No interruption

Threshold

The value against which the specified statistic is compared.

Required: Yes

Type: Double

Update requires: No interruption

Unit The unit for the alarm's associated metric.

You can specify the following values: Seconds | Microseconds | Milliseconds | Bytes | Kilobytes | Megabytes | Gigabytes | Terabytes | Bits | Kilobits | Megabits | Gibits | Terabits | Percent | Count | Bytes/Second | Kilobytes/Second | Megabytes/Second | Gigabytes/Second | Terabytes/Second | Bits/Second | Kilobits/Second | Megabits/Second | Gigabits/Second | Terabits/Second | Count/Second | None

Required: No

Type: String

Update requires: No interruption

Return Values

When you specify an AWS::CloudWatch::Alarm type as an argument to the Ref function, AWS CloudFormation returns the value of the AlarmName.

AWS::ElasticLoadBalancing::LoadBalancer

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-elb.html>

The AWS::ElasticLoadBalancing::LoadBalancer type creates a LoadBalancer.



If this resource has a public IP address and is also in a VPC that is defined in the same template, you must use the DependsOn attribute to declare a dependency on the VPC-gateway attachment.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::ElasticLoadBalancing::LoadBalancer"
2 Properties:
3   AccessLoggingPolicy:
4     <AccessLoggingPolicy>
5   AppCookieStickinessPolicy:
6     - <AppCookieStickinessPolicy>
7   AvailabilityZones:
8     - <String>
9   ConnectionDrainingPolicy:
10    <ConnectionDrainingPolicy>
11   ConnectionSettings:
12    <ConnectionSettings>
13   CrossZone: <Boolean>
14   HealthCheck:
15    <HealthCheck>
16   Instances:
17    - <String>
18   LBCookieStickinessPolicy:
19    - <LBCookieStickinessPolicy>
20   LoadBalancerName: <String>
21   Listeners:
22    - <Listener>
23   Policies:
24    - <ElasticLoadBalancing Policy>
25   Scheme: <String>
26   SecurityGroups:
27    <Security Group>
```

```
28     Subnets:  
29         - <String>  
30     Tags  
31         - <Resource Tag>
```

Properties

AccessLoggingPolicy

Captures detailed information for all requests made to your load balancer, such as the time a request was received, client's IP address, latencies, request path, and server responses.

Required: No

Type: Elastic Load Balancing AccessLoggingPolicy

Update requires: No interruption

AppCookieStickinessPolicy

Generates one or more stickiness policies with sticky session lifetimes that follow that of an application-generated cookie. These policies can be associated only with HTTP/HTTPS listeners.

Required: No

Type: A list of AppCookieStickinessPolicy objects.

Update requires: No interruption

AvailabilityZones

The Availability Zones in which to create the load balancer. You can specify the AvailabilityZones or Subnets property, but not both.

Note: For load balancers that are in a VPC, specify the Subnets property.

Required: No

Type: List of strings

Update requires: Replacement if you did not have an Availability Zone specified and you are adding one or if you are removing all Availability Zones. Otherwise, update requires no interruption.

ConnectionDrainingPolicy

Whether deregistered or unhealthy instances can complete all in-flight requests.

Required: No

Type: Elastic Load Balancing ConnectionDrainingPolicy

Update requires: No interruption

ConnectionSettings

Specifies how long front-end and back-end connections of your load balancer can remain idle.

Required: No

Type: Elastic Load Balancing ConnectionSettings

Update requires: No interruption

CrossZone

Whether cross-zone load balancing is enabled for the load balancer. With cross-zone load balancing, your load balancer nodes route traffic to the back-end instances across all Availability Zones. By default the CrossZone property is false.

Required: No

Type: Boolean

Update requires: No interruption

HealthCheck

Application health check for the instances.

Required: No

Type: ElasticLoadBalancing HealthCheck Type.

Update requires: Replacement if you did not have a health check specified and you are adding one or if you are removing a health check. Otherwise, update requires no interruption.

Instances

A list of EC2 instance IDs for the load balancer.

Required: No

Type: List of strings

Update requires: No interruption

LBCookieStickinessPolicy

Generates a stickiness policy with sticky session lifetimes controlled by the lifetime of the browser (user-agent), or by a specified expiration period. This policy can be associated only with HTTP/HTTPS listeners.

Required: No

Type: A list of LBCookieStickinessPolicy objects.

Update requires: No interruption

LoadBalancerName

A name for the load balancer. If you don't specify a name, AWS CloudFormation generates a unique physical ID and uses that ID for the load balancer. The name must be unique within your set of load balancers.

Note: If you specify a name, you cannot perform updates that require replacement of this resource. You can perform updates that require no or some interruption. If you must replace the resource, specify a new name.

Required: No

Type: String

Update requires: Replacement

Listeners

One or more listeners for this load balancer. Each listener must be registered for a specific port, and you cannot have more than one listener for a given port.

Note: If you update the property values for a listener specified by the `Listeners` property, AWS CloudFormation will delete the existing listener and create a new one with the updated properties. During the time that AWS CloudFormation is performing this action, clients will not be able to connect to the load balancer.

Required: Yes

Type: A list of ElasticLoadBalancing Listener Property Type objects.

Update requires: No interruption

Policies

A list of elastic load balancing policies to apply to this elastic load balancer. Specify only back-end server policies.

Required: No

Type: A list of ElasticLoadBalancing policy objects.

Update requires: No interruption

Scheme

For load balancers attached to an Amazon VPC, this parameter can be used to specify the type of load balancer to use. Specify `internal` to create an internal load balancer with a DNS name that resolves to private IP addresses or `internet-facing` to create a load balancer with a publicly resolvable DNS name, which resolves to public IP addresses.

Note: If you specify `internal`, you must specify subnets to associate with the load balancer, not Availability Zones.

Required: No

Type: String

Update requires: Replacement

SecurityGroups

A list of security groups assigned to your load balancer within your virtual private cloud (VPC).

Required: No

Type: A list of security groups

Update requires: No interruption

Subnets

A list of subnet IDs in your virtual private cloud (VPC) to attach to your load balancer. Do not specify multiple subnets that are in the same Availability Zone. You can specify the `AvailabilityZones` or `Subnets` property, but not both.

Required: No

Type: List of strings

Update requires: Replacement if you did not have an subnet specified and you are adding one or if you are removing all subnets. Otherwise, update requires no interruption. To update the load balancer to another subnet that is in the same Availability Zone, you must do two updates. You must first update the load balancer to use a subnet in different Availability Zone. After the update is complete, update the load balancer to use the new subnet that is in the original Availability Zone.

Tags

An arbitrary set of tags (key-value pairs) for this load balancer.

Required: No

Type: AWS CloudFormation Resource Tags

Update requires: No interruption

Return Values

Ref

When the logical ID of this resource is provided to the `Ref` intrinsic function, `Ref` returns the resource name. For example, `mystack-myelb-1WQN7BJGDB5YQ`.

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. The following are the available attributes and sample return values.

CanonicalHostedZoneName

The name of the Amazon Route 53 hosted zone that is associated with the load balancer.

Note: If you specify internal for the Elastic Load Balancing scheme, use `DNSName` instead. For an internal scheme, the load balancer doesn't have a `CanonicalHostedZoneName` value.

Example: `mystack-myelb-15HMABG9ZCN57-1013119603.us-east-1.elb.amazonaws.com`

CanonicalHostedZoneNameID

The ID of the Amazon Route 53 hosted zone name that is associated with the load balancer.

Example: Z3DZXE0Q79N41H

DNSName

The DNS name for the load balancer.

Example: mystack-myelb-15HMABG9ZCN57-1013119603.us-east-1.elb.amazonaws.com

SourceSecurityGroup.GroupName

The security group that you can use as part of your inbound rules for your load balancer's back-end Amazon EC2 application instances.

Example: amazon-elb

SourceSecurityGroup.OwnerAlias

The owner of the source security group.

Example: amazon-elb-sg

Examples

A load balancer with a health check and access logs

```
1 ElasticLoadBalancer:
2   Type: AWS::ElasticLoadBalancing::LoadBalancer
3   Properties:
4     AvailabilityZones:
5       Fn::GetAZs: ''
6     Instances:
7       - Ref: Ec2Instance1
8       - Ref: Ec2Instance2
9     Listeners:
10      - LoadBalancerPort: '80'
11        InstancePort:
12          Ref: WebServerPort
13        Protocol: HTTP
14      HealthCheck:
15        Target:
16          Fn::Join:
17            - ''
18            - - 'HTTP:'
19              - Ref: WebServerPort
20              - "/"
21      HealthyThreshold: '3'
```

```
22      UnhealthyThreshold: '5'
23      Interval: '30'
24      Timeout: '5'
25  AccessLoggingPolicy:
26      S3BucketName:
27          Ref: S3LoggingBucket
28      S3BucketPrefix: MyELBLogs
29      Enabled: 'true'
30      EmitInterval: '60'
31  DependsOn: S3LoggingBucketPolicy
```

A load balancer with access logging enabled

The following sample snippet creates an Amazon S3 bucket with a bucket policy that allows the load balancer to store information in the `Logs/AWSLogs/<AWS account number>/` folder. The load balancer also includes an explicit dependency on the bucket policy, which is required before the load balancer can write to the bucket.

```
1  S3LoggingBucket:
2      Type: AWS::S3::Bucket
3  S3LoggingBucketPolicy:
4      Type: AWS::S3::BucketPolicy
5      Properties:
6          Bucket:
7              Ref: S3LoggingBucket
8          PolicyDocument:
9              Version: '2012-10-17'
10         Statement:
11             - Sid: ELBAccessLogs20130930
12                 Effect: Allow
13                 Resource:
14                     Fn::Join:
15                         - ''
16                         - - 'arn:aws:s3:::'
17                             - Ref: S3LoggingBucket
18                             - "/"
19                             - Logs
20                             - "/AWSLogs/"
21                             - Ref: AWS::AccountId
22                             - "/*"
23         Principal:
24             Ref: ElasticLoadBalancingAccountID
```

```
25      Action:  
26          - s3:PutObject  
27 ElasticLoadBalancer:  
28     Type: AWS::ElasticLoadBalancing::LoadBalancer  
29     Properties:  
30         AvailabilityZones:  
31             Fn::GetAZs: ''  
32         Listeners:  
33             - LoadBalancerPort: '80'  
34                 InstancePort: '80'  
35                 Protocol: HTTP  
36         HealthCheck:  
37             Target: HTTP:80/  
38             HealthyThreshold: '3'  
39             UnhealthyThreshold: '5'  
40             Interval: '30'  
41             Timeout: '5'  
42         AccessLoggingPolicy:  
43             S3BucketName:  
44                 Ref: S3LoggingBucket  
45                 S3BucketPrefix: Logs  
46                 Enabled: 'true'  
47                 EmitInterval: '60'  
48     DependsOn: S3LoggingBucketPolicy
```

AWS::RDS::DBSubnetGroup

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-rds-dbsubnet-group.html>

The AWS::RDS::DBSubnetGroup type creates an RDS database subnet group. Subnet groups must contain at least two subnets in two different Availability Zones in the same region.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::RDS::DBSubnetGroup"
2 Properties:
3   DBSubnetGroupDescription: <String>
4   DBSubnetGroupName: <String>
5   SubnetIds:
6     - <String>
7   Tags:
8     - <Resource Tag>
```

Properties

DBSubnetGroupDescription

The description for the DB Subnet Group.

Required: Yes

Type: String

Update requires: No interruption

DBSubnetGroupName

The name for the DB Subnet Group. This value is stored as a lowercase string.

Constraints: Must contain no more than 255 letters, numbers, periods, underscores, spaces, or hyphens. Must not be default.

Required: No

Type: String

Update requires: Replacement

SubnetIds

The EC2 Subnet IDs for the DB Subnet Group.

Required: Yes

Type: List of String values

Update requires: No interruption

Tags The tags that you want to attach to the RDS database subnet group.

Required: No

Type: A list of resource tags in key-value format.

Update requires: No interruption

Return Value

Ref

When you pass the logical ID of an AWS::RDS::DBSubnetGroup resource to the intrinsic Ref function, the function returns the name of the DB subnet group, such as mystack-mydbsubnetgroup-0a12bc456789de0fg.

Example

```
1 AWSTemplateFormatVersion: "2010-09-09"
2 Resources:
3   myDBSubnetGroup:
4     Type: "AWS::RDS::DBSubnetGroup"
5     Properties:
6       DBSubnetGroupDescription: "description"
7       SubnetIds:
8         - "subnet-7b5b4112"
9         - "subnet-7b5b4115"
10      Tags:
11        -
12          Key: "String"
13          Value: "String"
```

AWS::RDS::DBInstance

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-rds-database-instance.html>

The AWS::RDS::DBInstance type creates an Amazon Relational Database Service (Amazon RDS) DB instance.



If a DB instance is deleted or replaced during an update, AWS CloudFormation deletes all automated snapshots. However, it retains manual DB snapshots. During an update that requires replacement, you can apply a stack policy to prevent DB instances from being replaced.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::RDS::DBInstance"
2 Properties:
3   AllocatedStorage: <String>
4   AllowMajorVersionUpgrade: <Boolean>
5   AutoMinorVersionUpgrade: <Boolean>
6   AvailabilityZone: <String>
7   BackupRetentionPeriod: <String>
8   CharacterSetName: <String>
9   CopyTagsToSnapshot: <Boolean>
10  DBClusterIdentifier: <String>
11  DBInstanceClass: <String>
12  DBInstanceIdentifier: <String>
13  DBName: <String>
14  DBParameterGroupName: <String>
15  DBSecurityGroups:
16    - <String>
17  DBSnapshotIdentifier: <String>
18  DBSubnetGroupName: <String>
19  Domain: <String>
20  DomainIAMRoleName: <String>
21  Engine: <String>
22  EngineVersion: <String>
23  Iops: <Number>
24  KmsKeyId: <String>
```

```
25 LicenseModel: <String>
26 MasterUsername: <String>
27 MasterUserPassword: <String>
28 MonitoringInterval: <Integer>
29 MonitoringRoleArn: <String>
30 MultiAZ: <Boolean>
31 OptionGroupName: <String>
32 Port: <String>
33 PreferredBackupWindow: <String>
34 PreferredMaintenanceWindow: <String>
35 PubliclyAccessible: <Boolean>
36 SourceDBIdentifier: <String>
37 SourceRegion: <String>
38 StorageEncrypted: <Boolean>
39 StorageType: <String>
40 Tags:
41     <Resource Tag>
42 Timezone: <String>
43 VPCSecurityGroups:
44     - <String>
```

Properties

AllocatedStorage

The allocated storage size, specified in gigabytes (GB).

If any value is set in the `Iops` parameter, `AllocatedStorage` must be at least 100 GB, which corresponds to the minimum `Iops` value of 1,000. If you increase the `Iops` value (in 1,000 IOPS increments), then you must also increase the `AllocatedStorage` value (in 100-GB increments).

Required: Conditional. This property is required except when you specify the `DBClusterIdentifier` property or when you create a read replica from AWS CloudFormation by using the `AWS::RDS::DBInstance` resource. In these cases, don't specify this property.

Type: String

Update requires: No interruption

AllowMajorVersionUpgrade

If you update the `EngineVersion` property to a version that's different from the DB instance's current major version, set this property to true.

Required: No

Type: Boolean

Update requires: No interruption

AutoMinorVersionUpgrade

Indicates that minor engine upgrades are applied automatically to the DB instance during the maintenance window. The default value is true.

Required: No

Type: Boolean

Update requires: No interruption or some interruptions.

AvailabilityZone

The name of the Availability Zone where the DB instance is located. You can't set the AvailabilityZone parameter if the MultiAZ parameter is set to true.

Required: No

Type: String

Update requires: Replacement

BackupRetentionPeriod

The number of days during which automatic DB snapshots are retained.

If this DB instance is deleted or replaced during an update, AWS CloudFormation deletes all automated snapshots. However, it retains manual DB snapshots.

Required: No

Type: String

Update requires: No interruption or some interruptions.

CharacterSetName

For supported engines, specifies the character set to associate with the DB instance.

If you specify the DBSnapshotIdentifier or SourceDBInstanceIdentifier property, don't specify this property. The value is inherited from the snapshot or source DB instance.

Required: No

Type: String

Update requires: Replacement

CopyTagsToSnapshot

Indicates whether to copy all of the user-defined tags from the DB instance to snapshots of the DB instance. By default, Amazon RDS doesn't copy tags to snapshots. Amazon RDS doesn't copy tags with the aws:: prefix unless it's the DB instance's final snapshot (the snapshot when you delete the DB instance).

Required: No

Type: Boolean

Update requires: No interruption

DBClusterIdentifier

The name of an existing DB cluster that this instance is associated with. If you specify this property, specify `aurora` for the `Engine` property and don't specify any of the following properties: `AllotedStorage`, `BackupRetentionPeriod`, `CharacterSetName`, `DBName`, `DBSecurityGroups`, `MasterUsername`, `MasterUserPassword`, `OptionGroupName`, `PreferredBackupWindow`, `PreferredMaintenanceWindow`, `Port`, `SourceDBInstanceIdentifier`, `StorageType`, or `VPCSecurityGroups`.

Amazon RDS assigns the first DB instance in the cluster as the primary, and additional DB instances as replicas.

If you specify this property, the default deletion policy is `Delete`. Otherwise, the default deletion policy is `Snapshot`.

Required: No

Type: String

Update requires: Replacement

DBInstanceClass

The name of the compute and memory capacity classes of the DB instance.

Required: Yes

Type: String

Update requires: Some interruptions

DBInstanceIdentifier

A name for the DB instance. If you specify a name, AWS CloudFormation converts it to lowercase. If you don't specify a name, AWS CloudFormation generates a unique physical ID and uses that ID for the DB instance.

If you specify a name, you cannot perform updates that require replacement of this resource. You can perform updates that require no or some interruption. If you must replace the resource, specify a new name.

Required: No

Type: String

Update requires: Replacement

DBName

The name of the DB instance that was provided at the time of creation, if one was specified. This same name is returned for the life of the DB instance.

If you specify the `DBSnapshotIdentifier` property, AWS CloudFormation ignores this property.

If you restore DB instances from snapshots, this property doesn't apply to the MySQL, PostgreSQL, or MariaDB engines.

Required: No

Type: String

Update requires: Replacement

DBParameterGroupName

The name of an existing DB parameter group or a reference to an AWS::RDS::DBParameterGroup resource created in the template.

Required: No

Type: String

Update requires: No interruption or some interruptions. If any of the data members of the referenced parameter group are changed during an update, the DB instance might need to be restarted, which causes some interruption. If the parameter group contains static parameters, whether they were changed or not, an update triggers a reboot.

DBSecurityGroups

A list of the DB security groups to assign to the DB instance. The list can include both the name of existing DB security groups or references to AWS::RDS::DBSecurityGroup resources created in the template.

If you set DBSecurityGroups, you must not set VPCSecurityGroups, and vice versa. Also, note that the EC2VpcId property exists only for backwards compatibility with older regions and is no longer recommended for providing security information to an RDS DB instance. Instead, use VPCSecurityGroups.

If you specify this property, AWS CloudFormation sends only the following properties (if specified) to Amazon RDS: AllocatedStorage, AutoMinorVersionUpgrade, AvailabilityZone, BackupRetentionPeriod, CharacterSetName, DBInstanceClass, DBName, DBParameterGroupName, DBSecurityGroups, DBSubnetGroupName, Engine, EngineVersion, Iops, LicenseModel, MasterUsername, MasterUserPassword, MultiAZ, OptionGroupName, PreferredBackupWindow, PreferredMaintenanceWindow

All other properties are ignored. Specify a virtual private cloud (VPC) security group if you want to submit other properties, such as StorageType, StorageEncrypted, or KmsKeyId. If you're already using the DBSecurityGroups property, you can't use these other properties by updating your DB instance to use a VPC security group. You must recreate the DB instance.

Required: No

Type: List of String values

Update requires: No interruption

DBSnapshotIdentifier

The name or Amazon Resource Name (ARN) of the DB snapshot that's used to restore the DB instance. If you're restoring from a shared manual DB snapshot, you must specify the ARN of the snapshot.

By specifying this property, you can create a DB instance from the specified DB snapshot. If the `DBSnapshotIdentifier` property is an empty string or the `AWS::RDS::DBInstance` declaration has no `DBSnapshotIdentifier` property, AWS CloudFormation creates a new database. If the property contains a value (other than an empty string), AWS CloudFormation creates a database from the specified snapshot. If a snapshot with the specified name doesn't exist, AWS CloudFormation can't create the database and it rolls back the stack.

Some DB instance properties aren't valid when you restore from a snapshot, such as the `MasterUsername` and `MasterUserPassword` properties.

If you specify this property, AWS CloudFormation ignores the `DBName` property.

Required: No

Type: String

Update requires: Replacement

DBSubnetGroupName

A DB subnet group to associate with the DB instance. If you update this value, the new subnet group must be a subnet group in a new VPC.

If there's no DB subnet group, then the instance isn't a VPC DB instance.

Required: No

Type: String

Update requires: Replacement

Domain

For an Amazon RDS DB instance that's running Microsoft SQL Server, the Active Directory directory ID to create the instance in. Amazon RDS uses Windows Authentication to authenticate users that connect to the DB instance.

If you specify this property, you must specify a SQL Server engine for the `Engine` property.

Required: No

Type: String

Update requires: No interruption

DomainIAMRoleName

The name of an IAM role that Amazon RDS uses when calling the AWS Directory Service APIs.

Required: No

Type: String

Update requires: No interruption

Engine

The database engine that the DB instance uses. This property is optional when you specify the `DBSnapshotIdentifier` property to create DB instances.

If you specify `aurora` as the database engine, you must also specify the `DBClusterIdentifier` property.

If you've specified `oracle-se` or `oracle-se1` as the database engine, you can update the database engine to `oracle-se2` without the database instance being replaced.

Required: Conditional

Type: String

Update requires: Replacement

EngineVersion

The version number of the database engine that the DB instance uses.

Required: No

Type: String

Update requires: Some interruptions

Iops The number of I/O operations per second (IOPS) that the database provisions. The value must be equal to or greater than 1000.

If you specify this property, you must follow the range of allowed ratios of your requested IOPS rate to the amount of storage that you allocate (IOPS to allocated storage). For example, you can provision an Oracle database instance with 1000 IOPS and 200 GB of storage (a ratio of 5:1), or specify 2000 IOPS with 200 GB of storage (a ratio of 10:1).

Required: Conditional. If you specify `io1` for the `StorageType` property, you must specify this property.

Type: Number

Update requires: No interruption

KmsKeyId

The ARN of the AWS Key Management Service (AWS KMS) master key that's used to encrypt the DB instance, such as `arn:aws:kms:us-east-1:012345678910:key/abcd1234-a123-456a-a12b-a123b4cd56ef`. If you enable the `StorageEncrypted` property but don't specify this property, AWS CloudFormation uses the default master key. If you specify this property, you must set the `StorageEncrypted` property to `true`.

If you specify the `SourceDBIdentifier` property, the value is inherited from the source DB instance if the read replica is created in the same region. If you specify this property when you create a read replica from an unencrypted DB instance, the read replica is encrypted.

If you create an encrypted read replica in a different AWS Region, then you must specify a KMS key for the destination AWS Region. KMS encryption keys are specific to the region that they're created in, and you can't use encryption keys from one region in another region.

If you specify `DBSecurityGroups`, AWS CloudFormation ignores this property. To specify both a security group and this property, you must use a VPC security group.

Required: No

Type: String

Update requires: Replacement

LicenseModel

The license model of the DB instance.

If `DBSecurityGroups` is specified, updating the license model requires its replacement.

Required: No

Type: String

Update requires: No interruption

MasterUsername

The master user name for the DB instance.

If you specify the `SourceDBInstanceIdentifier` or `DBSnapshotIdentifier` property, don't specify this property. The value is inherited from the source DB instance or snapshot.

Required: Conditional

Type: String

Update requires: Replacement

MasterUserPassword

The master password for the DB instance.

If you specify the `SourceDBInstanceIdentifier` or `DBSnapshotIdentifier` property, don't specify this property. The value is inherited from the source DB instance or snapshot.

Required: Conditional

Type: String

Update requires: No interruption

MonitoringInterval

The interval, in seconds, between points when Amazon RDS collects enhanced monitoring metrics for the DB instance. To disable metrics collection, specify `0`.

Required: Conditional. If you specify the `MonitoringRoleArn` property, specify a value other than `0` for `MonitoringInterval`.

Type: Integer

Update requires: No interruption or some interruptions.

MonitoringRoleArn

The ARN of the AWS Identity and Access Management (IAM) role that permits Amazon RDS to send enhanced monitoring metrics to Amazon CloudWatch, for example, arn:aws:iam::123456789012:role/emaccess.

Required: Conditional. If you specify a value other than `0` for the `MonitoringInterval` property, specify a value for `MonitoringRoleArn`.

Type: String

Update requires: No interruption

MultiAZ

Specifies if the database instance is a multiple Availability Zone deployment. You can't set the `AvailabilityZone` parameter if the `MultiAZ` parameter is set to `true`. Amazon Aurora storage is replicated across all the Availability Zones and doesn't require the `MultiAZ` option to be set.

Required: No

Type: Boolean

Update requires: No interruption

OptionGroupName

The option group that this DB instance is associated with.

Required: No

Type: String

Update requires: No interruption

Port The port for the instance.

Required: No

Type: String

Update requires: Replacement

PreferredBackupWindow

The daily time range during which automated backups are performed if automated backups are enabled, as determined by the `BackupRetentionPeriod` property.

Required: No

Type: String

Update requires: No interruption

PreferredMaintenanceWindow

The weekly time range (in UTC) during which system maintenance can occur.

This property applies when AWS CloudFormation initially creates the DB instance. If you use AWS CloudFormation to update the DB instance, those updates are applied immediately.

Required: No

Type: String

Update requires: No interruption or some interruptions.

PubliclyAccessible

Indicates whether the DB instance is an internet-facing instance. If you specify true, AWS CloudFormation creates an instance with a publicly resolvable DNS name, which resolves to a public IP address. If you specify false, AWS CloudFormation creates an internal instance with a DNS name that resolves to a private IP address.

The default behaviour value depends on your VPC setup and the database subnet group.

If this resource has a public IP address and is also in a VPC that is defined in the same template, you must use the DependsOn attribute to declare a dependency on the VPC-gateway attachment.

If you specify DBSecurityGroups, AWS CloudFormation ignores this property. To specify a security group and this property, you must use a VPC security group.

Required: No

Type: Boolean

Update requires: Replacement

SourceDBInstanceIdentifier

If you want to create a read replica DB instance, specify the ID of the source DB instance. Each DB instance can have a limited number of read replicas.

The SourceDBInstanceIdentifier property determines whether a DB instance is a read replica. If you remove the SourceDBInstanceIdentifier property from your template and then update your stack, AWS CloudFormation deletes the read replica and creates a new DB instance (not a read replica).

If you specify a source DB instance that uses VPC security groups, we recommend that you specify the VPCSecurityGroups property. If you don't specify the property, the read replica inherits the value of the VPCSecurityGroups property from the source DB when you create the replica. However, if you update the stack, AWS CloudFormation reverts the replica's VPCSecurityGroups property to the default value because it's not defined in the stack's template. This change might cause unexpected issues.

Read replicas don't support deletion policies. AWS CloudFormation ignores any deletion policy that's associated with a read replica.

If you specify SourceDBInstanceIdentifier, don't set the MultiAZ property to true, and don't specify the DBSnapshotIdentifier property. You can't deploy read replicas in multiple Availability Zones, and you can't create a read replica from a snapshot.

Don't set the `BackupRetentionPeriod`, `DBName`, `MasterUsername`, `MasterUserPassword`, and `PreferredBackupWindow` properties. The database attributes are inherited from the source DB instance, and backups are disabled for read replicas.

If the source DB instance is in a different region than the read replica, specify an ARN for a valid DB instance.

For DB instances in Amazon Aurora clusters, don't specify this property. Amazon RDS automatically assigns writer and reader DB instances.

Required: No

Type: String * *Update requires**: Replacement

SourceRegion

The ID of the region that contains the source DB instance for the read replica.

Required: No

Type: String

Update requires: Replacement

StorageEncrypted

Indicates whether the DB instance is encrypted.

If you specify the `DBClusterIdentifier`, `DBSnapshotIdentifier`, or `SourceDBInstanceIdentifier` property, don't specify this property. The value is inherited from the cluster, snapshot, or source DB instance.

Required: Conditional. If you specify the `KmsKeyId` property, you must enable encryption.

Type: Boolean

Update requires: Replacement

StorageType

The storage type associated with this DB instance.

Required: No

Type: String

Update requires: Some interruptions

Tags An arbitrary set of tags (key–value pairs) for this DB instance.

Required: No

Type: AWS CloudFormation Resource Tags

Update requires: No interruption

Timezone

The time zone of the DB instance, which you can specify to match the time zone of your applications.

Required: No

Type: String

Update requires: Replacement

VPCSecurityGroups

A list of the VPC security group IDs to assign to the DB instance. The list can include both the physical IDs of existing VPC security groups and references to AWS::EC2::SecurityGroup resources created in the template.

If you set VPCSecurityGroups, you must not set DBSecurityGroups, and vice versa.

You can migrate a DB instance in your stack from an RDS DB security group to a VPC security group, but you can't revert to using an RDS security group after you establish a VPC security group membership.

When you migrate your DB instance to VPC security groups, if your stack update rolls back because the DB instance update fails or because an update fails in another AWS CloudFormation resource, the rollback fails because it can't revert to an RDS security group.

To use the properties that are available when you use a VPC security group, you must recreate the DB instance. If you don't, AWS CloudFormation submits only the property values that are listed in the DBSecurityGroups property.

To avoid this situation, migrate your DB instance to using VPC security groups only when that is the only change in your stack template.

Required: No

Type: List of String values

Update requires: No interruption

Updating and Deleting AWS::RDS::DBInstance Resources

Updating DB Instances

When properties labeled “*Update requires:* Replacement” are updated, AWS CloudFormation first creates a replacement DB instance, then changes references from other dependent resources to point to the replacement DB instance, and finally deletes the old DB instance.

It's recommended that you take a snapshot of the database before updating the stack. If you don't, you lose the data when AWS CloudFormation replaces your DB instance. To preserve your data, perform the following procedure:

- Deactivate any applications that are using the DB instance so that there's no activity on the DB instance.
- Create a snapshot of the DB instance. For more information about creating DB snapshots, see [Creating a DB snapshot](#).
- If you want to restore your instance using a DB snapshot, modify the updated template with your DB instance changes and add the DBSnapshotIdentifier property with the ID of the DB snapshot that you want to use.
- Update the stack.

Deleting DB Instances

You can set a deletion policy for your DB instance to control how AWS CloudFormation handles the instance when the stack is deleted. For Amazon RDS DB instances, you can choose to *retain* the instance, to *delete* the instance, or to *create a snapshot* of the instance. The default AWS CloudFormation behaviour depends on the `DBClusterIdentifier` property:

- For `AWS::RDS::DBInstance` resources that don't specify the `DBClusterIdentifier` property, AWS CloudFormation saves a snapshot of the DB instance.
- For `AWS::RDS::DBInstance` resources that do specify the `DBClusterIdentifier` property, AWS CloudFormation deletes the DB instance.

Return Values

Ref

When you provide the RDS DB instance's logical name to the `Ref` intrinsic function, `Ref` returns the `DBInstanceIdentifier`. For example: `mystack-mydb-ea5ugmfvuaxg`.

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. The following are the available attributes and sample return values.

- `Endpoint.Address`
The connection endpoint for the database. For example: `mystack-mydb-1apw1j4phy1rk.cg034hpkmmt.us-east-2.rds.amazonaws.com`.
- `Endpoint.Port` The port number on which the database accepts connections. For example: 3306.

Examples

DBInstance with a set MySQL version, Tags and DeletionPolicy

This example shows how to set the MySQL version that has a `DeletionPolicy` Attribute set. With the `DeletionPolicy` set to `Snapshot`, AWS CloudFormation takes a snapshot of this DB instance before deleting it during stack deletion. A tag that contains a friendly name for the database is also set.

```
1 MyDB:
2   Type: "AWS::RDS::DBInstance"
3   Properties:
4     DBName:
5       Ref: "DBName"
6     AllocatedStorage:
7       Ref: "DBAllocatedStorage"
8     DBInstanceClass:
9       Ref: "DBInstanceClass"
10    Engine: "MySQL"
11    EngineVersion: "5.5"
12    MasterUsername:
13      Ref: "DBUser"
14    MasterUserPassword:
15      Ref: "DBPassword"
16    Tags:
17      -
18        Key: "Name"
19        Value: "My SQL Database"
20    DeletionPolicy: "Snapshot"
```

DBInstance with Provisioned IOPS

This example sets a provisioned IOPS value in the `Iops` property. Note that the `AllocatedStorage` property is set according to the 10:1 ratio between IOPS and GiBs of storage.

```
1 MyDB:  
2   Type: "AWS::RDS::DBInstance"  
3   Properties:  
4     AllocatedStorage: "100"  
5     DBInstanceClass: "db.m1.small"  
6     Engine: "MySQL"  
7     EngineVersion: "5.5"  
8     Iops: "1000"  
9     MasterUsername:  
10    Ref: "DBUser"  
11    MasterUserPassword:  
12    Ref: "DBPassword"
```

Cross-Region Encrypted Read Replica

The following example creates an encrypted read replica from a cross-region source DB instance.

```
1 AWSTemplateFormatVersion: 2010-09-09  
2 Description: RDS Storage Encrypted  
3 Parameters:  
4   SourceDBInstanceIdentifier:  
5     Type: String  
6   DBInstanceType:  
7     Type: String  
8   SourceRegion:  
9     Type: String  
10  Resources:  
11    MyKey:  
12      Type: 'AWS::KMS::Key'  
13      Properties:  
14        KeyPolicy:  
15          Version: 2012-10-17  
16          Id: key-default-1  
17          Statement:  
18            - Sid: Enable IAM User Permissions  
19            Effect: Allow  
20            Principal:  
21              AWS: !Join  
22                - ''  
23                - - 'arn:aws:iam::'  
24                - !Ref 'AWS::AccountId'  
25                - ':root'
```

```
26      Action: 'kms:*'
27      Resource: '*'
28 MyDBSmall:
29     Type: 'AWS::RDS::DBInstance'
30     Properties:
31       DBInstanceClass: !Ref DBInstanceType
32       SourceDBInstanceIdentifier: !Ref SourceDBInstanceIdentifier
33       SourceRegion: !Ref SourceRegion
34       KmsKeyId: !Ref MyKey
35 Outputs:
36   InstanceId:
37     Description: InstanceId of the newly created RDS Instance
38     Value: !Ref MyDBSmall
```

AWS::EC2::Instance

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html>

The AWS::EC2::Instance resource creates an EC2 instance.

If an Elastic IP address is attached to your instance, AWS CloudFormation reattaches the Elastic IP address after it updates the instance.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

```
1 Type: "AWS::EC2::Instance"
2 Properties:
3   Affinity: <String>
4   AvailabilityZone: <String>
5   BlockDeviceMappings:
6     - <EC2 Block Device Mapping>
7   CreditSpecification: <CreditSpecification>
8   DisableApiTermination: <Boolean>
9   EbsOptimized: <Boolean>
10  ElasticGpuSpecifications: [ <ElasticGpuSpecification>, ... ]
11  HostId: <String>
12  IamInstanceProfile: <String>
13  ImageId: <String>
14  InstanceInitiatedShutdownBehavior: <String>
15  InstanceType: <String>
16  Ipv6AddressCount: <Integer>
17  Ipv6Addresses:
18    - <IPv6 Address Type>
19  KernelId: <String>
20  KeyName: <String>
21  Monitoring: <Boolean>
22  NetworkInterfaces:
23    - <EC2 Network Interface>
24  PlacementGroupName: <String>
25  PrivateIpAddress: <String>
26  RamdiskId: <String>
27  SecurityGroupIds:
28    - <String>
29  SecurityGroups:
```

```
30      - <String>
31  SourceDestCheck: <Boolean>
32  SsmAssociations:
33      - <SSMAssociation>
34  SubnetId: <String>
35  Tags:
36      - <Resource Tag>
37  Tenancy: <String>
38  UserData: <String>
39  Volumes:
40      - <EC2 MountPoint>
41  AdditionalInfo: <String>
```

Properties

Affinity

Indicates whether Amazon Elastic Compute Cloud (Amazon EC2) always associates the instance with a dedicated host. If you want Amazon EC2 to always restart the instance (if it was stopped) onto the same host on which it was launched, specify `host`. If you want Amazon EC2 to restart the instance on any available host, but to try to launch the instance onto the last host it ran on (on a best-effort basis), specify `default`.

Required: No

Type: String

Update requires: No interruption

AvailabilityZone

Specifies the name of the Availability Zone in which the instance is located.

Required: No. If not specified, an Availability Zone will be automatically chosen for you based on the load balancing criteria for the region.

Type: String

Update requires: Replacement

BlockDeviceMappings

Defines a set of Amazon Elastic Block Store block device mappings, ephemeral instance store block device mappings, or both.

Required: No

Type: A list of Amazon EC2 Block Device Mapping Property.

Update requires: Replacement. If you change only the `DeleteOnTermination` property for one or more block devices, update requires No interruption.

CreditSpecification

Specifies the credit option for CPU usage of a T2 instance.

Required: No

Type: Amazon EC2 Instance CreditSpecification.

Update requires: No interruption

DisableApiTermination

Specifies whether the instance can be terminated through the API.

Required: No

Type: Boolean

Update requires: No interruption

EbsOptimized

Specifies whether the instance is optimized for Amazon Elastic Block Store I/O. This optimization provides dedicated throughput to Amazon EBS and an optimized configuration stack to provide optimal EBS I/O performance.

Required: No. By default, AWS CloudFormation specifies `false`.

Type: Boolean

Update requires: Some interruptions for Amazon EBS-backed instances

Update requires: Replacement for instance store-backed instances

ElasticGpuSpecifications

Specifies the Elastic GPUs. An Elastic GPU is a GPU resource that you can attach to your instance to accelerate the graphics performance of your applications.

Required: No

Type: List of Amazon EC2 Instance ElasticGpuSpecification

Update requires: Replacement

HostId

If you specify `host` for the `Affinity` property, the ID of a dedicated host that the instance is associated with. If you don't specify an ID, Amazon EC2 launches the instance onto any available, compatible dedicated host in your account. This type of launch is called an untargeted launch. Note that for untargeted launches, you must have a compatible, dedicated host available to successfully launch instances.

Required: No

Type: String

Update requires: No interruption

IamInstanceProfile

The name of an instance profile or a reference to an AWS::IAM::InstanceProfile resource.

Required: No

Type: String

Update requires: No interruption

ImageId

Provides the unique ID of the Amazon Machine Image (AMI) that was assigned during registration.

Required: Yes

Type: String

Update requires: Replacement

InstanceInitiatedShutdownBehavior

Indicates whether an instance stops or terminates when you shut down the instance from the instance's operating system shutdown command. You can specify stop or terminate.

Required: No

Type: String

Update requires: No interruption

InstanceType

The instance type, such as t2.micro. The default type is m3.medium.

Required: No

Type: String

Update requires: Some interruptions for Amazon EBS-backed instances

Update requires: Replacement for instance store-backed instances

Ipv6AddressCount

The number of IPv6 addresses to associate with the instance's primary network interface. Amazon EC2 automatically selects the IPv6 addresses from the subnet range. To specify specific IPv6 addresses, use the Ipv6Addresses property and don't specify this property.

Required: No

Type: Integer

Update requires: Replacement

Ipv6Addresses

One or more specific IPv6 addresses from the IPv6 CIDR block range of your subnet to associate with the instance's primary network interface. To specify a number of IPv6 addresses, use the Ipv6AddressCount property and don't specify this property.

Required: No

Type: List of EC2 NetworkInterface Ipv6Addresses

Update requires: Replacement

KernelId

The kernel ID.

Required: No

Type: String

Update requires: Some interruptions for Amazon EBS-backed instances

Update requires: Replacement for instance store-backed instances

KeyName

Provides the name of the Amazon EC2 key pair.

Required: No

Type: String

Update requires: Replacement

Monitoring

Specifies whether detailed monitoring is enabled for the instance.

Required: No

Type: Boolean

Update requires: No interruption

NetworkInterfaces

A list of embedded objects that describes the network interfaces to associate with this instance.

If you use this property to point to a network interface, you must terminate the original interface before attaching a new one to allow the update of the instance to succeed.

If this resource has a public IP address and is also in a VPC that is defined in the same template, you must use the DependsOn attribute to declare a dependency on the VPC-gateway attachment.

Required: No

Type: A list of EC2 NetworkInterface Embedded Property Type

Update requires: Replacement

PlacementGroupName

The name of an existing placement group that you want to launch the instance into (for cluster instances).

Required: No

Type: String

Update requires: Replacement

PrivateIpAddress

The private IP address for this instance.

If you make an update to an instance that requires replacement, you must assign a new private IP address. During a replacement, AWS CloudFormation creates a new instance but doesn't delete the old instance until the stack has successfully updated. If the stack update fails, AWS CloudFormation uses the old instance in order to roll back the stack to the previous working state. The old and new instances cannot have the same private IP address.

If you're using Amazon VPC, you can use this parameter to assign the instance a specific available IP address from the subnet (for example, 10.0.0.25). By default, Amazon VPC selects an IP address from the subnet for the instance.

Required: No

Type: String

Update requires: Replacement

RamdiskId

The ID of the RAM disk to select. Some kernels require additional drivers at launch. Check the kernel requirements for information about whether you need to specify a RAM disk.

Required: No

Type: String

Update requires: Some interruptions for Amazon EBS-backed instances

Update requires: Replacement for instance store-backed instances

SecurityGroupIds

A list that contains the security group IDs for VPC security groups to assign to the Amazon EC2 instance. If you specified the `NetworkInterfaces` property, do not specify this property.

Required: Conditional. Required for VPC security groups.

Type: List of String values

Update requires: No interruption for instances that are in a VPC.

Update requires: Replacement for instances that are not in a VPC.

SecurityGroups

Valid only for Amazon EC2 security groups. A list that contains the Amazon EC2 security groups to assign to the Amazon EC2 instance. The list can contain both the name of existing Amazon EC2 security groups or references to `AWS::EC2::SecurityGroup` resources created in the template.

Required: No

Type: List of String values

Update requires: Replacement.

SourceDestCheck

Controls whether source/destination checking is enabled on the instance. Also determines if an instance in a VPC will perform network address translation (NAT).

A value of “true” means that source/destination checking is enabled, and a value of “false” means that checking is disabled. For the instance to perform NAT, the value must be “false”.

Required: No

Type: Boolean

Update requires: No interruption

SsmAssociations

The Amazon EC2 Systems Manager (SSM) document and parameter values to associate with this instance. To use this property, you must specify an IAM role for the instance.

You can currently associate only one document with an instance.

Required: No

Type: List of Amazon EC2 Instance SsmAssociations.

Update requires: No interruption

SubnetId

If you’re using Amazon VPC, this property specifies the ID of the subnet that you want to launch the instance into. If you specified the NetworkInterfaces property, do not specify this property.

Required: No

Type: String

Update requires: Replacement

Tags An arbitrary set of tags (key–value pairs) for this instance.

Required: No

Type: AWS CloudFormation Resource Tags

Update requires: No interruption.

Tenancy

The tenancy of the instance that you want to launch, such as default, dedicated, or host. If you specify a tenancy value of dedicated or host, you must launch the instance in a VPC.

Required: No

Type: String

Update requires: No interruption if this property was set to dedicated and you change it to host or vice versa.

Update requires: Replacement for all other changes.

UserData

Base64-encoded MIME user data that is made available to the instances.

Required: No

Type: String

Update requires: Some interruptions for Amazon EBS-backed instances.

For EBS-backed instances, changing the `UserData` stops and then starts the instance; however, Amazon EC2 doesn't automatically run the updated `UserData`. To update configurations on your instance, use the `cfn-hup` helper script.

Update requires: Replacement for instance store-backed instances.

Volumes

The Amazon EBS volumes to attach to the instance.

Before detaching a volume, unmount any file systems on the device within your operating system. If you don't unmount the file system, a volume might get stuck in a busy state while detaching.

Required: No

Type: A list of EC2 MountPoints.

Update requires: No interruption

AdditionalInfo

Reserved.

Required: No

Type: String

Update requires: Some interruptions for Amazon EBS-backed instances

Update requires: Replacement for instance store-backed instances

Return Values

Ref

When you pass the logical ID of an `AWS::EC2::Instance` object to the intrinsic `Ref` function, the object's `InstanceId` is returned. For example: `i-636be302`.

Fn::GetAtt

Fn::GetAtt returns a value for a specified attribute of this type. The following are the available attributes and sample return values.

AvailabilityZone

The Availability Zone where the specified instance is launched. For example: us-east-1b.

You can retrieve a list of all Availability Zones for a region by using the Fn::GetAZs intrinsic function.

PrivateDnsName

The private DNS name of the specified instance. For example: ip-10-24-34-0.ec2.internal.

PublicDnsName

The public DNS name of the specified instance. For example: ec2-107-20-50-45.compute-1.amazonaws.com.

PrivateIp

The private IP address of the specified instance. For example: 10.24.34.0.

PublicIp

The public IP address of the specified instance. For example: 192.0.2.0.

Examples

EC2 Instance with an EBS Block Device Mapping

```
1 AWSTemplateFormatVersion: "2010-09-09"
2   Description: "Ec2 block device mapping"
3   Resources:
4     MyEC2Instance:
5       Type: "AWS::EC2::Instance"
6       Properties:
7         ImageId: "ami-79fd7eee"
8         KeyName: "testkey"
9         BlockDeviceMappings:
10           - DeviceName: "/dev/sdm"
11             Ebs:
12               VolumeType: "io1"
13               Iops: "200"
14               DeleteOnTermination: "false"
15               VolumeSize: "20"
16             - DeviceName: "/dev/sdk"
17               NoDevice: {}
```

Automatically Assign a Public IP Address

You can associate a public IP address with a network interface only if it has a device index of 0 and if it is a new network interface (not an existing one).

```
1 Ec2Instance:
2   Type: "AWS::EC2::Instance"
3   Properties:
4     ImageId:
5       Fn::FindInMap:
6         - "RegionMap"
7         - Ref: "AWS::Region"
8         - "AMI"
9     KeyName:
10    Ref: "KeyName"
11   NetworkInterfaces:
12     - AssociatePublicIpAddress: "true"
13     DeviceIndex: "0"
14     GroupSet:
15       - Ref: "myVPCEC2SecurityGroup"
16     SubnetId:
17       Ref: "PublicSubnet"
```

Chapter 6: Stack Policy

When you create a stack, all update actions are allowed on all resources. By default, anyone with stack update permissions can update all of the resources in the stack. During an update, some resources might require an interruption or be completely replaced, resulting in new physical IDs or completely new storage. You can prevent stack resources from being unintentionally updated or deleted during a stack update by using a stack policy. A stack policy is a JSON document that defines the update actions that can be performed on designated resources.⁶³

After you set a stack policy, all of the resources in the stack are protected by default. To allow updates on specific resources, you specify an explicit `Allow` statement for those resources in your stack policy. You can define only one stack policy per stack, but, you can protect multiple resources within a single policy. A stack policy applies to all AWS CloudFormation users who attempt to update the stack. You can't associate different stack policies with different users.

A stack policy applies only during stack updates. It doesn't provide access controls like an AWS Identity and Access Management (IAM) policy. Use a stack policy only as a fail-safe mechanism to prevent accidental updates to specific stack resources. To control access to AWS resources or actions, use IAM.

Example Stack Policy

The following example stack policy prevents updates to the `ProductionDatabase` resource:

```
{  
  "Statement" : [  
    {  
      "Effect" : "Allow",  
      "Action" : "Update:*",  
      "Principal": "*",  
      "Resource" : "*"  
    },  
    {  
      "Effect" : "Deny",  
      "Action" : "Update:*",  
      "Principal": "*",  
      "Resource" : "LogicalResourceId/ProductionDatabase"  
  ]  
}
```

⁶³<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/protect-stack-resources.html>

```
    }  
]  
}
```

Defining a Stack Policy

When you create a stack, no stack policy is set, so all update actions are allowed on all resources. To protect stack resources from update actions, define a stack policy and then set it on your stack. A stack policy is a JSON document that defines the AWS CloudFormation stack update actions that AWS CloudFormation users can perform and the resources that the actions apply to. You set the stack policy when you create a stack, by specifying a text file that contains your stack policy or typing it out. When you set a stack policy on your stack, any update not explicitly allowed is denied by default.

You define a stack policy with five elements: **Effect**, **Action**, **Principal**, **Resource**, and **Condition**.

Effect

Determines whether the actions that you specify are denied or allowed on the resource(s) that you specify. You can specify only **Deny** or **Allow**.

Example:

```
1   "Effect": "Deny"
```

If a stack policy includes overlapping statements (both allowing and denying updates on a resource), a **Deny** statement always overrides an **Allow** statement. To ensure that a resource is protected, use a **Deny** statement for that resource.

Action

Specifies the update actions that are denied or allowed.

Update:Modify: Specifies update actions during which resources might experience no interruptions or some interruptions while changes are being applied. All resources maintain their physical IDs.

Update:Replace: Specifies update actions during which resources are recreated. AWS CloudFormation creates a new resource with the specified updates and then deletes the old resource. Because the resource is recreated, the physical ID of the new resource might be different.

Update:Delete: Specifies update actions during which resources are removed. Updates that completely remove resources from a stack template require this action.

Update:*: Specifies all update actions. The asterisk is a wild card that represents all update actions.

The following example shows how to specify just the replace and delete actions:

```
1     "Action": [ "Update:Replace", "Update:Delete"]
```

To allow all update actions except for one, use `NotAction`. For example, to allow all update actions except for `Update:Delete`, use `NotAction`, as shown in this example:

```
1   {
2     "Statement" : [
3       {
4         "Effect": "Allow",
5         "NotAction": "Update:Delete",
6         "Principal": "*",
7         "Resource": "*"
8       }
9     ]
10 }
```

Principal

The `Principal` element specifies the entity that the policy applies to. This element is required but supports only the wild card (*), which means that the policy applies to all principals.

Resource

Specifies the logical IDs of the resources that the policy applies to. To specify types of resources, use the `Condition` element.

To specify a single resource, use its logical ID. For example:

```
1   "Resource": [ "LogicalResourceId/myEC2instance" ]
```

You can use a wild card with logical IDs. For example, if you use a common logical ID prefix for all related resources, you can specify all of them with a wild card:

```
1   "Resource" : [ "LogicalResourceId/CriticalResource*" ]
```

You can also use a `Not` element with resources. For example, to allow updates to all resources except for one, use a `NotResource` element to protect that resource:

```
1  {
2      "Statement" : [
3          {
4              "Effect" : "Allow",
5              "Action" : "Update:*",
6              "Principal": "*",
7              "NotResource" : "LogicalResourceId/ProductionDatabase"
8          }
9      ]
10 }
```

When you set a stack policy, any update not explicitly allowed is denied. By allowing updates to all resources except for the `ProductionDatabase` resource, you deny updates to the `ProductionDatabase` resource.

Conditions

Specifies the resource type that the policy applies to. To specify the logical IDs of specific resources, use the `Resource` element.

You can specify a resource type, such as all EC2 and RDS DB instances, as shown in the following example:

```
1  {
2      "Statement" : [
3          {
4              "Effect": "Deny",
5              "Principal": "*",
6              "Action": "Update:*",
7              "Resource": "*",
8              "Condition": {
9                  "StringEquals": {
10                      "ResourceType": ["AWS::EC2::Instance", "AWS::RDS::DBInstance"]
11                  }
12              }
13          },
14          {
15              "Effect": "Allow",
16              "Principal": "*",
17              "Action": "Update:*",
18              "Resource": "*"
19          }
20      ]
21 }
```

The `Allow` statement grants update permissions to all resources and the `Deny` statement denies updates to EC2 and RDS DB instances. The `Deny` statement always overrides `Allow` actions.

You can use a wild card with resource types. For example, you can deny update permissions to all Amazon EC2 resources, such as instances, security groups, and subnets, by using a wild card, as shown in the following example:

```
1 "Condition" : {  
2     "StringLike" : {  
3         "ResourceType" : [ "AWS::EC2::*" ]  
4     }  
5 }
```

You must use the `StringLike` condition when you use wild cards.

Setting a Stack Policy

You can use the console or AWS CLI to apply a stack policy when you create a stack. You can also use the AWS CLI to apply a stack policy to an existing stack. After you apply a stack policy, you can't remove it from the stack, but you can use the AWS CLI to modify it.

Stack policies apply to all AWS CloudFormation users who attempt to update the stack. You can't associate different stack policies with different users.

To set a stack policy when you create a stack (console):

1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
2. On the **CloudFormation Stacks** page, choose **Create Stack**.



3. In the **Create Stack** wizard, on the **Options** page, expand the **Advanced** section.

The screenshot shows the 'Stack Configuration' section of the AWS CloudFormation console. It includes fields for 'Termination Protection' (set to 'Disabled'), 'Timeout' (set to 0 minutes), 'Rollback on failure' (set to 'Yes'), and a large text area for 'Stack policy'. Below the text area are two options: 'Upload policy file' (selected) and 'Choose File' (button). A note at the bottom says 'No file chosen'.

Termination Protection ⓘ Enabled Disabled

Timeout ⓘ Minutes

Rollback on failure ⓘ Yes No

Stack policy ⓘ Enter policy

Upload policy file
Choose File No file chosen

[Learn more](#)

4. Choose **Enter policy** and then type the policy in the text box, or select **Upload policy file**, click **Choose File** and then choose the file that contains the stack policy.

To set a stack policy when you create a stack (CLI):

Use the `aws cloudformation create-stack` command with the `--stack-policy-body` option to type in a modified policy or the `--stack-policy-url` option to specify a file containing the policy.

To set a stack policy on an existing stack (CLI only):

Use the `aws cloudformation set-stack-policy` command with the `--stack-policy-body` option to type in a modified policy or the `--stack-policy-url` option to specify a file containing the policy.



To add a policy to an existing stack, you must have permission to the AWS CloudFormation `SetStackPolicy` action.

Updating Protected Resources

To update protected resources, create a temporary policy that overrides the stack policy and allows updates on those resources. Specify the override policy when you update the stack. The override

policy doesn't permanently change the stack policy.

To update protected resources, you must have permission to use the AWS CloudFormation SetStackPolicy action.



During a stack update, AWS CloudFormation automatically updates resources that depend on other updated resources. For example, AWS CloudFormation updates a resource that references an updated resource. AWS CloudFormation makes no physical changes, such as the resources' ID, to automatically updated resources, but if a stack policy is associated with those resources, you must have permission to update them.

To update a protected resource (console):

1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
2. Select the stack that you want to update, choose **Actions**, and then choose **Update Stack**.
3. If you modified the stack template, specify the location of the updated template. If not, choose **Use current template**.
 - For a template stored locally on your computer, choose **Upload a template to Amazon S3**. Choose **Choose File** to navigate to the file, select it, and then choose **Next**.
 - For a template stored in an Amazon S3 bucket, choose **Specify an Amazon S3 URL**. Type or paste the URL for the template, and then choose **Next**.
If you have a template in a versioning-enabled bucket, you can specify a specific version of the template, such as <https://s3.amazonaws.com/templates/myTemplate.template?versionId=123ab1cdeKdOW5IH4GAcYbEngcpTJTDW>.
4. If your template contains parameters, on the **Specify Parameters** page, enter or modify the parameter values, and then choose **Next**.
AWS CloudFormation populates each parameter with the value that is currently set in the stack except for parameters declared with the NoEcho attribute. You can use current values for those parameters by choosing **Use existing value**.
5. On the **Options** page, choose the file that contains the overriding stack policy or type a policy, and then choose **Next**. The override policy must specify an **Allow** statement for the protected resources that you want to update.
For example, to update all protected resources, specify a temporary override policy that allows all updates:

```
1  {
2      "Statement" : [
3          {
4              "Effect" : "Allow",
5              "Action" : "Update:*",
6              "Principal": "*",
7              "Resource" : "*"
8          }
9      ]
10 }
```



AWS CloudFormation applies the override policy only during this update. The override policy doesn't permanently change the stack policy.



6. Review the stack information and the changes that you submitted.

In the **Review** section, check that you submitted the correct information, such as the correct parameter values or template URL. If your template contains IAM resources, choose **I acknowledge that this template may create IAM resources** to specify that you want to use IAM resources in the template.

In the **Preview your changes** section, check that AWS CloudFormation will make all the changes that you expect. For example, 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.

7. Choose 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`.

To update a protected resource (CLI):

Use the `aws cloudformation update-stack` command with the `--stack-policy-during-update-body` option to type in a modified policy or the `--stack-policy-during-update-url` option to specify a file containing the policy.



AWS CloudFormation applies the override policy only during this update. The override policy doesn't permanently change the stack policy.

Modifying a Stack Policy To protect additional resources or to remove protection from resources, modify the stack policy. For example, when you add a database that you want to protect to your

stack, add a Deny statement for that database to the stack policy. To modify the policy, you must have permission to use the SetStackPolicy action.

Use the AWS CLI to modify stack policies.

To modify a stack policy (CLI):

Use the `aws cloudformation set-stack-policy` command with the `--stack-policy-body` option to type in a modified policy or the `--stack-policy-url` option to specify a file containing the policy.

You can't delete a stack policy. To remove all protection from all resources, you modify the policy to explicitly allow all actions on all resources. The following policy allows all updates on all resources:

```
{  
  "Statement" : [  
    {  
      "Effect" : "Allow",  
      "Action" : "Update:*",  
      "Principal": "*",  
      "Resource" : "*"  
    }  
  ]  
}
```

More Example Stack Policies

The following example policies show how to prevent updates to all stack resources and to specific resources, and prevent specific types of updates.

Prevent Updates to All Stack Resources

To prevent updates to all stack resources, the following policy specifies a Deny statement for all update actions on all resources.

```
{  
  "Statement" : [  
    {  
      "Effect" : "Deny",  
      "Action" : "Update:*",  
      "Principal": "*",  
      "Resource" : "*"  
    }  
  ]  
}
```

Prevent Updates to a Single Resource

The following policy denies all update actions on the database with the MyDatabase logical ID. It allows all update actions on all other stack resources with an Allow statement. The Allow statement doesn't apply to the MyDatabase resource because the Deny statement always overrides allow actions.

```
{  
  "Statement" : [  
    {  
      "Effect" : "Deny",  
      "Action" : "Update:*",  
      "Principal": "*",  
      "Resource" : "LogicalResourceId/MyDatabase"  
    },  
    {  
      "Effect" : "Allow",  
      "Action" : "Update:*",  
      "Principal": "*",  
      "Resource" : "*"  
    }  
  ]  
}
```

You can achieve the same result as the previous example by using a default denial. When you set a stack policy, AWS CloudFormation denies any update that is not explicitly allowed. The following policy allows updates to all resources except for the ProductionDatabase resource, which is denied by default.

```
{  
  "Statement" : [  
    {  
      "Effect" : "Allow",  
      "Action" : "Update:*",  
      "Principal": "*",  
      "NotResource" : "LogicalResourceId/ProductionDatabase"  
    }  
  ]  
}
```



There is risk in using a default denial. If you have an Allow statement elsewhere in the policy (such as an Allow statement that uses a wildcard), you might unknowingly grant update permission to resources that you don't intend to. Because an explicit denial overrides any allow actions, you can ensure that a resource is protected by using a Deny statement.

Prevent Updates to All Instances of a Resource Type

The following policy denies all update actions on the RDS DB instance resource type. It allows all update actions on all other stack resources with an Allow statement. The Allow statement doesn't apply to the RDS DB instance resources because a Deny statement always overrides allow actions.

```
{  
    "Statement" : [  
        {  
            "Effect" : "Deny",  
            "Action" : "Update:*",  
            "Principal": "*",  
            "Resource" : "*",  
            "Condition" : {  
                "StringEquals" : {  
                    "ResourceType" : [ "AWS::RDS::DBInstance" ]  
                }  
            }  
        },  
        {  
            "Effect" : "Allow",  
            "Action" : "Update:*",  
            "Principal": "*",  
            "Resource" : "*"  
        }  
    ]  
}
```

Prevent Replacement Updates for an Instance

The following policy denies updates that would cause a replacement of the instance with the MyInstance logical ID. It allows all update actions on all other stack resources with an Allow statement. The Allow statement doesn't apply to the MyInstance resource because the Deny statement always overrides allow actions.

```
{  
  "Statement" : [  
    {  
      "Effect" : "Deny",  
      "Action" : "Update:Replace",  
      "Principal": "*",  
      "Resource" : "LogicalResourceId/MyInstance"  
    },  
    {  
      "Effect" : "Allow",  
      "Action" : "Update:*",  
      "Principal": "*",  
      "Resource" : "*"  
    }  
  ]  
}
```

Prevent Updates to Nested Stacks

The following policy denies all update actions on the AWS CloudFormation stack resource type (nested stacks). It allows all update actions on all other stack resources with an Allow statement. The Allow statement doesn't apply to the AWS CloudFormation stack resources because the Deny statement always overrides allow actions.

```
{  
  "Statement" : [  
    {  
      "Effect" : "Deny",  
      "Action" : "Update:*",  
      "Principal": "*",  
      "Resource" : "*",  
      "Condition" : {  
        "StringEquals" : {  
          "ResourceType" : ["AWS::CloudFormation::Stack"]  
        }  
      }  
    },  
    {  
      "Effect" : "Allow",  
      "Action" : "Update:*",  
      "Principal": "*",  
      "Resource" : "*"
```

```
    }  
]  
}
```

Appendix A: Converting JSON to YAML

JSON to YAML

You can use the following Python script to convert a JSON CloudFormation template into a YAML template.

```
$ vi json2yaml.py
```

```
1 #!/usr/bin/env python \
2
3
4 import argparse
5 import json
6 import yaml
7
8 def parse_arguments():
9     """This method is used to parse arguments """
10    parser = argparse.ArgumentParser(description='json to yaml converter')
11    parser.add_argument('--input', '-i', required=True, help='input file')
12    parser.add_argument('--output', '-o', required=True, help='output file')
13    args = parser.parse_args()
14    return args
15
16 def main(args):
17    with open(args.input) as input_file, open(args.output, "w") as output_file:
18        data = json.load(input_file)
19        yaml.safe_dump(data, output_file, allow_unicode=True, default_flow_style\
20=False)
21
22 if __name__ == "__main__":
23    PARSED_ARGS = parse_arguments()
24    main(PARSED_ARGS)
```

To use the script:

```
$ python json2yaml.py -i <input file> -o <output file>
```

<Input file> is the JSON template file you want to convert into YAML and <output file> is the output file name.

YAML to JSON

You can use the following script to convert a YAML CloudFormation template into a JSON template.

```
$ vi yaml2json.py
```

```
1 #!/usr/bin/env python \
2
3
4 import argparse
5 import json
6 import yaml
7
8 def parse_arguments():
9     """This method is used to parse arguments """
10    parser = argparse.ArgumentParser(description='yaml to json converter')
11    parser.add_argument('--input', '-i', required=True, help='input file')
12    parser.add_argument('--output', '-o', required=True, help='output file')
13    args = parser.parse_args()
14    return args
15
16 def main(args):
17    with open(args.input) as input_file, open(args.output, "w") as output_file:
18        data = yaml.load(input_file)
19        json.dump(data, output_file, sort_keys=True, indent=4)
20
21 if __name__ == "__main__":
22    PARSED_ARGS = parse_arguments()
23    main(PARSED_ARGS)
```

To use the script:

```
$ python yaml2json.py -i <input file> -o <output file>
```

<Input file> is the YAML template file you want to convert into JSON and <output file> is the output file name.

Appendix B: CLI Commands

With the AWS Command Line Interface (CLI), you can create, monitor, update and delete stacks from your system's terminal. You can also use the AWS CLI to automate actions through scripts.⁶⁴



If you use Windows PowerShell, AWS also offers the AWS Tools for Windows PowerShell.

Creating a Stack

To create a stack you run the `aws cloudformation create-stack` command. You must provide the stack name, the location of a valid template, and any input parameters.

Parameters are separated with a space and the key names are case sensitive. If you mistype a parameter key name when you run `aws cloudformation create-stack`, AWS CloudFormation doesn't create the stack and reports that the template doesn't contain that parameter.

If you specify a local template file, AWS CloudFormation uploads it to an Amazon S3 bucket in your AWS account. AWS CloudFormation creates a unique bucket for each region in which you upload a template file. The buckets are accessible to anyone with Amazon S3 permissions in your AWS account. If an AWS CloudFormation-created bucket already exists, 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. By default, `aws cloudformation describe-stacks` returns parameter values. To prevent sensitive parameter values such as passwords from being returned, include a `NoEcho` property set to `TRUE` in your AWS CloudFormation template.

The following example creates the `myteststack` stack:

⁶⁴<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-using-cli.html>

```
PROMPT> aws cloudformation create-stack --stack-name myteststack --template-body\  
file:///home/testuser/mytemplate.json --parameters ParameterKey=Parm1,Parameter\  
Value=test1 ParameterKey=Parm2,ParameterValue=test2  
{  
    "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/myteststack/3\  
30b0120-1771-11e4-af37-50ba1b98bea6"  
}
```

Describing and Listing Your Stacks

You can use two AWS CLI commands to get information about your AWS CloudFormation stacks: `aws cloudformation list-stacks` and `aws cloudformation describe-stacks`.

`aws cloudformation list-stacks`

The `aws cloudformation list-stacks` command enables you to get a list of any of the stacks you have created (even those which have been deleted up to 90 days). You can use an option to filter results by stack status, such as `CREATE_COMPLETE` and `DELETE_COMPLETE`. The `aws cloudformation list-stacks` command returns summary information about any of your running or deleted stacks, including the name, stack identifier, template, and status.



The `aws cloudformation list-stacks` command returns information on deleted stacks for 90 days after they have been deleted.

The following example shows a summary of all stacks that have a status of `CREATE_COMPLETE`:

```
PROMPT> aws cloudformation list-stacks --stack-status-filter CREATE_COMPLETE  
[  
{  
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/mytestst\  
ack/  
644df8e0-0dff-11e3-8e2f-5088487c4896",  
    "TemplateDescription": "AWS CloudFormation Sample Template S3_Bucket: Sa\  
mple template showing how to create a publicly accessible S3 bucket. **WARNING**\  
This template creates an  
S3 bucket. You will be billed for the AWS resources used if you create a stack f\  
rom this template.",  
    "StackStatusReason": null,  
    "CreationTime": "2013-08-26T03:27:10.190Z",  
    "StackName": "myteststack",
```

```
        "StackStatus": "CREATE_COMPLETE"
    }
]
```

aws cloudformation describe-stacks

The `aws cloudformation describe-stacks` command provides information on your running stacks. You can use an option to filter results on a stack name. This command returns information about the stack, including the name, stack identifier, and status.

The following example shows summary information for the `myteststack` stack:

```
PROMPT> aws cloudformation describe-stacks --stack-name myteststack
{
    "Stacks": [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/myte\ststack/a69442d0-0b8f-11e3-8b8a-500150b352e0",
            "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample\\ template showing how to create a publicly accessible S3 bucket. **WARNING** Thi\\ s template creates an S3 bucket.  
You will be billed for the AWS resources used if you create a stack from this te\\ mplate.",
            "Tags": [],
            "Outputs": [
                {
                    "Description": "Name of S3 bucket to hold website content",
                    "OutputKey": "BucketName",
                    "OutputValue": "myteststack-s3bucket-jssofifi1zie2w"
                }
            ],
            "StackStatusReason": null,
            "CreationTime": "2013-08-23T01:02:15.422Z",
            "Capabilities": [],
            "StackName": "myteststack",
            "StackStatus": "CREATE_COMPLETE",
            "DisableRollback": false
        }
    ]
}
```

If you don't use the `--stack-name` option to limit the output to one stack, information on all your running stacks is returned.

Stack Status Codes

You can specify one or more stack status codes to list only stacks with the specified status codes. The following list describes each stack status code:

CREATE_COMPLETE

Successful creation of one or more stacks.

CREATE_IN_PROGRESS

Ongoing creation of one or more stacks.

CREATE_FAILED

Unsuccessful creation of one or more stacks. View the stack events to see any associated error messages. Possible reasons for a failed creation include insufficient permissions to work with all resources in the stack, parameter values rejected by an AWS service, or a timeout during resource creation.

DELETE_COMPLETE

Successful deletion of one or more stacks. Deleted stacks are retained and viewable for 90 days.

DELETE_FAILED

Unsuccessful deletion of one or more stacks. Because the delete failed, you might have some resources that are still running; however, you cannot work with or update the stack. Delete the stack again or view the stack events to see any associated error messages.

DELETE_IN_PROGRESS

Ongoing removal of one or more stacks.

REVIEW_IN_PROGRESS

Ongoing creation of one or more stacks with an expected StackId but without any templates or resources. A stack with this status code counts against the maximum possible number of stacks.

ROLLBACK_COMPLETE

Successful removal of one or more stacks after a failed stack creation or after an explicitly canceled stack creation. Any resources that were created during the create stack action are deleted.

ROLLBACK_FAILED

Unsuccessful removal of one or more stacks after a failed stack creation or after an explicitly canceled stack creation. Delete the stack or view the stack events to see any associated error messages.

ROLLBACK_IN_PROGRESS

Ongoing removal of one or more stacks after a failed stack creation or after an explicitly cancelled stack creation.

UPDATE_COMPLETE

Successful update of one or more stacks.

UPDATE_COMPLETE_CLEANUP_IN_PROGRESS

Ongoing removal of old resources for one or more stacks after a successful stack update. For stack updates that require resources to be replaced, AWS CloudFormation creates the new resources first and then deletes the old resources to help reduce any interruptions with your stack. In this state, the stack has been updated and is usable, but AWS CloudFormation is still deleting the old resources.

UPDATE_IN_PROGRESS

Ongoing update of one or more stacks.

UPDATE_ROLLBACK_COMPLETE

Successful return of one or more stacks to a previous working state after a failed stack update.

UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS

Ongoing removal of new resources for one or more stacks after a failed stack update. In this state, the stack has been rolled back to its previous working state and is usable, but AWS CloudFormation is still deleting any new resources it created during the stack update.

UPDATE_ROLLBACK_FAILED

Unsuccessful return of one or more stacks to a previous working state after a failed stack update. You can delete the stack or contact customer support to restore the stack to a usable state.

UPDATE_ROLLBACK_IN_PROGRESS

Ongoing return of one or more stacks to the previous working state after failed stack update.

Viewing Stack Event History

You can track the status of the resources AWS CloudFormation is creating and deleting with the `aws cloudformation describe-stack-events` command. The amount of time to create or delete a stack depends on the complexity of your stack.

In the following example, a sample stack is created from a template file by using the `aws cloudformation create-stack` command. After the stack is created, the events that were reported during stack creation are shown by using the `aws cloudformation describe-stack-events` command.

The following example creates a stack with the name `myteststack` using the `sampletemplate.json` template file:

```
PROMPT> aws cloudformation create-stack --stack-name myteststack --template-body \
file:///home/local/test/sampletemplate.json
[
  {
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/mytestst\
ack/466df9e0-0dff-08e3-8e2f-5088487c4896",
    "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample tem\
plate showing how to create a publicly accessible S3 bucket. **WARNING** This te\
mplate creates an S3 bucket.
You will be billed for the AWS resources used if you create a stack from this te\
mplate.",
    "Tags": [],
    "Outputs": [
      {
        "Description": "Name of S3 bucket to hold website content",
        "OutputKey": "BucketName",
        "OutputValue": "myteststack-s3bucket-jssofi1zie2w"
      }
    ],
    "StackStatusReason": null,
    "CreationTime": "2013-08-23T01:02:15.422Z",
    "Capabilities": [],
    "StackName": "myteststack",
    "StackStatus": "CREATE_COMPLETE",
    "DisableRollback": false
  }
]
```

The following example describes the myteststack stack:

```
PROMPT> aws cloudformation describe-stack-events --stack-name myteststack
{
  "StackEvents": [
    {
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/myte\
ststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
      "EventId": "af67ef60-0b8f-11e3-8b8a-500150b352e0",
      "ResourceStatus": "CREATE_COMPLETE",
      "ResourceType": "AWS::CloudFormation::Stack",
      "Timestamp": "2013-08-23T01:02:30.070Z",
      "StackName": "myteststack",
      "PhysicalResourceId": "arn:aws:cloudformation:us-east-1:123456789012\  
"
```

```
:stack/myteststack/a69442d0-0b8f-11e3-8b8a-500150b352e0",
    "LogicalResourceId": "myteststack"
},
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/myte\
ststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
    "EventId": "S3Bucket-CREATE_COMPLETE-1377219748025",
    "ResourceStatus": "CREATE_COMPLETE",
    "ResourceType": "AWS::S3::Bucket",
    "Timestamp": "2013-08-23T01:02:28.025Z",
    "StackName": "myteststack",
    "ResourceProperties": "{\"AccessControl\":\"PublicRead\"}",
    "PhysicalResourceId": "myteststack-s3bucket-jssofi1zie2w",
    "LogicalResourceId": "S3Bucket"
},
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/myte\
ststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
    "EventId": "S3Bucket-CREATE_IN_PROGRESS-1377219746688",
    "ResourceStatus": "CREATE_IN_PROGRESS",
    "ResourceType": "AWS::S3::Bucket",
    "Timestamp": "2013-08-23T01:02:26.688Z",
    "ResourceStatusReason": "Resource creation Initiated",
    "StackName": "myteststack",
    "ResourceProperties": "{\"AccessControl\":\"PublicRead\"}",
    "PhysicalResourceId": "myteststack-s3bucket-jssofi1zie2w",
    "LogicalResourceId": "S3Bucket"
},
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/myte\
ststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
    "EventId": "S3Bucket-CREATE_IN_PROGRESS-1377219743862",
    "ResourceStatus": "CREATE_IN_PROGRESS",
    "ResourceType": "AWS::S3::Bucket",
    "Timestamp": "2013-08-23T01:02:23.862Z",
    "StackName": "myteststack",
    "ResourceProperties": "{\"AccessControl\":\"PublicRead\"}",
    "PhysicalResourceId": null,
    "LogicalResourceId": "S3Bucket"
},
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/myte\

```

```

ststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
    "EventId": "a69469e0-0b8f-11e3-8b8a-500150b352e0",
    "ResourceStatus": "CREATE_IN_PROGRESS",
    "ResourceType": "AWS::CloudFormation::Stack",
    "Timestamp": "2013-08-23T01:02:15.422Z",
    "ResourceStatusReason": "User Initiated",
    "StackName": "myteststack",
    "PhysicalResourceId": "arn:aws:cloudformation:us-east-1:123456789012\
:stack/myteststack/a69442d0-0b8f-11e3-8b8a-500150b352e0",
    "LogicalResourceId": "myteststack"
}
]
}

```



You can run the `aws cloudformation describe-stack-events` command while the stack is being created to view events as they are reported.

The most recent events are reported first. The following table describe the fields returned by the `aws cloudformation describe-stack-events` command:

Field	Description
EventId	Event identifier
StackName	Name of the stack that the event corresponds to
StackId	Identifier of the stack that the event corresponds to
LogicalResourceId	Logical identifier of the resource
PhysicalResourceId	Physical identifier of the resource
ResourceProperties	Properties of the resource
ResourceType	Type of the resource
Timestamp	Time when the event occurred
ResourceStatus	The status of the resource, which can be one of the following status codes: CREATE_COMPLETE CREATE_FAILED CREATE_IN_PROGRESS DELETE_COMPLETE DELETE_FAILED DELETE_IN_PROGRESS DELETE_SKIPPED UPDATE_COMPLETE UPDATE_FAILED UPDATE_IN_PROGRESS. The DELETE_SKIPPED status applies to resources with a deletion policy attribute of retain.
ResourceStatusReason	More information on the status

Listing Resources

Immediately after you run the `aws cloudformation create-stack` command, you can list its resources using the `aws cloudformation list-stack-resources` command. This command lists a summary of each resource in the stack that you specify with the `--stack-name` parameter. The report includes a summary of the stack, including the creation or deletion status.

The following example shows the resources for the `myteststack` stack:

```
PROMPT> aws cloudformation list-stack-resources --stack-name myteststack
{
    "StackResourceSummaries": [
        {
            "ResourceStatus": "CREATE_COMPLETE",
            "ResourceType": "AWS::S3::Bucket",
            "ResourceStatusReason": null,
            "LastUpdatedTimestamp": "2013-08-23T01:02:28.025Z",
            "PhysicalResourceId": "myteststack-s3bucket-sample",
            "LogicalResourceId": "S3Bucket"
        }
    ]
}
```

AWS CloudFormation reports resource details on any running or deleted stack. If you specify the name of a stack whose status is `CREATE_IN_PROCESS`, AWS CloudFormation reports only those resources whose status is `CREATE_COMPLETE`.



The `aws cloudformation describe-stack-resources` command returns information on deleted stacks for 90 days after they have been deleted.

Retrieving a Template

AWS CloudFormation stores the template you use to create your stack as part of the stack. You can retrieve the template from AWS CloudFormation using the `aws cloudformation get-template` command.



The `aws cloudformation get-template` command returns the deleted stacks templates for up to 90 days after the stack has been deleted.

The following example shows the template for the `myteststack` stack:

```
PROMPT> aws cloudformation get-template --stack-name myteststack
{
  "TemplateBody": {
    "AWSTemplateFormatVersion": "2010-09-09",
    "Outputs": {
      "BucketName": {
        "Description": "Name of S3 bucket to hold website content",
        "Value": {
          "Ref": "S3Bucket"
        }
      }
    },
    "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample template showing how to create a publicly accessible S3 bucket. **WARNING** This template creates an S3 bucket.
You will be billed for the AWS resources used if you create a stack from this template.",
    "Resources": {
      "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
          "AccessControl": "PublicRead"
        }
      }
    }
  }
}
```

The output contains the entire template body, enclosed in quotation marks.

Validating a Template

To check your template file for syntax errors, you can use the `aws cloudformation validate-template` command.



The `aws cloudformation validate-template` command is designed to check only the syntax of your template. It does not ensure that the property values you have specified for a resource are valid for that resource. Nor does it determine the number of resources that will exist when the stack is created. To check the operational validity, you need to attempt to create the stack. There is no sandbox or test area for AWS CloudFormation stacks, so you are charged for the resources you create during testing.

During validation, AWS CloudFormation first checks if the template is valid JSON. If it isn't, AWS CloudFormation checks if the template is valid YAML. If both checks fail, AWS CloudFormation returns a template validation error. You can validate templates locally by using the --template-body parameter, or remotely with the --template-url parameter. The following example validates a template in a remote location:

```
PROMPT> aws cloudformation validate-template --template-url https://s3.amazonaws\
.com/cloudformation-templates-us-east-1/S3_Bucket.template
{
    "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample templat\
e showing how to create a publicly accessible S3 bucket. **WARNING** This templat\
e creates an S3 bucket.
You will be billed for the AWS resources used if you create a stack from this te\
mplate.",
    "Parameters": [],
    "Capabilities": []
}
```

The expected result is no error message, with information about all parameters listed.

The following example shows an error with a local template file:

```
PROMPT> aws cloudformation validate-template --template-body file:///home/local/\\
test/samplename.json
{
    "ResponseMetadata": {
        "RequestId": "4ae33ec0-1988-11e3-818b-e15a6df955cd"
    },
    "Errors": [
        {
            "Message": "Template format error: JSON not well-formed. (line 11, c\
olumn 8)",
            "Code": "ValidationError",
            "Type": "Sender"
        }
    ],
    "Capabilities": [],
    "Parameters": []
}
A client error (ValidationError) occurred: Template format error: JSON not well-\
formed. (line 11, column 8)
```

Uploading Local Artifacts to an S3 Bucket

For some resource properties that require an Amazon S3 location (a bucket name and filename), you can specify local references instead. For example, you might specify the S3 location of your AWS Lambda function's source code or an Amazon API Gateway REST API's OpenAPI (formerly Swagger) file. Instead of manually uploading the files to an S3 bucket and then adding the location to your template, you can specify local references, called local artifacts, in your template and then use the package command to quickly upload them. A local artifact is a path to a file or folder that the package command uploads to Amazon S3. For example, an artifact can be a local path to your AWS Lambda function's source code or an Amazon API Gateway REST API's OpenAPI file.

If you specify a file, the command directly uploads it to the S3 bucket. After uploading the artifacts, the command returns a copy of your template, replacing references to local artifacts with the S3 location where the command uploaded the artifacts. Then, you can use the returned template to create or update a stack.

If you specify a folder, the command creates a .zip file for the folder, and then uploads the .zip file. If you don't specify a path, the command creates a .zip file for the working directory, and uploads it. You can specify an absolute or relative path, where the relative path is relative to your template's location.



You can use local artifacts only for resource properties that the package command supports.

The following template specifies the local artifact for a Lambda function's source code. The source code is stored in the user's /home/user/code/lambdafunction folder.

Original Template

```
AWSTemplateFormatVersion: '2010-09-09'  
Transform: 'AWS::Serverless-2016-10-31'  
Resources:  
  MyFunction:  
    Type: 'AWS::Serverless::Function'  
    Properties:  
      Handler: index.handler  
      Runtime: nodejs4.3  
      CodeUri: /home/user/code/lambdafunction
```

The following command creates a .zip file containing the function's source code folder, and then uploads the .zip file to the root folder of the my-bucket bucket.

Package Command

```
aws cloudformation package --template /path_to_template/template.json --s3-bucket my-bucket --output json > packaged-template.json
```

The command saves the template that it generates to the path specified by the `--output` option. The command replaces the artifact with the S3 location, as shown in the following example:

Resulting Template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'
Resources:
  MyFunction:
    Type: 'AWS::Serverless::Function'
    Properties:
      Handler: index.handler
      Runtime: nodejs4.3
      CodeUri: s3://mybucket/lambdafunction.zip
```

Quickly Deploying Templates with Transforms

AWS CloudFormation requires you to use a change set to create a template that includes transforms. Instead of independently creating and then executing a change set, use the `aws cloudformation deploy` command. When you run this command, it creates a change set, executes the change set, and then terminates. This command reduces the numbers of required steps when you create or update a stack that includes transforms.

The following command creates a new stack by using the `my-template.json` template.

```
aws cloudformation deploy --template /path_to_template/my-template.json --stack-name my-new-stack --parameter-overrides Key1=Value1 Key2=Value2
```

Deleting a Stack

To delete a stack, you run the `aws cloudformation delete-stack` command. You must specify the name of the stack that you want to delete. When you delete a stack, you delete the stack and all of its resources.

The following example deletes the `myteststack` stack:

```
PROMPT> aws cloudformation delete-stack --stack-name myteststack
```

Appendix C: Intrinsic Function

AWS CloudFormation provides several built-in functions that help you manage your stacks. Use intrinsic functions in your templates to assign values to properties that are not available until runtime.⁶⁵



You can use intrinsic functions only in specific parts of a template. Currently, you can use intrinsic functions in resource properties, metadata attributes, and update policy attributes.

Fn::Base64

The intrinsic function Fn::Base64 returns the Base64 representation of the input string. This function is typically used to pass encoded data to Amazon EC2 instances by way of the UserData property.

Declaration

Syntax for the full function name:

```
"Fn::Base64": <valueToEncode>
```

Syntax for the short form:

```
!Base64 <valueToEncode>
```

If you use the short form and immediately include another function in the <valueToEncode> parameter, use the full function name for at least one of the functions. For example, the following syntax is invalid:

```
!Base64 !Sub <string>
!Base64 !Ref <logical_ID>
```

Instead, use the full function name for at least one of the functions, as shown in the following examples:

⁶⁵<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html>

```
!Base64
"Fn::Sub": <string>

"Fn::Base64":
!Sub <string>
```

Parameters

<valueToEncode>

The string value you want to convert to Base64.

Return Value

The original string, in Base64 representation.

Example

```
"Fn::Base64": AWS::CloudFormation
```

Supported Functions

You can use any function that returns a string inside the Fn::Base64 function.

Condition Functions

You can use intrinsic functions, such as Fn::If, Fn::Equals, and Fn::Not, to conditionally create stack resources. These conditions are evaluated based on input parameters that you declare when you create or update a stack. After you define all your conditions, you can associate them with resources or resource properties in the Resources and Outputs sections of a template.

You define all conditions in the Conditions section of a template except for Fn::If conditions. You can use the Fn::If condition in the metadata attribute, update policy attribute, and property values in the Resources section and Outputs sections of a template.

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 less capabilities to save costs. With conditions, you can define which resources are created and how they're configured for each environment type.



You can only reference other conditions and values from the Parameters and Mappings sections of a template. For example, you can reference a value from an input parameter, but you cannot reference the logical ID of a resource in a condition.

Associating a Condition

To conditionally create resources, resource properties, or outputs, you must associate a condition with them. Add the Condition: key and the logical ID of the condition as an attribute to associate a condition, as shown in the following snippet. AWS CloudFormation creates the NewVolume resource only when the CreateProdResources condition evaluates to true.

```
1 NewVolume:  
2   Type: AWS::EC2::Volume  
3   Condition: CreateProdResources  
4   Properties:  
5     Size: 100  
6     AvailabilityZone: !GetAtt EC2Instance.AvailabilityZone
```

For the Fn::If function, you only need to specify the condition name. The following snippet shows how to use Fn::If to conditionally specify a resource property. If the CreateLargeSize condition is true, AWS CloudFormation sets the volume size to 100. If the condition is false, AWS CloudFormation sets the volume size to 10.

```
1 NewVolume:  
2   Type: "AWS::EC2::Volume"  
3   Properties:  
4     Size:  
5       !If [CreateLargeSize, 100, 10]  
6     AvailabilityZone: !GetAtt: Ec2Instance.AvailabilityZone  
7   DeletionPolicy: Snapshot
```

You can also use conditions inside other conditions. The following snippet is from the `Conditions` section of a template. The `MyAndCondition` condition includes the `SomeOtherCondition` condition:

```
1 MyAndCondition: !And  
2   - !Equals ["sg-mysggroup", !Ref "ASecurityGroup"]  
3   - !Condition SomeOtherCondition
```

Fn::And

Returns `true` if all the specified conditions evaluate to `true`, or returns `false` if any one of the conditions evaluates to `false`. `Fn::And` acts as an AND operator. The minimum number of conditions that you can include is 2, and the maximum is 10.

Declaration

Syntax for the full function name:

`"Fn::And": [<condition>]`

Syntax for the short form:

`!And [<condition>]`

Parameters

<condition>

A condition that evaluates to `true` or `false`.

Example

The following `MyAndCondition` evaluates to `true` if the referenced security group name is equal to `sg-mysggroup` and if `SomeOtherCondition` evaluates to `true`:

```

1 MyAndCondition: !And
2   - !Equals ["sg-mysgggroup", !Ref ASecurityGroup]
3   - !Condition SomeOtherCondition

```

Fn::Equals

Compares if two values are equal. Returns `true` if the two values are equal or `false` if they aren't.

Declaration

Syntax for the full function name:

```
"Fn::Equals": [<value_1>, <value_2>]
```

Syntax for the short form:

```
!Equals [<value_1>, <value_2>]
```

Parameters

<value>

A value of any type that you want to compare.

Example

The following `UseProdCondition` condition evaluates to `true` if the value for the `EnvironmentType` parameter is equal to `prod`:

```

1 UseProdCondition:
2   !Equals [!Ref EnvironmentType, prod]

```

Fn::If

Returns one value if the specified condition evaluates to `true` and another value if the specified condition evaluates to `false`. Currently, AWS CloudFormation supports the `Fn::If` intrinsic function in the metadata attribute, update policy attribute, and property values in the Resources section and Outputs sections of a template. You can use the `AWS::NoValue` pseudo parameter as a return value to remove the corresponding property.

Declaration

Syntax for the full function name:

```
"Fn::If": [<condition_name>, <value_if_true>, <value_if_false>]
```

Syntax for the short form:

```
!If [<condition_name>, <value_if_true>, <value_if_false>]
```

Parameters

<condition_name>

A reference to a condition in the Conditions section. Use the condition's name to reference it.

<value_if_true>

A value to be returned if the specified condition evaluates to true.

<value_if_false>

A value to be returned if the specified condition evaluates to false.

Examples

Example 1

The following snippet uses an Fn::If function in the SecurityGroups property for an Amazon EC2 resource. If the CreateNewSecurityGroup condition evaluates to true, AWS CloudFormation uses the referenced value of NewSecurityGroup to specify the SecurityGroups property; otherwise, AWS CloudFormation uses the referenced value of ExistingSecurityGroup.

```
1 SecurityGroups:  
2   - !If [CreateNewSecurityGroup, !Ref NewSecurityGroup, !Ref ExistingSecurityGro\  
3     up]
```

Example 2

In the Output section of a template, you can use the Fn::If function to conditionally output information. In the following snippet, if the CreateNewSecurityGroup condition evaluates to true, AWS CloudFormation outputs the security group ID of the NewSecurityGroup resource. If the condition is false, AWS CloudFormation outputs the security group ID of the ExistingSecurityGroup resource.

```
1 Outputs:  
2   SecurityGroupId:  
3     Description: Group ID of the security group used.  
4     Value: !If [CreateNewSecurityGroup, !Ref NewSecurityGroup, !Ref ExistingSecu\  
5       rityGroup]
```

Example 3

The following snippet uses the AWS::NoValue pseudo parameter in an Fn::If function. The condition uses a snapshot for an Amazon RDS DB instance only if a snapshot ID is provided. If the UseDBSnapshot condition evaluates to true, AWS CloudFormation uses the DBSnapshotName parameter value for the DBSnapshotIdentifier property. If the condition evaluates to false, AWS CloudFormation removes the DBSnapshotIdentifier property.

```

1 MyDB:
2   Type: "AWS::RDS::DBInstance"
3   Properties:
4     AllocatedStorage: 5
5     DBInstanceClass: db.m1.small
6     Engine: MySQL
7     EngineVersion: 5.5
8     MasterUsername: !Ref DBUser
9     MasterUserPassword: !Ref DBPassword
10    DBParameterGroupName: !Ref MyRDSParamGroup
11    DBSnapshotIdentifier:
12      !If [UseDBSnapshot, !Ref DBSnapshotName, !Ref "AWS::NoValue"]

```

Example 4

The following snippet provides an auto scaling update policy only if the RollingUpdates condition evaluates to true. If the condition evaluates to false, AWS CloudFormation removes the AutoScalingRollingUpdate update policy.

```

1 UpdatePolicy:
2   AutoScalingRollingUpdate:
3     !If
4       - RollingUpdates
5       -
6         MaxBatchSize: 2
7         MinInstancesInService: 2
8         PauseTime: PT0M30S
9         - !Ref "AWS::NoValue"

```

Fn::Not

Returns true for a condition that evaluates to `false` or returns `false` for a condition that evaluates to true. `Fn::Not` acts as a NOT operator.

Declaration

Syntax for the full function name:

`"Fn::Not": [<condition>]`

Syntax for the short form:

`!Not [<condition>]`

Parameters

<condition>

A condition such as Fn::Equals that evaluates to true or false.

Example

The following MyNotCondition condition evaluates to true if the value for the EnvironmentType parameter is not equal to prod:

```
1 MyNotCondition:  
2   !Not [!Equals [!Ref EnvironmentType, prod]]
```

Fn::Or

Returns true if any one of the specified conditions evaluate to true, or returns false if all of the conditions evaluates to false. Fn::Or acts as an OR operator. The minimum number of conditions that you can include is 2, and the maximum is 10.

Declaration

Syntax for the full function name:

```
"Fn::Or": [<condition>, ...]
```

Syntax for the short form:

```
!Or [<condition>, ...]
```

Parameters

<condition>

A condition that evaluates to true or false.

Example

The following MyOrCondition evaluates to true if the referenced security group name is equal to sg-mysggroup or if SomeOtherCondition evaluates to true:

```
1 MyOrCondition:  
2   !Or [!Equals [sg-mysggroup, !Ref ASecurityGroup], Condition: SomeOtherCondition\  
3   n]
```

Supported Functions

You can use the following functions in the Fn::If condition:

- Fn::Base64
- Fn::FindInMap
- Fn::GetAtt
- Fn::GetAZs
- Fn::If
- Fn::Join
- Fn::Select
- Ref

You can use the following functions in all other condition functions, such as Fn::Equals and Fn::Or:

- Fn::FindInMap
- Ref
- Other condition functions



You can find sample templates for conditions here:

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/conditions-sample-templates.html>

Fn::FindInMap

The intrinsic function Fn::FindInMap returns the value corresponding to keys in a two-level map that is declared in the `Mappings` section.

Declaration

Syntax for the full function name:

```
"Fn::FindInMap": [ <MapName>, <TopLevelKey>, <SecondLevelKey> ]
```

Syntax for the short form:

```
!FindInMap [ <MapName>, <TopLevelKey>, <SecondLevelKey> ]
```

Parameters

<MapName>

The logical name of a mapping declared in the `Mappings` section that contains the keys and values.

<TopLevelKey>

The top-level key name. Its value is a list of key-value pairs.

<SecondLevelKey>

The second-level key name, which is set to one of the keys from the list assigned to `<TopLevelKey>`.

Return Value:

The value that is assigned to `<SecondLevelKey>`.

Example

The following example shows how to use Fn::FindInMap for a template with a `Mappings` section that contains a single map, `RegionMap`, that associates AMIs with AWS regions.

- The map has 5 top-level keys that correspond to various AWS regions.
- Each top-level key is assigned a list with two second level keys, “32” and “64”, that correspond to the AMI’s architecture.
- Each of the second-level keys is assigned an appropriate AMI name.

The example template contains an AWS::EC2::Instance resource whose ImageId property is set by the Fn::FindInMap function.

MapName is set to the map of interest, "RegionMap" in this example. TopLevelKey is set to the region where the stack is created, which is determined by using the "AWS::Region" pseudo parameter. SecondLevelKey is set to the desired architecture, "32" for this example.

Fn::FindInMap returns the AMI assigned to Fn::FindInMap. For a 32-bit instance in us-east-1, Fn::FindInMap would return "ami-6411e20d".

```
1 Mappings:
2   RegionMap:
3     us-east-1:
4       32: "ami-6411e20d"
5       64: "ami-7a11e213"
6     us-west-1:
7       32: "ami-c9c7978c"
8       64: "ami-cfc7978a"
9     eu-west-1:
10      32: "ami-37c2f643"
11      64: "ami-31c2f645"
12    ap-southeast-1:
13      32: "ami-66f28c34"
14      64: "ami-60f28c32"
15    ap-northeast-1:
16      32: "ami-9c03a89d"
17      64: "ami-a003a8a1"
18 Resources:
19   myEC2Instance:
20     Type: "AWS::EC2::Instance"
21     Properties:
22       ImageId: !FindInMap [ RegionMap, !Ref "AWS::Region", 32 ]
23       InstanceType: m1.small
```

Supported Functions

You can use the following functions in a Fn::FindInMap function:

- Fn::FindInMap
- Ref

Fn::GetAtt

The intrinsic function Fn::GetAtt returns the value of an attribute from a resource in the template.

Declaration

Syntax for the full function name:

```
"Fn::GetAtt": [ <logicalNameOfResource>, <attributeName> ]
```

Syntax for the short form:

```
!GetAtt <logicalNameOfResource>.<attributeName>
```

Parameters

<logicalNameOfResource>

The logical name of the resource that contains the attribute you want.

<attributeName>

The name of the resource-specific attribute whose value you want. See the resource's reference page <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html> for details about the attributes available for that resource type.

Return Value

The attribute value.

Example

This example returns a string containing the DNS name of the LoadBalancer with the logical name MyLB.

```
!GetAtt MyLB.DNSName
```

Supported Functions

For the Fn::GetAtt logical resource name, you cannot use any functions. You must specify a string that is a resource logical ID.

For the Fn::GetAtt attribute name, you can use the Ref function.

Attributes

To see the list of attributes that can be retrieved by Fn::GetAtt function go to <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-getatt.html>.

Fn::GetAZs

The intrinsic function Fn::GetAZs returns an array that lists Availability Zones for a specified region. Because customers have access to different Availability Zones, the intrinsic function Fn::GetAZs enables template authors to write templates that adapt to the calling user's access. That way you don't have to hard-code a full list of Availability Zones for a specified region.



For the EC2-Classic platform, the Fn::GetAZs function returns all Availability Zones for a region. For the EC2-VPC platform, the Fn::GetAZs function returns only Availability Zones that have a default subnet unless none of the Availability Zones has a default subnet; in that case, all Availability Zones are returned.

IAM permissions

The permissions that you need in order to use the Fn::GetAZs function depend on the platform in which you're launching Amazon EC2 instances. For both platforms, you need permissions to the Amazon EC2 `DescribeAvailabilityZones` and `DescribeAccountAttributes` actions. For EC2-VPC, you also need permissions to the Amazon EC2 `DescribeSubnets` action.

Declaration

Syntax for the full function name:

```
"Fn::GetAZs": <region>
```

Syntax for the short form:

```
!GetAZs <region>
```

Parameters

<region>

The name of the region for which you want to get the Availability Zones.

You can use the AWS::Region pseudo parameter to specify the region in which the stack is created. Specifying an empty string is equivalent to specifying AWS::Region.

Return Value

The list of Availability Zones for the region.

Examples

For these examples, AWS CloudFormation evaluates Fn::GetAZs to the following array; assuming that the user has created the stack in the us-east-1 region:

```
[ "us-east-1a", "us-east-1b", "us-east-1c" ]  
"Fn::GetAZs": ""  
"Fn::GetAZs": { Ref: "AWS::Region" }  
"Fn::GetAZs": "us-east-1"
```

Specify a Subnet's Availability Zone

The following example uses Fn::GetAZs to specify a subnet's Availability Zone:

```
1 mySubnet:  
2   Type: "AWS::EC2::Subnet"  
3   Properties:  
4     VpcId:  
5       !Ref VPC  
6     CidrBlock: 10.0.0.0/24  
7     AvailabilityZone:  
8       Fn::Select:  
9         - 0  
10        - Fn::GetAZs: ""
```

Supported Functions

You can use the Ref function in the Fn::GetAZs function.

Fn::ImportValue

The intrinsic function Fn::ImportValue returns the value of an output exported by another stack. You typically use this function to create cross-stack references.



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.

Declaration

You can use the full function name:

```
"Fn::ImportValue": <sharedValueToImport>
```

Alternatively, you can use the short form:

```
? ImportValue <sharedValueToImport>
```

Parameters

<sharedValueToImport>

The stack output value that you want to import.

Return Value

The stack output value.

Example

```
1 Fn::ImportValue:  
2   !Sub "${NetworkStackName}-SecurityGroupID"
```

Supported Functions

You can use the following functions in the Fn::ImportValue function. The value of these functions can't depend on a resource.

- Fn::Base64
- Fn::FindInMap
- Fn::If
- Fn::Join
- Fn::Select
- Fn::Sub
- Ref

Fn::Join

The intrinsic function Fn::Join appends a set of values into a single value, separated by the specified delimiter. If a delimiter is the empty string, the set of values are concatenated with no delimiter.

Declaration

Syntax for the full function name:

```
"Fn::Join": [ <delimiter>, [ <comma-delimited list of values> ] ]
```

Syntax for the short form:

```
!Join [ <delimiter>, [ <comma-delimited list of values> ] ]
```

Parameters

<delimiter>

The value you want to occur between fragments. The delimiter will occur between fragments only. It will not terminate the final value.

<ListOfValues>

The list of values you want combined.

Return Value

The combined string.

Example

The following example returns: “a:b:c”.

```
"Fn::Join": [ ":", [ a, b, c ] ]
```

Supported Functions

For the Fn::Join delimiter, you cannot use any functions. You must specify a string value.

For the Fn::Join list of values, you can use the following functions:

- Fn::Base64
- Fn::FindInMap
- Fn::GetAtt

- Fn::GetAZs
- Fn::If
- Fn::Join
- Fn::Select
- Ref

Fn::Select

The intrinsic function Fn::Select returns a single object from a list of objects by index.



Fn::Select does not check for null values or if the index is out of bounds of the array. Both conditions will result in a stack error, so you should be certain that the index you choose is valid, and that the list contains non-null values.

Declaration

Syntax for the full function name:

```
"Fn::Select": [ <index>, <listOfObjects> ]
```

Syntax for the short form:

```
!Select [ <index>, <listOfObjects> ]
```

Parameters

<index>

The index of the object to retrieve. This must be a value from zero to N-1, where N represents the number of elements in the array.

<listOfObjects>

The list of objects to select from. This list must not be null, nor can it have null entries.

Return Value

The selected object.

Examples

Basic Example

The following example returns: "grapes".

```
!Select [ "1", [ "apples", "grapes", "oranges", "mangoes" ] ]
```

Comma-delimited List Parameter Type

You can use Fn::Select to select an object from a CommaDelimitedList parameter. You might use a CommaDelimitedList parameter to combine the values of related parameters, which reduces the total number of parameters in your template. For example, the following parameter specifies a comma-delimited list of three CIDR blocks:

```
1 Parameters:  
2   DbSubnetIpBlocks:  
3     Description: "Comma-delimited list of three CIDR blocks"  
4     Type: CommaDelimitedList  
5     Default: "10.0.48.0/24, 10.0.112.0/24, 10.0.176.0/24"
```

To specify one of the three CIDR blocks, use `Fn::Select` in the Resources section of the same template, as shown in the following sample snippet:

```
1 Subnet0:  
2   Type: "AWS::EC2::Subnet"  
3   Properties:  
4     VpcId: !Ref VPC  
5     CidrBlock: !Select [ 0, !Ref DbSubnetIpBlocks ]
```

Supported Functions

For the `Fn::Select` index value, you can use the `Ref` and `Fn::FindInMap` functions.

For the `Fn::Select` list of objects, you can use the following functions:

- `Fn::FindInMap`
- `Fn::GetAtt`
- `Fn::GetAZs`
- `Fn::If`
- `Ref`

Fn::Sub

The intrinsic function Fn::Sub substitutes variables in an input string with values that you specify. In your templates, you can use this function to construct commands or outputs that include values that aren't available until you create or update a stack.

Declaration

The following sections show the function's syntax.

Syntax for the full function name:

```
1 "Fn::Sub":  
2   - <String>  
3   - { <Var1Name>: <Var1Value>, <Var2Name>: <Var2Value> }
```

Syntax for the short form:

```
1 !Sub  
2   - <String>  
3   - { <Var1Name>: <Var1Value>, <Var2Name>: <Var2Value> }
```

If you're substituting only template parameters, resource logical IDs, or resource attributes in the *String* parameter, don't specify a variable map.

Syntax for the full function name:

```
"Fn::Sub": <String>
```

Syntax for the short form:

```
!Sub <String>
```

Parameters

<String>

A string with variables that AWS CloudFormation substitutes with their associated values at runtime. Write variables as \${*MyVarName*}. Variables can be template parameter names, resource logical IDs, resource attributes, or a variable in a key-value map. If you specify only template parameter names, resource logical IDs, and resource attributes, don't specify a key-value map.

If you specify template parameter names or resource logical IDs, such as \${InstanceTypeParameter}, AWS CloudFormation returns the same values as if you used the Ref intrinsic function. If you specify resource attributes, such as \${MyInstance.PublicIp}, AWS CloudFormation returns the same values as if you used the Fn::GetAtt intrinsic function.

To write a dollar sign and curly braces (\${}) literally, add an exclamation point (!) after the open curly brace, such as \${!Literal}. AWS CloudFormation resolves this text as \${Literal}.

<VarName>

The name of a variable that you included in the <String> parameter.

<VarValue>

The value that AWS CloudFormation substitutes for the associated variable name at runtime.

Return Value

AWS CloudFormation returns the original string, substituting the values for all of the variables.

Examples

The following examples demonstrate how to use the Fn::Sub function.

UserData Commands

The following example uses Fn::Sub to substitute the AWS CloudFormation stack name and the AWS region pseudo parameters for the actual stack name and region at runtime.

```
1 UserData:  
2   "Fn::Base64":  
3     !Sub |  
4       #!/bin/bash -xe  
5       yum update -y aws-cfn-bootstrap  
6       /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource LaunchConfig\  
7       --configsets wordpress_install --region ${AWS::Region}  
8       /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource WebServ\  
9       erGroup --region ${AWS::Region}
```

Fn::Sub with a Mapping

The following example uses a mapping to substitute the \${Domain} variable with the resulting value from the Ref function.

```
1 Name: !Sub
2   - www.${Domain}
3   - { Domain: !Ref RootDomainName }
```

Supported Functions

You can use the following functions in the Fn::Sub function:

- Fn::Base64
- Fn::FindInMap
- Fn::GetAtt
- Fn::GetAZs
- Fn::If
- Fn::Join
- Fn::Select
- Ref

Ref

The intrinsic function `Ref` returns the value of the specified parameter or resource.

- When you specify a parameter's logical name, it returns the value of the parameter.
- When you specify a resource's logical name, it returns a value that you can typically use to refer to that resource, such as a physical ID.

When you are declaring a resource in a template and you need to specify another template resource by name, you can use the `Ref` to refer to that other resource. In general, `Ref` returns the name of the resource. For example, a reference to an `AWS::AutoScaling::AutoScalingGroup` returns the name of that Auto Scaling group resource.

For some resources, an identifier is returned that has another significant meaning in the context of the resource. An `AWS::EC2::EIP` resource, for instance, returns the IP address, and an `AWS::EC2::Instance` returns the instance ID.



You can also use `Ref` to add values to Output messages.

Declaration

Syntax for the full function name:

```
Ref: <logicalName>
```

Syntax for the short form:

```
!Ref <logicalName>
```

Parameters

`<logicalName>`

The logical name of the resource or parameter you want to dereference.

Return Value

The physical ID of the resource or the value of the parameter.

Example

The following resource declaration for an Elastic IP address needs the instance ID of an EC2 instance and uses the `Ref` function to specify the instance ID of the `MyEC2Instance` resource:

```
1 MyEIP:  
2   Type: "AWS::EC2::EIP"  
3   Properties:  
4     InstanceId: !Ref MyEC2Instance
```

Supported Functions

You cannot use any functions in the `Ref` function. You must specify a string that is a resource logical ID.

Resource Return

To see the list of sample values returned by `Ref` for particular AWS CloudFormation resources go to <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-ref.html>. For more information about `Ref` return values for a particular resource or property, refer to the documentation for that resource or property <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html>.

Appendix D: Resource Attribute

You can add `Resource Attribute` to a resource to control additional behaviors and relationships.⁶⁶

CreationPolicy Attribute

Associate the `CreationPolicy` attribute with a resource to prevent its status from reaching `create complete` until AWS CloudFormation receives a specified number of success signals or the timeout period is exceeded. To signal a resource, you can use the `cfn-signal` helper script or `SignalResource` API. AWS CloudFormation publishes valid signals to the stack events so that you track the number of signals sent.

The creation policy is invoked only when AWS CloudFormation creates the associated resource. Currently, the only AWS CloudFormation resources that support creation policies are:

- `AWS::AutoScaling::AutoScalingGroup`
- `AWS::EC2::Instance`
- `AWS::CloudFormation::WaitCondition`

Use the `CreationPolicy` attribute when you want to wait on resource configuration actions before stack creation proceeds. For example, if you install and configure software applications on an EC2 instance, you might want those applications to be running before proceeding. In such cases, you can add a `CreationPolicy` attribute to the instance, and then send a success signal to the instance after the applications are installed and configured. For a detailed example, see <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/deploying.applications.html>.

Syntax

⁶⁶<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-product-attribute-reference.html>

```
1 CreationPolicy:  
2   AutoScalingCreationPolicy:  
3     MinSuccessfulInstancesPercent: <Integer>  
4   ResourceSignal:  
5     Count: <Integer>  
6     Timeout: <String>
```

Properties

AutoScalingCreationPolicy

For an Auto Scaling group *replacement update*, specifies how many instances must signal success for the update to succeed.

- **MinSuccessfulInstancesPercent**

Specifies the percentage of instances in an Auto Scaling replacement update that must signal success for the update to succeed. You can specify a value from 0 to 100. AWS CloudFormation rounds to the nearest tenth of a percent. For example, if you update five instances with a minimum successful percentage of 50, three instances must signal success.

If an instance doesn't send a signal within the time specified by the **Timeout** property, AWS CloudFormation assumes that the instance wasn't created.

Default: 100

Type: Integer

Required: No

ResourceSignal

When AWS CloudFormation creates the associated resource, configures the number of required success signals and the length of time that AWS CloudFormation waits for those signals.

- **Count**

The number of success signals AWS CloudFormation must receive before it sets the resource status as CREATE_COMPLETE. If the resource receives a failure signal or doesn't receive the specified number of signals before the timeout period expires, the resource creation fails and AWS CloudFormation rolls the stack back.

Default: 1

Type: Integer

Required: No

- **Timeout**

The length of time that AWS CloudFormation waits for the number of signals that was specified in the **Count** property. The timeout period starts after AWS CloudFormation starts creating the resource, and the timeout expires no sooner than the time you specify

but can occur shortly thereafter. The maximum time that you can specify is 12 hours. The value must be in ISO8601 *duration format*⁶⁷, in the form: "PT#H#M#S", where each # is the number of hours, minutes, and seconds, respectively. For best results, specify a period of time that gives your instances plenty of time to get up and running. A shorter timeout can cause a rollback.

Default: PT5M (5 minutes)

Type: String

Required: No

Examples

Auto Scaling Group

The following example shows how to add a creation policy to an Auto Scaling group. The creation policy requires three success signals and times out after 15 minutes.

```
1 AutoScalingGroup:
2   Type: AWS::AutoScaling::AutoScalingGroup
3   Properties:
4     AvailabilityZones:
5       Fn::GetAZs: ''
6     LaunchConfigurationName:
7       Ref: LaunchConfig
8     DesiredCapacity: '3'
9     MinSize: '1'
10    MaxSize: '4'
11    CreationPolicy:
12      ResourceSignal:
13        Count: '3'
14      Timeout: PT15M
15    UpdatePolicy:
16      AutoScalingScheduledAction:
17        IgnoreUnmodifiedGroupSizeProperties: 'true'
18      AutoScalingRollingUpdate:
19        MinInstancesInService: '1'
20        MaxBatchSize: '2'
21        PauseTime: PT1M
22        WaitOnResourceSignals: 'true'
23    LaunchConfig:
24      Type: AWS::AutoScaling::LaunchConfiguration
25      Properties:
```

⁶⁷http://en.wikipedia.org/wiki/ISO_8601#Durations

```
26     ImageId: ami-16d18a7e
27     InstanceType: t2.micro
28     UserData:
29         "Fn::Base64":
30             !Sub |
31                 #!/bin/bash -xe
32                 yum update -y aws-cfn-bootstrap
33                 /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} \
34                 --resource AutoScalingGroup --region ${AWS::Region}
```

WaitCondition

The following example shows how to add a creation policy to a wait condition.

```
1 WaitCondition:
2   Type: AWS::CloudFormation::WaitCondition
3   CreationPolicy:
4     ResourceSignal:
5       Timeout: PT15M
6       Count: 5
```

DeletionPolicy Attribute

With the `DeletionPolicy` attribute you can preserve or (in some cases) backup a resource when its stack is deleted. You specify a `DeletionPolicy` attribute for each resource that you want to control. If a resource has no `DeletionPolicy` attribute, AWS CloudFormation deletes the resource by default.

To keep a resource when its stack is deleted, specify `Retain` for that resource. You can use `retain` for any resource. For example, you can retain a nested stack, S3 bucket, or EC2 instance so that you can continue to use or modify those resources after you delete their stacks.



If you want to modify resources outside of AWS CloudFormation, use a retain policy and then delete the stack. Otherwise, your resources might get out of sync with your AWS CloudFormation template and cause stack errors.

For resources that support snapshots, such as `AWS::RDS::DBInstance` and `AWS::EC2::Volume`, you can specify `Snapshot` to have AWS CloudFormation create a snapshot before deleting the resource.

The following snippet contains an Amazon S3 bucket resource with a `Retain` deletion policy. When this stack is deleted, AWS CloudFormation leaves the bucket without deleting it.

```
1 AWSTemplateFormatVersion: '2010-09-09'  
2 Resources:  
3   myS3Bucket:  
4     Type: AWS::S3::Bucket  
5     DeletionPolicy: Retain
```

DeletionPolicy Options

Delete

AWS CloudFormation deletes the resource and all its content if applicable during stack deletion. You can add this deletion policy to any resource type. By default, if you don't specify a `DeletionPolicy`, AWS CloudFormation deletes your resources. For Amazon S3 buckets, you must delete all objects in the bucket for deletion to succeed.

Retain

AWS CloudFormation keeps the resource without deleting the resource or its contents when its stack is deleted. You can add this deletion policy to any resource type. Note that when AWS CloudFormation completes the stack deletion, the stack will be in `Delete_Complete` state; however, resources that are retained continue to exist and continue to incur applicable charges until you delete those resources.

Snapshot

For resources that support snapshots (`AWS::EC2::Volume`, `AWS::RDS::DBInstance`, `AWS::RDS::DBCluster`, and `AWS::Redshift::Cluster`), AWS CloudFormation creates a snapshot for the resource before deleting it. Note that when AWS CloudFormation completes the stack deletion, the stack will be in the `Delete_Complete` state; however, the snapshots that are created with this policy continue to exist and continue to incur applicable charges until you delete those snapshots.

DependsOn Attribute

With the DependsOn attribute you can specify that the creation of a specific resource follows another. When you add a DependsOn attribute to a resource, that resource is created only after the creation of the resource specified in the DependsOn attribute. You can use the DependsOn attribute with any resource. Here are some typical uses:

- Determine when a wait condition goes into effect. For more information, see <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-waitcondition.html>.
- Declare dependencies for resources that must be created or deleted in a specific order. For example, you must explicitly declare dependencies on gateway attachments for some resources in a VPC.
- Override default parallelism when creating, updating, or deleting resources. AWS CloudFormation creates, updates, and deletes resources in parallel to the extent possible. It automatically determines which resources in a template can be parallelized and which have dependencies that require other operations to finish first. You can use DependsOn to explicitly specify dependencies, which overrides the default parallelism and directs CloudFormation to operate on those resources in a specified order.



During a stack update, resources that depend on updated resources are automatically updated. AWS CloudFormation makes no changes to the automatically updated resources, but if a stack policy is associated with those resources, you must be permitted to update them.

Syntax

The DependsOn attribute can take a single string or list of strings.

```
DependsOn: [ String, ... ]
```

Example

The following template contains an AWS::EC2::Instance resource with a DependsOn attribute that specifies myDB, an AWS::RDS::DBInstance. When AWS CloudFormation creates this stack, it first creates myDB, then creates Ec2Instance.

```
1 AWSTemplateFormatVersion: '2010-09-09'
2 Mappings:
3   RegionMap:
4     us-east-1:
5       AMI: ami-76f0061f
6     us-west-1:
7       AMI: ami-655a0a20
8     eu-west-1:
9       AMI: ami-7fd4e10b
10    ap-northeast-1:
11      AMI: ami-8e08a38f
12    ap-southeast-1:
13      AMI: ami-72621c20
14 Resources:
15   Ec2Instance:
16     Type: AWS::EC2::Instance
17     Properties:
18       ImageId:
19         Fn::FindInMap:
20           - RegionMap
21           - Ref: AWS::Region
22           - AMI
23     DependsOn: myDB
24   myDB:
25     Type: AWS::RDS::DBInstance
26     Properties:
27       AllocatedStorage: '5'
28       DBInstanceClass: db.m1.small
29       Engine: MySQL
30       EngineVersion: '5.5'
31       MasterUsername: MyName
32       MasterUserPassword: MyPassword
```

When a DependsOn attribute is required

VPC-gateway attachment

Some resources in a VPC require a gateway (either an Internet or VPN gateway). If your AWS CloudFormation template defines a VPC, a gateway, and a gateway attachment, any resources that require the gateway are dependent on the gateway attachment. For example, an Amazon EC2 instance with a public IP address is dependent on the VPC-gateway attachment if the VPC and InternetGateway resources are also declared in the same template.

Currently, the following resources depend on a VPC-gateway attachment when they have an associated public IP address and are in a VPC:

- Auto Scaling groups
- Amazon EC2 instances
- Elastic Load Balancing load balancers
- Elastic IP addresses
- Amazon RDS database instances
- Amazon VPC routes that include the Internet gateway

A VPN gateway route propagation depends on a VPC-gateway attachment when you have a VPN gateway

The following snippet shows a sample gateway attachment and an Amazon EC2 instance that depends on a gateway attachment:

```
1 GatewayToInternet:  
2   Type: AWS::EC2::VPCGatewayAttachment  
3   Properties:  
4     VpcId:  
5       Ref: VPC  
6     InternetGatewayId:  
7       Ref: InternetGateway  
8 EC2Host:  
9   Type: AWS::EC2::Instance  
10  DependsOn: GatewayToInternet  
11  Properties:  
12    InstanceType:  
13      Ref: EC2InstanceType  
14    KeyName:  
15      Ref: KeyName  
16    ImageId:  
17      Fn::FindInMap:  
18        - AWSRegionArch2AMI  
19        - Ref: AWS::Region  
20        - Fn::FindInMap:  
21          - AWSServiceType2Arch  
22          - Ref: EC2InstanceType  
23          - Arch  
24    NetworkInterfaces:  
25      - GroupSet:  
26        - Ref: EC2SecurityGroup
```

```
27     AssociatePublicIpAddress: 'true'
28     DeviceIndex: '0'
29     DeleteOnTermination: 'true'
30     SubnetId:
31       Ref: PublicSubnet
```

Amazon ECS Service and Auto Scaling Group

When you use Auto Scaling or Amazon Elastic Compute Cloud (Amazon EC2) to create container instances for an Amazon ECS cluster, the Amazon ECS service resource must have a dependency on the Auto Scaling group or Amazon EC2 instances, as shown in the following snippet. That way the container instances are available and associated with the Amazon ECS cluster before AWS CloudFormation creates the Amazon ECS service.

```
1 service:
2   Type: AWS::ECS::Service
3   DependsOn:
4     - ECSAutoScalingGroup
5   Properties:
6     Cluster:
7       Ref: ECSCluster
8     DesiredCount: 1
9     LoadBalancers:
10    - ContainerName: simple-app
11      ContainerPort: 80
12      LoadBalancerName:
13        Ref: EcsElasticLoadBalancer
14      Role:
15        Ref: ECSServiceRole
16      TaskDefinition:
17        Ref: taskdefinition
```

IAM Role Policy

Resources that make additional calls to AWS require a service role, which permits a service to make calls to AWS on your behalf. For example, the `AWS::CodeDeploy::DeploymentGroup` resource requires a service role so that AWS CodeDeploy has permissions to deploy applications to your instances. When you have a single template that defines a service role, the role's policy (by using the `AWS::IAM::Policy` or `AWS::IAM::ManagedPolicy` resource), and a resource that uses the role, add a dependency so that the resource depends on the role's policy. This dependency ensures that the policy is available throughout the resource's lifecycle.

For example, imagine that you have a template with a deployment group resource, a service role, and the role's policy. When you create a stack, AWS CloudFormation won't create the deployment

group until it creates the role's policy. Without the dependency, AWS CloudFormation can create the deployment group resource before it creates the role's policy. If that happens, the deployment group will fail to create because of insufficient permissions.

If the role has an embedded policy, don't specify a dependency. AWS CloudFormation creates the role and its policy at the same time.

Metadata Attribute

The `Metadata` attribute enables you to associate structured data with a resource. By adding a `Metadata` attribute to a resource, you can add data in JSON or YAML to the resource declaration. In addition, you can use intrinsic functions (such as `GetAtt` and `Ref`), parameters, and pseudo parameters within the `Metadata` attribute to add those interpreted values.



AWS CloudFormation does not validate the syntax within the `Metadata` attribute.

You can retrieve this data using the AWS command `aws cloudformation describe-stack-resource` or the `DescribeStackResource` action.

Example

The following template contains an Amazon S3 bucket resource with a `Metadata` attribute.

```
1 AWSTemplateFormatVersion: '2010-09-09'
2 Resources:
3   MyS3Bucket:
4     Type: AWS::S3::Bucket
5     Metadata:
6       Object1: Location1
7       Object2: Location2
```

UpdatePolicy Attribute

Use the `UpdatePolicy` attribute to specify how AWS CloudFormation handles updates to the `AWS::AutoScaling::AutoScalingGroup` resource. AWS CloudFormation invokes one of three update policies depending on the type of change you make or whether a scheduled action is associated with the Auto Scaling group.

- The `AutoScalingReplacingUpdate` and `AutoScalingRollingUpdate` policies apply when you make the following changes:
 - The Auto Scaling group's `AWS::AutoScaling::LaunchConfiguration`.
 - The Auto Scaling group's `VPCZoneIdentifier` property
 - Updating an Auto Scaling group that contains instances that don't match the current `LaunchConfiguration`.
- If both the `AutoScalingReplacingUpdate` and `AutoScalingRollingUpdate` policies are specified, setting the `WillReplace` property to true gives `AutoScalingReplacingUpdate` precedence.
- The `AutoScalingScheduledAction` policy applies when you update a stack that includes an Auto Scaling group with an associated scheduled action.

AutoScalingReplacingUpdate Policy

To specify how AWS CloudFormation handles replacement updates for an Auto Scaling group, use the `AutoScalingReplacingUpdate` policy. This policy enables you to specify whether AWS CloudFormation replaces an Auto Scaling group with a new one or replaces only the instances in the Auto Scaling group.



Before attempting an update, ensure that you have sufficient Amazon EC2 capacity for both your old and new Auto Scaling groups.

Syntax

```
1 UpdatePolicy:  
2   AutoScalingReplacingUpdate:  
3     WillReplace: <Boolean>
```

Properties

WillReplace

Specifies whether an Auto Scaling group and the instances it contains are replaced during an update. During replacement, AWS CloudFormation retains the old group until it finishes

creating the new one. If the update fails, AWS CloudFormation can roll back to the old Auto Scaling group and delete the new Auto Scaling group.

While AWS CloudFormation creates the new group, it doesn't detach or attach any instances. After successfully creating the new Auto Scaling group, AWS CloudFormation deletes the old Auto Scaling group during the cleanup process.

When you set the `WillReplace` parameter, remember to specify a matching `CreationPolicy`. If the minimum number of instances (specified by the `MinSuccessfulInstancesPercent` property) don't signal success within the `Timeout` period (specified in the `CreationPolicy` policy), the replacement update fails and AWS CloudFormation rolls back to the old Auto Scaling group.

Type: Boolean

Required: No

AutoScalingRollingUpdate Policy

To specify how AWS CloudFormation handles rolling updates for an Auto Scaling group, use the `AutoScalingRollingUpdate` policy. Rolling updates enable you to specify whether AWS CloudFormation updates instances that are in an Auto Scaling group in batches or all at once.



If you have an Auto Scaling group with rolling updates and scheduled actions enabled, you must suspend the scheduled actions before you can update the group. Use the `SuspendProcesses` property to do this.

Syntax

```
1 UpdatePolicy:  
2   AutoScalingRollingUpdate:  
3     MaxBatchSize: <Integer>  
4     MinInstancesInService: <Integer>  
5     MinSuccessfulInstancesPercent: <Integer>  
6     PauseTime: <String>  
7     SuspendProcesses:  
8       - <List of processes>  
9     WaitOnResourceSignals: <Boolean>
```

Properties

MaxBatchSize

Specifies the maximum number of instances that AWS CloudFormation updates.

Default: 1

Type: Integer

Required: No

MinInstancesInService

Specifies the minimum number of instances that must be in service within the Auto Scaling group while AWS CloudFormation updates old instances.

Default: 0

Type: Integer

Required: No

MinSuccessfulInstancesPercent

Specifies the percentage of instances in an Auto Scaling rolling update that must signal success for an update to succeed. You can specify a value from 0 to 100. AWS CloudFormation rounds to the nearest tenth of a percent. For example, if you update five instances with a minimum successful percentage of 50, three instances must signal success.

If an instance doesn't send a signal within the time specified in the `PauseTime` property, AWS CloudFormation assumes that the instance wasn't updated.

If you specify this property, you must also enable the `WaitOnResourceSignals` and `PauseTime` properties.

Default: 100

Type: Integer

Required: No

PauseTime

The amount of time that AWS CloudFormation pauses after making a change to a batch of instances to give those instances time to start software applications. For example, you might need to specify `PauseTime` when scaling up the number of instances in an Auto Scaling group.

If you enable the `WaitOnResourceSignals` property, `PauseTime` is the amount of time that AWS CloudFormation should wait for the Auto Scaling group to receive the required number of valid signals from added or replaced instances. If the `PauseTime` is exceeded before the Auto Scaling group receives the required number of signals, the update fails. For best results, specify a time period that gives your applications sufficient time to get started. If the update needs to be rolled back, a short `PauseTime` can cause the rollback to fail.

Specify `PauseTime` in the ISO8601 duration format (in the format PT#H#M#S, where each # is the number of hours, minutes, and seconds, respectively). The maximum `PauseTime` is one hour (PT1H).

Default: PT0S (zero seconds). If the `WaitOnResourceSignals` property is set to true, the default is PT5M.

Type: String

Required: No

SuspendProcesses

Specifies the Auto Scaling processes to suspend during a stack update. Suspending processes prevents Auto Scaling from interfering with a stack update. For example, you can suspend alarming so that Auto Scaling doesn't execute scaling policies associated with an alarm. For

valid values, see the `ScalingProcesses.member.N` parameter for the `SuspendProcesses` action in the *Auto Scaling API Reference*.

Default: Not specified

Type: List of Auto Scaling processes

Required: No

WaitOnResourceSignals

Specifies whether the Auto Scaling group waits on signals from new instances during an update. Use this property to ensure that instances have completed installing and configuring applications before the Auto Scaling group update proceeds. AWS CloudFormation suspends the update of an Auto Scaling group after new EC2 instances are launched into the group. AWS CloudFormation must receive a signal from each new instance within the specified `PauseTime` before continuing the update. To signal the Auto Scaling group, use the `cfn-signal` helper script or `SignalResource` API.

Default: false

Type: Boolean

Required: Conditional. If you specify the `MinSuccessfulInstancesPercent` property, you must also enable the `WaitOnResourceSignals` and `PauseTime` properties.

AutoScalingScheduledAction Policy

To specify how AWS CloudFormation handles updates for the `MinSize`, `MaxSize`, and `DesiredCapacity` properties when the `AWS::AutoScaling::AutoScalingGroup` resource has an associated scheduled action, use the `AutoScalingScheduledAction` policy.

With scheduled actions, the group size properties of an Auto Scaling group can change at any time. When you update a stack with an Auto Scaling group and scheduled action, AWS CloudFormation always sets the group size property values of your Auto Scaling group to the values that are defined in the `AWS::AutoScaling::AutoScalingGroup` resource of your template, even if a scheduled action is in effect.

If you do not want AWS CloudFormation to change any of the group size property values when you have a scheduled action in effect, use the `AutoScalingScheduledAction` update policy to prevent AWS CloudFormation from changing the `MinSize`, `MaxSize`, or `DesiredCapacity` properties unless you have modified these values in your template.

Syntax

```
1 UpdatePolicy:  
2   AutoScalingScheduledAction:  
3     IgnoreUnmodifiedGroupSizeProperties: <Boolean>
```

Properties

IgnoreUnmodifiedGroupSizeProperties

Specifies whether AWS CloudFormation ignores differences in group size properties between your current Auto Scaling group and the Auto Scaling group described in the `AWS::AutoScaling::AutoScalingGroup` resource of your template during a stack update. If you modify any of the group size property values in your template, AWS CloudFormation uses the modified values and updates your Auto Scaling group.

Default: false

Type: Boolean

Required: No

Examples

The following examples show how to add an update policy to an Auto Scaling group and how to maintain availability when updating metadata.

Add an UpdatePolicy to an Auto Scaling Group

The following example shows how to add an update policy. During an update, the Auto Scaling group updates instances in batches of two and keeps a minimum of one instance in service. Because the `WaitOnResourceSignals` flag is set, the Auto Scaling group waits for new instances that are added to the group. The new instances must signal the Auto Scaling group before it updates the next batch of instances.

```
1 ASG:
2   Type: 'AWS :: AutoScaling :: AutoScalingGroup'
3   Properties:
4     AvailabilityZones:
5       - us-east-1a
6       - us-east-1b
7     DesiredCapacity: '1'
8     LaunchConfigurationName:
9       Ref: LaunchConfig
10    MaxSize: '4'
11    MinSize: '1'
12    UpdatePolicy:
13      AutoScalingScheduledAction:
14        IgnoreUnmodifiedGroupSizeProperties: 'true'
15      AutoScalingRollingUpdate:
16        MinInstancesInService: '1'
17        MaxBatchSize: '2'
```

```
18     WaitOnResourceSignals: 'true'
19     PauseTime: PT10M
20 ScheduledAction:
21     Type: 'AWS : : AutoScaling : : ScheduledAction'
22     Properties:
23         AutoScalingGroupName:
24             Ref: ASG
25         DesiredCapacity: '2'
26         StartTime: '2017-06-02T20 : 00 : 00Z'
```

AutoScalingReplacingUpdate Policy

The following example declares a policy that forces an associated Auto Scaling group to be replaced during an update. For the update to succeed, a percentage of instances (specified by the `MinSuccessfulPercentParameter` parameter) must signal success within the `Timeout` period.

```
1 UpdatePolicy:
2     AutoScalingReplacingUpdate:
3         WillReplace: 'true'
4     CreationPolicy:
5         ResourceSignal:
6             Count:
7                 Ref: ResourceSignalsOnCreate
8                 Timeout: PT10M
9     AutoScalingCreationPolicy:
10        MinSuccessfulInstancesPercent:
11            Ref: MinSuccessfulPercentParameter
```

Maintain Availability When Updating the Metadata for the cfn-init Helper Script

When you install applications on your instances, you might use the `AWS::CloudFormation::Init` metadata key and the `cfn-init` helper script to bootstrap the instances in your Auto Scaling group. AWS CloudFormation installs the packages, runs the commands, and performs other bootstrapping actions described in the metadata.

When you update only the metadata (for example, when updating a package to another version), you can use the `cfn-hup` helper daemon to detect and apply the updates. However, the `cfn-hup` daemon runs independently on each instance. If the daemon happens to run at the same time on all instances, your application or service might be unavailable during the update. To guarantee availability, you can force a rolling update so that AWS CloudFormation updates your instances one batch at a time.



Forcing a rolling update requires AWS CloudFormation to create a new instance and then delete the old one. Any information stored on the old instance is lost.

To force a rolling update, change the logical ID of the launch configuration resource, and then update the stack and any references pointing to the original logic ID (such as the associated Auto Scaling group). AWS CloudFormation triggers a rolling update on the Auto Scaling group, replacing all instances.

Original Template

```
1 LaunchConfig:  
2   Type: AWS::AutoScaling::LaunchConfiguration  
3   Metadata:  
4     Comment: Install a simple PHP application  
5     AWS::CloudFormation::Init :  
6     ...
```

Updated Logical ID

```
1 LaunchConfigUpdateRubygemsPkg:  
2   Type: AWS::AutoScaling::LaunchConfiguration  
3   Metadata:  
4     Comment: Install a simple PHP application  
5     AWS::CloudFormation::Init:  
6     ...
```

Appendix E: Pseudo Parameters

Pseudo Parameters are parameters that are predefined by AWS CloudFormation. You do not declare them in your template. Use them the same way as you would a parameter, as the argument for the `Ref` function.⁶⁸

Example

The following snippet assigns the value of the `AWS::Region` pseudo parameter to an output value:

```
1 Outputs:  
2   MyStacksRegion:  
3     Value: !Ref "AWS::Region"
```

AWS::AccountId

Returns the AWS account ID of the account in which the stack is being created, such as 123456789012.

AWS::NotificationARNs

Returns the list of notification Amazon Resource Names (ARNs) for the current stack.

To get a single ARN from the list, use `Fn::Select`.

```
1 myASGrpOne:  
2   Type: AWS::AutoScaling::AutoScalingGroup  
3   Version: '2009-05-15'  
4   Properties:  
5     AvailabilityZones:  
6       - "us-east-1a"  
7     LaunchConfigurationName:  
8       Ref: MyLaunchConfiguration  
9     MinSize: '0'  
10    MaxSize: '0'  
11    NotificationConfigurations:
```

⁶⁸<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/pseudo-parameter-reference.html>

```

12     - TopicARN:
13         Fn::Select:
14             - '0'
15             - Ref: AWS::NotificationARNs
16     NotificationTypes:
17         - autoscaling:EC2_INSTANCE_LAUNCH
18         - autoscaling:EC2_INSTANCE_LAUNCH_ERROR

```

AWS::NoValue

Removes the corresponding resource property when specified as a return value in the Fn::If intrinsic function.

For example, you can use the AWS::NoValue parameter when you want to use a snapshot for an Amazon RDS DB instance only if a snapshot ID is provided. If the UseDBSnapshot condition evaluates to true, AWS CloudFormation uses the DBSnapshotName parameter value for the DBSnapshotIdentifier property. If the condition evaluates to false, AWS CloudFormation removes the DBSnapshotIdentifier property.

```

1 MyDB:
2   Type: AWS::RDS::DBInstance
3   Properties:
4     AllocatedStorage: '5'
5     DBInstanceClass: db.m1.small
6     Engine: MySQL
7     EngineVersion: '5.5'
8     MasterUsername:
9       Ref: DBUser
10    MasterUserPassword:
11      Ref: DBPassword
12    DBParameterGroupName:
13      Ref: MyRDSPParamGroup
14    DBSnapshotIdentifier:
15      Fn::If:
16          - UseDBSnapshot
17          - Ref: DBSnapshotName
18          - Ref: AWS::NoValue

```

AWS::Region

Returns a string representing the AWS Region in which the encompassing resource is being created, such as us-west-2.

AWS::StackId

Returns the ID of the stack as specified with the `aws cloudformation create-stack` command, such as `arn:aws:cloudformation:us-west-2:123456789012:stack/teststack/51af3dc0-da77-11e4-872e-1234567db123`.

AWS::StackName

Returns the name of the stack as specified with the `aws cloudformation create-stack` command, such as `teststack`.

Appendix F: Helper Scripts

AWS CloudFormation provides a set of Python helper scripts that you can use to install software and start services on an Amazon EC2 instance that you create as part of your stack. You can call the helper scripts directly from your template. The scripts work in conjunction with resource metadata that you define in the same template. The helper scripts run on the Amazon EC2 instance as part of the stack creation process.⁶⁹

The helper scripts are pre-installed on the latest versions of the Amazon Linux AMI. The helper scripts are also available from the Amazon Linux yum repository for use with other UNIX/Linux AMIs.

Currently, AWS CloudFormation provides the following helpers:

- `cfn-init`: Used to retrieve and interpret the resource metadata, installing packages, creating files and starting services.
- `cfn-signal`: A simple wrapper to signal an AWS CloudFormation `CreationPolicy` or `WaitCondition`, enabling you to synchronize other resources in the stack with the application being ready.
- `cfn-get-metadata`: A wrapper script making it easy to retrieve either all metadata defined for a resource or path to a specific key or subtree of the resource metadata.
- `cfn-hup`: A daemon to check for updates to metadata and execute custom hooks when the changes are detected.

These scripts are installed by default on the latest Amazon Linux AMI in `/opt/aws/bin`. They are also available in the Amazon Linux AMI yum repository for previous versions of the Amazon Linux AMI as well as via RPM for other Linux/Unix distributions. You can also install the scripts on Microsoft Windows (2008 or later) by using Python for Windows.

The scripts are not executed by default. You must include calls to execute specific helper scripts.

The AWS CloudFormation helper scripts are available from the following locations:

- The latest version of the Amazon Linux AMI has the AWS CloudFormation helper scripts installed by default in `/opt/aws/bin`.
- The AWS helper scripts are available in the Amazon Linux AMI yum repository (the package name is `aws-cfn-bootstrap`) for previous versions of the Amazon Linux AMI.
- The helpers are also available in other formats:

⁶⁹<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-helper-scripts-reference.html>

- <https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.amzn1.noarch.rpm>
- <https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz> to install the helper scripts via the Python easy-install tools. For Ubuntu, to complete installation, you must create a symlink: `ln -s /root/aws-cfn-bootstrap-latest/init/ubuntu/cfn-hup /etc/init.d/cfn-hup`.
- <https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.zip>
- 32 bit: <https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.msi> or 64 bit: <https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-win64-latest.msi> for installation on Microsoft Windows.
- The source for the scripts is available at <https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.src.rpm>, which can be used for Linux distributions other than the Amazon Linux AMI.

The helper scripts are updated periodically. If you use the helper scripts, ensure your launched instances are using the latest version of the scripts:

- Include the following command in the `UserData` property of your template before you call the scripts. This command ensures that you get the latest version:
`yum install -y aws-cfn-bootstrap`
- If you don't include the `yum install` command and you use the `cfn-init`, `cfn-signal`, or `cfn-get-metadata` scripts, then you'll need to manually update the scripts in each Amazon EC2 Linux instance using this command:
`sudo yum install -y aws-cfn-bootstrap`
- If you don't include the `yum install` command and you use the `cfn-hup` script, then you'll need to manually update the script in each Amazon EC2 Linux instance using these commands:
`sudo yum install -y aws-cfn-bootstrap`
`sudo /sbin/service cfn-hup restart`
- If you use the source code for the scripts to work with another version of Linux or a different platform, and you have created your own certificate trust store, you'll also need to keep the trust store updated.

cfn-init

Description

The `cfn-init` helper script reads template metadata from the `AWS::CloudFormation::Init` key and acts accordingly to:

- Fetch and parse metadata from CloudFormation
- Install packages
- Write files to disk
- Enable/disable and start/stop services



If you use `cfn-init` to update an existing file, it creates a backup copy of the original file in the same directory with a `.bak` extension. For example, if you update `/path/to/file_name`, the action produces two files: `/path/to/file_name.bak` contains the original file's contents and `/path/to/file_name` contains the updated contents.



`cfn-init` does not require credentials, so you do not need to use the `--access-key`, `--secret-key`, `--role`, or `--credential-file` options.

Syntax

```
cfn-init --stack|-s stack.name.or.id \
          --resource|-r logical.resource.id \
          --region region \
          --access-key access.key \
          --secret-key secret.key \
          --role rolename \
          --credential-file|-f credential.file \
          --configsets|-c config.sets \
          --url|-u service.url \
          --http-proxy HTTP.proxy \
          --https-proxy HTTPS.proxy \
          --verbose|-v
```

Options

Name	Description	Required
-s, --stack	Name of the Stack. <i>Type:</i> String <i>Default:</i> None <i>Example:</i> -s { "Ref" : "AWS::StackName" },	Yes
-r, --resource	The logical resource ID of the resource that contains the metadata <i>Type:</i> String <i>Example:</i> -r WebServerHost	Yes
--region	The AWS CloudFormation regional endpoint to use. <i>Type:</i> String <i>Default:</i> us-east-1 <i>Example:</i> --region ", { "Ref" : "AWS::Region" },	No
--access-key	AWS access key for an account with permission to call DescribeStackResource on CloudFormation. The credential file parameter supersedes this parameter. <i>Type:</i> String	No
--secret-key	AWS secret access key that corresponds to the specified AWS access key. <i>Type:</i> String	No
--role	The name of an IAM role that is associated with the instance. <i>Type:</i> String Condition: The credential file parameter supersedes this parameter.	No
-f, --credential-file	A file that contains both a secret access key and an access key. The credential file parameter supersedes the --role, --access-key, and --secret-key parameters. <i>Type:</i> String	No
-c, --configsets	A comma-separated list of configsets to run (in order). <i>Type:</i> String <i>Default:</i> default	No
-u, --url	The AWS CloudFormation endpoint to use. <i>Type:</i> String	No
--http-proxy	An HTTP proxy (non-SSL). Use the following format: http://<user>:<password>@<host>:<port> <i>Type:</i> String	No
--https-proxy	An HTTPS proxy. Use the following format: https://<user>:<password>@<host>:<port>	No

Name	Description	Required
-v	<p><i>Type:</i> String</p> <p>Verbose output. This is useful for debugging cases where cfn-init is failing to initialize.</p> <p>Note</p> <p>To debug initialization events, you should turn DisableRollback on. You can do this by using the CloudFormation console, selecting <i>Show Advanced Options</i>, and then setting “Rollback on failure” to “No”. You can then SSH into the console and read the logs at /var/log/cfn-init.log.</p>	No

Example

Amazon Linux Example

The following snippet shows the `UserData` property of an EC2 instance, which runs the `InstallAndRun` configset that is associated with the `WebServerInstance` resource.

```

1 UserData:
2   "Fn::Base64":
3     !Sub |
4       #!/bin/bash -xe
5       /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource LaunchConfig\
6       --configsets wordpress_install --region ${AWS::Region}

```

cfn-signal

Description

The `cfn-signal` helper script signals AWS CloudFormation to indicate whether Amazon EC2 instances have been successfully created or updated. If you install and configure software applications on instances, you can signal AWS CloudFormation when those software applications are ready.

You use the `cfn-signal` script in conjunction with a `CreationPolicy` or an Auto Scaling group with a `WaitOnResourceSignals` update policy. When AWS CloudFormation creates or updates resources with those policies, it suspends work on the stack until the resource receives the requisite number of signals or until the timeout period is exceeded. For each valid signal that AWS CloudFormation receives, AWS CloudFormation publishes the signals to the stack events so that you track each signal.

Syntax for Resource Signaling (Recommended)

If you want to signal AWS CloudFormation resources, use the following syntax.



`cfn-signal` does not require credentials, so you do not need to use the `--access-key`, `--secret-key`, `--role`, or `--credential-file` options.

```
cfn-signal --success|-s signal.to.send \
    --access-key access.key \
    --credential-file|-f credential.file \
    --exit-code|-e exit.code \
    --http-proxy HTTP.proxy \
    --https-proxy HTTPS.proxy \
    --id|-i unique.id \
    --region AWS.region \
    --resource resource.logical.ID \
    --role IAM.role.name \
    --secret-key secret.key \
    --stack stack.name.or.stack.ID \
    --url AWS CloudFormation.endpoint
```

Syntax for Use with Wait Condition Handle

If you want to signal a wait condition handle, use the following syntax.

```
cfn-signal --success|-s signal.to.send \
--reason|-r resource.status.reason \
--data|-d data \
--id|-i unique.id \
--exit-code|-e exit.code \
waitconditionhandle.url
```

Options

The options that you can use depend on whether you're signaling a creation policy or a wait condition handle. Some options that apply to a creation policy might not apply to a wait condition handle.

Name	Description	Required
--access-key (resource signaling only)	AWS access key for an account with permission to call the AWS CloudFormation SignalResource API. The credential file parameter supersedes this parameter. <i>Type:</i> String	No
-d, --data (wait condition handle only)	Data to send back with the <code>waitConditionHandle</code> . <i>Type:</i> String <i>Default:</i> blank	No
-e, --exit-code	The error code from a process that can be used to determine success or failure. If specified, the <code>--success</code> option is ignored. <i>Type:</i> String <i>Example:</i> <code>-e \$?</code> (for Linux), <code>-e %ERRORLEVEL%</code> (for Windows cmd.exe), and <code>-e \$lastexitcode</code> (for Windows PowerShell).	No
-f, --credential-file (resource signaling only)	A file that contains both a secret access key and an access key. The credential file parameter supersedes the <code>-role</code> , <code>--access-key</code> , and <code>--secret-key</code> parameters. <i>Type:</i> String	No
--http-proxy	An HTTP proxy (non-SSL). Use the following format: <code>http://<user>:<password>@<host>:<port></code> <i>Type:</i> String	No
--https-proxy	An HTTPS proxy. Use the following format: <code>https://<user>:<password>@<host>:<port></code> <i>Type:</i> String	No
-i, --id	The unique ID to send. <i>Type:</i> String <i>Default:</i> The ID of the Amazon EC2 instance. If the ID cannot be resolved, the machine's Fully Qualified Domain Name (FQDN) is returned.	No
-r, --reason (wait condition handle only)	A status reason for the resource event (currently only used on failure) - defaults to 'Configuration failed' if success is false <i>Type:</i> String	No
--region	The AWS CloudFormation regional endpoint to use.	No

Name	Description	Required
(resource signaling only)	<i>Type:</i> String <i>Default:</i> us-east-1	
--resource (resource signaling only)	The logical ID of the resource that contains the creations policy you want to signal. <i>Type:</i> String	Yes
--role (resource signaling only)	The name of an IAM role that is associated with the instance. <i>Type:</i> String Condition: The credential file parameter supersedes this parameter.	No
-s, --success	if true, signal SUCCESS, else FAILURE. <i>Type:</i> Boolean <i>Default:</i> true	No
--secret-key (resource signaling only)	AWS secret access key that corresponds to the specified AWS access key. <i>Type:</i> String	No
--stack	The stack name or stack ID that contains the resource you want to signal <i>Type:</i> String	Yes
-u, --url (resource signaling only)	The AWS CloudFormation endpoint to use. <i>Type:</i> String	No
waitcondition\ handle.url (wait condition handle only)	A presigned URL that you can use to signal success or failure to an associated WaitCondition	Yes

Example

Amazon Linux Example

A common usage pattern is to use `cfn-init` and `cfn-signal` together. The `cfn-signal` call uses the return status of the call to `cfn-init` (using the `$?` shell construct). If the application fails to install, the instance will fail to create and the stack will rollback.

```
1 MyInstance:
2   Type: AWS::EC2::Instance
3   Metadata:
4     AWS::CloudFormation::Init:
5       cfn-init information
6   Properties:
7     ImageId: ami-12345678
8     UserData:
9       "Fn::Base64":
10      !Sub |
11        #!/bin/bash -xe
12        yum update -y aws-cfn-bootstrap
13        /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource LaunchCo\
14 nfig --configsets wordpress_install --region ${AWS::Region}
15        /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource Web\
16 ServerGroup --region ${AWS::Region}
17   CreationPolicy:
18     ResourceSignal:
19       Timeout: PT5M
```

Additional Examples

Several AWS CloudFormation sample templates use `cfn-signal`, including the following templates.

- https://s3.amazonaws.com/cloudformation-templates-us-east-1/LAMP_Single_Instance.template
- https://s3.amazonaws.com/cloudformation-templates-us-east-1/WordPress_Single_Instance.template

cfn-get-metadata

Description

You can use the `cfn-get-metadata` helper script to fetch a metadata block from CloudFormation and print it to standard out. You can also print a sub-tree of the metadata block if you specify a key. However, only top-level keys are supported.



`cfn-get-metadata` does not require credentials, so you do not need to use the `--access-key`, `--secret-key`, or `--credential-file` options.

Syntax

```
cfn-get-metadata --access-key access.key \
                  --secret-key secret.key \
                  --credential-file|f credential.file \
                  --key|k key \
                  --stack|-s stack.name.or.id \
                  --resource|-r logical.resource.id \
                  --url|-u service.url \
                  --region region
```

Options

Name	Description	Required
-k, --key	For a key-value pair, returns the name of the key for the value that you specified. <i>Type</i> : String <i>Example</i> : For { "SampleKey1": "Key1", "SampleKey2": "Key2" }, cfn-get-metadata -k Key2 returns SampleKey2.	No
-s, --stack	Name of the Stack. <i>Type</i> : String <i>Default</i> : None <i>Example</i> : -s { "Ref" : "AWS::StackName" },	Yes
-r, --resource	The logical resource ID of the resource that contains the metadata <i>Type</i> : String <i>Example</i> : -r WebServerHost	Yes
--region	The region to derive the CloudFormation URL from. <i>Type</i> : String <i>Default</i> : None <i>Example</i> : --region , { "Ref" : "AWS::Region" },	No
--access-key	AWS access key for an account with permission to call DescribeStackResource on CloudFormation. The credential file parameter supersedes this parameter. <i>Type</i> : String	No
--secret-key	AWS secret access key that corresponds to the specified AWS access key. <i>Type</i> : String	No
-f, --credential-file	A file that contains both a secret access key and an access key. The credential file parameter supersedes the --role, --access-key, and --secret-key parameters. <i>Type</i> : String	No

cfn-hup

Description

The `cfn-hup` helper is a daemon that detects changes in resource metadata and runs user-specified actions when a change is detected. This allows you to make configuration updates on your running Amazon EC2 instances through the `UpdateStack` API action.

Syntax

```
cfn-hup --config|-c config.dir \
    --no-daemon \
    --verbose|-v
```

Options

Name	Description	Required
<code>-c</code> , <code>--config</code> <code>config.dir</code>	Specifies the path that the <code>cfn-hup</code> script looks for the <code>cfn-hup.conf</code> and the <code>hooks.d</code> directories. On Windows, the default path is <code><system_drive>\cfn</code> . On Linux, the default path is <code>/etc/cfn</code> .	No
<code>--no-daemon</code>	Specify this option to run the <code>cfn-hup</code> script once and exit.	No
<code>-v</code> , <code>--verbose</code>	Specify this option to use verbose mode.	No

cfn-hup.conf Configuration File

The `cfn-hup.conf` file stores the name of the stack and the AWS credentials that the `cfn-hup` daemon targets. The `cfn-hup.conf` file uses the following format:

```
1 [main]
2 stack=<stack-name-or-id>
```

Name	Description	Required
stack	A stack name or ID. <i>Type:</i> String	Yes
credential-file	An owner-only credential file, in the same format used for the command line tools. <i>Type:</i> String	No
region	The name of the AWS region containing the stack. <i>Example:</i> us-east-1	No
interval	The interval used to check for changes to the resource metadata in minutes <i>Type:</i> Number <i>Default:</i> 15	No
verbose	Specifies whether to use verbose logging. <i>Type:</i> Boolean <i>Default:</i> false	No

hooks.conf Configuration File

The user actions that the `cfn-hup` daemon calls periodically are defined in the `hooks.conf` configuration file. The `hooks.conf` file uses the following format:

```

1 [hookname]
2 triggers=post.add or post.update or post.remove
3 path=Resources.<logicalResourceId> (.Metadata or .PhysicalResourceId)(.<optional\>
4 Metadatapath>)
5 action=<arbitrary shell command>
6 runas=<runas user>
```

When the action is run, it is run in a copy of the current environment (that `cfn-hup` is in), with `CFN_OLD_METADATA` set to the previous value of path, and `CFN_NEW_METADATA` set to the current value.

The hooks configuration file is loaded at `cfn-hup` daemon startup only, so new hooks will require the daemon to be restarted. A cache of previous metadata values is stored at `/var/lib/cfn-hup/data/metadata_db` (not human readable)–you can delete this cache to force `cfn-hup` to run all `post.add` actions again.

Name	Description	Required
hookname	A unique name for this hook <i>Type:</i> String	Yes
triggers	A comma-delimited list of conditions to detect. <i>Valid values:</i> post.add, post.update, or post.remove <i>Example:</i> post.add, post.update	Yes
path	The path to the metadata object. Supports an arbitrarily deep path within the Metadata block.	Yes
action	An arbitrary shell command that is run as given.	Yes
runas	A user to run the commands as. Cfn-hup uses the su command to switch to the user.	Yes

Path format options:

Resources.<LogicalResourceId>

monitor the last updated time of the resource, triggering on any change to the resource.
Resources.<LogicalResourceId>.PhysicalResourceId

monitor the physical ID of the resource, triggering only when the associated resource identity changes (such as a new EC2 instance). Resources.<LogicalResourceId>.Metadata(.optional path)

monitor the metadata of a resource for changes (a metadata subpath may be specified to an arbitrarily deep level to monitor specific values).

hooks.d Directory

To support composition of several applications deploying change notification hooks, cfn-hup supports a directory named `hooks.d` that is located in the `hooks` configuration directory. You can place one or more additional hooks configuration files in the `hooks.d` directory. The additional hooks files must use the same layout as the `hooks.conf` file.

The `cfn-hup` daemon parses and loads each file in this directory. If any hooks in the `hooks.d` directory have the same name as a hook in `hooks.conf`, the hooks will be merged (meaning `hooks.d` will overwrite `hooks.conf` for any values that both files specify).

Example

In the following template snippet, AWS CloudFormation triggers the `cfn-auto-reloader.conf` hooks file when you change the `AWS::CloudFormation::Init` resource that is associated with the `LaunchConfig` resource.

```
1 ...
2 LaunchConfig:
3   Type: "AWS::AutoScaling::LaunchConfiguration"
4   Metadata:
5     AWS::CloudFormation::Init:
6 ...
7     /etc/cfn/hooks.d/cfn-auto-reloader.conf:
8       content: !Sub |
9         [cfn-auto-reloader-hook]
10        triggers=post.update
11        path=Resources.LaunchConfig.Metadata.AWS::CloudFormation::Init
12        action=/opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --reso\
13        urce LaunchConfig --configsets wordpress_install --region ${AWS::Region}
14        mode: "000400"
15        owner: "root"
16        group: "root"
17 ...
```