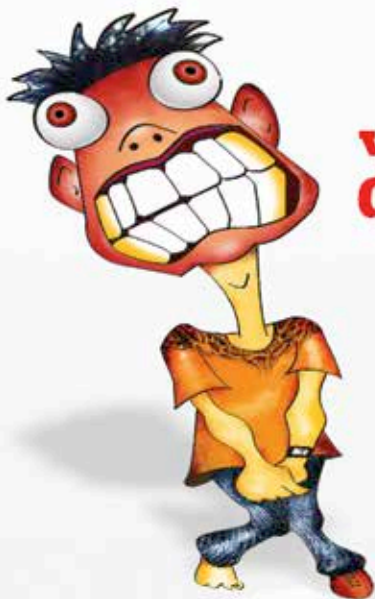


to
**Amazon
Web Services**



- An introduction to AWS services
- IAM, VPC and account configuration of AWS
- Amazon in the cloud
- AWS databases and integrating them into your app
- Open your app to the world!
- Serverless code!
- The AWS data game
- AWS language tools
- Learning and recognising in AWS
- AWS IoT - Greengrass
- AWS MTurk
- CloudWatch and more



**www.
digit.in/forum**

Join the forum to
express your views
and resolve your
differences in a more
civilised way.

**digit.in
FORUM**

Post your queries
and get instant
answers to all
your technology
related questions



One of the most active online technology forums
not only in India but world-wide

**JOIN
NOW**

digit.in



AMAZON WEB SERVICES

powered by

digit
YOUR TECHNOLOGY NAVIGATOR

CHAPTERS

AMAZON WEB SERVICES

JANUARY 2018

05

PAGE

An introduction to AWS Services

A myriad of services are offered by Amazon under the umbrella of AWS. Here's a little introduction to them all.

11

PAGE

IAM,VPC and Account config of AWS

One of the most used AWS services. VPC allows you to set up an isolated network within AWS.

18

PAGE

Amazon in the cloud

Get started with your own configurable web server instance with Amazon EC2.

28

PAGE

AWS Databases and integrating them into your app

Whether you're starting off with a new project or migrating an existing project onto AWS, there's a database for you.

35

PAGE

Open your app to the world!

Open your application to the world using Route 53, AWS' very own Domain Name System.

CREDITS

The people behind this book

EDITORIAL

Executive Editor

Robert Sovereign-Smith

Asst. Technical Editor

Mithun Mohandas

Writer

Arnab Mukherjee

Contributing Writers

Arunava Roy Choudhury

Ashish Dhiman

Poulami Dey

Prashant Panchal

Sumedh Rasal

DESIGN

Sr. Art Director

Anil VK

Visualiser

Baiju NV

40
PAGE

Serverless code!

AWS Step Functions makes it easy to coordinate the components of distributed applications and microservices using visual workflows.

48
PAGE

The AWS data game

Sift through massive amounts of data using Amazon Kinesis - the big data solution available on AWS.

55
PAGE

AWS Language tools

Everyone's getting a chat bot and they're quite easy to build, especially if you know Amazon Lex.

64
PAGE

Learning and recognising in AWS

Thinking of building your own predictive model on AWS? Amazon Machine Learning is what you need.

74
PAGE

AWS IOT - Greengrass

Use Greengrass to get all your connected devices to communicate with each other.

80
PAGE

AWS Mturk

Don't like the way machines handle your data? You can always get a human to do it instead.

88
PAGE

CloudWatch and more

Keep an eye on your AWS applications just so that any untoward incidents don't end up giving you a bill shock.

© 9.9 Mediaworx Pvt. Ltd.

Published by 9.9 Mediaworx Pvt. Ltd.

No part of this book may be reproduced, stored, or transmitted in any form or by any means without the prior written permission of the publisher.

January 2018

Free with Digit. If you have paid to buy this Fast Track from any source other than 9.9 Mediaworx Pvt. Ltd., please write to editor@digit.in with details

Custom publishing

If you want us to create a customised Fast Track for you in order to demystify technology for your community, employees or students contact editor@digit.in

e-mail




COVER DESIGN: PETERSON PU

A computer in the clouds

Amazon is big - there is no doubt about that. From selling everything, to providing within the hour delivery of everyday stuff, to delivering TV shows and movies to your phones and laptops, it offers a lot of services. One could say that it has solely popularised smart speakers with the Echo devices. But there is another part of Amazon that not many know about, and yet it's the fastest growing business within Amazon with the highest operating income. We are talking about AWS.

Amazon Web Services is that division of Amazon which essentially provides cloud computing services. Yes, we did oversimplify when we said that - AWS provides almost every cloud-based service under the sun across its broad product portfolio. You can buy storage space to hold a huge database, bandwidth to host a website, or processing power to run complex AI workloads. AWS helps companies and individuals, big and small, to stop worrying about the infrastructure and enables them to implement their ideas exactly as they'd want to.

Earlier this year, when Amazon posted its earnings, it was evident that AWS was over three times as big as its nearest competitor. From early adopters like Netflix to big names such as Adobe, Airbnb, IMDb, Samsung, Ubisoft and many more, AWS is used by everyone. It is the popular belief that AWS runs the internet. While that may not essentially be true, and AWS might be a big fish swimming in an enormous ocean, there's no denying the true possibility of what can be accomplished with their services - many of which are available for you to try for free!

And that is what this book is all about. From introducing you to the complex AWS portfolio to taking you through what it means to have a completely operational application in the cloud, this FastTrack aims to be the one - and only - starting point you'll ever need to the amazing (and mind boggling) world of Amazon Web Services. 



Amazon Web Services

A myriad of services are offered by Amazon under the umbrella of AWS. Here's a little introduction to them all.

Technology is by far the most rapidly growing domain in the last two decades or so. Ever since the compute explosion happened and multi-core and multi-threaded computers were created, the capabilities of computing have gone from amazing to downright mind boggling in almost no time.

Today, we are deciphering thousands of years old ancient text, predicting asteroid impacts (the dinosaurs would be jealous) and even sequencing our own DNA to come up with new and improved medicines, and even our own selves. It seems technology has reached a limit where we are no longer in absolute

control but coexist with what is by many definitions a complex organism unto itself. But where are the brains of this organism?

Years into the technology explosion we still have the same approach towards hardware and platforms to run our advanced software. Mainframes are still bulky and expensive to maintain and a computer with processing speed in Teraflops is still inaccessible to an average individual.

This was perhaps the problem statement that challenged the people at Amazon back in 2006. Hence Amazon Web Services (AWS) was formed, with a singular purpose to overcome the limitations of physical infrastructure and provide the ease and flexibility of cloud based managed services and platforms. AWS has evolved into a multi-billion dollar subsidiary of Amazon. Today it has well over a thousand services offered on its platform ranging from a simple Virtual Machine (VM) to an incredibly powerful video analysis tool.

Developers are no longer limited to ideas that are feasible only on minimal hardware and can rent very powerful machines and pay for them by usage or by the hour solving problems that would not have been conceivable a few years ago. Entrepreneurs likewise, do not need to wait for a technical implementation of an idea (that can take months if not years), but can simply use one of the managed services on offer.

The AWS TimeLine:

2006

- AWS releases with Simple Storage Service (S3).
- Simple Queue Service (SQS) is released in production.
- Amazon launches Amazon Elastic Compute Cloud (EC2), allowing users to rent virtual computers on which to run their own computer applications.

2007

- Amazon goes global, launches S3 in Europe, reducing latency and bandwidth for European users.

2008

- AWS launches Amazon CloudFront, a content delivery network (CDN).

2009

- Amazon launches Amazon Elastic MapReduce, which allows researchers, data analysts, and developers to easily process vast amounts of data.

- Amazon introduces Elastic Load Balancing, Auto Scaling, and Amazon CloudWatch.

2010

- Amazon.com migrates its retail web services to AWS.
- AWS launches Amazon Route 53, a scalable and highly available DNS.

2013

- Amazon announces G2 instances, a new Amazon Elastic Compute Cloud (EC2) instance type designed for applications that require 3D graphics capabilities.
- Amazon releases Amazon Kinesis, a service for real-time processing of streaming data.

2014

- AWS launches AWS Lambda, its Functions as a Service (FaaS) tool. With Lambda, AWS customers can define and upload functions with specific triggers and execution code.

2015

- AWS launches Snowball, a physical appliance with 50 TB of storage and a Kindle on the side. Customers can get a Snowball delivered to their premises that they can fill with data and then ship it back to Amazon.

2016

- Amazon Polly text-to-speech product

Services on AWS:

The total number of services offered by AWS as of the date of writing this article stands at more than 2000. The ecosystem of services offered on the platform are broadly divided into the following categories:

- Compute
- Storage
- Database
- Migration
- Networking & Content Delivery
- Developer Tools
- Management Tools

- Media Tools
- Machine Learning
- Analytics
- Security, Identity & Compliance
- Mobile Services
- AR & VR
- Internet of Things
- Game Development

In this tutorial, we will primarily focus on the services under Compute, Storage, Database & Machine Learning. Readers are encouraged to study and play around with all the other services on offer as well. (especially as many of them have a free usage tier!).

Getting Started on AWS

Setting up an account

Setting up an account on AWS is easy for an independent developer and only takes a few minutes.

- Navigate to aws.amazon.com and click on “Sign Up”.
- Fill out the account information and click “Next”.
- You will need to add a Credit Card to your account to which AWS bills your costs (if any) at the end of each month.
- Once you’re done, login to the console at <https://console.aws.amazon.com/console/home>

Some components of the AWS Console:

Services Tab

The services tab near the top left corner of the console serves as a quick navigation bar between services.

Resource Groups Tab

The “Resource Groups” tab is used to view a group of resources on AWS that share a common tag. This can be particularly useful when organizing resources (services or machines) into logical blocks or smaller systems of software (eg: Production & Development ecosystems).

Start by signing up

Region Tab

The “Region” tab is default set to “US East (Virginia)”. This basically means that all services and machines that you request from AWS will be hosted from the AWS Data Center present at that location. This is particularly helpful as often, for highly available and fault tolerant systems, it is critical to have services available across multiple regions.

Support Tab

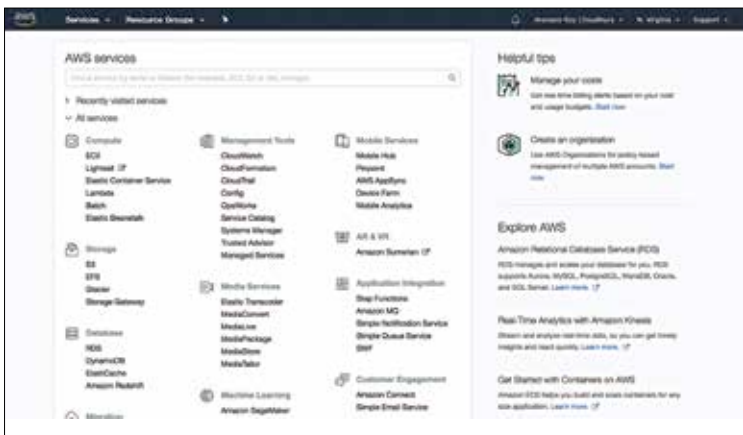
The “Support” tab can be used to access the Support Center where experts can hear your queries and provide solutions. It also links to documentation for all the services, training material for many of them and a forum.

Alerts Tab:

The “Alerts” tab provides information on all the fixes and issues that are part of AWS deployments as well as scheduled changes (eg: releases).

Build a solution

The “Build a Solution” panel is a great tool to quickly configure and deploy a service as it allows a wizard based creation of a service that is easy and intuitive to follow. Often, using this tool would drastically lower the time taken to deploy a service simply because it streamlines the process of configuring it. Most novice developers start off from here when trying to deploy a service.



A glimpse at all AWS services right on your dashboard

Learn to build

A more comprehensive and detailed approach to “Build a solution”, the “Learn to Build” panel offers a set of step-by-step instruction manual with detail training and manual on deploying the most common and user friendly services (host static website, register a domain on Route53 etc). This is the go-to tool for all beginners if they want to get an interactive tool to learn AWS, while creating an actual live service in the process.

Or you could just read this tutorial!

Setting up your local machine for AWS development

The capabilities of AWS allow even server-less(!) deployment. But Amazon Web Services are after all a set of programs unto themselves and are not always feasible to be accessed over the interactive console.

Enter AWS CLI or the AWS Command Line Interface – a unified tool to manage your AWS services. With just one tool, you can control most AWS services from the command line, write scripts for automation of jobs (like deploying multiple instances of your application or taking backups). Setting up AWS CLI is easy too. For Mac and Linux users, simply install using pip by the following command:

- `pip install awscli`

* this requires python 2.6.5 or higher as a dependency

For Windows users, download the installer from the url: <http://dgit.in/AWS32CLI> or <http://dgit.in/AWS64CLI> for your 32-bit or 64-bit machine and run it. You can now connect to and maintain your services on amazon simply from the command line. To do this, you would need to configure your AWS CLI for credentials (explain later in chapters) and then run the relevant commands.

We now have a context on Amazon Web Services, a brief primer on its history as well a basic understanding of how to get started with an account on amazon, including the AWS Console. Let us now begin to look at each of the services in greater detail. **d**



AWS VPC

One of the most used AWS services.
VPC allows you to set up an isolated network within AWS.

Virtual Private Cloud is nothing but a dedicated virtual network for someone's AWS account. It gives you the ability to launch AWS resources in a virtual network that you define. Now what is a virtual network? It is a technology that facilitates the centralised control of one or more devices. These devices may be in different physical locations.

A VPC is logically isolated from other virtual networks. You can use VPC to launch AWS resources and configure the network by modifying IP address range, creating subnets, and modifying security settings as per the need.

Key features of AWS VPC are follows:

Secure - VPC is logically isolated from other virtual networks. Along with this, AWS provides some advanced features for security like security group, Network access control lists (ACLs), flow logs etc.

- The Security Group plays the role of a firewall for EC2 instances. Whenever an instance is launched in VPC, the user can associate one more security group with it.
- The Network access control list is the firewall for subnets.
- Flow logs log information of incoming and outgoing IP traffic to the VPC. Additionally, you can also go for dedicated instances that run on hardware, assigned to your instance only.

Simple - It is very easy to create a VPC. You just need to follow some basic steps using Amazon's console. Select a network setup according to your requirements and your VPC is ready. It will automatically create subnets, routing tables, security groups etc. for you. You don't need to invest your time setting up and can instead start launching your application directly.

Scalability and Reliability - VPC comes with the same scalability and reliability as any other of AWS's resources. You can easily launch other AWS resources in your VPC like EC2 instances, S3 bucket, and DynamoDB. You will of course, have to pay as per usage.

Connectivity - Another advantage VPC has is its connectivity options. You can connect with other VPCs, or data centres, or the internet as per your needs.

- If your application needs an internet connection, you can launch it onto a public subnet.
- If your application doesn't need to be directly accessible via the internet, you can use a private subnet. You can access the internet without exposing the IP addresses of your instances by routing through a NAT gateway. NAT sends traffic to the internet with some public IP addresses and keeps the mapping of private and public IP addresses to itself.
- You can have a standard VPN connection to your corporate networks or multiple VPCs can also be connected through a private connection within AWS.

Usability of VPC:

- If you have a simple web application, you can host it in AWS VPC and make use of AWS security systems like Security Group. You can have a public subnet for communicating with internet.

- For multi-tier web applications, you can launch your web server on a public subnet and data server using a private subnet, where external traffic cannot go.
- AWS VPC also gives you the opportunity to extend your network into the cloud.
- AWS VPC is highly scalable.



Use VPC to hide certain features of your app from the Internet

You can launch your web server with a public subnet, which will communicate with the internet, and instances in other subnets can communicate with your corporate network for data storage.

Getting Started with VPC:

You can launch your own VPC by following some basic steps.

- **Create the VPC** – You can create a VPC using the Amazon VPC wizard in the Amazon VPC console. It will create a VPC and attach an internet gateway to it. After that it will create a subnet and associate a route table to that subnet. You can also view information about the VPC, subnet, internet gateways, and routing table on Amazon's VPC console after creation.
- **Create a Security Group** – The next step is to create your own security group. To use Security Group, you need to add inbound and outbound rules for controlling incoming and outgoing traffic. Whenever you launch any instances in your VPC, just specify this security group. All these rules will be applied to your instance. Later, if you want to modify the Security Group, rules will be modified and applied accordingly. But this step is not mandatory. If you do not specify a security group while launching your instances, they will be associated with a default security group defined by AWS.
- **Launch an instance** – This is the most important step where you launch your instances. After choosing the instance type, you will get an option to 'Configure Instance Details'. Here you need to choose your VPC from a network list and a subnet from a subnet list. After this, you can add name tags using the Add Tag option. Next, you select your security group from the 'Configure Security group page' option. Your instance is now ready to be launched.

- **Assign elastic IP Address to your instance** – If your instance needs to communicate with internet, you need only attach an elastic IP to your instance.
- **Clean up** – Last but not least, clean up your resources before you delete your VPC. You should terminate all instances you have running in that VPC. Keep in mind that when deleting a VPC, all other components like the subnet, routing table, and Security Group associated with it will also be deleted.

AWS IAM:

Identity and Access Management (IAM) is a web service that allows you to manage access to AWS services and resources in a beautiful yet simple way. Using IAM you can give access of your AWS resources to authorised and authentic persons only.

Now, you already need to sign-in in order to access AWS services or resources. So why do we need an additional access management service?

We need this because this is the ideal way to secure our AWS resources.

The identity you created while creating an AWS account has access to all resources and services. It's known as the root user. Ideally, the root user's credentials should not be used for daily tasks. According to AWS "adhere to the best practice of using the root user only to create your first IAM user."

IAM Features:

Following are the key features of IAM.

- Using IAM you can easily give other people access to your AWS resources without sharing your password.
- Different users can have different permissions depending on their responsibility. For example, you can give some users complete access to EC2 instances while another user may have only read access to Amazon S3.
- If you want to impose extra security for your AWS instances, you can use multi-factor authentication for individual users.
- Many AWS services work well while IAM is integrated with them.
- You can also give temporary access to the users who are already registered in your corporate network.
- IAM replicates data across many data servers within Amazon for achieving consistency.
- And finally, you will not be additionally charged for using this service. You just need to pay for the resources used by your users.

Working Principle of IAM:

IAM provides the infrastructure to manage access to your AWS resources. Now let's see how IAM works.

- **Principal** - An entity that can perform an action on AWS resources can be called a principal. Users who have access to AWS resources, an application that has access to AWS account, all are principals.
- **Request** - When a principal tries to perform an action on any resource then this process is called a request. Requests contain data like -
 1. Operations that a principal wants to perform.
 2. Resources on the operation which will be performed.
 3. Information regarding the principal.
- **Authentication** - For sending any request, a principal should be always authenticated. This means any principal should have a username and password for its unique identification. You can impose multi-factor authentication for some critical resources in your AWS account. Then some other information like secret key should be shared along with the username and password.
- **Authorisation** - To send any request, a principal must be authorised to do so. You can grant different level of access to different principals. AWS root user has access to all the resources by default.
- **Action** - Actions are the operations principals want to perform. After it is verified that principal is authentic and authorised, AWS approves the requested action.
- **Resources** - Principals perform actions using resources. It can be anything from EC2 instances, to S3 bucket, to an IAM user.

Overview of IAM users:

Using IAM policy you can impose customised security system in your AWS account. Different users can have different level of access. Broadly we can categorise users into three groups.

- **Root user** - By default, this user has access to every resource. This is the user you have created when creating the AWS account. This user creates the first IAM user.
- **IAM users** - These are the users who can access AWS resources depending on their rights.
- **Federating existing users** - You can give temporary access to the users who are already authenticated in your corporate network by federating existing identities into AWS.

IAM Best practises:

You should adhere to some IAM best practices to achieve better security for your AWS resources. Those practices are as follows.

- Your root user should be used only to create the first IAM user, and not for anything else.
- Create individual IAM users for each person you want to give access to. Do not share credentials.
- Try to use policies that are already defined by AWS while assigning permissions.
- Create Security groups to assign permissions to IAM users.
- Grant access only to the resources that are required.
- Your passwords should always be strong and as per industry standard.
- Keep track of changes to your AWS account.
- Credentials should be changed periodically.
- Use multi-factor authentication for your critical resources.



Be careful with your user groups in IAM

AWS Config:

After you've set up your VPC and created IAM user groups, you can launch all the AWS instances you require. But things do not end here. You need to monitor, assess, audit, and evaluate your AWS resource configuration to check whether they are running according to your expectations. This can get rigorous and hectic.

AWS comes with a solution for this, AWS Config. It helps you to automate this process. Using AWS Config, you can do the following jobs very easily.

- You can check if AWS resource configuration is complied with your desired setting.
- Audit configuration related data.
- Get notifications when any resource is modified.
- Check relationships between the resources.

Whenever there is change in configuration of your AWS resource, AWS Config records this change and normalises this change in a consistent format. Then it checks if the configuration is aligned with your needs. You can view the results of these evaluations on a dashboard.



AWS config allows you to monitor your IAM user group.

Benefits of AWS Config:

- **Continuous Monitoring** - Continuous monitoring is a key feature of AWS config. Whenever there is a change you will be notified. You can also define rules using AWS config. If any resource violates any rule, config will notify you. AWS config evaluates your resource configurations continuously.
- **Continuous Assessments** - AWS Config not only monitors, but also helps you assess changes. Another advantage of config is its visual display of evaluation results.
- **Operational Troubleshooting** - AWS config maintains historical records of your configuration changes. If some resource change causes problems due to integration with other resources, you can go through historical data to find the best solution.
- **Security Analysis** - You can also monitor your IAM user group using AWS Config. It will give you clear picture of which user has what controls. If you feel there should be a change in control, you can do that as well. Additionally, it will also give you historical data.

So, if you have a web application and are wondering where to launch it and how to secure it, AWS has the solution. Just launch your instances in VPC and secure it with IAM. **d**



EC2 (Elastic Cloud Compute)

Get started with your own configurable web server instance with EC2.

EC2 is one of the most important services of AWS and is often referred to as the backbone of AWS. EC2 is a cloud service which allows you to launch configurable compute capacity in the cloud. You can increase and decrease the capacity based on your requirement.

1.1 Launching an EC2 Instance in AWS

There are multiple ways to launch an EC2 instance in the cloud

1. AWS EC2 Dashboard
2. AWS CLI
3. AWS SDK

The most commonly used way to launch an EC2 instance is using the AWS EC2 dashboard and this is what we are going to learn.

Once you sign in to the AWS Management Console, go to the EC2 dashboard and click on Launch Instance to launch a new EC2 instance. Launching an instance consists of multiple steps mentioned below.

Step 1 - Select an AMI for your EC2 instance:

AMI is an Amazon Machine Image, which basically consists information about the OS of your virtual machine and the default software packages included in the machine. You may select any of the default AMI's available from the list.



Start off by picking AMIs which are pre-configured OS images in EC2.

Step 2 - Selecting an Instance Type:

Once you select the AMI, you will need to select the Instance Type. Instance type is the compute unit that you want to launch. It contains information like number of cores, memory, and I/O about the virtual machine. There are various instance types that are available in the list which you can select as per your needs. We will select t2.micro for now.

Step 3 - Configure Instance Details:

Once you have selected the instance type, you will see the Configure Instance screen where you can configure how many instances you want to launch, network configuration etc. At the moment, you just need to worry about

Step 2: Choose an Instance Type

Review 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 | Amazon Linux 2 AMIs

Currently selected: 12 vCPUs (Instance ID: 1), 1 vCPU (EC2-Optimized), 100 GB (memory), 100 GB (storage)

Family	Type	vCPUs	Memory (MiB)	Instance Storage (GiB)	EBS-Optimized Available	Network Performance	Price per Hour
General purpose	t2.micro	1	1	0	Yes	Low to Moderate	Yes
General purpose	t2.xlarge	8	16	0	Yes	Low to Moderate	Yes
General purpose	t2.medium	2	4	0	Yes	Low to Moderate	Yes
General purpose	t2.large	2	8	0	Yes	Low to Moderate	Yes
General purpose	t2.xlarge	4	16	0	Yes	Moderate	Yes
General purpose	t2.2xlarge	8	32	0	Yes	Moderate	Yes
General purpose	m4.xlarge	2	8	0	Yes	Moderate	Yes

Buttons: Cancel, Previous, **Next: Configure Instance Details**, Next: Configure Instance Details

Pick the configuration that's best suited to your compute needs.

the number of instances that you want to launch and keep the rest of the options as it is. We will learn about other configuration details later.

Step 4 - Configure Security Group:

Once you have selected the Instance Type, skip Step 4 and 5. Click on the Next button and go to Step 6 directly – selection of a Security Group. Security Group consists of rules which control network traffic for your instance. There are inbound and outbound traffic rules that you can modify to change what traffic can reach your instance and what your instance can access. You can create a new Security Group or select an existing Security Group. We are going to create a new Security Group test-security with inbound rules. For example, first inbound rule i.e. SSH 22 ensures that once I launch my instance, I should be able to ssh into it. Second inbound rule in the list makes the HTTP 8080 PORT of the instance to be accessible to anyone. You can add as many rules as you like and link as many Security Groups to the instances as you like. In case of multiple Security groups, the combination of rules of all of the security groups linked to an instance will be applied. Once done, click on the Review and Launch button.

Step 5 - Review and Launch:

Review all the information before finally launching the instance. If everything seems OK, click on the Launch button. It will open a dialog to select a Key-Pair for your instance. The Key-Pair is a key which has information about logging into the instance using ssh. You can use the same Key-Pairs for multiple instances also. We are going to create a new Key-Pair here, with



Use Security Group settings to determine who can access your front-end and back-end.

name test-t2-micro. To create a new Key-Pair select the Create a new key pair option, enter the name of the Key-Pair and click on the Download Key Pair button. It will download the test-t2-micro.pem file which contains a private key needed to access the instance. You should save this file as this is the only time you can download this Key-Pair and this is mandatory to ssh into the launched instance.

The final step is to click on the Launch Instances button. Once the instance is ready you should be able to see something like this on the EC2 Dashboard.

1.2 Connect to EC2 Instance through SSH

The most basic task that you would need to do after launching the EC2 is connecting to that instance through SSH. You would need the private key that you have stored in the previous steps to do so.



You can only get your private key file once. Keep it safe.

i. If you are using a new Key-Pair, then go to the directory where the Key-Pair is located using Terminal and type the following command:

- `chmod 400 test-t2-micro.pem`

This command makes your Key-Pair hidden and it can now be used to SSH into your EC2 instance.

ii. Use this command to SSH into the EC2 instance:

- `ssh -i "test-t2-micro.pem" <user>@<instance.public.ip>`

Note that the `<user>` and `<instance.public.ip>` have to be replaced with the actual values. You can get this information by selecting your instance on EC2 Dashboard and clicking on the Connect button.



All your instance details can be accessed at anytime via the dashboard.

You are now logged into the EC2 instance that you launched in the last step. It will SSH you into the home directory of your instance. You can now use this instance as a virtual machine.

1.3 Other ways to Launch and Connect to EC2

In the previous exercises, you learned about how to launch a new EC2 instance using the EC2 Dashboard and how to connect to the instance using SSH. But this is not the only way to launch an EC2 instance and as mentioned earlier there are other ways to connect to Launch a new EC2 instance i.e. by using Amazon CLI or by using AWS SDK. You might need to use any of these ways to launch instances depending upon your use case. Now that you know how to launch an instance using one way, it will be very easier to understand other ways as well.

In the second step of the exercise, we learned about how to connect to an EC2 instance using Terminal. But there are some tools like WinSCP and

FileZilla which can connect to an instance and provide a File Explorer-like interface to exchange files between your local machine and an EC2 instance.

1.4 Instance purchasing options

Now that you know about launching EC2 instances, this section introduces the pricing and purchasing aspect of the EC2 instances. Prices of EC2 instances are directly dependent on the compute, memory and I/O. Higher you go above after three parameters, higher you need to pay. But there are multiple purchasing options available in AWS which could drastically optimise the cost you actually pay by simply selecting the suitable purchasing option for your needs. We are not going to go in details but we'll explore some commonly available options. They are:

- 1. On demand instances :** Pay as per you use. You get charged for the number of instance hours you used the instance for. When you launch an instance without any purchasing option, it gets launched in this category. This is the default choice for launching an instance.
- 2. Reserved instances :** If you purchase reserved instances, Amazon offers a significant discount on the actual price of the instances. The plans are available for one or three years. So, if you are launching some instances and plan to use it for a very long period of time, you can go for this plan.
- 3. Spot instances :** Spot instances are available at a very low price (specific to availability zone) as compared to On Demand and Reserved instances. The reason for their low price is that Amazon identifies the free instances in a particular availability zone and puts them into a common pool and assigns a price to that instance. The price changes as per the demand of the instance. You can place a bid for a spot instance (usually lower than on-demand price) and if the bid is higher than the current Spot Price of the instance, the instance will be launched. The instance will keep on running till the Spot Price remains lower than the bid price put by you. Once the Spot price exceeds the bid price, you will get a two minutes notice and the instance will terminate automatically. These are not the best choice but can be used for running some very low priority logs like log processing etc.
- 4. Dedicated host :** A dedicated host is a physical server with an EC2 instance capacity, fully dedicated for your use. This is an expensive option but if your data is very confidential that you can't take a risk of running an instance on a shared machine, this is a best choice for you.

2. S3 (Simple Storage Service)

Simple Storage Service commonly referred to as S3 is a cloud storage service where you can store, retrieve data at anytime from anywhere on the Internet. S3 comes with a simple web interface where you can see all the files you have saved on S3 and use it to upload or download.

2.1 Creating a bucket in S3

Every object in S3 is stored inside a bucket. Bucket is like a Folder on AWS cloud which has a unique name across AWS S3 users. So, if you have created a bucket with some name, no one will be able to use the same bucket name. Once a bucket is created, you cannot rename it. We will now see how we can create an S3 bucket and store data on the cloud.

- i. Go to the Amazon S3 Dashboard and click on Create bucket button.
- ii. Select the Bucket name and the Region where you would like to create the bucket. Although it is not mandatory but you should always use the bucket name compliant with a DNS address. The reason for DNS compliant bucket name is that S3 bucket acts like a web address and if you like, you can also use the S3 bucket to host a static website or some files.
- iii. Leave the properties as it is for the moment and go to the Set Permissions step. Make sure that your User ID or Group ID has permission

Setting up a new S3 bucket is super easy.

to Read and Write the Object data. You can change these permissions later on if you desire.

- iv. Once done, you will be able to see an empty bucket in the bucket list. We will now upload some files into this bucket.

2.2 Uploading data to S3 bucket

Data is always stored in the form of objects in a bucket. So, each file is a unique object and has a Key associated with it. You can create multiple folders inside a bucket and upload files into those folders. There are multiple ways to upload/download data on S3 but we are first going to upload using the Web Interface.

2.2.1 Upload using S3 web interface

- i. Open the bucket by clicking on it and click on the Upload button. Then click on Add files. Select a file from your local machine and add it to the list. You can select multiple files to upload. We have added a test-new-file.txt to upload which we will be able to see in S3 after uploading.
- ii. If you click on the file, you will be able to see an Overview. You can make it public so that it can be publicly accessible using a web url, download it to any local machine, move or copy this file anywhere in the bucket.

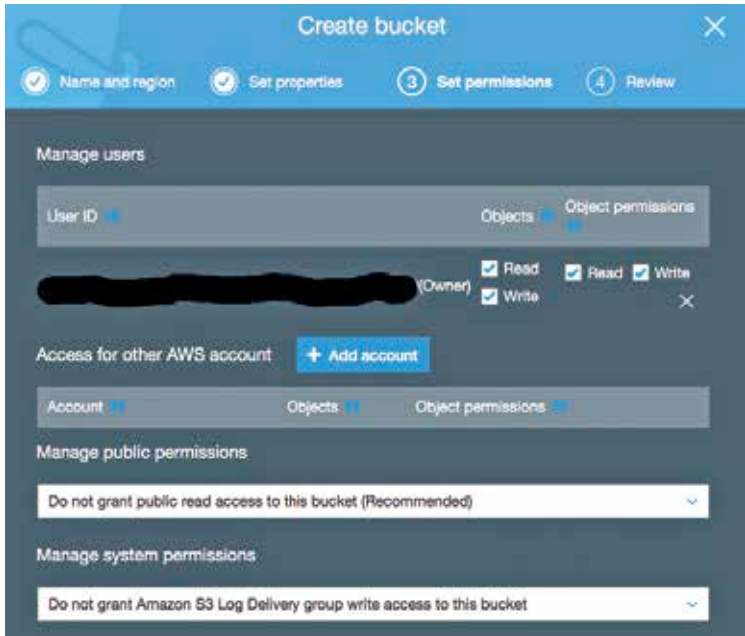
2.2.2 Upload using AWS CLI

In practice, the amount of data that we need to upload to S3 or download from S3 can be huge, so using the Web Interface is not the most efficient way to do so. In such cases, we can make use of AWS CLI which is the most efficient way to deal with the S3 data.

AWS command line interface is a tool which can be used to manage various AWS services using the command line. It provides us with a set of commands to efficiently exchange the data with the S3 bucket. It can be used for managing other services apart from S3 but we just need to know a couple of cp commands for our use case.

You can download and install the AWS CLI on your local machine and then we can get started with the following steps to exchange data with S3.

- i. The very first thing we need to do is configure AWS CLI. Type “aws configure” in the terminal and it will prompt for your account details. Only enter AWS Access Key ID and AWS Secret Access Key from your IAM access key in security credentials. You can create a new access key if you don't have any by going to IAM service in AWS Dashboard.



Decide which users can access your S3 bucket.

ii. To list the files in a bucket, use the command:

- `aws s3 ls s3://<bucket-name>`

<bucket-name> above has to be replaced with the actual bucket name.

iii. To copy the files from your local machine to the S3 Bucket or from the S3 bucket to your local machine, use the following command:

- `aws s3 cp <local-folder> s3://<bucket-name>/<folder-name> --recursive`

This command copies the contents of your local folder recursively to the folder inside the S3 bucket. This is clearly not possible using the web interface provided by AWS Dashboard for uploading files. Replace <local-folder> with the local folder you want to upload, <bucket-name> with the name of the S3 bucket where you want to copy the folder, and <folder-name> with the destination folder/path in S3 bucket. --recursive ensures that all files including subfolders are copied to the destination. You can change the order of the S3 path and local folder name if you want to copy files from S3 to local. If you open your S3 Bucket on AWS Dashboard, you will see the files have been copied successfully.


2.3 S3 Storage Management

There are different store management operations that are available as a part of S3 which can help in cost optimisation, analytics and cleanup of data. If you click on Management tab in S3 bucket, it looks something like this

Let's look at some of these options:

1. **Lifecycle** : S3 was designed primarily for rapid retrieval of data. But this fast retrieval comes with a significant cost and if the data becomes huge, it can be pretty expensive. Amazon has a solution for that too. You can create various life-cycle rules to move the data out of S3. If your data is not relevant to you after a certain time period, you can add an expiration rule and S3 will clean up the particular folder for you after the data becomes older than a certain threshold. If you don't want to permanently delete the data, another option is moving the data to Glacier. Glacier is a very cheap data store which costs as little as \$0.01 per GB per month where you can put the old data of S3 but it takes 3-5 hours to retrieve the data.
2. **Replication** : You can add cross region replication rules in case you want to have faster data access across multiple regions.
3. **Analytics** : This Amazon S3 analytics feature observes data access patterns and helps you determine when to transition from the less frequently accessed STANDARD storage to the STANDARD_IA (IA, for infrequent access) storage class. You can set it up on the entire bucket or a prefix path inside the bucket.
4. **Metrics** : Normally, AWS provides you with storage metrics once per day, free of cost but if you want to opt for request metrics per minute you can opt for this paid metrics option.

3. EC2 & S3

Often, you will have a situation where you would need to establish communication between various AWS services and one such common scenario will be communication between EC2 and S3. You have learned EC2 and S3 in the above sections and after having a hands-on tutorial, you should now have become quite familiar with them. You have also learned how to make use of AWS CLI to exchange the files between S3 bucket and local machine. Similarly, EC2 is like a local machine but it is running on the cloud. By doing SSH into the EC2 machine you can use the same strategies that you have learned in section 2.2.2 of this tutorial to establish communication between S3 and EC2 and transfer whatever data you need from S3 to EC2 or vice-versa. 



Databases on AWS

Whether you're starting off with a new project or migrating an existing project onto AWS, there's a database for you.

In the last chapter, we saw how to launch EC2 (Electric Compute Cloud) instances of varying processing power as well as save data as files (objects) on the S3(Simple Storage Service) platform.

However, most applications do not store data in the form of files, at least not directly. Nearly all applications that store frequently accessed information store them in a database of one form or another. A database allows structuring of information and querying it, while abstracting the underlying storage implementation and search details.

Amazon offers a number of solutions to maintaining and storing your databases. Let us take a look at them one by one.

Bare Linux Machines as DB Servers

As most of you would have already guessed, you CAN in fact use the EC2 machine configured in the previous chapter as a database server (although not a very good one with such minimal specs).

This is mostly similar to what we do today where we load a VM and then install the database (Oracle, MySQL etc) on it, then run it from within the VM. Let us take a look at how we can achieve this using the EC2 machine that we configured earlier.

Start by connecting via ssh to your machine:

- `ssh -i "test-t2-micro.pem" <user>@<instance.public.ip>`

Next, install one or more required dependencies.

For this example, we will try to install MySQL DB on the EC2 instance. To do this, we need to get the installer for MySQL from the official registry, the command for this is as follows:

- `> sudo yum localinstall http://repo.mysql.com/mysql-community-release-el6-3.noarch.rpm`
- `> sudo yum install mysql-community-server`

That's it! The mysql-community-server has been installed to your EC2 machine. To start the service, simply run

- `>sudo service mysqld start`

Your MySQL server should be running on the default port on this machine. You can connect to this using any client and the default user and password.

[Please note: The security group configured for this EC2 machine must allow traffic on the port the server is listening to, to check this, select the instance from EC2 console, click on "Actions">Networking>Security Groups]

Try connecting to the machine using any DB Client on port 3306 with username "root" and no password. You have now successfully set up a database that can be accessible as long as the EC2 machine is. This database can now be used in your application to store and retrieve data without worrying about downtime and bootups.

However, a few concerns remain. Nobody is taking backups for you, there are no database management jobs that are running for it, the indexes are not being rebuilt. In short, the database is not being administered properly, it just exists.

Of course, one can argue that this is where the database admin walks in and flexes his muscles looking all-important and goes about doing his routine backups and optimization. Alternatively, you can leverage one of the many services on offer from AWS that does many of these for you. Let us take a look at what's on offer and the advantages they provide.

Relational Database Service (RDS)

AWS offers a completely managed version of multiple databases as a service. It is completely resizable and varies on the data you store such that you do not need to provision machines for it separately. You also end up paying only for the storage space and I/O operations that you use and not for the underlying VMs that are used. In addition to that, most database administration tasks are handled by Amazon itself including regular backups, point in time recoveries and trigger optimizations.

This allows the user to focus more on the data and its usage rather than getting tangled up in non-functional maintenance activities. Let us take a look at how to deploy and connect to a database instance on RDS.

Setup a database on RDS

- Open the “Services” tab from the AWS Console and navigate to ‘RDS’ under Databases.
- This should open up the Amazon RDS Console and default to the Dashboard. The dashboard lists all the running instances of RDS, click on any of them to view their details.
If there are none running, an additional popup introducing RDS opens up with the button “Get Started Now”. Click on this to open up a wizard that creates and launches your first RDS instance.
- The first screen on the wizard allows you to select the underlying database engine. AWS currently supports six database engines: Aurora, Oracle, PostgreSQL, MySQL, MariaDB and Microsoft SQL Server. Select any one of these according to your requirement and click ‘Next’. [Please note: Aurora is not supported under the ‘free tier’ and will be billed as per usage]
- The next screen brings up the DB details like engine version, licenses to be used, as well as the size of the underlying db instance (micro, medium, large etc). You can choose to select a multi-availability zone deployment as well which would allow your database to become more fault tolerant with lower latency. Next, select the storage class you want to use, (SSD

or Magnetic) and the amount of storage you want to allocate to your db and finally supply an identifier, a master username and password for root access to the db. Click 'Next' to continue.

- Finally, configure the security groups and VPC (if any) under which the DB should be available, select a name for the database (eg: empdb) and click on 'Launch'. Note there are a number of parameters related to the database administration that you can tune on this page, like backup scheduling and retention etc but for now continue with the defaults.

Clicking on launch will finish the wizard and create the database for you, navigate back to the RDS dashboard and you will see your instance listed there. To connect to it, simply use the endpoint provided under the connect tab and the port and you should be able to connect and access it exactly like a normal database!

Once the backup scheduler runs you will see that a snapshot of the database has been created. To view the list of available snapshots click on the tab labelled "Snapshots" on the left panel of the RDS console.

You should see a list of snapshots taken till date. Clicking on them will allow you to restore or copy them to a separate location or even migrate to a bigger database when required. These are listed under "Snapshot Actions".

Congratulations! You have now successfully created a managed RDS instance running your choice of database engine. It allows minimal management from the user and increased availability and scalability on the underlying infrastructure all the while respecting the same interface contracts of



the underlying engine. This is the recommended approach of setting up a relational database on AWS for production use. Do remember to check out the pricing page of RDS for information on how usage is charged.

DynamoDB

RDS is a great option to store relational information, but what if you do not have such data? What if you only have unstructured data to store? Non relational databases have been used to store and query such data before. An example of such a database would be MongoDB or Cassandra. So the question is, does Amazon Web Services also offer a managed database for non relational data. You bet it does!

DynamoDB is a fast and flexible NoSQL database service that provides consistent, millisecond latency at any scale. It is a fully managed database and supports both document and key-value store models. DynamoDB has a flexible data model and its reliable performance, along with automatic scaling of throughput capacity, allows it to stand out in the world of noSQL databases.

Let us now try to create our own table on DynamoDB and access it.



Creating a Table on DynamoDB

- Open the “Services” tab from the AWS Console and navigate to ‘DynamoDB’ under Databases.
- This should open up the DynamoDB Console and default to the ‘Tables’ list. It lists all the created DynamoDB tables, click on any of them to view their details. If there are none, an additional popup introducing DynamoDB opens

up with the button “Get Started Now”. Click on this to open up a wizard that creates your first DynamoDB table.

- The first screen on the wizard requires you to enter a name for the table and a primary key. The primary key is used to hash the data across hosts for scalability. Keep the default settings check box checked as it allows for minimal configuration for the table. Note that the dialog box below it says that auto scaling is not enabled by default, this can be overridden by changing the default settings. Click on ‘Create’ to create the table.
- The table is now created and you are redirected to the tables list. To check the items of the table open the ‘Items’ tab and search the items by typing in a query and clicking ‘Start search’.

To connect to DynamoDB table you can use one of the available SDKs for the popular languages available on AWS.

ElastiCache

Although frequently accessed data was traditionally stored in databases (relational or otherwise) the usual latency for disk-based operations associated with most databases became a blocker for some high throughput applications in recent years. This led to the rise of ‘in-memory databases’, a class of databases that store (most) data in-memory instead of on disks allowing for much lower latencies.

Nowadays, most highly available and fluid applications use in-memory databases in one way or another (usually to cache some data, ticker updates etc). Just like traditional databases, amazon also offers a managed variant of the two most popular in-memory databases Redis & Memcached.

Let us take a look at how to setup a Redis engine on ElastiCache.

Setting up ElastiCache:

- Open the “Services” tab from the AWS Console and navigate to ‘ElastiCache’ under Databases. This should open up the ElastiCache Console and default to the ‘Tables’ list. It lists all the created ElastiCache instances, click on any of them to view their details. If there are none, an additional popup introducing ElastiCache opens up with the button “Get Started Now”. Click on this to open up a wizard that creates your first ElastiCache in-memory database.
- The on screen wizard requires you to select either Redis or Memcached as the underlying in-memory database engine, here choose the Redis engine and then select a name for the cluster.

Create your Amazon ElastiCache cluster

Cluster engine Redis
 In-memory data structure store used as database, cache and message broker. ElastiCache for Redis offers built-in Auto Failover and enhanced security.
☐ **Cluster Mode enabled**

☐ **Memorized**
 High performance, distributed memory object caching system, intended for use in speeding up dynamic web applications.

Redis settings

Name

Description

Engine version compatibility 3.2.10

Port 6379

Parameter group default.redis3.2

Node type cache.m1.large (10.5 GB)

Number of replicas 2

Click launch to start the database cluster.

[Note: Just like DynamoDB there are a number of advanced settings that you can tweak here, such as shard count, replication factor etc, but for the purposes of this tutorial let's keep this at default settings] Once launched the Redis cluster is available on its configured port (default is 6379). To connect to it via aws-cli from a machine, enter the following command:

- `>redis-cli -h <hostname> -p <port>`
- Eg: `redis-cli -h test-cache.us-east-1.cache.amazonaws.com -p 6379`

This should allow you to login to the remote Redis server using the REPL (Read,Eval,Print and Loop) mode, where you can type further commands to modify data.

You have now successfully setup a managed Redis cluster, that can be used in all applications by specifying the given host and port. The cluster is fully managed, including auto scaling and distribution of shards and can be made available over multiple AZs.

In this chapter we saw how to install a database server on a barebone Linux machine and connect to it. We also worked on setting up our own managed relational database on Amazon RDS, a non-relational database on DynamoDB as well an in-memory database on AWS ElastiCache. Hopefully, we now have a better idea on how we can leverage the different database technology services offered by Amazon for our use cases. Needless to say, it is worth spending the time and effort to look at the pricing of each individual service on offer and compare it with alternatives (eg: own DB on EC2 vs managed RDS instance) for the optimum choice in each case. **d**



Route 53

Open your application to the world using Route 53, AWS' very own Domain Name System.

AWS Route 53 is a cloud based web service for Domain Name system and it comes with high scalability and availability like other AWS resources. For understanding what is AWS Route 53 first we need to gain some basic knowledge on Domain and Domain Name System (DNS).

A domain name system is a hierarchical and decentralized naming system for your websites. Suppose you want to launch your website or your web application. What is the first thing you need to do? You need to register the name of your website, which is known as the domain name. It is an identity of your website on the internet, which is mapped to your IP by a DNS.

When you search for a website over the Internet, a DNS translates the name of the site into a numeric IP address like 192.10.10.1.

Features of Route 53

Let's have a look at different components and features of a DNS.

- **DNS Query:** When you search anything on internet you type the name of the website in the address bar of your browser. This is known as a DNS query. Typical response of a DNS query is the IP address of the resource you have asked for.
- **DNS Resolver:** Your DNS query does not reach directly to the name server. An intermediary working between them is known as a DNS resolver.
- **Domain Name System:** It is the Name Server network responsible for domain names that helps worldwide devices in communicating.
- **Name Server:** Job of name servers is to translate the domain names in your DNS query into IP addresses.

Routing of Internet Traffic to Your Application using Route 53

Each device communicates with other devices using IP address on the internet. But you just need to type the name of the web site; rest of the things are taken care by DNS. Route 53 helps in making that connection between domain names and IP addresses.

Configuration of Route 53 for routing traffic for your Domain

You need to follow below steps to configure your Route 53

- First you need to register your domain name using Amazon Route 53 console. Users will use this domain name to access your resource on the internet.
- Then Route 53 will create a public hosted zone which will have the same name as your domain.
- Then you should create your resource record set in that hosted zone created by Route 53. This is the set of records that contains detailed information regarding how traffic would be routed to your resources. It includes following information –
 1. **Name:** name of the records that will be routed by Amazon Route 53. Every name must end with host zone's name.
 2. **Type:** Type specifies the resource type to which traffic will be routed. For example, to route to a webserver type will be A.
 3. **Value:** Value contains the detailed information depending on type.

Routing of traffic by Route 53

Now you have successfully configured Route 53 to route traffic to your resources. Your job is done. Let see how Route 53 works.

- A user wants to visit your website (for example your site address is www.myapp.com). He enters www.myapp.com in his browser
- At first this request goes to the DNS resolver which is managed by the user's Internet Service Provider.
- Then this resolver sends this request to a DNS root name server. This root name server gives the name server address for the top-level-domain. In our example, it is '.com'.
- Now, the DNS resolver sends the request to the name server for this TLD.
- TLD name server returns the name of Amazon Route 53 name servers associated with your web site as response.
- Next task of resolver is to send this request to Route 53. Now route 53 will check hosted zone and finally return the IP address for www.myapp.com. DNS resolver also caches the IP address for this site for a period.
- Now user's browser has the IP address and sends the request to it.
- In return it receives the web page and displays it in the browser. All these activities are done within a few seconds.

Health check of your resources by Route 53

An interesting feature offered by AWS Route 53 is Health Check. You can use Route 53 for continuous monitoring and health checking of your AWS resources. It checks if your resources are reachable and functional by sending requests in regular intervals over the Internet. You can also receive notifications if any of your resources stop working.

- You should define how the health check should work.
 1. First mention the IP addresses of the resources that you want to be monitored.



A DNS is your gateway to the Internet.

2. Amazon Route 53 will follow the protocol given by you. For example, you can choose TCP, HTTP or *HTTPS* as per your requirement.
 3. You have the freedom to define the interval of the health check request.
 4. Finally Route 53 can also set a CloudWatch alarm for you. Whenever any resource is detected as unhealthy you will receive a notification. CloudWatch internally uses Amazon SNS to send notifications.
- After you have configured all the values for the health check, Route 53 starts sending requests to your resources in the given interval. Until the resources stop responding it is considered that the resource is healthy.
 - Now if a resource does not respond to health check request, the request is considered as failed and Route 53 starts keeping a count of the failed requests. If this count crosses a certain threshold value, then the resource is considered as unhealthy. This threshold value needs to be predefined by you. But if the resource starts responding before the threshold value is reached Route 53 will reset the count to 0 and will not notify you.
 - Suppose a resource is detected as unhealthy. Then if you are subscribed for receiving notifications it will notify you. Don't worry if you are not configured for notifications, you can still check the health status of your resources in Amazon Route 53 console.

Customisation of Health Check

You can also customize this health check as per your needs.

- Suppose you have multiple web servers for your application so that you don't need to worry if one server is down. Others can easily handle its work. Then you configure Route 53 such that you will be notified only if a certain number of resources are unhealthy.
- You can also configure your Route 53 such that it will route traffic to the healthy resources only.

Route 53 Health Check Type

Route 53 health check can be categorized in three types

- Health check of an endpoint: This is the basic type of health check. It just checks if your resources are available and functional or not.
- Health check of CloudWatch alarm: You can configure a health check for Cloudwatch alarm also.
- Health check to monitor other health checks: You can also configure a health check for monitoring other health checks created for your endpoint resources.

Getting Started with Route 53

You need to follow some guideline for proper utilization your Route 53. Let's see how it needs to be configured.

Setting Up Amazon Route 53


- Sign in AWS account with your root user. Your root user by default have access to all AWS resources.
- Next step is to create an IAM user group. It is not recommended to use root user for any day to day work. So, you should create an IAM user group and give it access to use Route 53
- Then set up AWS command line interface. You can access AWS services using AWS CLI.
- Route 53 provides many APIs for your use. But it will be better if you download the SDK for your programming language. SDKs have some advantages over these APIs. It makes authentication processes much simpler yet stronger and also makes integrations easier.

Start your Amazon Route 53

- First step is to register a domain for your website.
- Then configure an S3 bucket so that it can host a website.
- Now create your website. And it should be uploaded to your S3 bucket.
- Set up the other configuration such that DNS traffic for your domain name will be routed to your bucket.
- Last but not the least set up configuration for health check. You should also subscribe for Cloudwatch notification.
- Now check if your website is working as per your expectations.

So, it can be concluded that three basic features offered by Amazon Route 53 are

- **Register Domain name**
- **Routing internet traffic to AWS resources**
- **Health check of resources**

You can use any of the said services or combination of the services. For example, you have a simple web application. You can go for Domain name registration and routing of internet traffic. But if your application is critical and needs to be always available you should go for all the said services. So, choose the services you need and open your app to the world! 



AWS Lambda

Lambda and Step Functions – Serverless code!

What is AWS Lambda?

You have worked very hard on a difficult piece of code and now are confident with it. You want this code to be executed every time an event (say API call) occurs. Why would you now need to write non-productive and time-consuming server deployment, thread management and concurrency handling code, allocate resources and memory for it? Would it not be simply great if by some magic you could just ‘give’ your code to a genie that handles all of this for you? That is the beauty and power of AWS Lambda. Lambda gives you the freedom to run your code without managing any server. It runs your code on a highly available compute environment and also, manages servers, maintains operating system, monitor your code and do logging for you. Lambda will always scale your code automatically based on the number of requests that hit the lambda

service. You just need to pay for the computing time spent by the lambda in running your code.

As of today, lambda supports Java, Node.js, C# and Python, all of which can be used to quickly write and deploy application logic. This logic can then be executed in response to events called “Triggers” that are sent from a variety of sources like S3, CloudWatch, API Gateway etc.

Lambda gives you the provision of building Serverless applications in a matter of minutes, that are guaranteed to scale and only charge for its usage.

Things to know about Lambda:

- The program or code you run on AWS Lambda is known as a Lambda Function. Lambda functions are ready to be invoked as soon as they are created. Along with your code you need to specify some additional information like function name and resource requirement. These functions are not bounded to the resources required to run them so they are also called Stateless functions. Lambda functions can be configured to be integrated with other AWS resources like Amazon S3 bucket.
- Your Lambda function needs to be triggered by an event or function. Any AWS service which triggers your Lambda function is called as Event Source.
- After your Lambda function is triggered it may call any other AWS service. We call these services as Downstream Resources.
- AWS has its own model that defines serverless application. It is called AWS Serverless Application Model or AWS SAM.
- AWS provides three key tools that can be used for interacting with AWS Lambda services:
 1. Lambda Console
 2. AWS CLI
 3. SAM Local

Key Features of Lambda:

- **Custom backend services** – You can easily add some custom logic to your AWS resources or backend services using Lambda functions. Suppose you have an Amazon S3 bucket. You need to run some activity whenever data is changed in your S3 bucket. Just write a lambda function that will be triggered by that data change.
- **Build your own code** – Using lambda you can include any third-party library. No need to learn a new language, new tool or any new framework.
- **Automated administration** – You supply the code and Lambda runs

your code in a highly available environment. All the maintenance and administration related work is taken care by Lambda. It also does logging and monitoring of your code.

- **Fault tolerance** – Lambda has built in fault tolerance capability. Since there are no servers to manage, virtually no downtime occurs.
- **Automated scaling** – AWS lambda scales your code automatically. For example, if usage of your code increases, you need not worry about increasing the servers. Lambda will take care of that.
- **Run complex or long tasks easily** – You can coordinate with multiple lambda functions for running a complex or long tasks. AWS Step function will help you to build a workflow and run these tasks.
- **Integrated security model** – you can integrate your Lambda with IAM service provided by AWS. IAM service will take care of access management on behalf of you. Your AWS resources will be secured.
- **Pay per use** – You will be charged only for the time that has been taken for computation by your code.
- **Flexible resource allocation** – You will have the freedom for defining memory allocation for your code. Other resources like CPU, network bandwidth will be allocated accordingly.

When to use Lambda:

Before going with Lambda, you need to understand when you should really use lambda. Below are some use cases appropriate for AWS Lambda.

- If you need to process your data after certain events like shifts in date, some requests made by end-users, data change in your data store. You can go for Lambda.
- Another scenario where Lambda fits perfectly is real time file processing. If some codes need to be executed as soon as a file is uploaded, you should definitely go for Lambda.
- Serverless backend can be built using Lambda when you need to handle a web request or 3rd party API request.
- Lambda can also be used to perform data validation, data sorting and data transformation triggered by some modification in your data store.
- You can make your web application highly scalable and tolerant to faults using Lambda.

Lambda function versioning and aliases:

AWS Lambda comes with an advanced feature of versioning and aliases to

make the developers' life much easier. You can store and publish more than one version of your lambda function. It gives you the flexibility of working with different versions of the same Lambda function in different stages of your development workflow. However, a version is like a snapshot of the code and once published, it cannot be altered to make changes you need to make a new one. Every version of Lambda function will have a unique Amazon resource name (ARN) for its identification. You can also make aliases for each and every version of the lambda functions.

Lambda and CloudWatch:

Lambda comes with CloudWatch to make the process of monitoring it much easier. CloudWatch keeps an eye on your Lambda functions on behalf of you. Lambda keeps track of the number of requests that come in, as well as the number of request that result in errors. You can also configure for custom CloudWatch alarms on the basis of these counts.

Request rates and error rates for each function can be viewed using AWS Lambda console or CloudWatch console.

Generally, we insert logging statements in our program to check if they are working as expected or not. Now these logs can be pushed to the CloudWatch log associated with your Lambda function. Lambda does this automatically when it has permission to write in CloudWatch log. You need to grant this access through your IAM user group.

Getting started with AWS Lambda:

Now let's get started with Lambda services

- At first, you need to login into your AWS account.
- Then set up and configure your AWS Command line interface.
- Next step is to install SAM Local. This tool makes your Lambda function development easier.
- Now create a simple Lambda function using AWS Lambda console.
- For testing, you need to invoke your Lambda function manually.
- And the final step is to verify the result returned by your function.
- For better utilisation also check CloudWatch metrics.

Some best practices for AWS Lambda:

Function code –

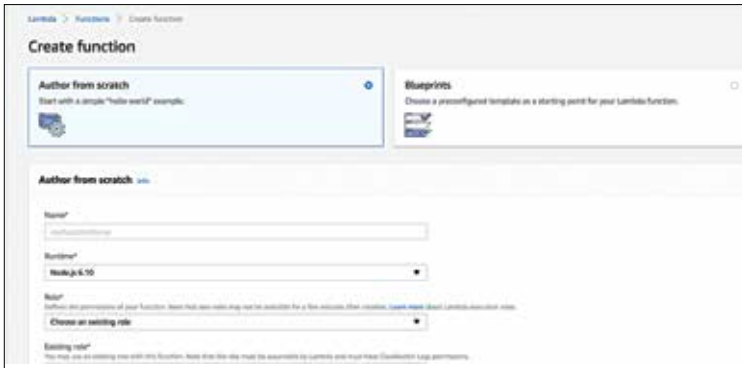
1. Your Lambda handler is the entry point of your Lambda function. It should be always separated from the core logic section for making your

code unit-testable. For Example, let see a Lambda function in Node.js.

```

•   exports.myHandler = function(event, context, callback)
{
•   var foo = event.foo;
•   var bar = event.bar;
•   var result = MyLambdaFunction (foo, bar);
•
•   callback(null, result);
• }
•
• function MyLambdaFunction (foo, bar) {
•   // MyLambdaFunction logic here

```



There are numerous pre-configured lambda templates to choose from.

2. Try to improve performance of your Lambda function by re using container.
3. You should always use an environmental variable for passing parameters.
4. Dependencies in your functions should be controlled by packaging them all with deployment package. But also, keep this in mind that the size of package should not exceed certain limits.
5. It is better not to use recursive code.

Function configuration –

1. Both performance and load testing should be done for your Lambda function.

2. Your IAM user group should be restrictive. It should grant the access as required.
3. All the Lambda functions that are not being used should be deleted.

Alarm and Metrics –

1. You should use AWS Lambda metrics and configure CloudWatch alarm.
2. Integrate your program logging with CloudWatch logs for better performance.

Use Lambda VPC –

1. If your Lambda function needs to access any resources within VPC then you should go for Lambda VPC.



Once you've created your lambda function, you can then set up triggers.

Event Invokes –

1. Your Lambda function should be tested with different batch sizes and record sizes.
2. You should use dead letter queues for finding async errors.

Limits of AWS Lambda:

- Lambda has certain limitations in terms of memory allocation range, disc capacity, number of processors and threads. You can browse these limitations from the tables in the next page.
- If any of the limit exceeds Lambda function will fail with exceeds limit error. The only solution to this error is reducing the size of your code.

AWS Step Function:

Most applications perform a series of complicated tasks that should be divided in different components. Each component performs some specified task. AWS Step Function is a service that helps you co-ordinate among the components of your application using a visual workflow. It comes with a graphical console. There you can visualise every component and steps performed by each component.

State of each step is also logged by Step function. So, whenever anything goes wrong you can easily diagnose and debug your functions.

Like other AWS resources, it also comes with a console. Your task can be run anywhere that has access to AWS. You can access Step function through this console.

AWS LAMBDA RESOURCE LIMITS PER INVOCATION

Resource	Limits
Memory allocation range	Minimum = 128 MB / Maximum = 3008 MB (with 64 MB increments)
Ephemeral disk capacity	512 MB
Number of file descriptors	1,024
Number of processes and threads (combined total)	1,024
Maximum execution duration per request	300 seconds
Request body payload size (synchronous invocation)	6 MB
Request body payload size (asynchronous invocation)	128 K

AWS LAMBDA ACCOUNT LIMITS PER REGION

Resource	Default Limit
Concurrent executions	1000

AWS LAMBDA DEPLOYMENT LIMITS

Item	Default Limit
Lambda function deployment package size	50 MB
Total size of all the deployment packages per region	75 GB
Size of code/dependencies that you can zip into a deployment package	250 MB
Total size of environment variables set	4 KB

Key features of Step Function:

- Concept of Step function is inspired by tasks and state machines.
- Step function comes with a visual display. Step Function Console shows the graphical view of your workflow. It makes the debugging process easier.
- State machines are defined using Amazon's own language called Amazon State language. It is nothing but a simple Json based language.

Some use cases of Step Function:


Any computational problem or business process can be divided in some steps. Step function help co-ordinate among those steps.

- You can use step function for data processing application where you need to unify data from multiple sources.
- Now in IT industry everyone is talking about automation. Everything should be automated. You can use step function to automate your software release process. It will help you in continuous and automatic deployment of your software.
- You can use Step Function for E-commerce where you need to automate critical business process.

Getting started with Step Function:

You need to follow the below steps to create a step function:

- First you need to create a lambda function and define IAM group.
- Then create Lambda State machine with this Lambda function and start a new execution.
- Now create a job status poller.
- Create a task timer and start state machine using CloudWatch event.
- Finally create a step function API.
- Your step function is ready to use.

We can conclude that AWS Lambda gives you the opportunity to run your application without providing a server or physical compute resources explicitly. You need not worry about server management, operating system maintenance, logging or monitoring your application. Just go and deploy your code. You can concentrate your efforts only in building a great and reliable application. Along with Lambda, Step Function gives you the provision of graphical visualisation of your application. You can also keep track of each and every step performed by your application. So, let's build cost effective serverless application! 



Big Data and Kinesis

Sift through massive amounts of data using Amazon Kinesis.

The term “Big Data” was coined way back in 1998 by a scientist in the silicon valley, but only in 2013 was it finally adopted into the Oxford English Dictionary indicating that the technology had finally come of age. Today, Big Data remains one of the most hyped buzzwords in the industry. But when can data be deemed ‘Big’?. To put this into perspective consider this, in 2018 the world is expected to generate roughly 50 Terabytes of data every second! A lot of this data is never going to be utilised in any manner, and even usable and meaningful data will take a long time to process, clean, and store, before it becomes fit for use. This means that valuable insights might be delayed (or worse, lost) simply because we do not have the capability to process the volume of data required to extract it.

Kinesis

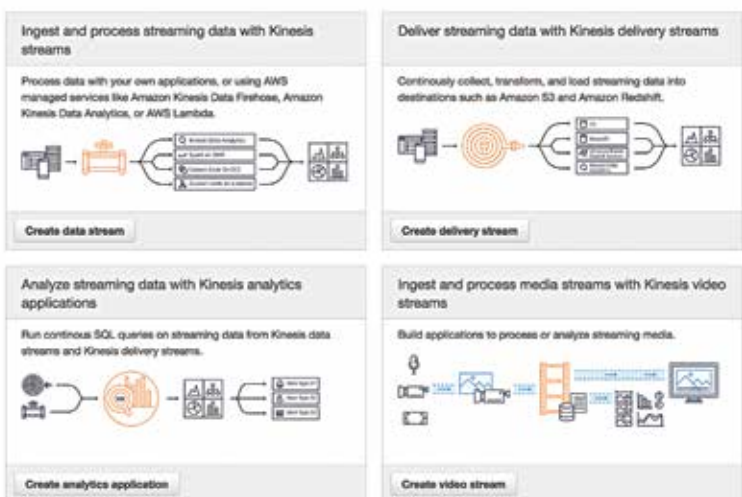
Most of our problem solving techniques hinge on our ability to take informed decisions (informed being the keyword here) and without proper and meaningful information at the right time, doing that becomes a very difficult task. Processing large volumes of continuously flowing data in real-time is a very complex and computationally exclusive task. This is where Amazon Kinesis comes in.

Kinesis simplifies the task of processing continuous data streams from a number of sources, extracting meaningful data from them. It allows easy processing of video & data streams and querying of the streaming data with simple SQL syntax.

To create a stream select the Kinesis data stream from “Services” and click on “Create” under the Streams tab. A wizard opens up which allows you to fill the necessary information to create your data stream. A stream represents an ordered sequence of data records. When you create a stream, you must specify a stream name and a shard count. A stream consists of one or more shards; each shard is a group of data records.

Once you click ‘launch’, a new data stream is created and ready for you to ingest data.

Amazon Kinesis resources



A little primer on what Kinesis does.

Now let us take a look at how to go about flowing data along your stream.

A Kinesis data stream usually has at least two components, a Data Producer and a Data Consumer, we will briefly touch upon these two and their uses in the following sections.



Start off your journey with Kinesis.

Data Producer

The data producer can be anything that adds data to Streams, such as an EC2 instance, client browser, or mobile device. In most real-life scenarios, the data consumer and data producers are two separate applications using Streams to communicate asynchronously.

To add data into a stream, producers call the PutRecord operation. Each PutRecord call requires the stream name, a partition key, and the data record that the producer is adding to the stream. The stream name determines the stream in which the record will reside. The partition key is used to locate the shard of the stream that the data will then be added to.

The choice of partition key depends on the application. The only thing to note is that the number of partition keys should be much greater than the number of shards. A high number of partition keys to shards helps the stream distribute data evenly across the shards in your stream.

Data Consumer

Data consumers are workers that retrieve and process data from the stream. Each consumer reads data from a particular shard. Consumers pull data from a shard using the GetShardIterator and GetRecords operations.

A shard iterator denotes the position of the stream and shard from which the each consumer will read. A consumer gets a shard iterator when it starts reading from a stream or changes the position from which it reads records from a stream. To get a shard iterator, consumers must provide a stream name, shard ID, and shard iterator type. The shard iterator type allows it to specify where in the stream it would like to start reading from (such as from the start of the stream where the data is arriving in real-time). The stream has an option to return the records in a batch, whose size you can control using the limit parameter.

The data consumer creates a table in DynamoDB to maintain state information (such as checkpoints and worker-shard mapping) for the application. Each application has its own DynamoDB table. The data records, once processed, can then be sent to a more concrete storage like S3 or DynamoDB. Or, they can be used for real-time analytics and discarded later.

Kinesis Client Library (KCL)

Applications can also use the Kinesis Client Library (KCL) to simplify parallel processing of the stream. The library takes care of the complex tasks such as load-balancing across multiple instances, responding to failures, checkpointing, and reacting to resharding. The KCL enables you to focus on writing record processing logic.

The data consumer provides the KCL with position of the stream that it wants to read from, the library then calls `GetShardIterator` on behalf of the consumer. The consumer component also provides the client library with the processing logic for the incoming records using an important KCL interface called `IRecordProcessor`. The KCL calls `GetRecords` on behalf of the consumer and then processes those records as specified by `IRecordProcessor`.

Elastic Map Reduce (EMR)

MapReduce is a framework for processing large datasets in parallel using a large number of instances (nodes), collectively referred to as a cluster or a grid (based on location and specification details). MapReduce jobs can run on both file system records as well as database entries. A typical MapReduce job consists of two steps, Map and Reduce. In the “Map” step, each worker (node) applies the “`map()`” function to the data, and writes the output to a temporary storage. MapReduce supports distributed processing of the map and reduction operations. Provided that each mapping operation is independent of the others, all maps can be performed in parallel – though in reality this is limited by the number of CPUs allocated to each source.



Once you have your data ingested, use MapReduce to sort and filter it.

Next, in the “Reduce” step, the workers process each group of data, per key, in parallel.

A set of ‘reducers’ can also perform the reduction in parallel, provided that all outputs of the map operation that share the same key are presented to the same reducer at the same time. While this process can often appear inefficient compared to algorithms that are traditionally more sequential (since the reduction process need not be broken down then), MapReduce can be applied to much larger datasets. The parallelism also allows a possibility of recovering from partial failure of jobs during the operation. i.e. if one mapper or reducer fails, the work can be reassigned to another one. In many situations, the input data might already be distributed among many different servers, in which case the map job becomes greatly simplified by assigning map servers that would process the locally present input data.

AWS provides a fully managed framework that makes it easy, fast, and cost-effective to process large amounts of data across scalable Amazon EC2 instances. You can also run popular distributed frameworks such as Apache Spark, HBase, Presto, and Flink in Amazon EMR, as well as interact with data in other AWS data stores such as Amazon S3 and Amazon DynamoDB.

Amazon EMR can be easily applied to a number of big data applications, including log analysis, web indexing, data transformations (ETL), machine learning, financial analysis, scientific simulation, and bio-informatics. An EMR cluster can be set up on AWS in a matter of minutes. Let us take a look at how we can do that.

Setting up an EMR Cluster on AWS:

- Navigate to ‘EMR’ from the ‘Services’ tab. On opening the EMR console, you will be presented with a home screen that lists your current clusters. If there aren’t any, click on the “Get Started Now” button.
- Click on “Create Cluster” to bring up the wizard that allows you to create



Create your EMR cluster.

an EMR cluster. You will be asked to provide a cluster name, as well as the choice of starting up a cluster or a complete step function that would start the cluster, run the job and terminate it on completion. Let’s go with the second option.

- Configure the number of instances you would need as well as their capacities. Please note that the underlying EC2 instances will be charged separately.
- Once selected, you will be asked to configure the cluster type. To do a map reduce data streaming job, select the streaming option and configure the location of the mapper and reducer. Set the location of the output jobs and logs, and you are good to go.
- Click the next button to launch the cluster, you will see metrics of the cluster including the running time, current status, as well as ARN. You can monitor the cluster state and once the mapping and reducing jobs are complete, the cluster should automatically terminate.



Point out the location of the mapper and the reducer.

Thats it! You have now successfully launched an EMR cluster that has run a MapReduce job over a distributed system and stored the results in an S3 bucket.

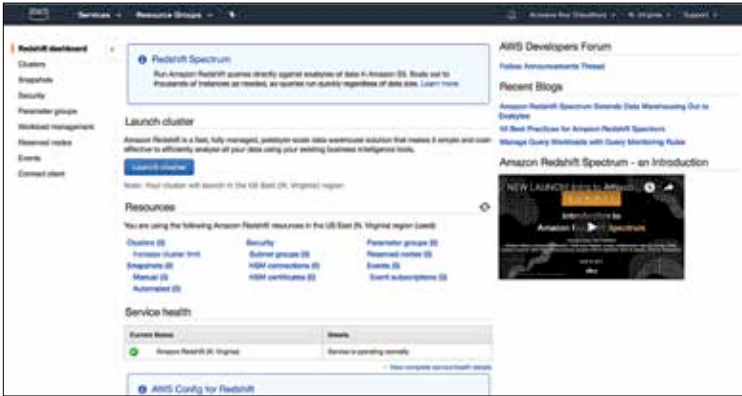
Redshift

Of course, all this processing and computation is meaningless unless you have a means of storing the big data in a way that allows you to access it as and when needed. Traditionally this was done either in file systems (ugh), or databases. Both of them however, need to be extensively tweaked to maintain speed and throughput in the face of the very high volume that big data can potentially generate. AWS can help you here too. The Redshift data warehouse service is a very fast and fully managed solution that scales to petabytes and makes it simple and cost-effective to efficiently analyse all your data without changing the choice of your Business Intelligence (BI) tools.

You can very easily setup your own redshift warehouse and start logging all your data into it.

Setting up AWS Redshift:

- Navigate to “Redshift” from the “Services” tab. On opening the Redshift console, you will be presented with a home screen that lists your



Launch a new Redshift cluster to save all your processed data.

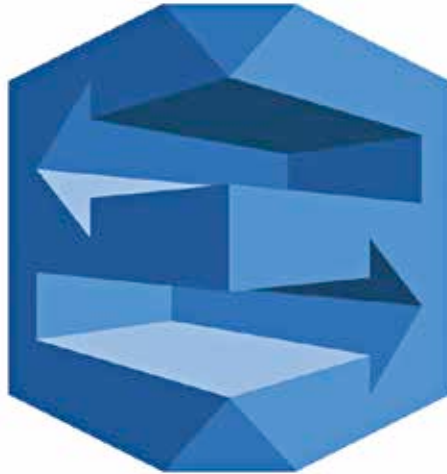
current warehouse clusters. If there aren't any, click on the “Launch Cluster” button.

- Fill in the details on the next screen such as the cluster identifier that uniquely identifies the redshift cluster, the ‘database name’ that can be used to connect to the redshift database, as well as the port that can be used for the connection (defaults to 5439). Set a master username and password.
- Next, set the node types (these vary in compute capacity and number of EC2 per node) and the number of nodes required for the cluster.
- Finally, on the next screen fill out the security details such as choice of VPC, security groups and availability zone etc. Click continue to deploy your cluster.

[Please Note: Deploying a cluster might incur high charges, unless you are eligible for the ‘free tier’. Please double check configuration before deploying.]

The cluster will take a few minutes to spin up. Once running, you can connect to it using a JDBC driver and the url `jdbc:redshift://<cluster-arn>:<port>/<database-name>`

Now you can start adding data to the cluster as well as connect all your BI tools to the redshift cluster for running analytics. **d**



Amazon Lex

Everyone's getting a chat bot and they're quite easy to build, especially if you know Amazon Lex.

Have you ever come across chat boxes on some websites which allow you to type in questions and help you navigate the website? Many such chats are run by computer programs known as 'bots' that process the user's questions, understand them, and then come up with answers. These bots or 'Chatbots' as they are more commonly referred to as, have been around for sometime and they have been put to a number of uses, from customer interactions, to interpreting user commands from more 'natural' conversations.

Amazon Lex

Amazon offers a service that allows you to build conversational interfaces into your applications using voice and text. This service, named ‘Amazon Lex’, uses the same deep learning engine that powers the now popular Amazon Alexa so that developers can leverage the full potential of the powerful engine for their own use case. Amazon Lex provides both natural language understanding (NLU) and automatic speech recognition (ASR) to enable easy development of highly engaging applications that can have almost lifelike conversations.

Getting Started with Amazon Lex

To start creating your first bot on Lex, navigate to the ‘Services’ tab and then go to Amazon Lex under it. Click on the ‘Create’ button under the list of bots.

A wizard opens up that lets you select either a template or a blank bot schema where you add the bot name. Let us select the ‘OrderFlowers’ bot template that creates a sample bot that can be used to accept orders for flowers. A list of sample sentences called ‘utterances’ would be populated for your bot such as “I would like to order flowers” or “I want to order some flowers”. These are sentences that would invoke the chat bot. You do not need to explicitly map all possible utterances, as amazon uses machine learning to effectively deduce the meaning of your sentence and tries to map it to one of the utterances. These utterances then fire a set of events known as Intents. Intents control the flow of application logic and choose the type of programming logic to execute for a given utterance. Additionally, an utterance might also contain slots that take in user data for processing, such as ‘I would like to order some roses’ would map to an utterance that contains a slot for the last word so that the application knows what kind of flowers the user is trying to order. The bot can also guide the user to add inputs using a series of sentences known as ‘prompts’. An example of a prompt would be “What kind of flowers would you like to order?”

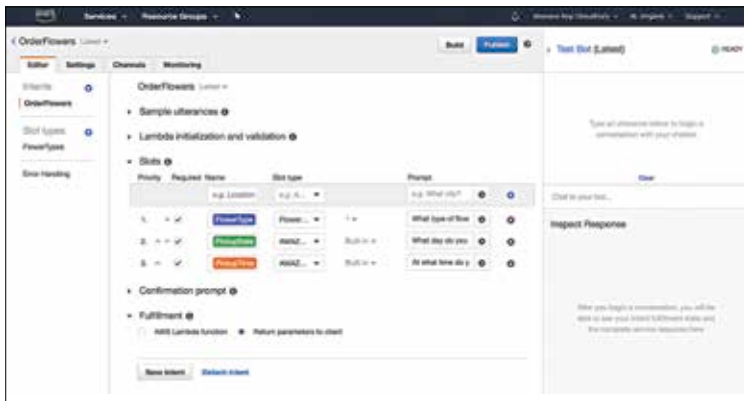
Once all the required data is collected, a bot can fire one or more ‘Confirmation prompts’ that are sentences that let the user



Build your own chat bot with Amazon Lex.

confirm or deny a task much like a submit button. An example of a confirmation prompt would be: 'Your roses are expected to be delivered by Sunday morning. Does that sound okay?' Once the user replies in affirmative, the task can be executed or else a failed task status is generated. Confirmation prompts differ from regular prompts in the sense that they would always ask a binary question that lets the user decide between two choices.

If a confirmation is received, an Amazon Lex bot can trigger a Lambda function by passing it the data it captured. The output of the lambda can then be propagated to the user from the bot. This would effectively allow the bot to carry out the user's 'request' and return an appropriate response for it. Like ordering flowers from a nearby store.

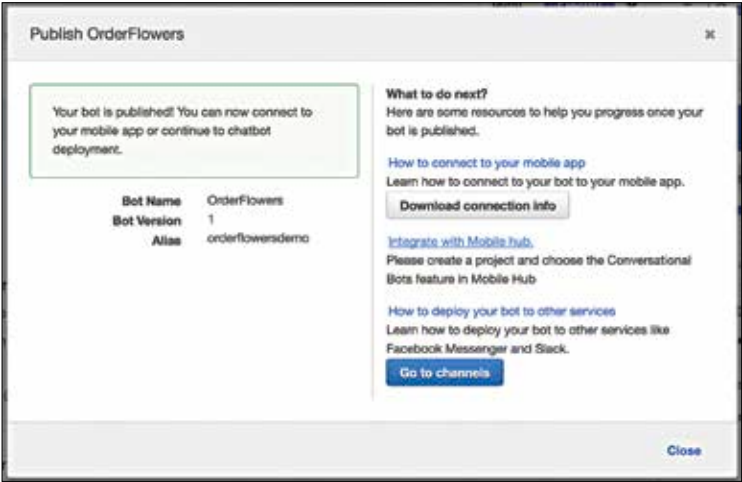


Set up utterances in your Lex bot to trigger responses.

Once a bot is ready, it can be integrated with any app or with an existing tool like Facebook Messenger, which will allow users to interact with the bot right from within the facebook messenger application.

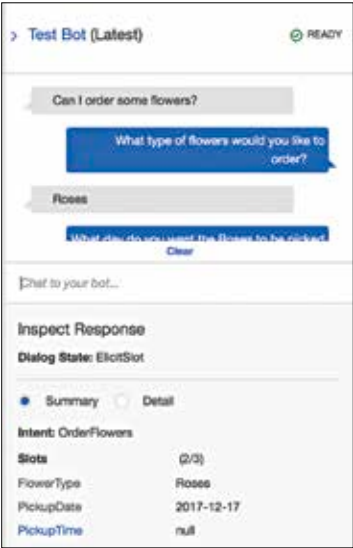
Integrating an Amazon Lex bot with Facebook Messenger:

- Open the newly created Amazon Lex bot from the dashboard and click on the channels tab.
- Choose Facebook under Chatbots. The wizard then displays the Facebook integration page.
- Fill up the following details on the integration page to continue:
 - Name of the Bot Association that you want to create, eg: Example-Association.



Once your bot is published, you can move to deployment.

- For Alias, choose the bot alias from the dropdown.
- For Verify token enter any string you choose (for example, ExampleToken). You will need to use this token later in the Facebook developer portal when you set up the webhook.
- For Page access token, type the app token that you have obtained from the Facebook developer portal.
- For App secret key, enter the app secret key that you have obtained from the Facebook developer portal.
- Click on Activate. The console then creates the bot channel association and returns a callback URL. Save this URL as you will need to set this on the Facebook app.
- On the Facebook developer portal, click on your app to select it. Choose



Test your Lex bot.

the Messenger product, and choose Setup webhooks in the ‘webhooks section’ of the page.

- On the webhook page of the wizard, fill up the following details:
 - For Callback URL, type the callback URL that you saved from the Amazon Lex bot.
 - For Verify Token, type the same token that you used in Amazon Lex (eg: ExampleToken).
 - Choose Verify and Save. This will save your configuration and run an initial test on the integration. If everything works well you should have the chat bot integrated within the Facebook app.

Amazon Polly

While amazon lex allows us to process and interpret natural conversations into commands, the results are still output as text. However, for a true conversational experience, dialogue needs to be bi-directional. That is, if we are capable of talking to our devices to get things done, they should be able to talk back to us! Amazon Polly allows us to do just that by providing a text-to-speech service that converts written text into realistic and lifelike speech thus making it possible for computers to converse with their human users.

Polly uses advanced deep learning technologies to synthesize speech that sounds like a human voice instead of a boring and unrelatable robotic translation. Amazon Polly also supports multiple languages and includes a variety of lifelike voices, so that you can tweak your speech-enabled applications to work in multiple locations and use the ideal voice for the local audience.

Amazon Polly has three main components:

- **Input text** – This is the input to the service. Here we provide the text to synthesise, and Amazon Polly returns an audio stream. You can provide the input as plain text or in Speech Synthesis Markup Language (SSML) format. SSML allows us to control various aspects of speech such as pronunciation, volume, pitch, and speech rate. For example you can add a whispered word within a sentence, or add a pause break after a phrase.
- **Available voices** – Amazon Polly provides multiple languages and a variety of voices for each of them. For most languages you can select from both male and female voices, and to do this you only need to specify the voice name when calling the service, and then the service uses this voice to convert the text to speech. However, Amazon Polly will not

be doing a translation for your text—the synthesised speech is in the language of the text.

- **Output format** – Amazon Polly can deliver the output speech in a variety of formats. For example, you might request the speech in the MP3 format to consume in web and mobile applications. Or, you might request the PCM output format for AWS IoT devices and telephony solutions.

Getting Started with Amazon Polly:

Now that we saw the main components of Amazon Polly, let us take a look at how to test it on AWS console and finally how we should go about adding it to our applications.

• Testing Polly on AWS Console

- To test Amazon Polly on AWS, open up the Services tab and navigate to Polly. Select the Text-To-Speech tab and choose plain-text or SSML based on your requirement.
- Type a single line of text in the box below, such as “Hello! I am Polly.”
- Select the region as English (Indian) and click ‘Listen to Speech’. You should be able to hear your sentence converted to speech.

The screenshot shows the 'Text-to-Speech' console interface. At the top, it says 'Listen, customize, and download speech. Integrate when you're ready.' Below this is a large text input box containing 'Hello! I am Polly.' To the right of the box are links for 'Show default text' and 'Clear text'. Below the input box, there are two tabs: 'Plain text' (selected) and 'SSML'. Under the 'Plain text' tab, there are dropdown menus for 'Language and Region' (set to 'English, Indian') and 'Voice' (with radio buttons for 'Adri, Female' and 'Fareena, Female', where 'Fareena, Female' is selected). To the right of these are buttons for 'Listen to speech' (blue), 'Download MP3' (grey), and 'Change file format' (blue). Below the main controls, there is a section titled 'Customize pronunciation' with a link to 'Learn more' and a text area for 'Apply lexicons' with an 'Upload lexicon' button.

Synthesize speech using Amazon Polly.

• Amazon Polly using AWS CLI

- To start using the Amazon Polly API, you would need to have Amazon

CLI installed on your machine. In case you don't have it, please install it before continuing further. The steps to install Amazon CLI have already been discussed in the introductory chapter.

- Once the AWS CLI is set up, try running the below example command:

```
• >aws polly synthesize-speech \
  --output-format mp3 \
  --voice-id Aditi \
  --text 'Hello, I am Polly.' \
  hello.mp3
```

In the above command, the call to `synthesize-speech` allows you to convert your text into audio. To do this, you provide sample text for the synthesis (`--text`), the voice to use (`--voice-id`), and the output format (`--output-format`). The command saves the resulting audio to the `hello.mp3` file in the current working directory. You can play the resulting `hello.mp3` file to verify the synthesised speech. To get the list of available voices use the `DescribeVoices` operation.

```
• >aws polly describe-voices
```

However, you would likelier need to use Polly from within your application. To do that, Amazon provides Software Development Kits or SDKs in multiple popular programming languages like Java, Python etc that can be used to integrate Polly within your application in minutes.

• AWS SDK for Python

Let us take a look at a sample Python application that uses Polly to convert a text into an mp3 file.

Start off by importing all the required dependencies for the application. These include `Session` class from `boto3` and `contextlib` from `closing`.

```
• from boto3 import Session
from botocore.exceptions import BotoCoreError, ClientError
from contextlib import closing
import os
import sys
import subprocess
from tempfile import gettempdir
```

Next, we create a session object by passing it a credentials profile and get the polly client from it.

```
• session = Session(profile_name="adminuser")
polly = session.client("polly")
```

Then we try to generate an mp3 response using the polly client by calling the “synthesize_speech” function and passing it the following parameters: Text=“Hello! I am Polly.” as the text to be converted OutputFormat=“mp3” and VoiceId=“Aditi” for the voice to be used.

```
• try:
    response = polly.synthesize_speech(Text="Hello! I am
Polly.", OutputFormat="mp3",VoiceId="Aditi")
except (BotoCoreError, ClientError) as error:
    # The service returned an error, exit gracefully
    print(error)
    sys.exit(-1)
```

If the response has returned, we check to see if it contains an audio stream, and finally if it does, we try to write the contents of the stream into a file named “Hello.mp3” to the temp directory and exit. Otherwise, if the response does not contain an audio stream we terminate the application.

```
# Access the audio stream from the response
if "AudioStream" in response:
    # Note: Closing the stream is important as the service
    throttles on the number of parallel connections.
```


```
    with closing(response["AudioStream"]) as stream:
        output = os.path.join(gettempdir(), "Hello.mp3")
        try:
            file.write(stream.read())
        except IOError as error:
            print(error)
            sys.exit(-1)
```

```
else:
    # The response didn't contain audio data, exit gracefully
```



```
print("Could not stream audio")  
sys.exit(-1)
```

The above application thus uses a set of credentials to connect to AWS Services using the AWS SDK for Python and then converts the input text ("Hello! I am Polly") into audio and proceeds to save it as a file named Hello.mp3 to a temporary directory.

Congratulations! You have now successfully created an Amazon Polly integrated application that can convert text into lifelike speech. For further exploration, we urge you to look into more details regarding SSML as it allows a much finer grained control over the speech nuances, such as pauses, pitch, volume etc. Also, another thing worth mentioning are 'Lexicons'. Lexicons are simply text to audio mappings that you can add to your Polly app (upto 5 each app) that allows it to speak a text in a certain way. This allows you to customise the output of the Polly engine to even greater detail. For example, you can add a lexicon to say 'toh-mah-to' instead of 'to-may-to' (or vice versa) when Polly encounters the word 'Tomato'. You could even add your own choice of words and their equivalent lexicons eg: 'Balle balle' or 'Oribaba'. The true potential of Amazon Lex and Polly come to light when both of them are used in unison to create an amazing chatbot that is capable of having a natural conversation with the user and responding to its requests in a way that till recently only humans could. 



Machine Learning, Rekognition and SageMaker

Thinking of building your own predictive model on AWS? Amazon Machine Learning is what you need.

A

busy street, thousands of commuters cross the area with only a single CCTV camera to cover it, when suddenly a flag is raised on the system for a potential terrorist presence in the area.

Within minutes, multiple quick response teams converge

and apprehend the now confirmed terrorist possibly preventing a number of future crimes and saving lives.

Amazingly, the helm of the operation, the person responsible for identification of the criminal is not a human at all, it is in fact, a computer program tasked with monitoring the secure CCTV channels across the city and looking for known criminals.

Sounds like the stuff of fiction? Welcome to Moscow, the year is 2017.

The last decade saw a new technique of computer programming emerge that would take the world by storm. Instead of programming a computer to specifically process a set of data in a certain way, what if we could show it a large set of input and outputs so that it is able to figure out (with some degree of approximation) a programming logic that would generate those results given the inputs? That is the essence of machine learning.

Unlike classical programming approaches, machine learning does not seek to explicitly determine the logic flow of the application and instead uses iterative mathematical optimisations to determine a near approximate function, known as a model that converts the inputs supplied to the mapped outputs. This model is then used to evaluate real life inputs with the assumption that it would generate an output that is close to the actual output.

A detailed study on machine learning is out of the scope of the tutorial but we will try to clarify most terminologies and processes as we go through our example. However, a basic understanding of machine learning and mathematics would definitely help the reader to better utilize the offerings on AWS.

Amazon Machine Learning (ML)

Amazon Machine Learning (Amazon ML), lets you build and train predictive models and host your applications in a scalable environment. It also allows you to process input datasets for better optimization of training.

Getting started with Amazon ML:

- To start creating your own model on Amazon ML, go to the Services tab and open 'Amazon Machine Learning' service from it. Click on the 'Get Started Now' button and you will be redirected to a standard setup wizard.
- First, you have to add training data for your model, to do this add your data to a S3 bucket and provide the file URL in the S3 data section on the input page. Alternatively you can also specify Redshift as a datasource for your model.



Head over to the AML dashboard to get started.



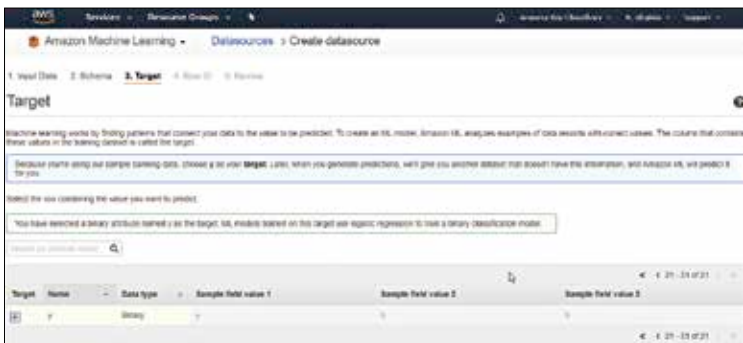
Feel Amazon ML your data.

- On the next screen, Amazon analyses your data to generate a schema that lists the input types (boolean, numeric etc). This is to better allow pre-processing of the data and make sure your training inputs are exactly the way they want to be.
- Next, we specify the target for your model, i.e. the output that you want to generate from the data. Of course, this has to be part of your training data as well and based on your model it can be a binary, categorical or numeric input.
- Finally, add a row identifier for your data if you have one. This is purely for referencing purposes and does not contribute to the model. Review your data on the next screen and if everything looks fine, you can complete the datasource configuration.
- Next, we configure the ML model. The first screen allows you to choose the ML model type (binary, categorical, numeric etc) and a name for it.



	Name	Data type	Sample field value 1	Sample field value 2	Sample field value 3
1	age	Numeric	age	44	33
2	job	Categorical	job	other job	business
3	marital	Categorical	married	married	married
4	education	Categorical	education	high school	education
5	default	Categorical	default	education	age

Sift through the schema details that AML generates based on your data.



Target	Name	Data type	Sample field value 1	Sample field value 2	Sample field value 3
Y		Binary	Y	Y	Y

Specify the target for your data model.

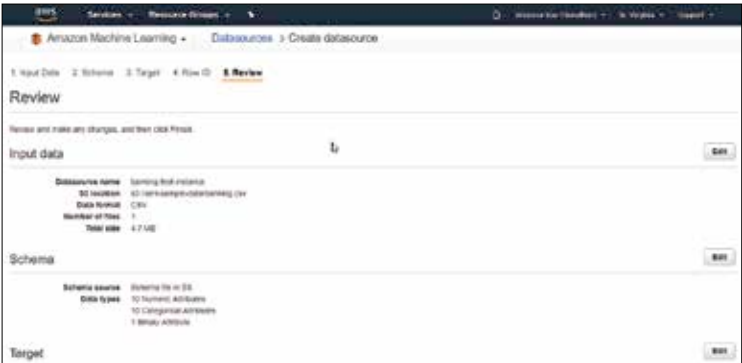


Row identifier (optional)	Name	Data type	Sample field value 1	Sample field value 2	Sample field value 3

A row identifier is optional but helpful in the long run.

The target output is also filled based on the input data source that we just configured.

- We then configure what we call a 'amazon recipe' for pre-processing of the input data as well as splitting it into training, test and validation datasets. Pre-processing also allows us to run basic operations on data



Finally review your data model before proceeding forward.



Configure the ML model type and give it a name.

such as normalization, filling missing values with mean or computing Cartesian product of two variables as a means to better process and fit our training data.

- Click next and deploy the ML model and you have a live machine learning model that can now churn out predictions based on input. It can accept both batch predictions as well as requests over a rest API.
- You can test your model by clicking on try out prediction (batch or real-time) and run a couple of test cases.

We have now successfully created and deployed our own machine learning model that takes in input data from a S3 source, processes it, and then creates a model by training on it.

A lot of these steps, if done manually require significant knowledge of



Summary of your data model prior to deployment.



Test your model by letting a batch prediction run a couple of test cases.

mathematics and time, but amazon machine learning service abstracts most of it to allow developers to concentrate mostly on the problem statement rather than the implementation. In fact, with the help of this service, even someone with little to no knowledge of machine learning can set up an at least functional (if not good) machine learning application.

Rekognition

Now that we saw how to create predictive models using machine learning, we would be very excited about its possibilities, football fans for instance can try to think of a model that would accurately predict scores or wins in a match or governments can use weather data from sensors to try to predict crop harvest cycles. However, a large number of real world problems require humans to 'look' at it with our own eyes. These problems cannot be

re-written as a set of data points for a program to use. An example of such a problem would be our ability to recognise faces. Even though infants are able to do it (to some degree at least) we find it hard to quantify how exactly we recognise a face we have seen before. What factors are at play when we are trying to do it and which face similarity qualifies as recognition. These problems when translated for computers require them to 'look' at images in a certain context and come up with answers. They differ from data related problems to some degree in the approach towards their solution. Amazon offers a set of services under its 'Rekognition' service to tackle machine learning problems that involve images. We will now take a look at how to go about building applications on it.

Amazon Rekognition allows you to just provide an image (or video) as input to the Rekognition API, and the service can then identify certain objects, people, text, scenes, and activities within it, as well as detect any inappropriate content. It also provides highly accurate facial analysis and facial recognition and can detect, and compare faces for a wide variety of use cases such as user verification, security, etc. Rekognition uses a highly scalable, deep learning technology developed by Amazon's computer vision scientists that is capable of analysing a huge number of images and videos daily, and just like Amazon ML requires almost no machine learning expertise to use. It is designed as a simple and easy to use API that can quickly analyse any image or video file stored in Amazon S3. Amazon Rekognition is always learning from continuous feedback data that is automatically leveraged for all future runs.

Getting Started with Amazon Rekognition

- To start using the Amazon Rekognition API, you would need to have Amazon CLI installed on your machine. In case you don't have it, please install it before continuing further. The steps to install Amazon CLI have already been discussed in the introductory chapter.
- Once the AWS CLI is set up, try running the below command with a test image to ensure that Rekognition is working correctly.
- ```
>aws rekognition detect-labels --image "S3Object={Bucket=<test-bucket>,Name=example.jpg}" --region us-west-2
```

This would run Rekognition against an example image located at `s3://test-bucket/example.jpg` and return the labels of objects found in it (person, building etc).



- To use rekognition using the CLI we specify a number of parameters as listed below:
  - Service type: This denotes the type of Rekognition service that you want to use, ie whether you are attempting to detect objects, analyze faces, recognize them or read text from an image. Some of the currently supported types of operations are compare-faces, detect-faces, detect-labels, detect-text, recognize-celebrities, etc.
  - Image location: This points to the location of the image in S3. The image must have permissions that allow it to be read by rekognition. It is also a good idea to host the images in a region that has rekognition available to prevent additional charges due to data transfer between regions
  - Region: This specifies the region against which rekognition should run. As mentioned earlier, this should (ideally) be the same region the image is hosted in, but as of today rekognition is only available in a select few regions.
  - Profile: This specifies the credentials profile that would be used by AWS CLI to make calls to AWS services.

An example request as below:

```
• aws rekognition detect-labels \
--image '{"S3Object":{"Bucket":"bucketname","Name":"object.
jpg"}}' \
--region us-east-1 \
--profile adminuser
```

Has the service type as detect-labels which is used for label detection, the image location as “bucket1” under S3 bucket name and object.jpg and image name. The specified region is us-east-1 and the profile credentials to be used is adminuser.

Of course, these APIs can also be called from within applications and Amazon provides Software Development Kits or SDKs in multiple popular programming languages like Java, Python, etc., that can be used to integrate Rekognition within your application in minutes.

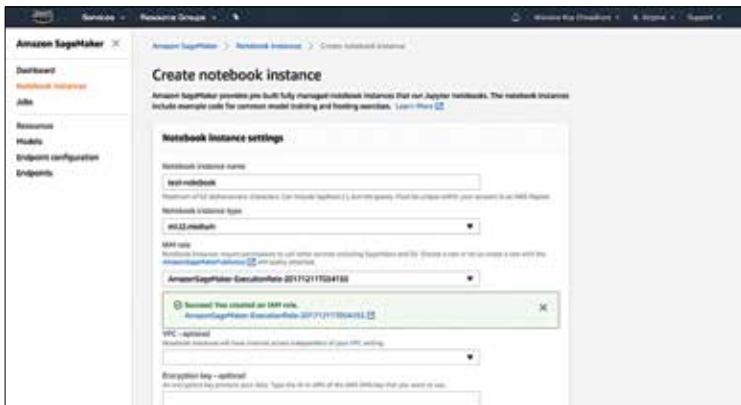
## SageMaker

In addition to these managed services, amazon also offers a much more open and sandbox like environment that can be used by data scientists and developers to train, test and deploy their own machine learning algorithms (or use some common ones) in a production ready scalable environment.

Amazon SageMaker allows directly deployment of machine learning models from training into a production-ready hosted environment. It provides an integrated Jupyter authoring notebook instance for easy access to your data sources for exploration and analysis, so that you do not have to have a separate server for it. It also provides common machine learning algorithms with native support for custom algorithms and frameworks. Amazon SageMaker also offers flexible distributed training options that allow you to autoscale your training jobs according to a specific workflow.

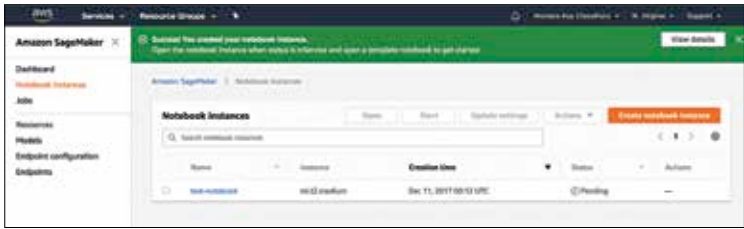
## Getting Started on Amazon SageMaker:

- Navigate to 'Amazon SageMaker' under Services tab and click on 'Create new notebook instance' on the SageMaker console. This allows you to create a new notebook instance that can be used to design your machine learning model.
- Fill out the notebook instance name and assign a role to it and click continue to successfully create the notebook. Optionally you can also specify an encryption mode for it.

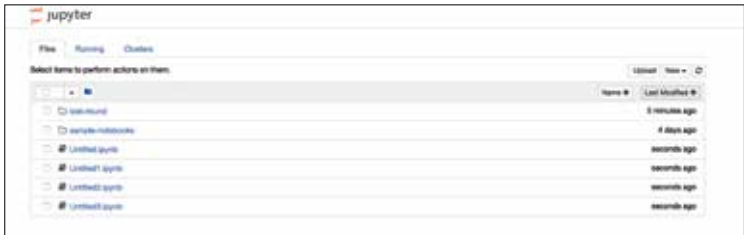


Create a new notebook instance on the SageMaker console.

- Once the notebook is created, you can choose to open it from the 'Notebook instances' tab list by clicking on the entry and then clicking the 'Open' button. This opens up the jupyter notebook where you can design the actual machine learning application.
- When the notebook is opened, click on the 'New' button and start writing out your algorithm in a platform of choice (spark, mxnet etc), you can



Your newly created Jupyter notebook.



You can write your own algorithm or pick one of the samples.

also download one of the samples from Amazon if you don't want to write your own.

- Once satisfied with the algorithm you can spin up a Amazon SageMaker training job that will start training your model on the input dataset you provide.
- Once the model is trained you can convert it into an endpoint from the dashboard by selecting the 'Endpoints' tab list and creating one that uses the current model.

Although this might appear similar to Amazon Machine Learning, SageMaker offers a much more finer grained control over the choice of algorithm, data transformations and output processing that the former. This of course would mean that a degree of machine learning expertise is required to effectively utilize SageMaker and hence it is primarily targeted at people who have some experience in this regard. **d**



# AWS IOT

Use Greengrass to get all your devices to communicate with each other.

**T**echnology has made our life more convenient and enjoyable. The Internet of Things enables physical devices, home appliances, vehicles, software to connect and communicate with each other. Aim of IoT is to bridge the gap between the physical world and digital world. The size of the IoT universe is expected to be 20 billion connected devices by 2020. AWS offers a set of services tuned to the specific domain of IoT.

### Key features of AWS IoT:

Below are some of the features that make AWS IoT stand apart.

- **AWS Device SDK** – You can connect your hardware devices or mobile application using AWS IoT device SDK. It helps your devices to connect, authenticate and exchange data.

- **Device Gateway** – Your devices connect with each other and AWS cloud securely using Device Gateway. Device gateway uses the publication/subscription model for exchanging messages. It gives IoT devices the provision of broadcasting data to multiple subscribers.
- **Authenticate** – IoT implements authentication and encryption at every point of the connection. It supports AWS' own method of authentication and also certificate based authentication. You can create certificates and deploy these certificates for your devices.
- **Registry** – AWS provides Registry for keeping track of your devices. Each and every device has a unique identifier in Registry. And you can also store attributes and capabilities of your devices in form of metadata.
- **Device Shadows** – AWS IoT allows you to create a virtual version of your devices called shadows. Shadows reflect the last reported state and future desired state of your device. Even if the device is offline, you can still get its last reported state from its shadow.
- **Rule Engine** – You can build an IoT application that will collect, analyse, process data and accordingly perform some actions on devices. All of these can be done with the help of rule engine. It will evaluate data and perform task as defined in your rule.

### Services of AWS IoT:

- **AWS IoT core** – AWS IoT core is the cloud platform for Internet of Things. It helps to create a secure and reliable connection between your devices and other cloud based AWS services. IoT core can also collect data coming from your devices and accordingly perform some action.
- **AWS IoT Device Management** – AWS IoT device management helps you to manage your devices including tasks such as onboarding and sending remote updates. Devices can be organised into groups and monitored using device management console.
- **AWS Greengrass** – Using AWS Greengrass, you can run local compute, send data through secure connection between devices, even when they are not connected to the internet.
- **AWS IoT Analytics** – You need not worry about the cost and complexity for building your own IoT analytics platform. AWS provides a service for running analytics known as AWS IoT analytics.
- **Amazon FreeRTOS** – It is a FreeRTOS based operating system for small microcontroller based devices. You can securely connect these devices to AWS IoT core or more powerful edge devices.

- **AWS IoT 1-Click** – AWS IoT 1-Click helps your devices to trigger AWS Lambda functions. All you need to do is download the 1-click mobile app and then select the device that will trigger the lambda function. Then create the Lambda function. IoT 1-Click also provides tracking reports for your deployed devices.
- **AWS IoT Button** – Simply a programmable button. The logic for this button can be coded in the cloud for performing some action. For example, this can be programmed to order your favourite food, remotely control your home appliance or book a cab.
- **AWS IoT Device Defender** – AWS IoT Device Defender is used for continuous auditing of the security policies on your devices.

### **AWS Greengrass:**

AWS Greengrass is the most powerful service offered by AWS IoT. It enables local execution of AWS Lambda functions. It can collect, analyse data and accordingly perform an action closer to the edge. AWS Greengrass uses the publication/subscription model for message transfer. When a connection is lost it can buffer messages so that incoming & outgoing message are preserved. Greengrass also protects user data as each and every device is securely authenticated and local devices are connected to the cloud through secure connections.

AWS Greengrass core is the central part of Greengrass. It consists of a message manager for routing messages between devices and the cloud. It also has a lambda runtime for running Lambda Functions. Then there's 'Thing Shadow' that contains last reported state of a device. And lastly, Deployment agent, which restarts Greengrass core if the AWS Greengrass Group Configuration is updated.

### **AWS Greengrass Groups:**

Aws Greengrass group consist of Greengrass core and the devices you want to communicate with Greengrass core. Greengrass group setting consist of IAM group role for AWS Greengrass, configuration for logging and configuration of local connection. Additionally, it also tracks information on connectivity of the Greengrass core.

### **How Does AWS Greengrass work?**

AWS Greengrass group builds connections between your devices and also with the IoT cloud. Greengrass group consist of a Greengrass core and

devices that you want to monitor or control. Greengrass core act as the central hub. It performs the local execution of Lambda functions, transfers messages between devices, enables device shadows etc. A device should run Linux and should support ARM for hosting Greengrass.

Greengrass core can work with other devices as well. Especially, those that have Amazon FreeRTOS or IoT Device SDK. These devices can trigger the execution of AWS Lambda function locally even if IoT-core is unable to connect with the cloud.

### **Greengrass ML inference:**

Now it has become possible to run Machine Learning locally with Greengrass Machine Learning inference capability. Let's say you are running a hotel, and want to locate your VIP customer and give them special treatment. What will you do? With Greengrass ML you can store a facial recognition model and run it on in-store cameras. When your camera identifies a VIP guest, it will trigger a lambda function. You can then inform your staff to tend to the VIP guest or display a greeting at the door. You need not to send huge amounts of data to the IoT cloud to achieve this. It can be done locally.

### **Benefits of AWS Greengrass:**

- You can have real-time response as the devices act on data locally. Should the need arise, they use advanced AWS features as well.
- It allows your devices to operate even when there is no internet connectivity.
- Your data is always secure with AWS Greengrass. It encrypts data for cloud communication and local device to device communication. It uses mutual device authentication and authorisation at every point of network.
- AWS Greengrass uses the same Lambda programming you are familiar with. Additionally, it gives you the provision of running that Lambda function locally.
- It is one of the most cost effective solutions for IoT. Greengrass performs some action on the data that your device collects. So, all the data need not to be transmitted to the application in cloud. This significantly lowers the cost and increases the quality of the data.

### **Some Use cases of Greengrass:**

- We have already seen in the previous section that you can use Greengrass in hospitality and retail industry for giving better treatment to your special guests.

- AWS Greengrass can also be used in traffic cameras. For example, it can count vehicles and pedestrians and accordingly adjust traffic signal time periods.
- You can use Greengrass in medical imaging equipment. Greengrass and machine learning inference can be utilised for making some prediction locally. It might speed up the process of medical diagnosis.

## Getting Started with AWS Greengrass:

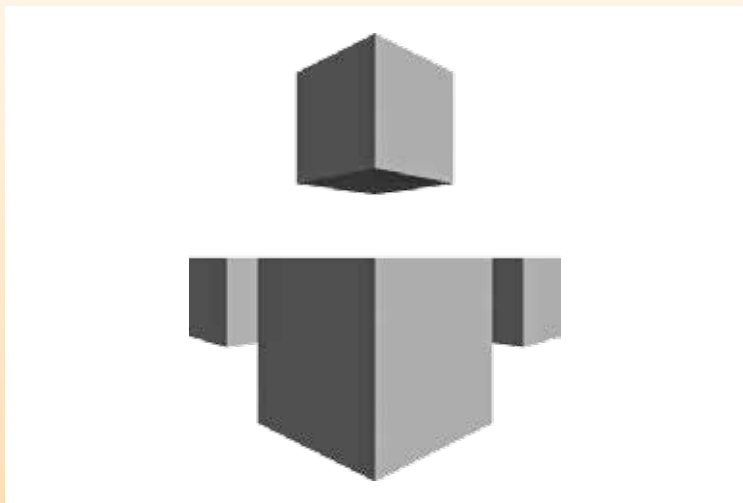
Let's check how you can create your own AWS Greengrass. Below are some steps you need to follow.

- After you log in to your AWS account first you need to set up the environment for your Greengrass.
  1. Set up a Raspberry Pi – Setup your Raspberry Pi and also check for all dependencies required for AWS Greengrass. If you already have a Raspberry Pi it will be better if you re-image it with Raspbian Jessie operating system.
  2. Set up EC2 instance – Launch your EC2 instance along with the appropriate security groups.
  3. Set up other dependency devices. So, the environment is ready to welcome your Greengrass.
- Install Greengrass core. You need to configure Greengrass in AWS IoT first.
  1. Set up your Greengrass group.
  2. Name your group
  3. Create a core function for your group
  4. Run a scripted easy group creation
  5. Connect your core device.
- Now start your Greengrass device.
- Next step is to create and configure a Lambda function in your Greengrass.
  1. On the group overview page select option Lambda and then choose “Use existing Lambda” and select the Lambda you have created in the previous step.
- You need to deploy your Lambda function and also corresponding subscription configuration. While this setup process is going on your AWS Greengrass should be connected to the Internet. And also check if your Greengrass daemon has been started or not.
- After your Lambda is deployed you need to test your Lambda function by publishing messages through AWS IoT console.



- Finally, your AWS IoT devices need to be created and tested.
  1. Select your group through the AWS IoT console and add a device in the group configuration page.
  2. Create a Registry entry for your device and create a certificate and policy according to your need.
  3. Now select a source and destination for publishing messages. As Greengrass uses publisher/subscriber model for messaging, your source will be the publisher and the target will be the subscriber of the messages.
  4. Then you need to install the correct IoT Device SDK. For example, you can install AWS IoT Device Python SDK from Git repository.
  5. Test if your publisher and subscriber can communicate with each other or not.
- Your basic set up for AWS Greengrass has been completed. But the last two steps will help you to utilise some advanced feature of AWS Greengrass such as device shadow.
  1. Two devices are required for this. You can use devices created in previous steps.
  2. Connect AWS Greengrass core to internet so that each AWS IoT device shadow will be in sync with AWS IoT
  3. Test communication between these devices when sync is enabled and also when sync is disabled.
- This is the last step which will guide you in the most important aspect of AWS Greengrass, accessing AWS cloud services. For example, your device will communicate with DynamoDB in AWS cloud.
  1. As you are accessing AWS Cloud resource as well as Lambda function you must create an IAM role that grant access for both DynamoDB and you Lambda function.
  2. Then you need to create Lambda function. You can also use the one you have already created.
  3. Configure subscription for communication.
  4. Last but not the least test the communication.

So, you are ready to go with AWS Greengrass. Build your own solution for real life problems. 



# Amazon Mechanical Turk

Don't like the way machines handle your data? You can always get a human to do it instead.

**T**his document provides a conceptual overview of Amazon Mechanical Turk for developers who want to write html scripts using the Amazon Mechanical Turk Requester API.

Amazon Mechanical Turk is a web service that provides an on-demand, scalable, human workforce to complete jobs that humans can do better than computers, like recognising objects in photographs or video-graph or any other task.

## The major sections of Amazon Mechanical Turk:

1. What is Mechanical Turk
2. Set up the accounts and tools for Mechanical Turk
3. Use the Mechanical Turk SDKs to write and publish a Human Intelligence Task (HIT)
4. Implementing Mechanical Turk

## What is Mechanical Turk?

Amazon Mechanical Turk software formalises job offers to the thousands of Workers willing to do work at their convenience. The software also retrieves work performed and compiles it for you, the Requester, who pays the Workers for satisfactory work (only). Optional qualification tests enable you to select competent Workers.

The kinds of tasks humans can complete better than computers include finding objects in photos, writing reviews of restaurants, movies, or businesses, translating text passages into foreign languages, getting the hours of operation of the business centre within a hotel, determining if a hotel is family-friendly, or telling you the most relevant search results.

### Salient features:

1. On-demand workforce—Amazon Mechanical Turk provides access to a virtual community of Workers.
2. Create jobs that Workers perform over the Internet—Advertise your job to the thousands of Amazon Mechanical Turk Workers around the world
3. Test and publish your jobs—Test your jobs in the Amazon Mechanical Turk sandbox and publish the revised jobs to the outside world.

## Amazon Mechanical Turk Concepts:

### Requester

A Requester is a company, organisation, or person that creates and submits tasks (HITs) to Amazon Mechanical Turk for Workers to perform.

### Human Intelligence Task

A (HIT) is a task that a Requester submits to Amazon Mechanical Turk for Workers to perform. Each HIT has a lifetime, specified by the Requester, that determines how long the HIT is available to Workers. A HIT also has an assignment duration, which is the amount of time a Worker has to complete a HIT after accepting it.

## Worker

A Worker is a person who performs the tasks specified by a Requester in an HIT.

## Assignment

An assignment specifies how many people can submit completed work for your HIT. When a Worker accepts a HIT, Amazon Mechanical Turk creates an assignment to track the work to completion. The assignment belongs exclusively to the Worker and guarantees that the Worker can submit results and be eligible for a reward until the time the HIT or assignment expires.

## Reward

A reward is the money that you, as a Requester, pay Workers for satisfactory work that they do on your HITs.

## Set up your development environment.

### Step 1: Sign Up for an AWS Account

To develop solutions using the Amazon Mechanical Turk web service, you must first sign up for an AWS account.

### AWS Security Credentials

AWS uses access keys to help protect your data when you access AWS programmatically. An access key consists of two parts: an access key ID, which is similar to a user name, and a secret access key, which is similar to a password.

You use access keys to sign programmatic requests that you make to the Amazon Mechanical Turk API using a supported AWS SDK. You can use these access keys in both, the sandbox and the production environments.

To manage your AWS security credentials:

1. Sign in to the Amazon Web Services website at <http://aws.amazon.com/security-credentials>.
2. Do one of the following:
  - a. If you've signed in using your root credentials, choose Continue to Security Credentials.
  - b. If you signed in as an IAM user, choose Access Keys (Access Key ID and Secret Access Key).
3. Choose Create New Access Key.
4. In Create Access Key, choose Download Key File.

5. The private key file (rootkey.csv) is automatically downloaded by your browser. Save the private key file in a safe place.

## Step 2: Create a Requester Account

Before you can use Amazon Mechanical Turk, you must have an Amazon Mechanical Turk Requester account.

To create and register a Requester account go to <https://requester.mturk.com>.

## Step 3: Link Your AWS account to your MTurk Requester account

Next, you will need to link your AWS Account to your MTurk Requester Account. This operation grants permission to your AWS Account to access your Requester account using the MTurk APIs.

1. Go to <https://requester.mturk.com/developer>.
2. Click Link your AWS Account and sign-in with your AWS Root user email address and password.

## Step 4: Create an IAM User

With AWS Identity and Access Management (IAM) you can securely control access to AWS services and resources for your users. Creating an IAM user allows you to use these credentials to access the Amazon Mechanical Turk API instead of using your root account credentials. Please note that Mechanical Turk does not currently support the use of IAM role credentials.

To create an IAM user

1. Sign in to the Amazon Web Services website at <http://aws.amazon.com/security-credentials>.
2. Choose Get Started with IAM Users.
3. Choose Create New Users.
4. Under Enter User Names, type names for each of your users.
5. Make sure Generate an access key for each user is selected and choose Create.
6. Choose Show User Security Credentials.
7. Choose Download Credentials.
8. Your browser automatically downloads the private key file (credentials.csv). Save the private key file in a safe place.

## Step 5: Set up the Developer Sandbox

You should test your HITs in the Amazon Mechanical Turk sandbox testing

environment to make sure they work as expected before publishing your HITs in the Mechanical Turk marketplace. The sandbox provides a testing environment where you can publish and work on HITs to try them out before publishing them in the Amazon Mechanical Turk Marketplace. The sandbox consists of a Requester sandbox website and a Worker sandbox website.

You will need to create a Requester account on the Requester sandbox website, which is located at <https://requestersandbox.mturk.com>. You'll also need to create a Worker account on the Worker sandbox website located at <https://workersandbox.mturk.com> to view your sandbox created HITs as a Worker. There is no charge for using the Mechanical Turk sandbox sites.

## Step 6: Set up an AWS SDK

The MTurk API can be accessed using the following AWS SDKs: Python/Boto (Boto3), Javascript (NodeJS or Browser), Java, .NET, Go, Ruby, PHP or C++.

Configure the AWS SDK to use the 'us-east-1' region. This is the region in which the MTurk API is available. Then connect to the MTurk Developer Sandbox and check your account balance (the Sandbox should always return a balance of \$10,000). To connect to the MTurk Developer Sandbox, set the API endpoint in your SDK to <https://mturk-requester-sandbox.us-east-1.amazonaws.com>.

## Step 7: Prepay for Your HITs

Before you use the production version of Amazon Mechanical Turk you need to prepay for the HITs you create. Otherwise, you can't post your HITs to Workers. To post your HITs to Workers, you must have money in your Prepaid HIT Balance to prepay for all of your HITs. You can use a credit or debit card or debit card to prepay for the HITs.

## Creating an HIT

The following procedure gives you an overview of creating, testing, publishing, and managing an HIT.

### 1. Define your HIT.

Construct your question in one of the Question and Answer Data Structure. The Question parameter accepts a XML string or HTML string.

### 2. Create HIT.

Build a new HIT with the CreateHIT operation or CreateHITwithHITType operation. Provide Title, Description, Keywords, and Question Details as outlined in the documentation.

### 3. Test your HIT.

Publish your HIT on the Amazon Mechanical Turk Developer Sandbox. The Amazon Mechanical Turk Developer Sandbox is a simulated environment that allows you to view your HIT as it would appear to Workers.

### 4. Publish your HIT on the Amazon Mechanical Turk production system.

This step makes your HIT available to Workers.

### 5. Workers accept your HIT and complete the assignment.

You can view the status of your HITs with AWS Shell or the AWS CLI.

### 6. Process the assignment results.

When a Worker completes an assignment, you can view the results, output the results to a file, and accept or reject the work. Accepting the work means that you agree to pay the Worker.

### 7. Manage your HIT.

You can extend the completion time for your HIT, expire the HIT early, add additional assignments, modify the HIT properties, or block Workers whose work does not meet your standards.

## Coding Resources

1. Click the Developer tab on the Requester website located at <https://requester.mturk.com/> to get to the Developer Resources page, which has links to sample code, documentation, the sandbox, and other helpful information.
2. You can use code samples as a means of understanding how to implement the Amazon Mechanical Turk API. For more information, go to Amazon Mechanical Turk Sample Code and Libraries.
3. Recommendation is you look at the Amazon Mechanical Turk Forum to get an idea of what other users are doing and to benefit from the questions they've asked.

## Configure my application to use the Developer Sandbox.

- To use the Developer Sandbox during development and testing, set the URL service endpoint in your code based on the web service interface you are using:
  - SOAP service endpoint: <http://mechanicalturk.sandbox.amazonaws.com/?Service=AWSMechanicalTurkRequester>
  - REST service endpoint: <http://mechanicalturk.sandbox.amazonaws.com>
- To use the Developer Sandbox for testing HITs using external questions, set the destination URL for the form action to be: <http://workersandbox.mturk.com/mturk/externalSubmit>

- To switch your application to the production Mechanical Turk system, set the URL service endpoint to:
  - SOAP service endpoint: <http://mechanicalturk.amazonaws.com?Service=AWSMechanicalTurkRequester>
  - REST service endpoint: <http://mechanicalturk.amazonaws.com>
- To switch your HITs using external questions to the production Mechanical Turk system, set the destination URL for the form action to be: <http://www.mturk.com/mturk/externalSubmit>

## Sample python example on how to setup your very own HIT in Mechanical Turk.

1. Use boto3 package to access all AWS services in python.
2. Create a MTurk client using boto3 and provide the sandbox endpoint. This way your HITs are created in the sandbox environment.

```
12 sendbot_url = "https://workersandbot.murk.com/turk"
13 sendbot_post_url = "externalSubmit"
14 sendbot_get_url = "getService"
15 sendbot_request_url = "https://turk-requester.amazonaws.com/"
16 st_url = "https://st.amazonaws.com/"
17 sqs_queue_name = "technical_turk_result"
```

```
mark = boto3.client('mark',aws_access_key_id=aws_access_key_id,aws_secret_access_key=aws_secret_access_key,region='us-east-1')
s3 = boto3.client('s3',aws_access_key_id=aws_access_key_id,aws_secret_access_key=aws_secret_access_key)
```

3. Create a html equivalent python string object containing details about your HIT.

[illegible]

```
167 #/usr/bin/ls
168 #Answer:Spells/Faculties
169 #/usr/bin/ls
170 #/usr/bin/ls
171 #Answer:Spells/Faculties
172 #/usr/bin/ls
173 #ls
174 #/usr/bin/ls
175 #/usr/bin/ls
176 #/usr/bin/ls
```

4. Create a form tag in the html question and make sure to submit the form to requester MTurk endpoint.
5. Create a HIT using the MTurk object and be sure to supply in the HIT



title, description, html question in string format, max assignment value, reward value, HIT time in seconds, maximum HIT expiration time.

```
222
223 response = mturk.create_hit(title=title, description=description, keywords=keywords, question=final_question,
224 assignment_duration_in_seconds=assignment_duration, lifetime_duration_in_seconds=lifetime_duration, reward=reward,
225 max_assignment_max_assignment)
```

6. On successful creation of HIT, you should get a HIT Group ID which can be used to load up the newly created HIT.
7. Create a notification for the HIT, so that once the task is submitted by each worker the requester would get a notification about the completion of the work. In the current example the notification is sent to SQS.

```
226 notification_response = mturk.update_notification_settings(
227 NotificationResponse={'HITID': HITid},
228 Notification={'Destination': sqs_queue_url, 'Transport': 'SQS', 'Version': '2014-06-30', 'EventTypes': ('AssignmentSubmitted')},
229 Assign=True)
```

8. Read the message from SQS and retrieve the HIT ID.
9. Using the MTurk object, read the HIT details by the HIT ID.
10. You should get back the value(s) submitted in the html form by the worker.
11. Access the answer provided by the worker and based on your quality threshold either accept or reject the HIT.



12. Use the MTurk object to either accept/reject the HIT by providing the HIT ID.

This is how you would create a HIT and post completion of the HIT you can either accept/reject the work. Once all testing is completed you need to switch the URLs to the production URLs of Mechanical Turk. **d**



# CloudWatch and more

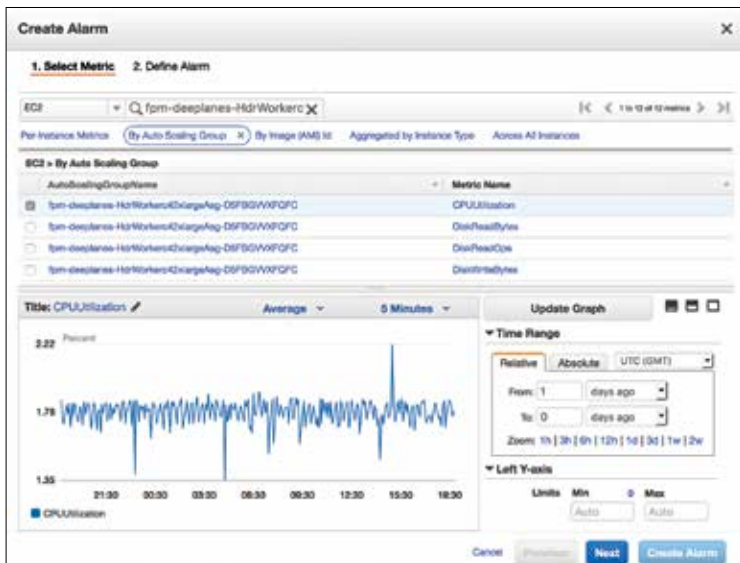
Keep an eye on your AWS applications just so that any untoward incidents don't end up giving you a bill shock.

**A**mazon CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS. You can use Amazon CloudWatch to collect and track metrics, collect and monitor log files, set alarms, and automatically react to changes in your AWS resources. Amazon CloudWatch can monitor AWS resources such as Amazon EC2 instances, Amazon DynamoDB tables, and Amazon RDS DB instances, as well as custom metrics generated by your applications and services, and any log files that your applications generate.

Examine logs from different AWS services.

## Set Alarms

Set alarms on any of your metrics to receive notifications or take other automated actions when your metric crosses a specific threshold. You can use alarms to detect and close EC2 instances that are unused or underutilised.



Pick the metric and set trigger events for alarms.

## View Graphs and Statistics

With Amazon CloudWatch dashboards, you can create re-usable dashboards which allow you to monitor your AWS resources in one location. Metric data is kept for a period of fifteen months enabling you to view granular data right down to each minute of usage and also historical data.

## Monitor and React to Resource Changes

Amazon CloudWatch Events enables you to respond quickly to application availability issues or resource changes, with notifications from AWS services delivered in near-real-time. You simply write rules to indicate which events are of interest to your application and what automated action to take when a rule matches an event. You can, for example, invoke AWS Lambda functions or notify an Amazon Simple Notification Service (SNS) topic.

## AWS Cost Optimization

Are you using the most cost-effective techniques for your workloads? To make sure that you are, we will explore some popular cost optimisation techniques and good practices. Most of the services in AWS are instance based, so they are charged by the number of hours that the instance is utilised. A single instance hour refers to a regular clock hour for which an instance was available to you, regardless of usage. AWS' variety of services and pricing options offers a flexible way to manage your resources in an efficient way and still meet your business needs. These are some of the commonly used practices and techniques to optimise the cost for AWS services :

### 1. Proper Sizing of Instances

AWS offers more than 60 EC2 instance types. Going for a very large instance type with higher compute is not always a prudent choice. Properly sizing the instances means choosing the cheapest instance but still meeting the performance requirements of the application. Apart from this, you should monitor your production workflow frequently and if the instances are underutilised (i.e. not using the full compute capacity), you can downsize them to a lower instance type which can definitely save a lot of money for you.

Here is a very good example of over-provisioning the EC2 instances. Suppose you have started up and you are deploying your product on AWS. Your application can run on an m4.large (which costs 0.1 \$ per hour) instance but you haven't given much thought about right sizing and you don't want to compromise on performance, so, you go for m4.xlarge instance type (which costs \$0.2 per hour). If you run this instance for an year you spend \$1752 but if you would have provisioned it right at the beginning you would have paid half that amount i.e. \$876. So, just by going one size up, you are literally paying double the money.

There is an EC2 Right sizing solution created by AWS, which is a template and can be launched using Cloudformation. It runs your application on different instance types and creates a detailed report of usage, compute and I/O, at the end, which can be used to determine the right size of instance for your needs.

### 2. Buying Reserved Instances

If you are planning to use AWS for long term, purchasing a reserved capacity instead of using On Demand instances is always a wise choice. AWS offers a significant discount of up to 25% off on the price of On

Demand instance. You can buy a reserved capacity starting from one year to three years and you can choose between standard and convertible class of reserved instances.

If you buy standard reserved instances you can modify some of the attributes like instance size but you can't modify instance type. But if you buy convertible instance type you can change the instance type, family and tenancy during the term. You can choose between standard and convertible depending on your use case. If you buy a reserved instance capacity for certain availability zone/region you don't need to do anything special to apply the discount. When bill is generated your discount is automatically applied to the matching instances in that availability zone/region.

### **3. Using Spot Instances whenever you can**

For short term usage or doing some low priority computation, you should always go for Spot Instances. Spot instance are generally available at a very low cost and can reduce your spends by 90% compared to the same On Demand instance type. The only drawback with Spot Instances is that they tend to terminate on their own when the price exceeds the bid. If you still want to take advantage of spot prices but face the issue of frequent instance terminations, then you can go for Spot Fleet or Spot Instance with Duration. In a Spot Fleet instance, you can choose among different availability zones and set multiple bids. AWS makes sure that it launches the cheapest available instance out of the multiple bids in the Spot Fleet. Another option is using a spot instance with duration. So, while launching a spot instance, a duration has to be mentioned (1-to-6 hours) and it will not terminate till that duration is over.

### **4. Monitoring**

Amazon CloudWatch is a tool for monitoring your AWS services. You can use CloudWatch to collect and track metrics, monitor log files, set alarms, and automatically react to changes in your AWS resources. Amazon CloudWatch can also be used to gain system-wide visibility into resource utilisation, application performance, and operational health. AWS also provides a set of tools which can help in reducing cost. Tools like Cost Explorer, Budgets, Billing and Cost Management console can be used on a regular basis using to explore spends, billing and cost trends. There is also a Cost Optimization Monitor solution dashboard, which uses managed services to visualise detailed billing reports.

## 5. Few More Tips

There may be some other techniques which we might not have mentioned in the above list and can be used for cost optimising your AWS resources. You should always try to shut down the instances that are not in use. One simple solution to achieve this is to automate the startup and stop using automated scripts. Another way to save on cost if you are not able to fully utilise your Reserved capacity is, selling the unused Reserved capacity on the AWS Marketplace. Sometimes your unattached volumes, old snapshots and redundant data may cost you some money. You can also use automated scripts to find and clean such data.

## AWS Cost Explorer

AWS comes with a huge range of solutions for making developers' life easier. Starting from VPC EC2 for setting up your own cloud to AWS Greengrass to set up your IoT solution, AWS has a solution for your each and every need. With time, it becomes difficult to keep track of billing and managing cost for all AWS resources you've signed up for.

AWS Cost Explorer is a tool for viewing and managing your expenses for other AWS resources. It maintains historic data of your expenditure and you can view the same going back to the previous 13 months. It can also forecast how much you are going to spend in the next three months. Using Cost Explorer, you can visualise the pattern of your expenditure over time. So, you can quickly identify the areas where you can optimise your usage and or where you should invest more for getting better performance.

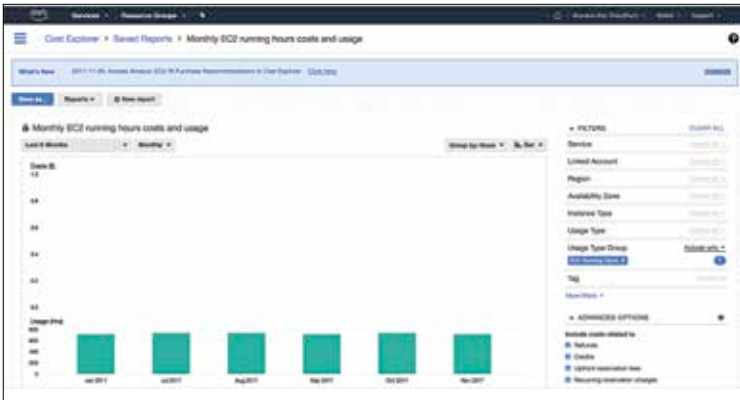


Get a breakdown of expenditure by time and service using Cost Explorer.

## Features of Cost Explorer:

- Cost Explorer comes with an easy-to-use interface to visualise as well as manage your spending and usage of AWS resources.

- It comes with some default reports that give you immediate view of cost analysis.
- You can set the custom period over which you want to visualise AWS cost and usage. It can range from monthly to daily periods depending on your requirement.
- Cost Explorer can also forecast how your expenditure is likely to be in next three months. So, accordingly, you can plan ahead.
- It can store historic data as a report for future reference.
- It supports a variety of filters for viewing AWS cost. You can view your report grouped by AWS resources you use or you can view which account uses an AWS resource the most.
- Ad-hoc analytics engines work behind AWS Cost Explorer and make it much more reliable.



Check expenditure trends over months.

## How to enable Cost explorer?

By default, every user has access to cost explorer. And the master user of the AWS account has the power to revoke access of other member user. To enable Cost explorer, you first need to sign in to AWS management Console with root credentials and go to Billing and Cost Management console. Then choose the option Cost explorer. After the “Welcome to Cost Explorer” page opens, just enable Cost Explorer.

As you launch Cost explorer, the current month's data will be available within 24 hours. Rest of the data will take some time. Cost and usage data is updated once every 24 hours.



## Control access to Cost explorer

You can control the access to Cost Explorer in a similar way as done for all other AWS resources. It can be enabled with root credentials only. As soon as you enable Cost Explorer all other users of your organization can view Cost Explorer. Now all members of your organization need not know the billing details. Ideally, you'd want to restrict access to a few users only. To achieve this, you just need to create an IAM User group. Then add those users to that group in order to grant them access.

## AWS Budget

In real life, we create a budget to control our expenditure and to keep expenses within a certain limit. Similarly, with AWS budget you can plan your service usage so that the service cost remains within a certain limit. You can set your own custom budget in terms of cost and usage. It will also alert you if the cost exceeds your budgeted amount. You can also customise the period of your AWS budget. It can range from monthly, quarterly, yearly or you can specify the start and end date of your budget. Your costs, subscription, or refunds can also be tracked through AWS budget.

AWS budgets can be categorised into the following types:

- Cost Budget – You can specify an amount that you want to spend on a device using this budget. For example, the amount you want to spend for your S3 bucket subscription.
- Usage Budget – Usage budget helps you to specify the time of usage. For example, how many hours you want to use your EC2 instance.
- RI Utilization – These types of budget work on a utilisation threshold defined by you and you will receive alerts whenever RIs are below this threshold.

## Creating A budget:

You need to follow the following steps for creating a budget.

- Choose the option Budgets under 'Billing and Cost' in the management console. You will get an option "Create budget".
- Specify budget type as well as the time period for your budget. You can create a monthly, quarterly or yearly budget. For RI utilisation budget, you can also go for daily budget. For cost or usage budget you should specify the start date, end date and amount. Start date and end date specification is optional.



Create a budget for a pre-determined time period.

- You can refine your budget by choosing some supported filters like service, usage type, usage type group etc.
- Next, configure the type of notification you would like to receive when service cost exceeds the budget amount. You need to specify the percentage of the budget at which you will be notified. So, it gives you the provision of being notified before your budget exceeds. For example, you want to be notified when 90% of the budget is exhausted. To get that notification, you need to mention 90 while configuring your notification. Notification will be sent via email or SNS topic or both.
- Finally click on the button “Create” for creating your budget.

You can also secure your budget and control access by using AWS IAM user group. Users who are given access using this IAM group can only view and modify your budget.

Now that we have seen some of the most commonly used services on the AWS Platform, we must realise that the scope of AWS extends well beyond simply providing infrastructure as a service. In fact, the very nature of Amazon Web Services (a cloud company) gives them access to petabytes of data. It should thus come as no surprise that Amazon has leveraged (and continues to do so) most of this information to come up with more and more services each day to better cater to our requirements. The latest launch of Amazon Sumerian, a drag-and-drop Virtual and Augmented Reality Platform generator is a grand affirmation to this fact.

Amazon is not alone in the cloud race, both Google Cloud and Microsoft Azure continue to be strong contenders and are fast catching up. But perhaps beyond all of this, we need to take a pause and wonder as to what technology would be capable of, with unlimited scales of growth in an age where human imagination itself is falling short of its achievements. **d**



All this and more in the  
world of Technology

VISIT  
NOW

▶ **digit.in**

www.facebook.com

Search

# Join 1 Mil + members of the digit community



http://www.facebook.com/thinkdigit

facebook

diGit.in

Digit

Like

Your favourite magazine on your social network. Interact with thousands of fellow Digit readers.

- Mail
- Info
- Search
- Current Item
- Subscriptions
- My Home
- Channel
- Photo
- Blog

http://www.facebook.com/IThinkGadgets

facebook

I Think Gadgets

Like



I Think Gadgets

An active community for those of you who love mobiles, laptops, cameras and other gadgets. Learn and share more on technology.

- Mail
- Info
- Photo
- Video
- Events
- Locations

http://www.facebook.com/GladtoBeaProgrammer

facebook

Glad to be a Programmer

Like



If you enjoy writing code, this community is for you. Be a part and find your way through development.

- Mail
- Info
- Photo
- Video
- Events

http://www.facebook.com/devworx.in

facebook

Devworx

Like



devworx, a niche community for software developers in India, is supported by 9.9 Media, publishers of Digit

- Mail
- Info
- Photo
- Video
- Events
- Locations