# NETWORKING

**OCTOBER 22, 2019**

## AWS Classroom Series – 22/Oct/2019

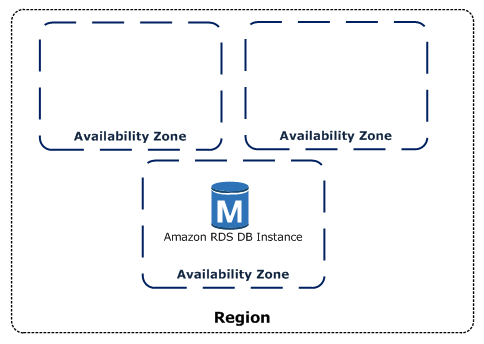
### Amazon Global Infrastructure

### References

1. [Global Infra](https://aws.amazon.com/about-aws/global-infrastructure/)
2. [Infra Map](https://www.infrastructure.aws/)

### Global Infra

1. Region:
   * Geographical Location
   * Regions code => <Continent>-<direction>-<regioncount>
   * Examples:
     + Mumbai => ap-south-1
     + NVirginia => us-east-1
     + Oregon => us-west-2
2. Availability Zone:
   * Actual place where you have Datacenters
   * Naming convention is <region-code><alphabet>
   * Eg: us-west-2a, us-west-2b, ap-south-1a

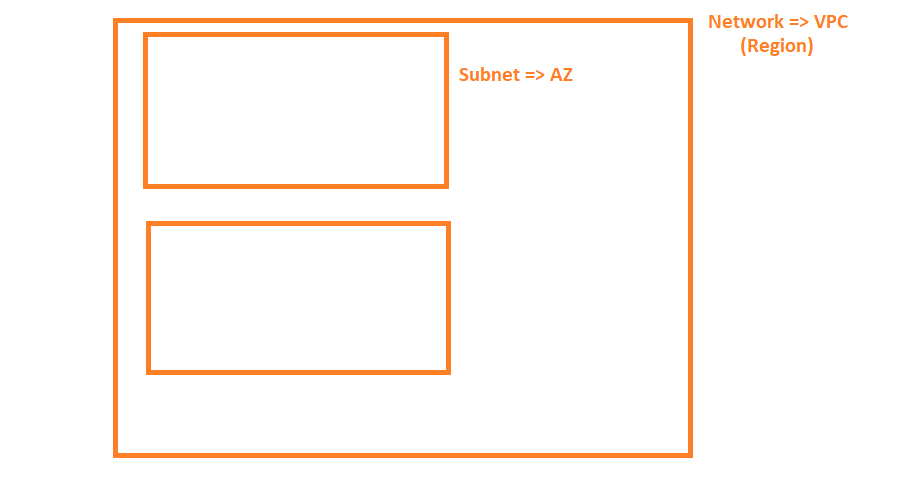


### Cloud Terms

1. Service:
   * Offerings by Amazon
   * Examples:
     + RDS => Databases
     + EC2 => Virtual Machines
     + VPC => Network
2. Resource:
   * Using Service whatever you have created is a Resource

### Example of Create a Network resource using VPC Service

* Network scope is region
* Subnets scope is Availability Zone



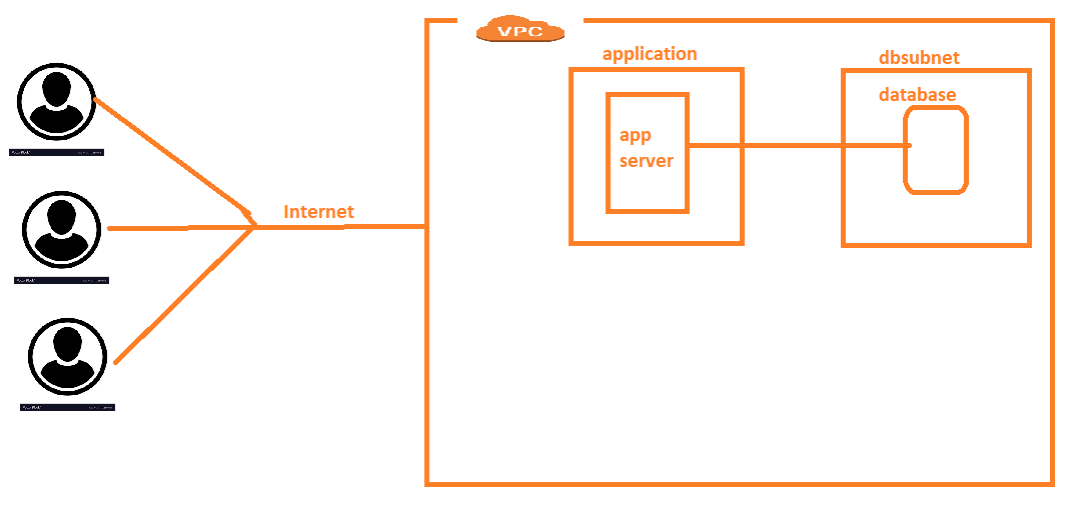
**OCTOBER 23, 2019**

## AWS Classroom Notes – 23/Oct/2019

### Why should I care about networking in AWS ?

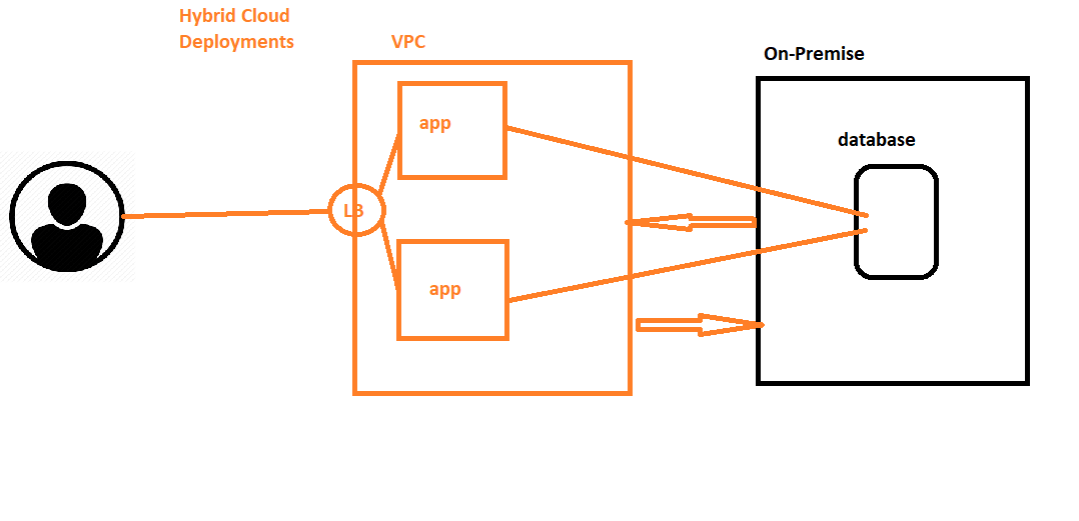
### Scenario-1

* Deploying a multi-tier application (Ecommerce application) with and without scaling



### Scenario-2

* Deploying an application which has some components on-premise (Hybrid)



### What needs to be understood in AWS Networking

* VPC (Network)
* Subnet (Sub-network)
* Private Networks and Public IP addressing
* Internet Gateways
* Security:
  + Security Groups
  + Network ACLs
* NAT:
  + NAT-instance
  + NAT-Gateway
* Route-tables
* Multi-Network Communications Over Private IPs
  + VPC Peering
  + VPNS
    - Point to Site
    - Site to Site
* Endpoints

### What should I use to create resources in Amazon

* Is this creation repetitive?
  + Yes:
    - AWS CLI (Command Line Interface)
    - Templating (Cloud Formation)
  + No:
    - AWS Console

### AWS CLI Installation

* Windows:
  + [Install Chocolatey](https://chocolatey.org/docs/installation)
  + Install Git For Windows: choco install git -y
  + Install AWS CLI: choco install awscli
* Linux:
  + [AWS CLI Installation](https://docs.aws.amazon.com/cli/latest/userguide/install-linux.html)

### AWS CLI Configuration

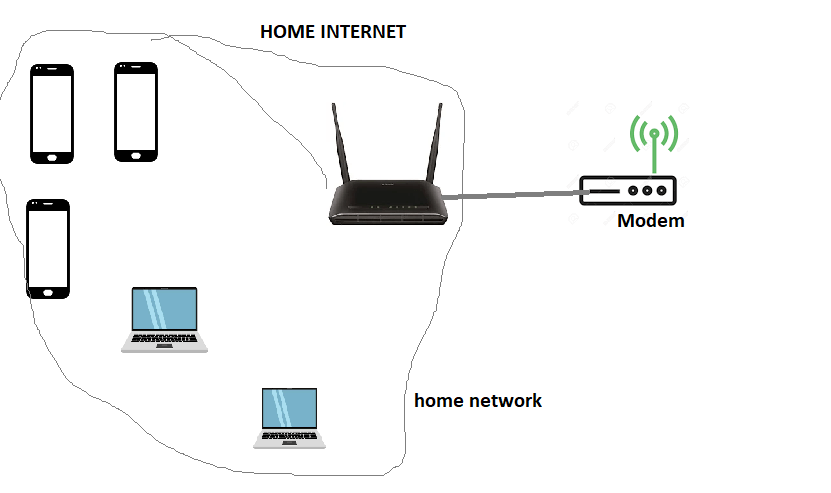
* AWS CLI needs credentials to connect to your AWS account.
* But these credentials are not email id & Password, rather they are ACCESS KEY and SECRET KEY.
* TO Generate ACCESS KEY and SECRET Key [Click Here](https://serverless-stack.com/chapters/create-an-iam-user.html)
* Test AWS cli connectivity using aws ec2 describe-vpcs

**OCTOBER 24, 2019**

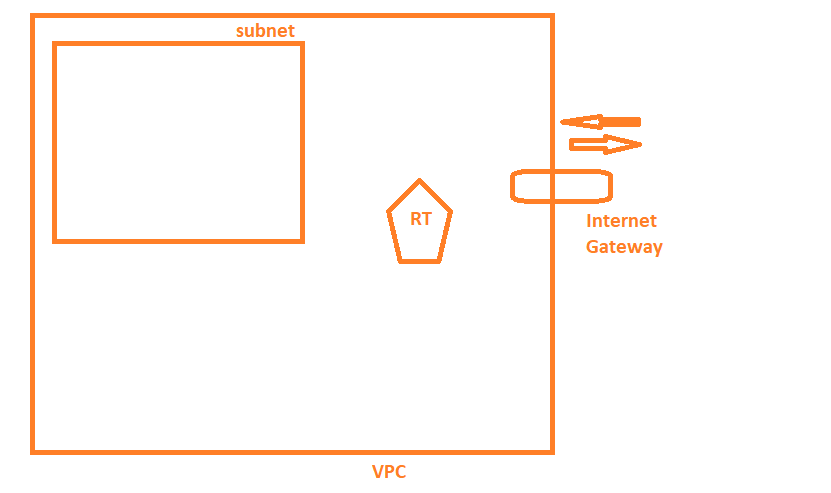
## AWS Classroom Series 24/Oct/2019 – Networking

### AWS Networking Basics

### Home Network



### AWS VPC



### Comparison

* Modem => Internet Gateway
* Router => Route Table
* Home Network => Subnet

### Create a Network with new Route Table and Internet Gateway

* Create a VPC in any region
* Create a subnet in the vpc
* Create and attach Internet gateway to VPC
* Create a Route Table
* Associate the subnet to Route Table
* Edit Routes in Route and add rule for any ip address (apart from your VPC range) to go to internet gateway

### Searching CLI’s

* Cheat sheet

aws cli <servicename>

google for aws cli vpc

* Command line basics

<command> [<subcommands>] [--argname argvalue] [argvalue]

ping google.com

docker create --name test helloworld

aws ec2 describe-vpcs

* AWS Command line

aws <servicename> <action> [args]

aws ec2 describe-vpcs

### AWS CLI Commands

aws ec2 create-vpc --cidr-block '192.168.0.0/16'

aws ec2 create-subnet --cidr-block '192.168.0.0/24' --vpc-id <vpc-id>

aws ec2 create-internet-gateway

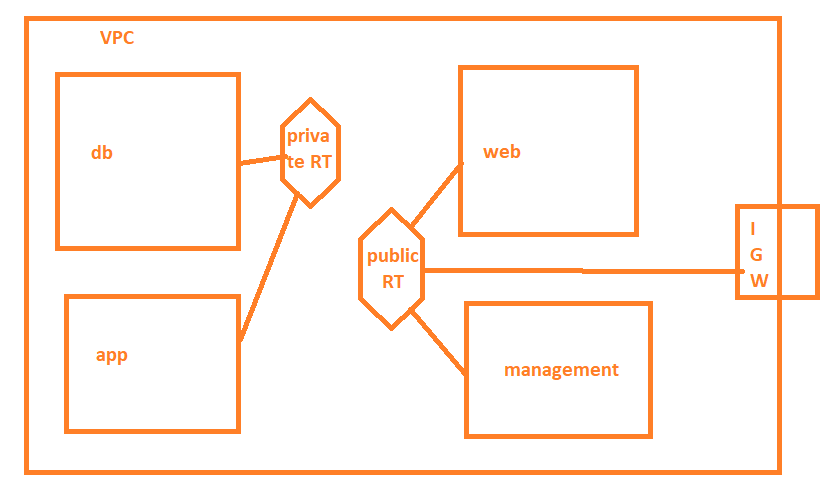
aws ec2 attach-internet-gateway --internet-gateway-id <igw-id> --vpc-id <vpc-id>

**OCTOBER 25, 2019**

## AWS Classroom Series – 25/Oct/2019

### Exercise

1. Create a VPC with cidr range 10.100.0.0/16
2. In this VPC create 4 subnets and then the cidr ranges can be
   * 10.100.0.0/24 => web
   * 10.100.1.0/24 => app
   * 10.100.2.0/24 => db
   * 10.100.3.0/24 => management
3. Create Internet Gateway
4. Create two route tables public and private
5. Public route table associate with web and management and connect to internet gateway
6. Private route table associate with app and db subnets and do not connect to internet gateway.



### Solution

# Creating VPC

aws ec2 create-vpc --cidr-block '10.100.0.0/16'

# "VpcId": "vpc-0eba8022e1953c115"

# Create Web Subnet

aws ec2 create-subnet --vpc-id "vpc-0eba8022e1953c115" --cidr-block "10.100.0.0/24"

# "SubnetId": "subnet-0c243a6309b465f46"

# Create app Subnet

aws ec2 create-subnet --vpc-id "vpc-0eba8022e1953c115" --cidr-block "10.100.1.0/24"

# "SubnetId": "subnet-02b7aae980d15b6ad"

# Create db Subnet

aws ec2 create-subnet --vpc-id "vpc-0eba8022e1953c115" --cidr-block "10.100.2.0/24"

# "SubnetId": "subnet-0d391ef304b2c9a39"

# Create management Subnet

aws ec2 create-subnet --vpc-id "vpc-0eba8022e1953c115" --cidr-block "10.100.3.0/24"

# "SubnetId": "subnet-0e4f54626c7fe7836"

# Create igw

aws ec2 create-internet-gateway

# "InternetGatewayId": "igw-0bac78a54c832207b"

# Atttach igw to VPC

aws ec2 attach-internet-gateway --internet-gateway-id "igw-0bac78a54c832207b" --vpc-id "vpc-0eba8022e1953c115"

# Create public route table

aws ec2 create-route-table --vpc-id "vpc-0eba8022e1953c115"

# "RouteTableId": "rtb-0ab89bf233a402434"

# Create route from public subnet to igw

aws ec2 create-route --route-table-id "rtb-0ab89bf233a402434" --gateway-id "igw-0bac78a54c832207b" --destination-cidr-block "0.0.0.0/0"

# Attach Public Route table to Web Subnet

aws ec2 associate-route-table --route-table-id "rtb-0ab89bf233a402434" --subnet-id "subnet-0c243a6309b465f46"

# "AssociationId": "rtbassoc-0642624b29b6df071"

# Attach Public Route table to management Subnet

aws ec2 associate-route-table --route-table-id "rtb-0ab89bf233a402434" --subnet-id "subnet-0e4f54626c7fe7836"

# "AssociationId": "rtbassoc-07d2d5f7cf543c479"

# Create Private Route table

aws ec2 create-route-table --vpc-id "vpc-0eba8022e1953c115"

# "RouteTableId": "rtb-0c0c41cdd523fc8a9"

# Attach Private Route table to app Subnet

aws ec2 associate-route-table --route-table-id "rtb-0c0c41cdd523fc8a9" --subnet-id "subnet-02b7aae980d15b6ad"

# "AssociationId": "rtbassoc-0c5252c37222ba28b"

# Attach Private Route table to db Subnet

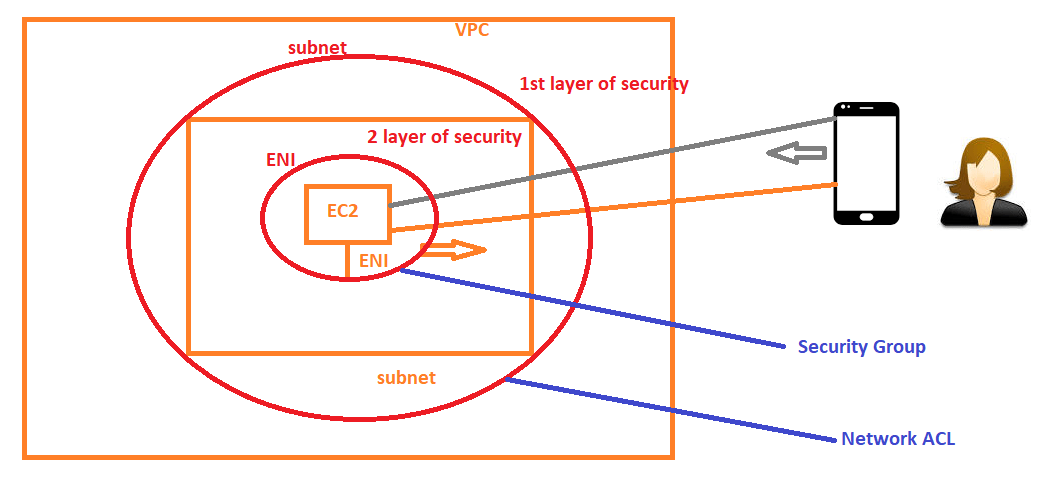
aws ec2 associate-route-table --route-table-id "rtb-0c0c41cdd523fc8a9" --subnet-id "subnet-0d391ef304b2c9a39"

# "AssociationId": "rtbassoc-04c8c8b7e00d16b4d"

**OCTOBER 27, 2019**

## AWS Classroom Series – 27/Oct/2019

### Security in AWS VPC



### Layered Security

* Security At Subnet
* Security At Network Interface

### Network ACL

* Is Security at subnet level.
* Here you write rules. Rules for ALLOW and DENY can be written.
* When you create a VPC, a default NACL is created which allows everything in both directions.
* Here for Rules we have priority
* Rules are broken down to two kinds
  + Incoming (Inbound, Ingress)
  + Outgoing (Outbound, Egress)
* When you write a rule
  + IP Address (Source IP or Destination IP): We use CIDR ranges to enter ip address rules. For eg 100.10.0.0/16 means any machine with 100.10.x.x, 0.0.0.0/0 means anywhere, 100.100.10.10/32 means only single ip 100.100.10.10
  + Protocol: TCP/UDP/ICMP etc..
  + Port: which port number to consider
  + ACTION: ALLOW/DENY
* If you have not written the rule it becomes deny by default (bcoz of priority evaluations)
* Scope of Network ACL is VPC, it means NACLs create for one VPC cannot be used in other VPCS (you need to recreate)

### NACL Rule Evaluation

* Important things to consider:
  + **\*** is Lowest Priority
  + **100** is Highest Priority
  + Lower the number higher the priority
* Whenever incoming Traffic is received on subnet, Evaluation starts from Lowest Number (Highest Priority) Rule in your NACL
  + Is this rules IP matching with Source IP. If no go to next rule. If yes continue
  + Is this rules Protocol matching with the request. If no got the next rule. If yes continue
  + Is this rules Port number matching with the request. If no go to the next rule. If yes continue
  + If this rules action is ALLOW/DENY, ALLOW/DENY the packet and stop process further

### Security Group

* It is Security at ENI(Elastic Network Interface) Level
* Here you write rules only for ALLOW
* Rules are broken down to two kinds
  + Incoming (Inbound, Ingress)
  + Outgoing (Outbound, Egress)
* When you write a rule
  + IP Address (Source IP or Destination IP): We use CIDR ranges to enter ip address rules. For eg 100.10.0.0/16 means any machine with 100.10.x.x, 0.0.0.0/0 means anywhere, 100.100.10.10/32 means only single ip 100.100.10.10
  + Protocol: TCP/UDP/ICMP etc..
  + Port: which port number to consider
* Scope of Security Group is VPC, it means Security Group create for one VPC cannot be used in other VPCS (you need to recreate)

### Exercise

* Create NACL rules which allows ssh to all incoming and everything outgoing in your vpc
* Attach NACL to all of your subnets
* Create two Security group which allows ssh and which doesnt have a rule about ssh in your vpc
* Now create a ec2 machine with allow ssh security group.
* Now create one more ec2 machine with deny ssh security group
* Check the connectivity
* In your ec2 machines install apache server

# ubuntu

sudo apt-get update

sudo apt-get install apache2 -y

# centos/redhat

sudo yum install httpd

sudo service httpd start

* try to access your server from browser http://<publicip>

**OCTOBER 29, 2019**

## AWS Classroom Series – 29/Oct/2019

### NAT Gateways

* Private subnets don’t have a route to IGW. which means no internet.
* What if i need internet to Private subnet.
* In traditional network we have NAT Servers.
* AWS offers NAT with the following options
  + NAT Instance:
    - EC2 machine which user creates with NAT AMI
    - Create a Static Public IP (Elastic IP) and associate with EC2 machine
    - If EC2 machine crashes, users responsibility to correct it.
  + NAT Gateways:
    - Managed by AWS, user needs to just specify he needs NAT Gateway
    - Elastic IP is automatically created or you can also use existing Elastic IPs
    - AWS does the scaling/management etc because user gets SLA on NAT Gateway

### Creating NAT Gateway

* Assumptions are you already have a vpc with 4 subnets. Two are private and Two are public.
* Create a NAT Gateway in Public subnet
* Add a route to Private Route Table which whenever the destination is 0.0.0.0/0 (Anywhere) forwards the packet to NAT Gateway.
* After doing this check, once cross check private subnets NACL and EC2 machines security group
* This setup should enable internet access to private subnets connected to a router.

### Egress only Internet Gateway.

* All the subnets in the VPC will be private with internet connections

### CLI to create NAT Gateway

* References
  + [Elastic IP](https://docs.aws.amazon.com/cli/latest/reference/ec2/allocate-address.html)
  + [NAT Creation](https://docs.aws.amazon.com/cli/latest/reference/ec2/create-nat-gateway.html)
  + [Create route for private router](https://docs.aws.amazon.com/cli/latest/reference/ec2/create-route.html)

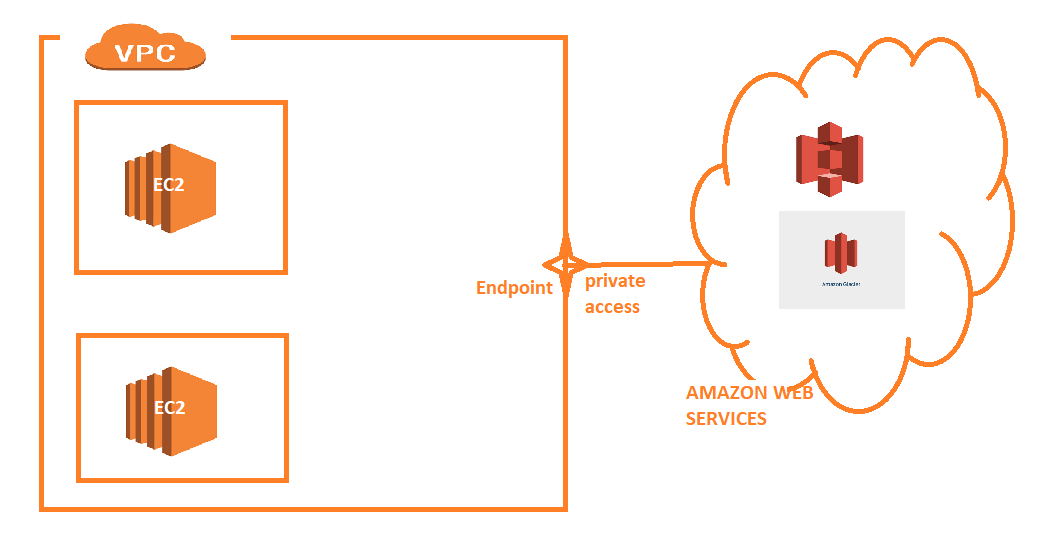
**OCTOBER 30, 2019**

## AWS Classroom notes-30 Oct 2019

### AWS Networking Endpoints

### Endpoint

* Endpoint provides private access to VPC resources for AWS Services

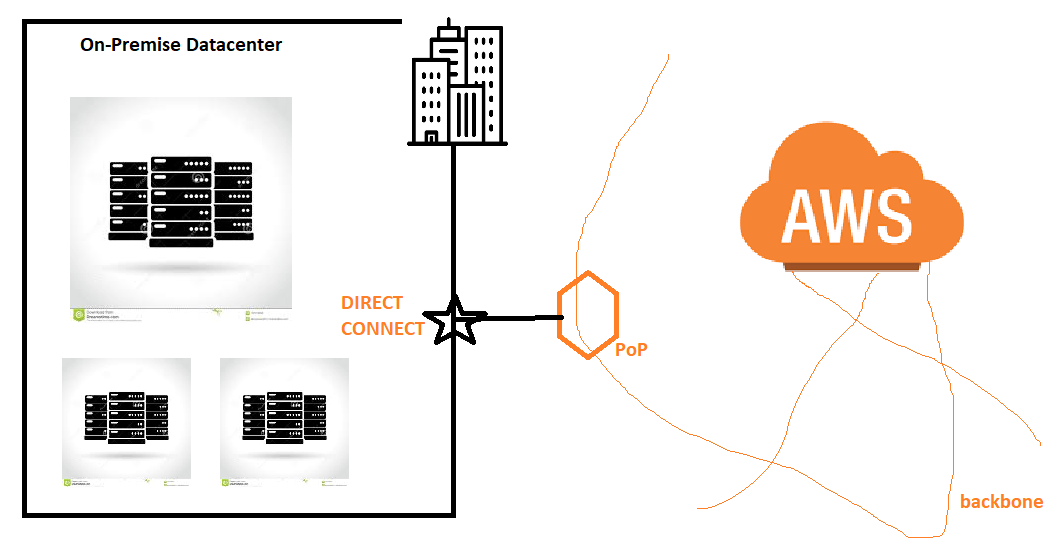


### Endpoint Types

* Interface Endpoints
  + An Elastic Network Interface will be created within your VPC cidr range to a list of Services mentioned over [here](https://docs.aws.amazon.com/vpc/latest/userguide/vpce-interface.html)
* Gateway Endpoints
  + A Gateway is created to connect to AWS Services (S3 and Dynamo DB)
  + To use Gateway Endpoint add the route to point to Gateway. (Much like Route to IGW or Route to NAT Gateway)

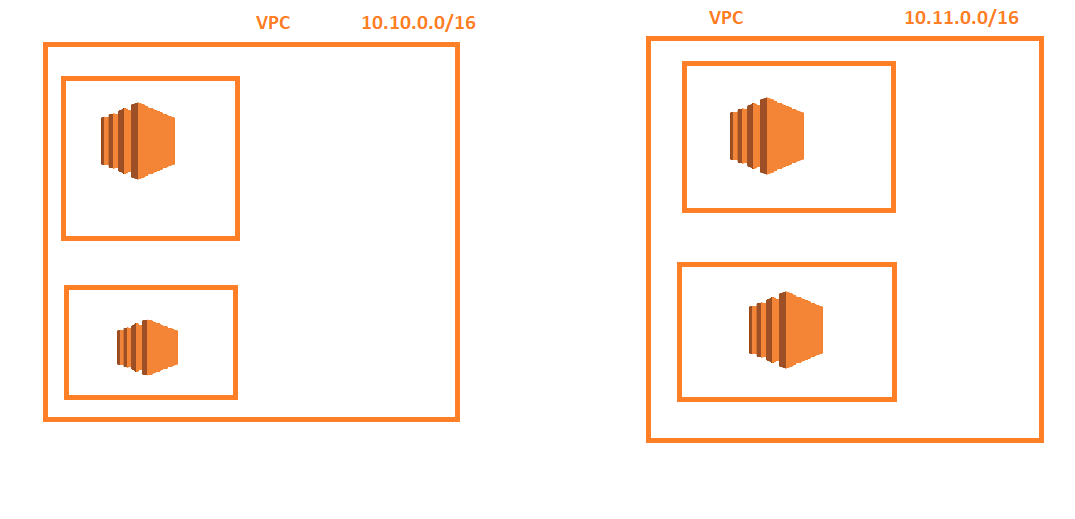
### How about OnPremise to AWS Services

### Direct Connect

* Using Direct Connect, On-Premises Data Centers can be connected to AWS Regions via PoP.
* To use Direct Connect, You need to go via AWS Partners supported in the Region. 

### Multi Network Connectivity in AWS

* Consider Account A has a VPC-A in Region A
* In VPC-A applications are running.
* This Application whenever it needs to send pager notifications, it needs to connect to another VPC (VPC-B) which is in Region B and Managed by Account B



### Problems with Connectivity

* Machines need to have Public IP and Connectivity to connect across VPCs.
* To solve this we need a way to connect from VPC-A to VPC-B using private IPS

### How to do that

1. IP Ranges should not collide
2. Since it is connectivity between two networks, VPN(Virtual Private Network) can be used.
3. For connectivity across AWS VPCs, then Amazon gives the Option of **VPC Peering**

**OCTOBER 31, 2019**

## AWS Classroom Series – 31/Oct/2019

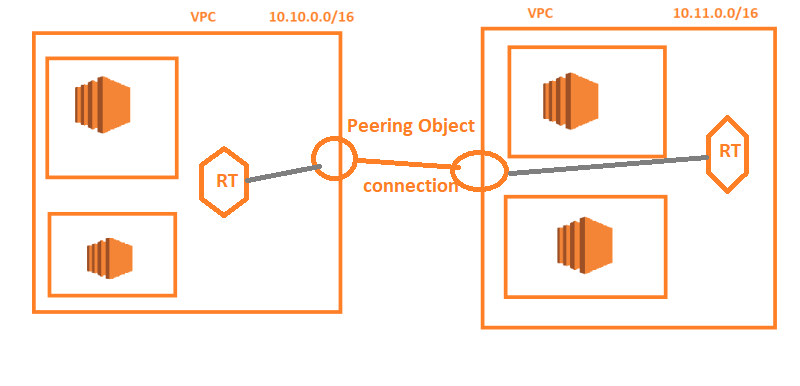
### VPC Peering

* Create a VPC with cidr range 192.168.0.0/16 and tag it with name vpcA
* Create two subnets in this vpcA.
* Create an Internet gateway and attach it to vpcA
* Now create a VPC with CIDR range 10.10.0.0/16 and tag it with name vpcB
* Create two subnets
* Create one ec2 machine in VPC A with public ip
* Create one ec2 machine in VPC B with only private ip
* Login into ec2 machine of vpc A and try to connect to ec2 machine of VPC B
* It should fail.
* Create a Peering Request from Vpc A to Vpc B.
* Approve the request in VPC B
* Now got the route table of VPC A and define a route

dest iprange 10.10.0.0/16

destination peering object

* Repeat the same thing from VPC B to Vpc A



**NOVEMBER 1, 2019**

## AWS Classroom notes 01/Nov/2019

### VPN in AWS

* On a broader note two kinds of VPNs are available
  + Point-to-Site:
    - Private Connectivity one device to the network remotely
    - Eg During Work from homes you generally connect laptop to office network
  + Site-to-Site
    - Private Connectivity from one site (office/network) to another site

### Site-to-Site VPN Single and Multiple Connection Examples

The following diagrams illustrate single and multiple Site-to-Site VPN connections.

### Single Site-to-Site VPN Connection

The VPC has an attached virtual private gateway, and your remote network includes a customer gateway, which you must configure to enable the Site-to-Site VPN connection. You set up the routing so that any traffic from the VPC bound for your network is routed to the virtual private gateway.


        VPN layout
      

### Single Site-to-Site VPN Connection with a Transit Gateway

The VPC has an attached transit gateway, and your remote network includes a customer gateway, which you must configure to enable the Site-to-Site VPN connection. You set up the routing so that any traffic from the VPC bound for your network is routed to the transit gateway.


        Single Site-to-Site VPN Connection with a Transit Gateway
      

### Multiple Site-to-Site VPN Connections

The VPC has an attached virtual private gateway, and your remote network includes a customer gateway, which you must configure to enable the Site-to-Site VPN connection. You set up the routing so that any traffic from the VPC bound for your network is routed to the virtual private gateway.

When you create multiple Site-to-Site VPN connections to a single VPC, you can configure a second customer gateway to create a redundant connection to the same external location. You can also use it to create Site-to-Site VPN connections to multiple geographic locations.


        Multiple Site-to-Site VPN layout
      

### Multiple Site-to-Site VPN Connections with a Transit Gateway

The VPC has an attached transit gateway, and your remote network includes a customer gateway, which you must configure to enable the Site-to-Site VPN connection. You set up the routing so that any traffic from the VPC bound for your network is routed to the transit gateway.

When you create multiple Site-to-Site VPN connections to a single VPC, you can configure a second customer gateway to create a redundant connection to the same external location. You can also use it to create Site-to-Site VPN connections to multiple geographic locations.


        Multiple Site-to-Site VPN connections with a Transit Gateway
      

# Identity and Access Management (IAM)

**NOVEMBER 5, 2019**

## AWS Classroom Series – 05/November

### Identity and Access Management (IAM)

* Is all about authentication and authorization.

### IAM terms

* Accountid: Every AWS account has a unique account id
* User:
  + Giving authentication to someone
* Group:
  + Categorizing users by some name
  + Eg: Admin, Developer, tester
* Role:
  + Is authentication and authorization given to AWS Service/Resource
* Policy:
  + Authorization.
  + Two kinds of Policies
    - AWS Managed Policies
    - User Managed Policies
  + Policies can be created from console using UI options/JSON

### AWS Resources Creations

* For any resource that is created in AWS a unique id is generated. That is called as **ARN(Amazon Resource Name)**
* ARN can be computed.
* AWS Provides patterns for finding ARN

### Policy evaluation when conflict arises

* Conflict of Policy can arise from
  + Multiple policies in User or Group
  + User Policy vs Group Policy
* In AWS Policy Evaluation Deny has more priority/Weight than ALLOW, so irrespective of position (User/Group), DENY is always the winner.

**NOVEMBER 6, 2019**

## AWS Classroom – Json Basics 06/Nov/2019

### Json

### History

* Prior to JSON, for configurations and transferring data
  + Plain Text => Human Readable
  + Binary => Machine Readable
  + XML => Both Machine and Human Readable

### Understanding JSon

* Json is collection of name value pairs
* Value of <name> can be text/number/boolean/list/object
* Text: "<name>": "<value>"
* Number: "<name>": <value>
* Boolean: "<name>": True/False
* list: "<name>": ["<value1>",......,"<valuen>"]
* object:

{

"name1": <value1>

..

..

"namen": <valuen>

}

* Sample Json tried in Class

{

'Movies': [

{

'name': '',

'language': [],

'Rating': 80,

'certificate': 'UA'

},

]

}

### References

* Json Tutorials:
  + [W3Schools](https://www.w3schools.com/js/js_json_intro.asp)
  + [Tutorials Point](https://www.tutorialspoint.com/json/index.htm)
* YAML Tutorials:
  + [Tutorials Point](https://www.tutorialspoint.com/yaml/index.htm)

### Using Json To Create IAM Policies

* A Strict Grammar is defined by AWS. [Refer Here](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_grammar.html) for full specifications

**NOVEMBER 7, 2019**

## AWS Classroom Series 07/Nov/2019

### Writing Simple IAM Policies

* For Grammar [Refer Here](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_grammar.html)

### S3 bucket use case

1. Give Permissions to user for S3 (Service Level Access)
2. Deny Permission to user on some bucket (Resource Level Access)
3. Give Permission to the user on s3 to read but not to write or delete or create

### Service Level Policies

* It applies to all the resources under that service. (Consider Use case 1)
* For every aws resource, we have an unique ARN and in the case of Service Level Access i can assume resource as \* (Any thing)
* Usecase 3 speaks about Service Level Access, BUt there are action based Restrictions

### Terms

* Service : Amazons offfering
* Resource: Whatever user Creates
* Actions: Different Operations of Resource

### Considerations To Write Policy

1. Policy Grammar
2. Which Service are you writing the policy around
3. Is the Resource Specific.
4. What are all the possible actions of this resource

### IAM Policy for Usecase 1

* To Get all the actions google aws iam <service> actions, in this case i will be querying with aws iam s3 actions

{

"Version" : "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": ["s3:\*"],

"Resource": "\*"

}

]

}

* To give Readonly access add the necessary actions to Statement => Action list

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"s3:Get\*",

"s3:Describe\*",

"s3:List\*"

],

"Resource": "\*"

}

]

}

* To Give Readonly on one resource and write on all others.
* First findout ARN for that Resource. google aws s3 arn
* Test this for read only on one bucket

{

"Version" : "2012-10-17",

"Statement": [

{

"Resource": "arn:aws:s3:::qt-s3-new-testing",

"Effect": "Deny",

"Action": [

"s3:AbortMultipartUpload", "s3:Create\*",

"s3:Delete\*", "s3:ObjectOwnerOverrideToBucketOwner",

"s3:Put\*", "s3:Replicate\*","s3:Restore\*", "s3:Replicate\*",

]

},

{

"Effect": "Allow",

"Action": ["s3:\*"],

"Resource": "\*"

}

]

}

**NOVEMBER 8, 2019**

## AWS Classroom Series – 08/Nov/2019

### IAM Policies

* Not Action:
  + All actions but the specified Actions
  + Eg: Deny write actions on s3
    - Not Action : ["s3:Put\*"], Effect: "Allow"
* Not Resource
  + All resources apart from specified resources
  + Eg: Deny Access to only specific s3 bucket
    - Not Resource: "specific s3 arn", Effect: "Allow"

### Exercise

* Create a IAM Policy which allows to
  + Do any thing on s3 bucket
  + Read, List ec2 machines (Ec2 Readonly)
  + Read, List RDS
* Solution:
  + Create a policy by combining statements from ec2 readonly, rds readonly and s3 fullaccess

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "s3:\*",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "ec2:Describe\*",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "elasticloadbalancing:Describe\*",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": [

"cloudwatch:ListMetrics",

"cloudwatch:GetMetricStatistics",

"cloudwatch:Describe\*"

],

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "autoscaling:Describe\*",

"Resource": "\*"

},

{

"Action": [

"rds:Describe\*",

"rds:ListTagsForResource",

"ec2:DescribeAccountAttributes",

"ec2:DescribeAvailabilityZones",

"ec2:DescribeInternetGateways",

"ec2:DescribeSecurityGroups",

"ec2:DescribeSubnets",

"ec2:DescribeVpcAttribute",

"ec2:DescribeVpcs"

],

"Effect": "Allow",

"Resource": "\*"

},

{

"Action": [

"cloudwatch:GetMetricStatistics",

"logs:DescribeLogStreams",

"logs:GetLogEvents"

],

"Effect": "Allow",

"Resource": "\*"

}

]

}

* Create an IAM Policy to give s3 full access to all resources but deny permission to one bucket "qt-iam-learning". ec2 readonly access, but start, stop and terminate to one ec2 machine

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": ["ec2:StartInstances", "ec2:StopInstances", "ec2:TerminateInstances"],

"Resource": "arn:aws:ec2:us-west-2:798279872530:instance/i-0a4a6259ceab6c6b1"

}

,

{

"Effect": "Deny",

"Action": "\*",

"Resource": "arn:aws:s3:::qt-s3-learn-again"

},

{

"Effect": "Allow",

"Action": "s3:\*",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "ec2:Describe\*",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "elasticloadbalancing:Describe\*",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": [

"cloudwatch:ListMetrics",

"cloudwatch:GetMetricStatistics",

"cloudwatch:Describe\*"

],

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "autoscaling:Describe\*",

"Resource": "\*"

}

]

}

* Create an IAM Policy for the user which gives full access for user to s3, ec2 and rds in Oregon (us-west-2) and deny access on all other regions

**NOVEMBER 10, 2019**

## AWS Classroom Series – 10/Nov/2019

### Federated Logins using AWS

* [Refer Here](https://aws.amazon.com/blogs/security/enabling-federation-to-aws-using-windows-active-directory-adfs-and-saml-2-0/)

### Directory Services in AWS

* [Refer Here](https://docs.aws.amazon.com/directoryservice/latest/admin-guide/what_is.html)

### Policy Evaluations

* Manual:
  + Create a test user attach to newly created policy
  + Login as a test user and verify manually
* Policy Summary:
  + Whenever a custom policy is created/update you can have a look at Resource Summary, Action Summary and Service Summary
* Policy Simulator:
  + Use policy simulator for the user hosted at [here](https://policysim.aws.amazon.com/)
  + Test the policies.
  + [Refer](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_testing-policies.html) this page

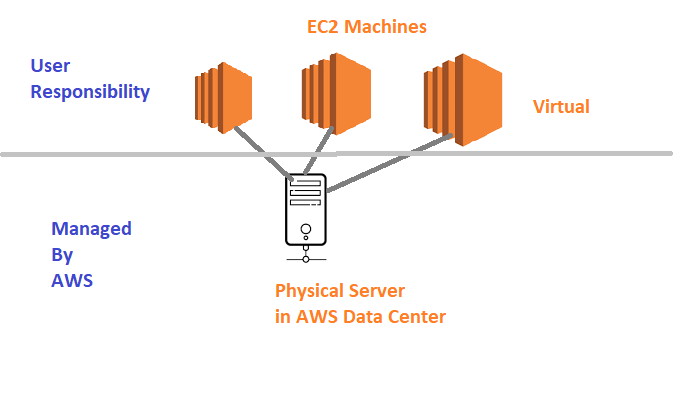
# Amazon Elastic Compute Cloud (Amazon EC2)

**NOVEMBER 12, 2019**

## AWS Classroom Series – 12/Nov/2019

### Virtual Machines in AWS

* Called as **EC2**



### Dependencies for EC2 instances on other AWS Services

* Network:
  + All aws ec2 instances belong to some subnet in VPC.
  + By default they get created in Default VPC
  + Security Group: just think of it as a firewall
* Storage:
  + What is type of hard disk which you want to use
  + Types:
    - Instance Store
    - EBS

### Other Things in EC2 to be considered

* AMI (Amazon Machine Image):
  + Which OS has to be used to create EC2 instance
  + Three kinds of AMIs
    - AWS Managed:
      * Managed by AWS with basic OS images and some extra softwares
    - Market Place:
      * Images created by Third Parties.
      * Extra licensing costs can be imposed
    - Custom Images:
      * Images created by users.
      * Generally will have your applications preconfigured.
* KeyPair:
  + AWS by default disables password authentication.
  + Keypair RSA based Private and Public keys
  + Can be created in AWS or Create them in your servers and import the public key to AWS.
* Instance Type:
  + number of Cpus
  + Size of RAM
  + Network Speed
  + Kind of Hardware
  + Number of Instance Stores

### Diagnostic Information about EC2 machines

* System Log:
  + Instance Settings => Get System Log
* Instane Screen Shot:
  + Instance Settings => Get Instance ScreenShot
* How to Change Instance Type of the EC2 Instance:
  + Shutdown the EC2 instance
  + Instance Settings => Change Instance Type
  + Note: requires downtime.

### Instance States

* running
* stopped
* terminated => Deleted
* Reboot

### Attaching Roles

* Roles can be attached and changed when ec2 machine is running.

### Execute Some Script Once the EC2 machine is created

* User-Data

**NOVEMBER 13, 2019**

## AWS Classroom Series – 13/Nov/2019

### Amazon Machine Image

* All Virtualizations require some image to start the Virtual Machine
* Amazon Machine Image is clone of Hard disk with OS and some metadata.
* Creating Amazon Machine Image:
  + Create an EC2 Instance
  + Configure EC2 instance with necessary software/configurations
  + Create an EC2 Image
* Amazon Machine Image will have Storage Costs.
* Automated way of Creating AMIs
  + using AWS CLI
  + Using Packer
* AMI will be created in only one Region. If you want a multi region AMI, Create AMI and copy across other regions(AMI id changes). Each Region has a unique AMI

### CLI TO Create AWS Ec2 AMI

* Create an IAM User with EC2 full access and Programmatic Access
* Install AWS CLI [Refer Here](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html) and Configure the user using

aws configure

* Create an Ec2 machine.

aws ec2 run-instances --count 1 --image-id ami-06d51e91cea0dac8d --instance-type t2.micro --key-name ec2learnings --security-group-ids sg-00d319006f01acfd5 --subnet-id subnet-030f585b

* After Launching an Ec2 machine, install apache2

sudo apt-get update

sudo apt-get install apache2 -y

* Create an AMI by executing the following CLI command

aws ec2 create-image --instance-id <instance id> --name my-apache-1

# make a note of AMI id

* Execute describe AMI till the status is Available

aws ec2 describe-image --image-id <image-idfromabovecommand>

* Now launch the Ec2 machine with your image

aws ec2 run-instances --count 1 --image-id <my-apache-1-id> --instance-type t2.micro --key-name ec2learnings --security-group-ids sg-00d319006f01acfd5 --subnet-id subnet-030f585b

* Now login into the machine and check the machine for the contents/configurations, they will match.

### Exercise: Create an Amazon Machine Image

* Create a ubuntu machine
* Login into the machine and install lamp stack. [Refer Here](https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-16-04)
* Manual Steps

sudo apt-get update

sudo apt-get install apache2 -y

sudo apt-get install php libapache2-mod-php php-mysql php-cli -y

# Create a file at /var/www/html/info.php

sudo nano /var/www/html/info.php

# Contents

<?php

phpinfo();

?>

sudo apt-get install stress -y

### Scaling

* Vertical Scaling:
  + Increasing the hardware resource configuration.
  + Downtime will be required
* Horizontal Scaling
  + Increase VMs or Servers
  + Doesn’t require a downtime for existing server

### Scaling vs Elasticity

* Scaling is all about increasing
* Elasticity is about increasing and decreasing as well
* In AWS we use Elasticity.
* Vertical Scaling Can be achieved by using Change Instance Type after shutdown of Ec2 instance (Stopped)
* Horizontal Scaling is achieved by Auto Scaling Groups

**NOVEMBER 14, 2019**

## AWS Classroom 14/Nov/2019

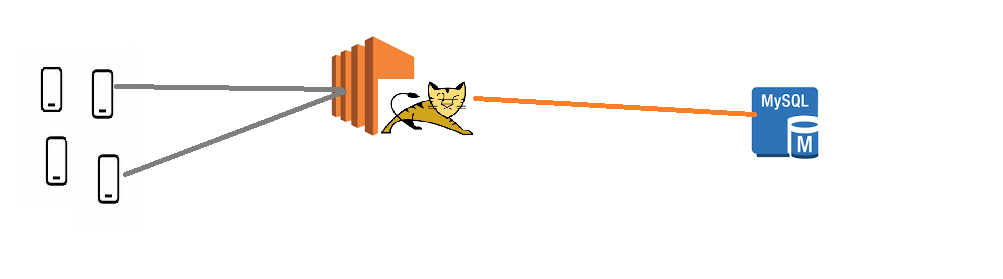
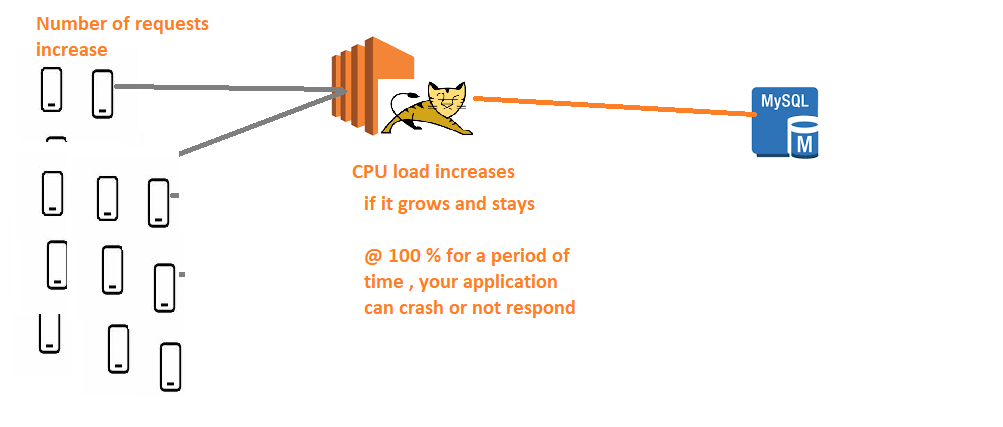
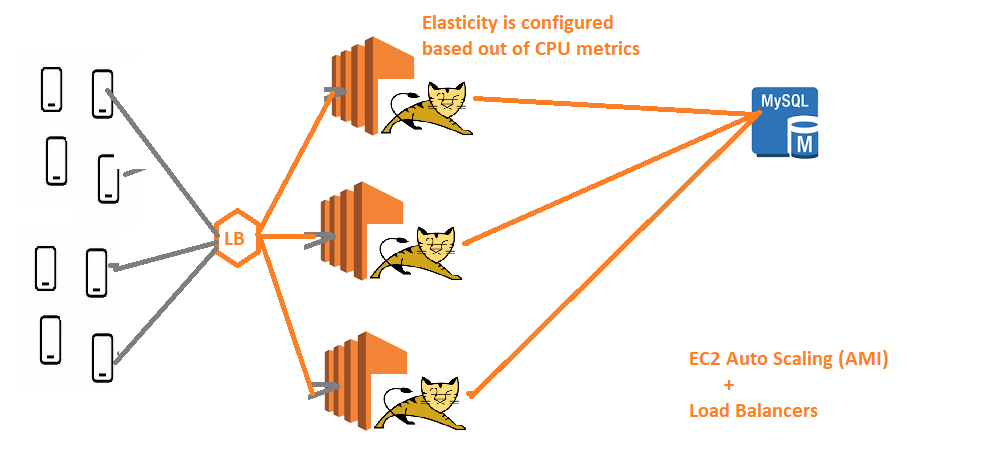
### EC2 Instance Types

* [Refer Here](https://aws.amazon.com/ec2/instance-types/)
* Instances Types are divided into Families
  + They Share same Hardware
  + Instance Families also have Generations
* Instance Types are Categorized into Purposes
  + General Purpose:
    - CPU to RAM Ratio is balanced
  + Compute Optimized:
    - High CPU to RAM
  + Memory Optimized:
    - High RAM to CPU
  + Storage Optimized:
    - Hard disk related optimizations
  + Accelerated Compute:
    - Parallel Processors

### EC2 Pricing Models

* Use [Amazon Pricing Calculator](https://calculator.s3.amazonaws.com/index.html)
* On Demand:
  + Create a machine at any point and delete the machine on demand
  + Billing:
    - Hourly: unit of billing is 1 hour
    - Per-Second: unit of billing is Second
  + EC2 Billing
    - Only when your instance is Running
  + Disk Billings:
    - This billings continue irrespective of instance state
  + This is the Costliest Model of Amazon.
* Reserved Instances:
  + Users give a commitment to Amazon for 1 or 3 years
  + They can pay the bills of EC2 machine with Commitment
    - All upfront (most discounted price)
    - Partial Upfront
    - Pay as you go (monthly bills)
  + Ec2 machines are charged for complete hours.
* Spot Instances:
  + Users can bid for instances
* Scheduled Instances:
  + Users can create a schedule for creating and deleting EC2 Machines
* Savings Plan:
  + Give a commitment on Cost.
  + I will be using ec2 machines of work 10$/hour for next one year

### Auto Scaling

* Basic Architecture 
* Problem with load increase 
* AWS Cloud Solution 

**NOVEMBER 15, 2019**

## AWS Classroom notes – 15/Nov/2019

### Auto Scaling

* Lets create an Auto Scaling Group with an AMI (apache2, php and stress)
* Create a Launch Configuration by selecting
  + AMI
  + Instance type
  + Storage
  + Security Group
  + KeyPair
* Create AutoScaling Group linked with Launch Configuration
  + Configure Increase Scenario (eg average cpu utilization > 75 for period of 5 minutes)
  + Configure Decrease Scenario (eg average cpu utilization <= 50 for period of 5 minutes)
* Then you should see the machine running
* Login into one machine to create artificial load using stress

stress --cpu 8 --io 4 --vm 2 --vm-bytes 128M -v --timeout 10m

* To have a closer look, create one more ssh login session and execute htop and observe cpu

### Load Balancers

* With Auto scaling group, you will have multiple servers where your application is running, we need to give unique ip/dns to the user
* For this we go with load Balancers
* In AWS, we have three kinds of Load balancers
  + Classic Load Balancer : Layer 4 and Layer 7
  + Application Load Balancer: Layer 7
  + Network Load Balancer: Layer 4
* Load Balancer needs to know which machines are healthy, to do this we have health checks
* Generally there are two kinds of LBs
  + Layer-4 Loadbalancers: LBs which are aware of ip and port
  + Layer-7 Loadbalancers: LB which are aware of http,url, ip and port
* [Refer here](https://freeloadbalancer.com/load-balancing-layer-4-and-layer-7/)

<https://freeloadbalancer.com/load-balancing-layer-4-and-layer-7/>

* Create a Classic Load balancer by configuring health checks
* Access the appplication using classic loadbalancers dns name.

**NOVEMBER 17, 2019**

## Aws Classroom Series – 17/Nov/2019

### User Data

* This is the script that gets executed once when the machine is created.
* UserData can be passed using Console or CLI
* UserData in linux systems need shebang

#!/bin/bash

sudo apt-get update && sudo apt-get install stress -y

* This user data is referred in other clouds as cloud-init

### Burstable Instance Types

* AWS instance type t-Series is burstable.
* CPU Credits will be given to machines when they are under utilized.
* [Refer Here](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/burstable-credits-baseline-concepts.html)

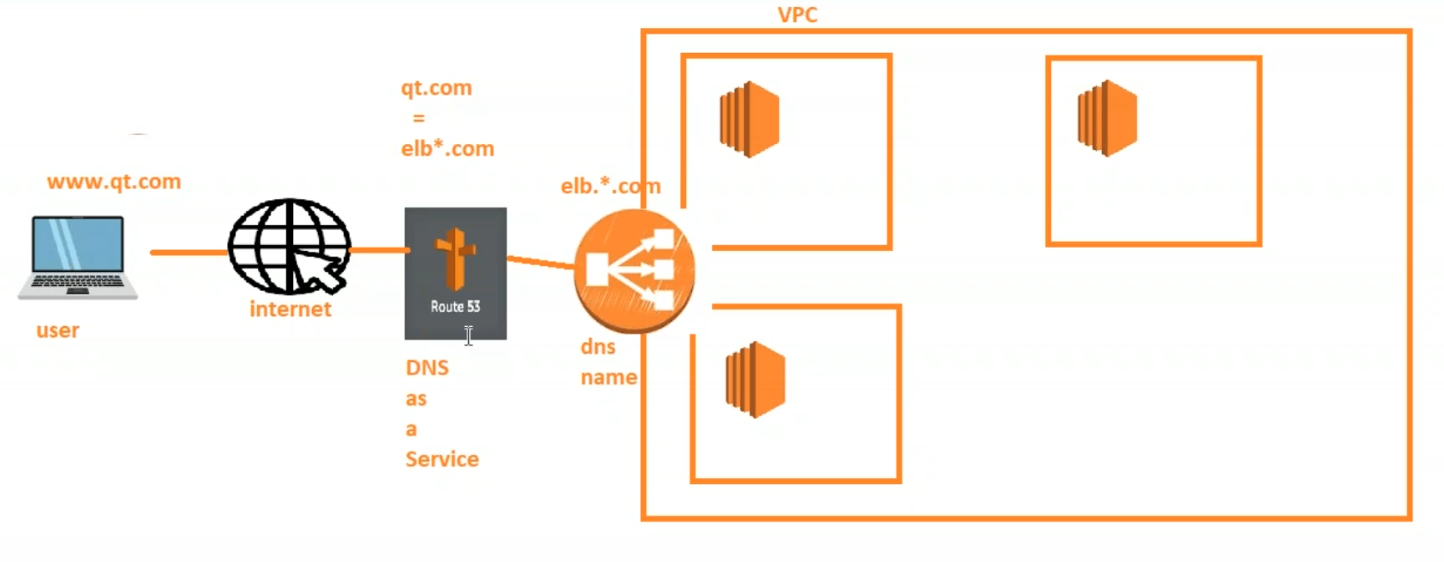
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/burstable-credits-baseline-concepts.html>

### Launch Templates

* Reusable Template for EC2 instance Creation with default values selected
* Saves time as you need not select options.

### Route 53

* This is DNS as a service
* Here we can add A-Records & C-Records to map domain name to loadbalancers or application gateways or ec2 machines.



### HTTPS in AWS Loadbalancers

* [Refer Here](https://aws.amazon.com/premiumsupport/knowledge-center/elb-ssl-tls-certificate-https/)

<https://aws.amazon.com/premiumsupport/knowledge-center/elb-ssl-tls-certificate-https/>

### Exercise

1. Create an auto scaling group with any image (lamp)
2. configure a load balancer
3. Create a Route 53 hosted zone for the domain
4. Create a Record Set with A-Record to point to Load Balancer
5. Ensure your Domain Register is using the nameservers in Route53 Record Set.

### CLI Exercise

* Create a ec2 machine with your ami
* Change the size of ec2 machine
* Delete the ec2 machine

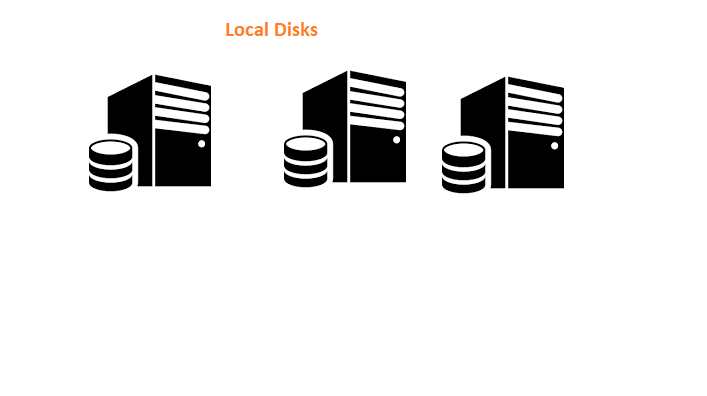
# Storage and Operators

**NOVEMBER 19, 2019**

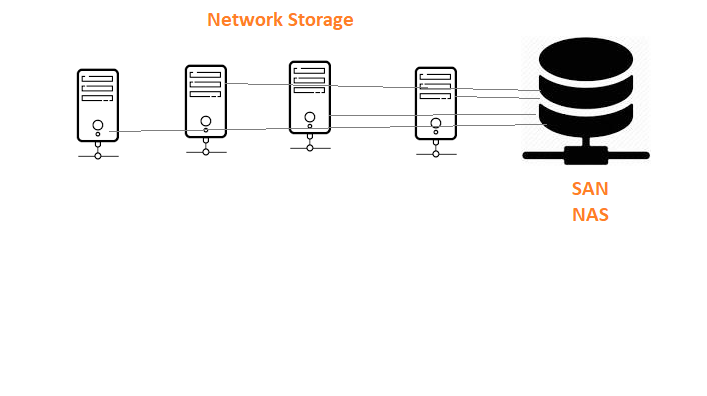
## AWS Classroom Series – Storage – 19/Nov/2019

### Storage in Enterprise

* Local Disk Storage:
  + Can be attached to only one server at any given time
  + Which are boot/OS disks



* Network Storage:
  + Can be attached to multiple machines
  + Is not about a boot disk



* Object Storage:
  + Storage connected via http protocols

### Other Considerations in Storage

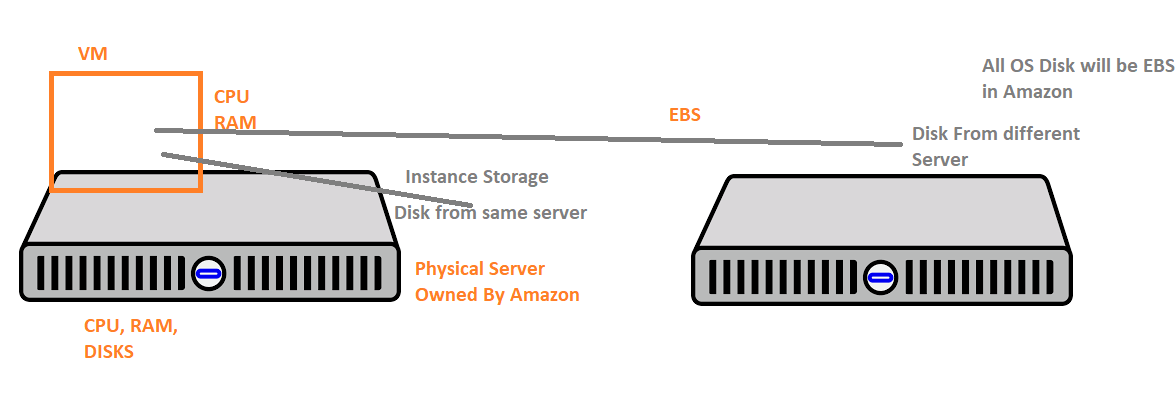
* Backup:
  + Quick Recovery
  + Quick
  + Data loss should be minimal
* Archive
  + Disaster Recovery
  + Will be slow
* Hardware Type:
  + Magnetic
  + HDD
  + SSD
* Speed
* Size

### Storage Solutions by AWS

* Disks:
  + Instance Store
  + Elastic Block Storage (EBS)
* Network Disks:
  + Elastic File Storage (EFS)
* Object Storage:
  + Simple Storage Service (S3)
* Backups:
  + Snapshots
* Archival:
  + Glacier
* MarketPlace:
  + Third party solutions from NetAPP, Pure Storage, EMC2

### Virtual Infrastructure in AWS

* In AWS we work on virtual machines & virtual storages.



* Virtual Machines are created on Physical Servers. The Disk of the Virtual Machine can be attached from
  + Same Physical Server => Instance Store
  + Different Physical Server in the Same AZ => EBS
* In AWS, for VM (EC2) the operating system disk (os disk/root disk) will always be EBS
* User will get options of Instance Store only for additional disks (Only for some instance types)

**NOVEMBER 20, 2019**

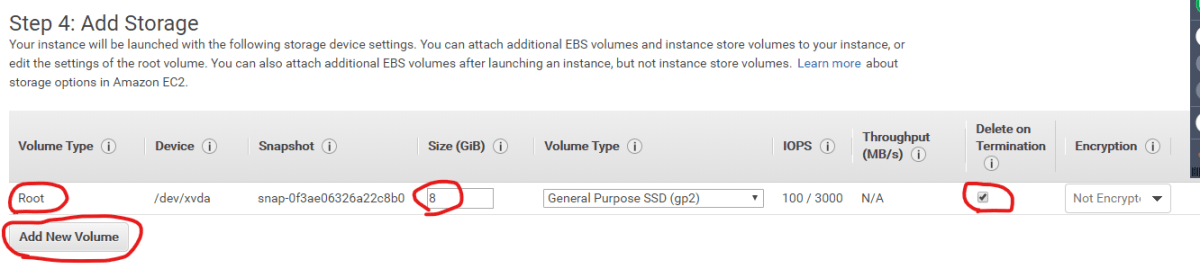
## AWS Classroom Series – Storage 20/Nov/2019

### Disk Storage in AWS

* Disk Storage for EC2 instances is provided by AWS using
  + Elastic Block Storage
  + Instance Store
* Disks are categorized as
  + Root Disk: Disk with Operating System
  + Disk
* Free Tier Plan
  + 30 GB of General Purpose Disk is free per month
* Root Disk is always EBS
* EC2 instance Type will provide whether instance store can be attached or not.
* Two kinds of Disks Depending on Data Persistence are
  + Ephemeral Disks: Instance Store
  + Non Ephemeral: EBS

### Creating a EC2 instance with Hard disk (Volume)

* Launch an Ec2 Instance by selecting AMI and click next till you see storage section



* Number of Disks can be added
* Size of Disk can be changed
* An option to choose not to delete the Volume after EC2 machine is terminated.

### Adding Volumes to Existing EC2 machines

* Create a Volume in the same AZ as your EC2 instance.
* Attach the volume to EC2 machine

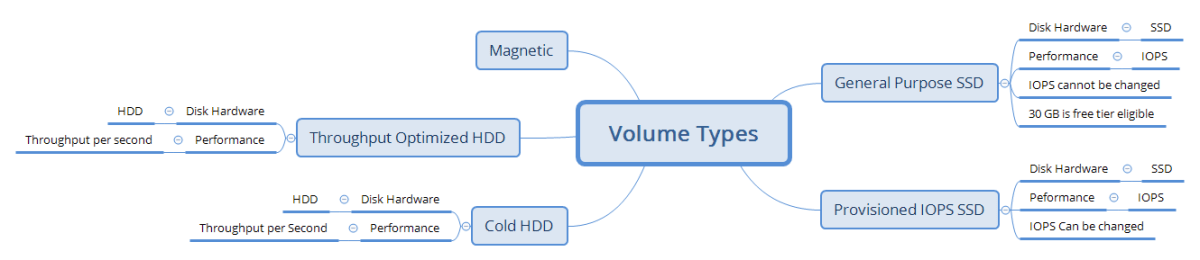
### Increase the Size of Volume while EC2 Instance is Running

* EBS Volumes are Elastic Volumes because they support change in disk size while the machine is running.

1 GiB = 1024

### Volume Types in EBS

* General Purpose SSD
* Provisioned IOPS
* Cold HDD
* Throughput Optimized HDD
* Magnetic



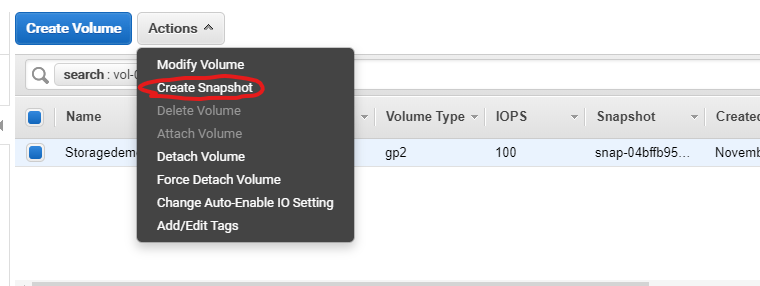
**NOVEMBER 21, 2019**

## AWS Classroom Notes – Storage – 21/Nov/2019

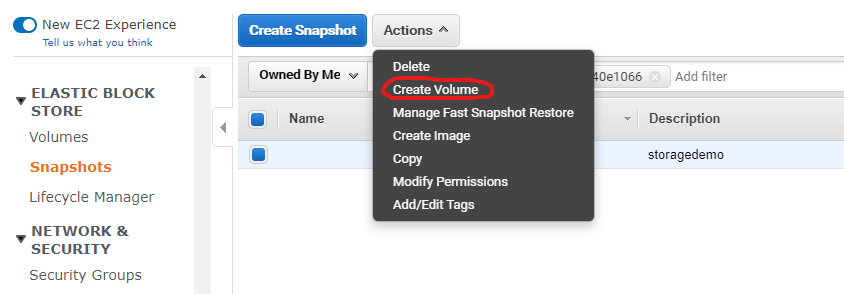
### Exercise

1. Create an EC2 instance with 10 GiB of EBS volume (Any Linux) (10 mins)
2. Backup of Volume.
3. From Backup create a new Volume (in same/different region)

### Creating Snapshot of Volume

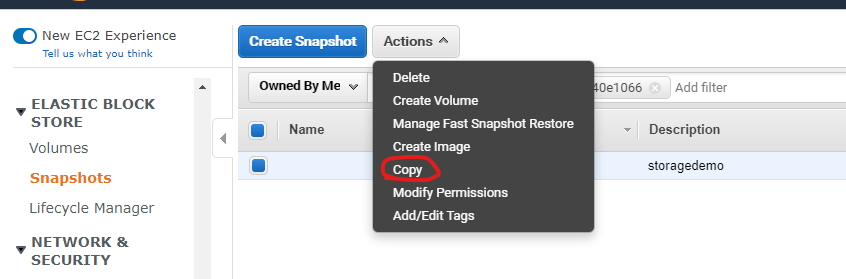
* Manual:
  + Creating a Snapshot of the Volume
  + 
* Automated:
  + AWS CLI to create a backup and then run this script periodically.
* Scheduled Backups

### Creating Volume from Snapshot

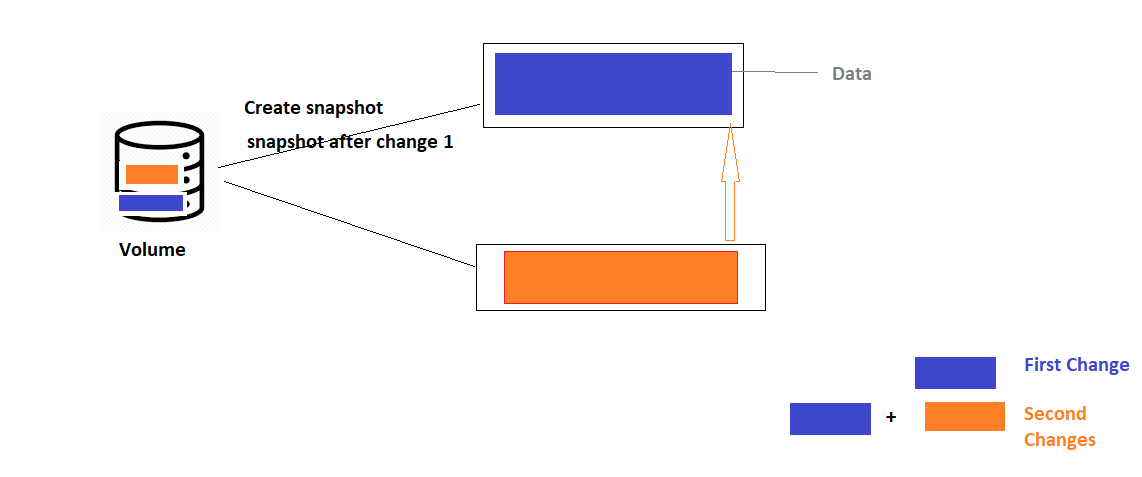


* Creates the volume from snapshot in the same region (where snapshot exists).
* User can select the AZ.

### Creating Volumes in Different Regions from Existing Snapshots

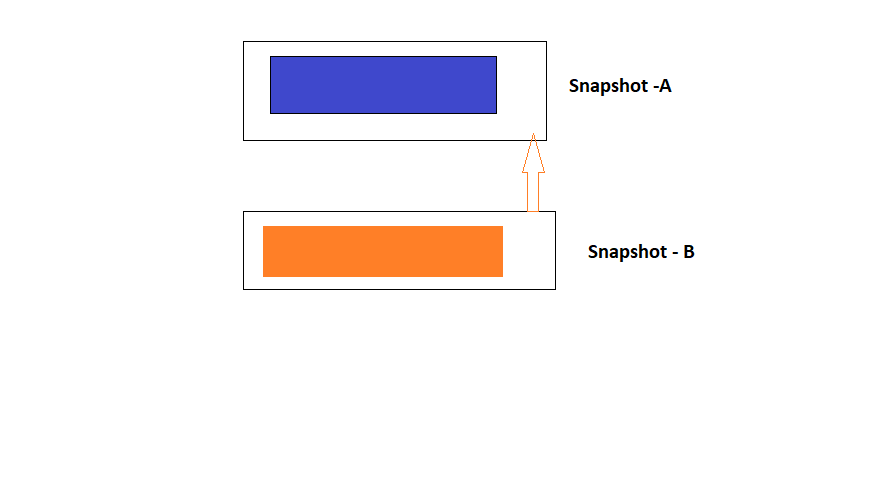
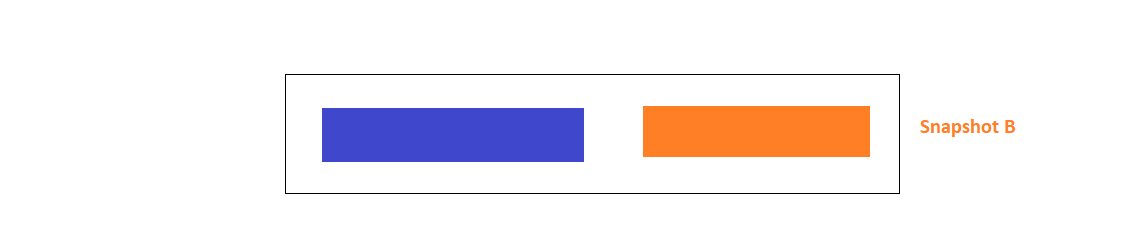
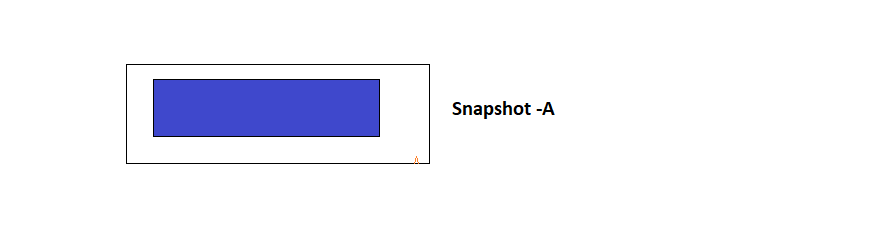
* Copy the snapshot to your desired Region
* Using the copied snapshot create Volume
* 

### How Snapshot Creation Works



* First snapshot of the volume is Full disk back up.
* All the snapshots after first will be incremental backups.

### How about Snapshot deletions

* Scenario: you have two snapshots as shown below
* 
* If any other snapshot is depending on the current snapshot to be deleted, then contents of the current snapshot will be copied to snapshots depending on it
* If there are no depending snapshots on current snapshot , then the Snapshot is deleted. (so the change will be lost)
* Deleting Snapshot A: 
* Deleting Snapshot B: 
* Deleting the Latest snapshots might lead to data loss, but older snapshots will not lead to data loss.

**NOVEMBER 22, 2019**

## AWS Classroom Series – 22/Nov/2019

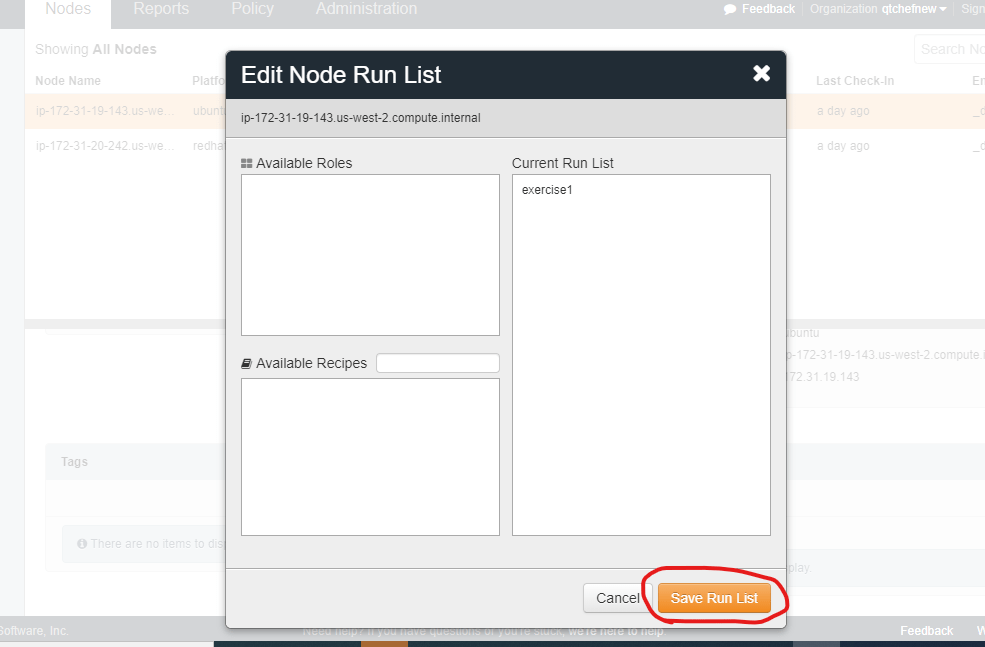
### General Purpose SSD VS Provisioned IOPS SSD

* Disk Speed will be varying in General Purpose SSD
* Provisioned IOPS speed is consistent.
* In Provisioned IOPS Max IOPS to Disk Size Ratio is 50:1.
* Minimum IOPS is 100
* In General Purpose SSD we have burstable volumes where you get credits for using hard disks less than (In terms of IO Activity) baseline. These credits get utilized when the disk is in need.

### Cost Overview.

* [Refer Here](https://calculator.s3.amazonaws.com/index.html) (<https://calculator.s3.amazonaws.com/index.html>)  for pricing calculator

### Automated Snapshot Creation

* Managed by Life cycle Manager.
* Select the tags for the volumes or instances
* Create a schedule of backups and number of snaphots to be retained. 

### Taking Snapshots from AWS CLI

* AWS CLI Configuration:
* Install AWS CLI. [Refer Here](https://serverless-stack.com/chapters/create-an-iam-user.html) (<https://serverless-stack.com/chapters/create-an-iam-user.html>)
* Create an AWS IAM User [Refer Here](https://serverless-stack.com/chapters/create-an-iam-user.html) (<https://serverless-stack.com/chapters/create-an-iam-user.html>)
* Configure AWS CLI

aws configure

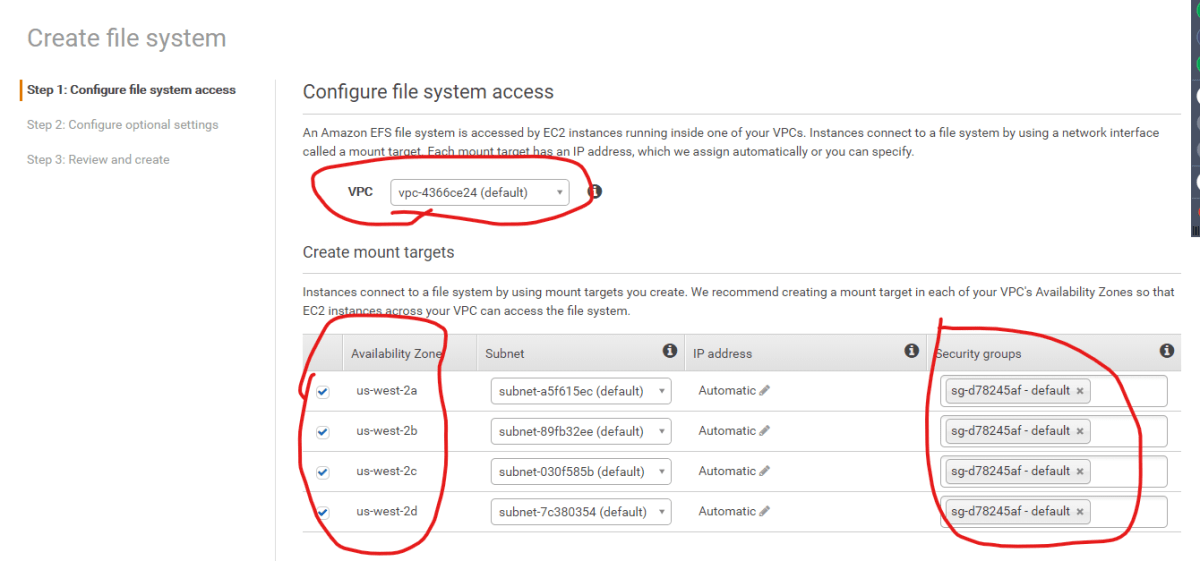
* Creating snapshots from cli.

aws ec2 create-snapshot --description 'for fun' --volume-id vol-03b1d00dc8be488f8

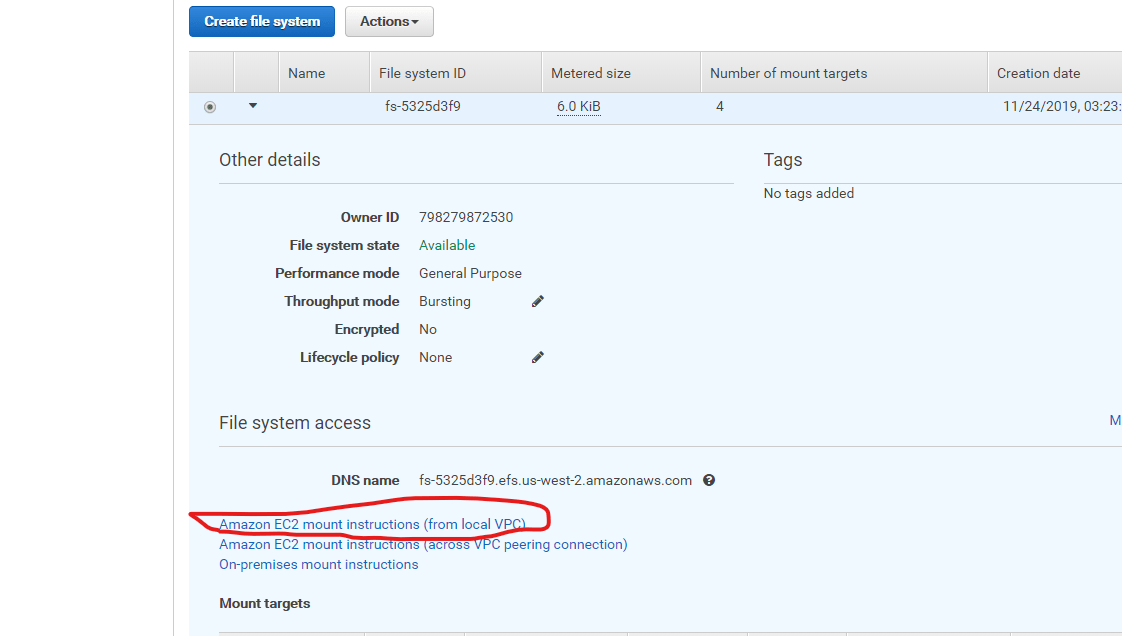
**NOVEMBER 24, 2019**

## AWS Classroom Notes – 24/Nov/2019

### Elastic File System

* Network Disk.
* While Creating EFS users should choose the az’s where EC2 instances are existing. 
* EFS is a Network Disk which can be mounted from any Linux EC2 Instance
* Whenenver EFS is created, user selects the AZs and Security Groups which determine the access to EFS.
* Once the EFS is created it can be mounted on AWS EC2 linux instances.

### Exercise:

* Create two linux instances with Amazon AMI in two different subnets
* Create a security group. (Open all is fine for now but not in enterprise usecases)
* Create an EFS with size of 1GB in default VPC and select all the AZ.
* Use Mount Instructions to mount the efs on to your ec2 instance. 
* Now do the tests by creating files or folders in one instance and checking in other instance.

**NOVEMBER 25, 2019**

## AWS Classroom Series – 25/Nov/2019

### Simple Storage Service(S3)

* Blob (Binary Large object) Storage
* Unlimited Storage.
* Each file/blob size cannot be greater than 5 TB
* S3 is organized into
  + Buckets
* Each Bucket has files (objects) and folders
* All the objects inside buckets can be accessed over https
* In S3 Buckets you can give permissions only to AWS (IAM) users or to everyone.
* All the objects are exposed over https, helps in achieving following use cases
  + Dropbox/Google Drive
  + Amazon Prime Video/Music/Netflix
  + Downloads in Websites

### Cost of S3

* S3 cost includes
  + Size of Storage
  + Accessing the Storage
* Depending on your use case AWS gives different Storage Classes
  + Standard Storage:
    - Paying more for storage
    - Less for Accessing
    - Redundancy levels will be high
  + Reduced Redundancy:
    - Redundancy Levels will be less
  + Infrequent Access:
    - Paying more for Accessing
    - Paying Less for Storage
  + Glacier:
    - Archival kind of storage.
    - Storage cost will be much less
    - Zero Access cost (bcoz you cannot access the data directly from glacier)
  + Intelligent Tiers:
    - Amazon will automatically choose storage class depending on usage.
* Default Storage class is standard

### Content Delivery Networks

* CDNs have
  + Origin Servers
  + Points Of Presence Servers.
* [Refer Here](https://infrastructure.aws/) (<https://infrastructure.aws/>)  for AWS Point of Presence Servers
* Data will be stored in S3 Buckets.
* For CDN
  + S3
  + Cloudfront (CDN)
  + Media Services.

### Static Web Sites

* Any website which is developed in html, javascript and css (Static Websites) can be directly hosted on s3 buckets.

**NOVEMBER 28, 2019**

## AWS Classroom Series – 28/Nov/2019

### AWS CLI Setup

* Install AWS Cli. [Refer Here](https://docs.aws.amazon.com/cli/latest/userguide/install-windows.html) (<https://docs.aws.amazon.com/cli/latest/userguide/install-windows.html>)
* Chocolatey: choco install awscli -y
* Create an IAM User by following instructions [from here](https://serverless-stack.com/chapters/create-an-iam-user.html)(<https://serverless-stack.com/chapters/create-an-iam-user.html>)

aws configure

# Enter Access Key, Secret Key and Region

### S3 CLI

* AWS S3 has two CLI commands
  + [aws s3](https://docs.aws.amazon.com/cli/latest/reference/s3/) (<https://docs.aws.amazon.com/cli/latest/reference/s3/>)
  + [aws s3api](https://docs.aws.amazon.com/cli/latest/reference/s3api/)(<https://docs.aws.amazon.com/cli/latest/reference/s3api/>)

### S3 CLI

* Available commands
  + cp => copy
  + ls => list
  + mb => make bucket
  + mv => move
  + rb => remove bucket
  + rm => remove
  + sync => sync source and destination
* aws cli syntax

aws s3 <s3 command> --param1 value

* s3 bucket uri syntax: s3://<bucketname>

### Examples

* Create bucket with name **qts3bucketfromcli**

aws s3 mb s3://qts3bucketfromcli

### Exercise using AWS CLI

1. Create any s3 bucket

aws s3 mb s3://qts3fortesting-1

1. Copy a file from local machine to aws s3 bucket

aws s3 cp '.\Advanced Agile Material.docx' s3://qts3fortesting-1

1. Copy a folder from local machine to aws s3 bucket

aws s3 cp --recursive .\ec2\ s3://qts3fortesting-1

1. Create a directory with following files
   * 1.txt
   * 2.txt
   * 3.xml
   * 4.json
2. Copy all the files to s3 bucket except 3.xml

aws s3 cp ".\testing\" s3://qts3fortesting-1 --recursive --exclude 3.xml

1. Create one more s3 bucket and copy the contents of the above bucket to new bucket

aws s3 mb s3://qts3fortesting-2

aws s3 cp s3://qts3fortesting-1/ s3://qts3fortesting-2 --recursive

1. remove the old bucket.

aws s3 rb s3://qts3fortesting-1/ --force

aws s3 rb s3://qts3fortesting-2/ --force

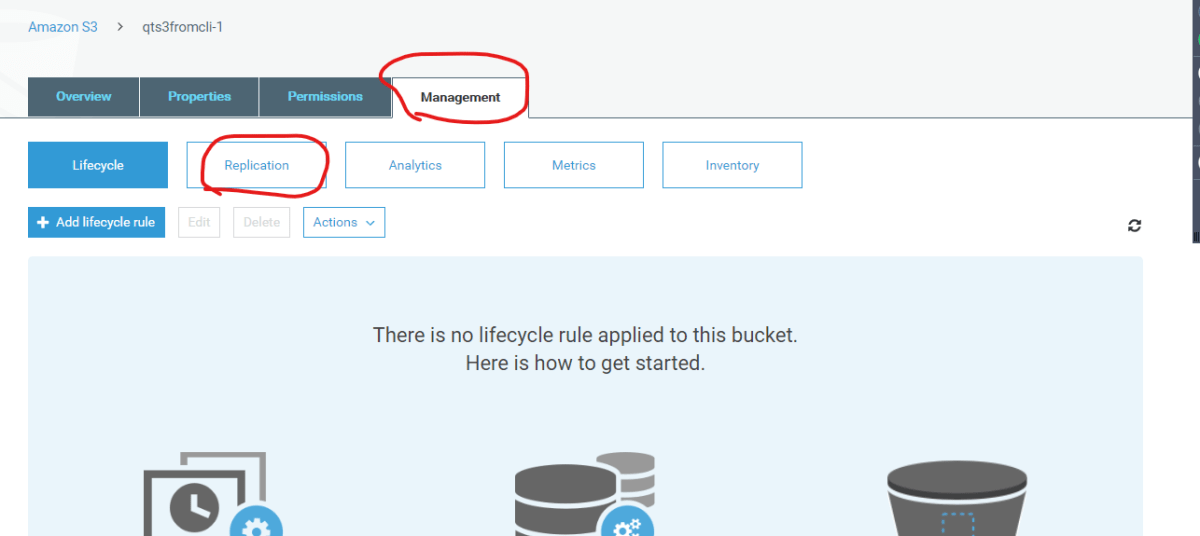
**NOVEMBER 30, 2019**

## AWS Classroom Series – 29/Nov/2019

### S3 CLI

* Create a script to create
  + AWS S3 bucket
  + Create a folder in s3 bucket
  + Sync any folder on your laptop to aws s3 bucket folder
* Create a script to sync s3 bucket to local folder
* use s3 api cli to list all the buckets
* use s3 api cli to list all the objects in a bucket

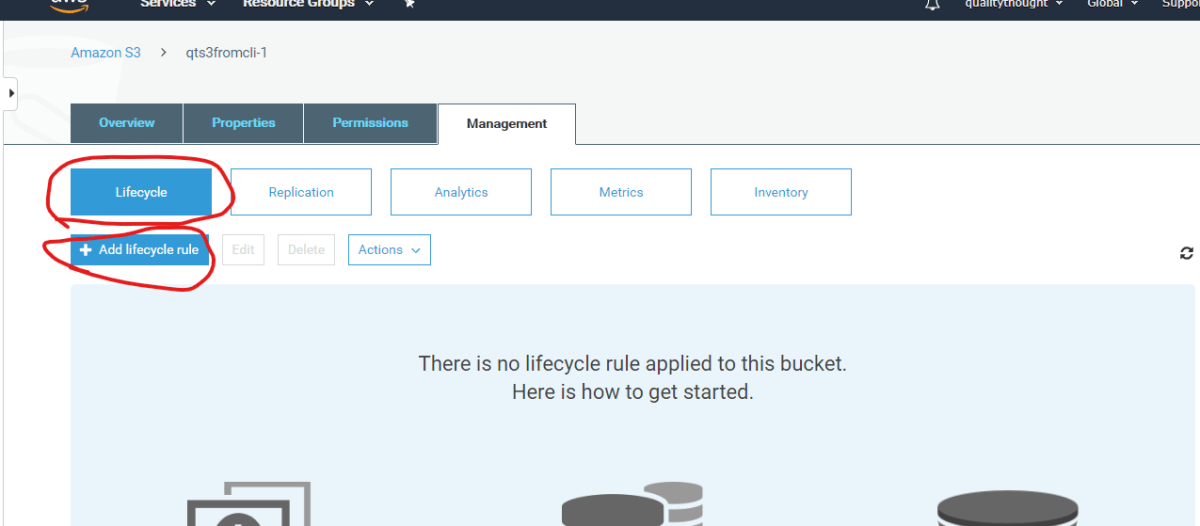
### Replication

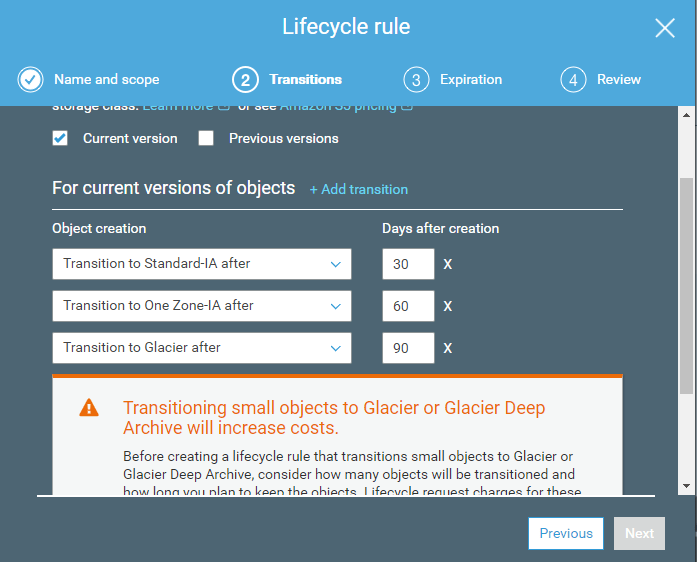
* All the contents of S3 bucket can be replicated to any other s3 bucket.
* Other Bucket can be from same account or different account 

### Storage Classes

* Standard
* Reduced Redundancy
* Infrequent Access
* Intelligent Tiering
* Glacier

### LifeCycle for changing storage classes automatically

* Change storage classes for cost & space optimization 



**DECEMBER 1, 2019**

## AWS Classroom Series – 01/Dec/2019

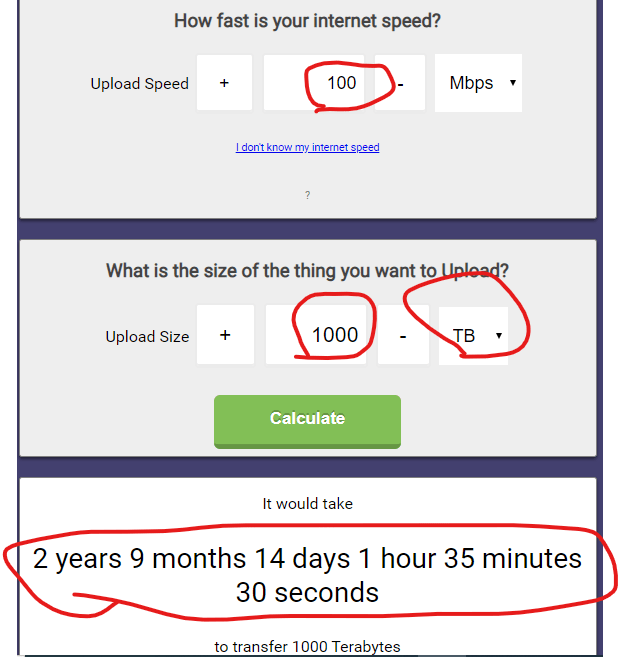
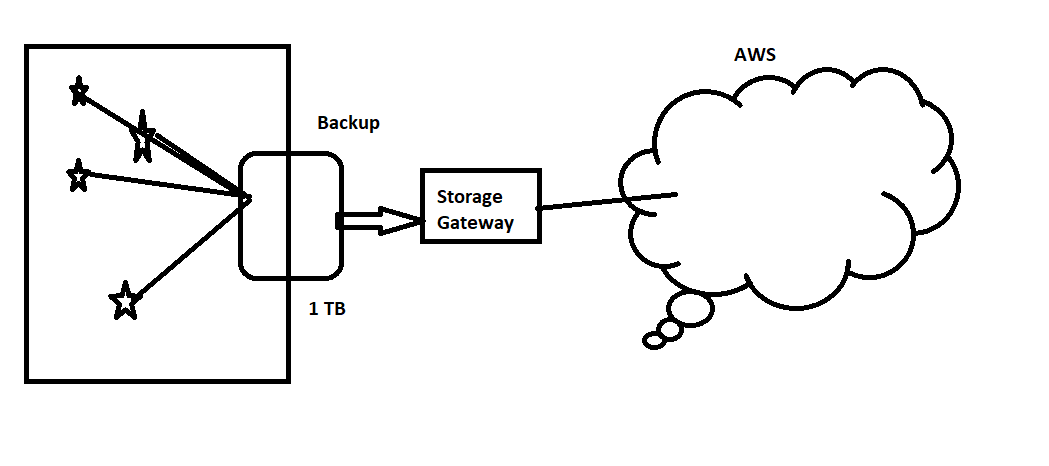
### Glacier vs Glacier Deep Archive

* Cost
* Move data to archive (Glacier), reading data is no longer possible.
* If you want access to data change storage class to Standard or Infrequent Access
* Deep Archive gives option to move the data from archive to other storage classes for a period of time. [Refer Here](https://aws.amazon.com/blogs/aws/new-amazon-s3-storage-class-glacier-deep-archive/) (<https://aws.amazon.com/blogs/aws/new-amazon-s3-storage-class-glacier-deep-archive/>)

### Moving Data to Glacier

* Life cycle rule
* glacier cli

### Transferring Data from onpremise to S3

* If the data is relative small, use aws s3 console or aws s3 cli
* If the data is huge, then lets estimate the time of upload using aws s3 cli or console 
* Amazon has [Snowball Product line](https://aws.amazon.com/snowball/). (<https://aws.amazon.com/snowball/>)
* AWS Snowball Devices [Refer Here](https://docs.aws.amazon.com/snowball/latest/ug/device-differences.html) (<https://docs.aws.amazon.com/snowball/latest/ug/device-differences.html> )
* AWS SnowMobile is upto 100 Peta Bytes
* Transferring the data consistently every day to AWS s3 is challenging, then AWS Storage Gateway can help. Fit Storage Gateway in your Data center and ensure you copy the data to storage gateway, this syncs the data to AWS. 

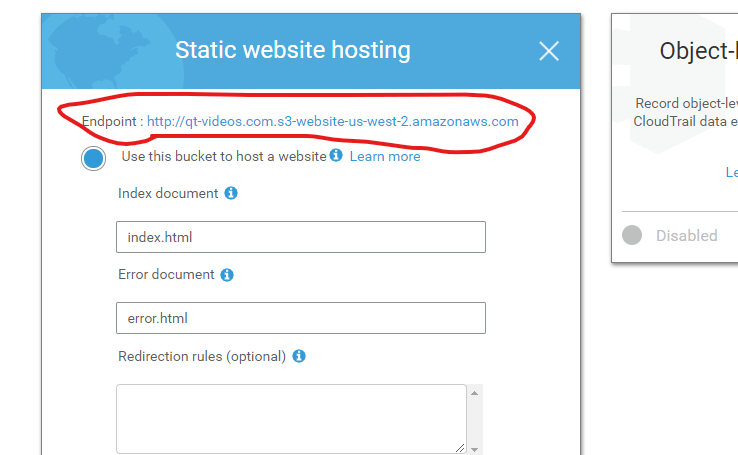
### Versioning in S3

* Versioning can be enabled at bucket level.
* Versioning can be suspended, no earlier versions are deleted post suspension.
* Deleting older versions is user responsibility.

### Default Access Policy

* By default public access is blocked.
* Each object can be made public

### Static Website Hosting

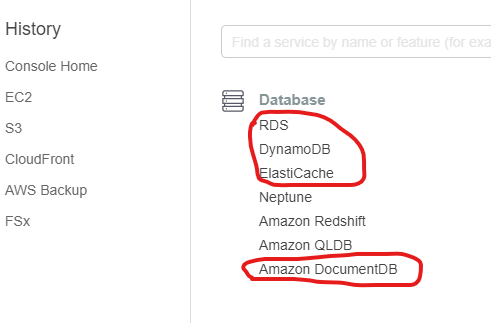
* Create a s3 bucket with name like domain names (example.com) and disable blocking all public access
* Create an index.html (home) page and upload to s3
* Create an Error.html (error) page and upload to s3
* Properties => Static Website Hosting and enter index and error html path
* Now make a note of endpoint url 

# DATABASES

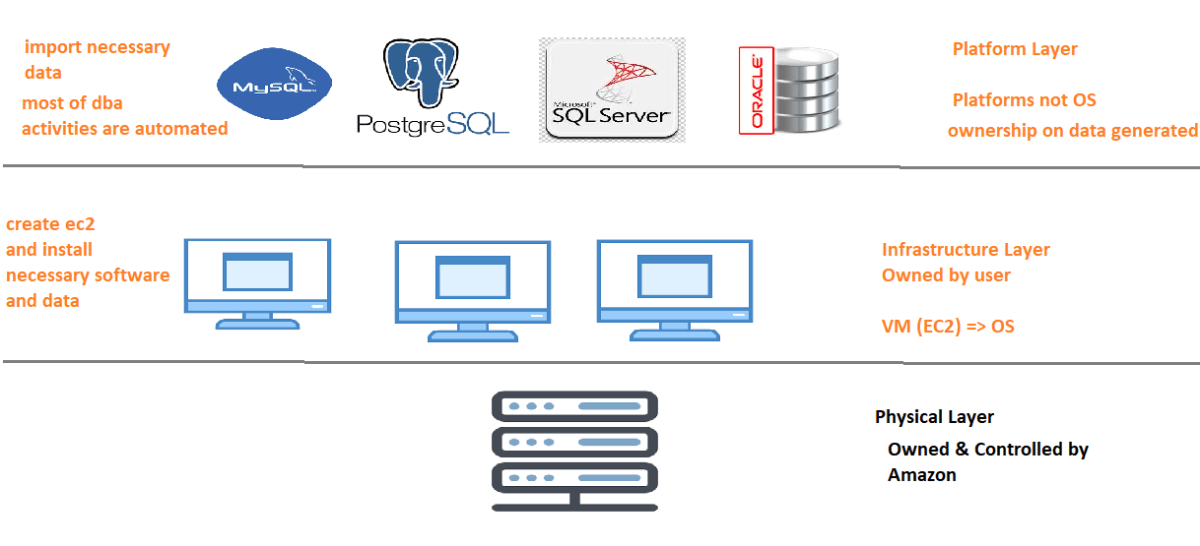
**DECEMBER 2, 2019**

## AWS Classroom Series – Database – 02/Dec/2019

### AWS Database Services

* Data Services of AWS 

### Service Models

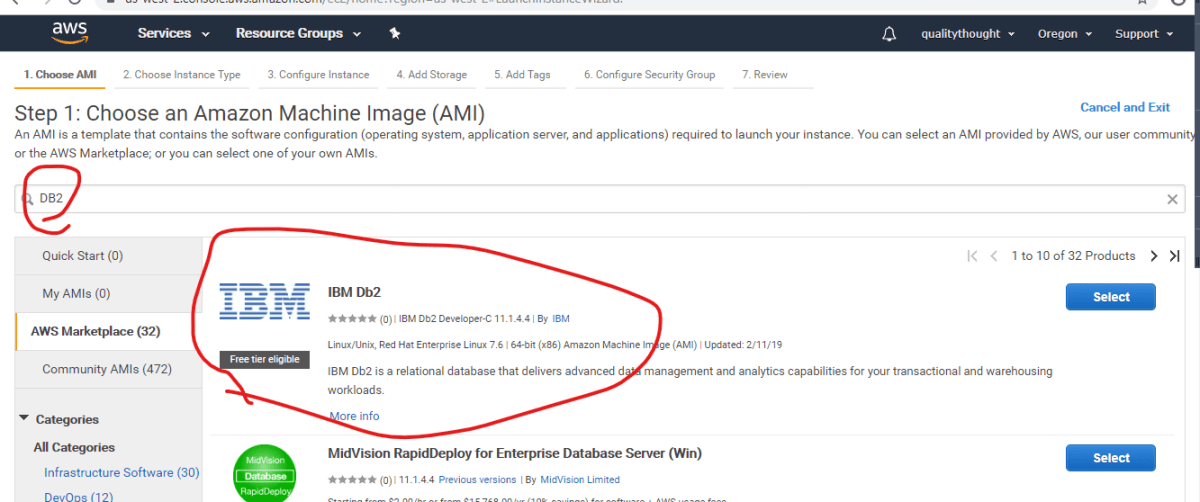


* Physical:
  + Completely owned by Amazon
  + users have no control
* Infrastructure
  + Create ec2 machines and install databases
  + Require DBA
  + OS patching, DB patching, networking is users responsibility
  + This is often referred as unmanaged
  + Pros:
    - Full control of DB
    - Any version can be installed
  + Cons:
    - Administrative Efforts are huge
* Platform
  + Create database platform (mysql/maria/Aurora, postgres/Aurora, SQL server, Oracle)
  + Minimal DBA activities
  + NO need of any patching. Users need to maintain only data
  + This is often referred as Managed Database Services
  + Pros:
    - Administrative Efforts are minimal
  + Cons:
    - Specific list of databases/versions are supported.

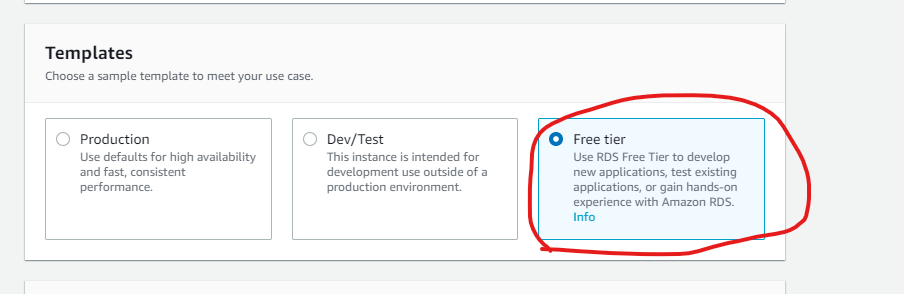
### RDS (Relational Database Services)

* This is managed Service for Relational Databases
* RDS supports
  + mysql
  + mariadb
  + SQL Server
  + Oracle
  + Postgres
* RDS introduces
  + Aurora for mysql
  + Aurora for Postgres
* RDS automates
  + Backup/Restore
  + Failover
  + Replications
* DBAs are required for Query Performance Tuning. Amazon gives necessary tooling support.

### AWS Marketplace Images

* These are AMIs (Amazon Machine Images) with Preinstalled Databases.
* But rest of the activites as shown below are user responsibilities.
  + License Mangement
  + Backup/Restore
  + Failover
  + Replications
* It is not recommended to use Marketplace for supported Databases.
* It is used to create only unsupported Databases.
* In the below image, IBM DB2 is shown as the marketplace image 

### Create a free tier mysql rds instance

* Database => mysql
* Database instance type => db.t2.micro
* Select template => Free Tier 

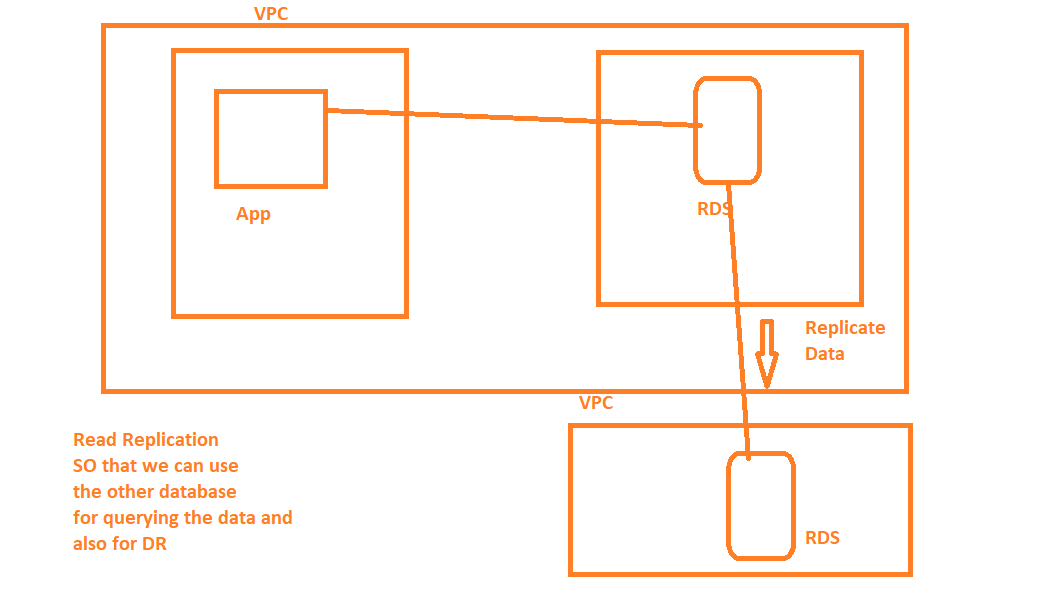
**DECEMBER 3, 2019**

## AWS Classroom Series – RDS-03/Dec/2019

### Relational Database Services

* RDS Features
  + Read Replication
  + Multi-AZ Deployment
  + Automatic Minor Version Upgrades
  + Database shutdown
  + Snapshots
* DB Subnet Groups:
  + Whenever RDS is created it needs at least two subnet-ids
  + DB Subnets help in selecting network and subnets (regions and AZs)
* Reserved Instances:
  + Give a long term commitment to Amazon for 1 or 3 years for RDS usage can save costs upto 58%

### Read Replication



* Promotion is giving write permissions to read replica.

### Multi-AZ Deployment

* Database is created in multiple AZs
* Due to this failover is automatic
* Costs will be on the higher side.

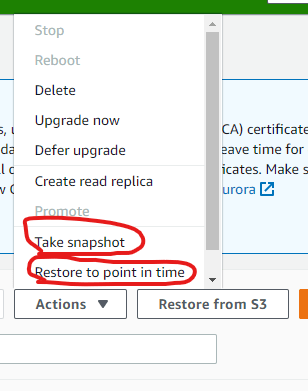
### Automatic Minor Version Upgrades

* User can enable this feature which upgrades minor releases but you have to suggest the time period when your database is used less (maintenance Window).

### Database Shutdown

* Database can be stopped or rebooted.
* Database can be shutdown only for 1 day continuously.

### Database Snapshots

* Backup of Database is called as snapshots
* Snapshots can be created manually or you can schedule them. 

### Exercise

* Create an RDS mysql instance with t2.micro (free tier)
* Ensure you select last 2 days bakcup.
* Create a Read Replica in other Region
* Promote Read Replica.
* Delete all the RDS instances and Snapshots

**DECEMBER 4, 2019**

## AWS Classroom Series – 04/Dec/2019

### Performance Insights

* Is the toolset for Query performance optimization
* Integrated with all the databases in AWS Console.

### NO SQL Databases

* Popular Databases
  + Mongo
  + Cassandra
* Amazon Offerings
  + Dynamo DB
  + Document Database (Mongo db)

### Cache Databases

* Elastic Cache
* Offering
  + Memcached
  + Redis

### Exercise from CLI

* Create an rds instance from AWS CLI
* Create a read replica
* create a snapshot
* Delete all of the above.
* Add an additional disk to existing linux and windows EC2 instance. (mkfs, mount, xfs/ext4)

**DECEMBER 5, 2019**

## AWS Classroom Series – 05/Dec/2019

### What is special about Aurora

* Aurora is complete database engine rewrite for
  + mysql
  + postgres
* [Refer Here](https://www.stratoscale.com/blog/dbaas/aurora-vs-rds/) (<https://www.stratoscale.com/blog/dbaas/aurora-vs-rds/>)
* [Refer Here](https://www.vividcortex.com/blog/three-things-that-differentiate-amazon-aurora-from-mysql) (<https://www.vividcortex.com/blog/three-things-that-differentiate-amazon-aurora-from-mysql>)

### Mounting a new ebs volume on ec2 machine

* For Linux [Refer Here](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-using-volumes.html) (<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-using-volumes.html>)
* For Windows [Refer Here](https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/ebs-using-volumes.html) (<https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/ebs-using-volumes.html>)

### Overview

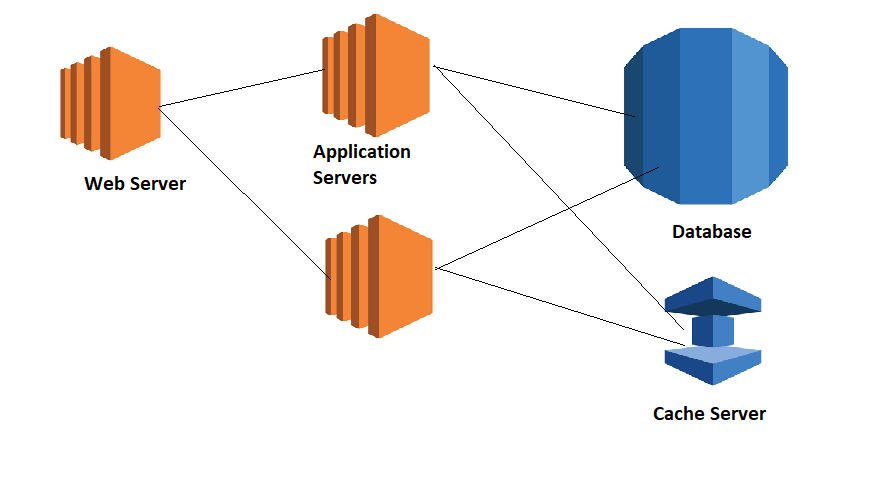
* Disks:
  + EBS
    - General Purpose
    - Provisioned IOPS
  + Instance Store
  + Elastic Volumes:
    - Increase the size of disk while instance is running
* Network Disks
  + EFS
* Blob Storage
  + S3
* S3:
  + Backup
  + Media
    - Cloudfront
  + Archive:
    - Glacier
    - Glacier Deep Archive
* Relational Databases
  + RDS
    - mySQL
    - Postgres
    - Oracle
    - Microsoft SQL Server
  + RDS Features
    - Read-Replication
    - Multi-AZ
    - Automated Backups
    - Manual Backups
  + NOSQL:
    - Dynamo DB
    - Document DB
  + Cache:
    - Memcached
    - Redis

# OPERATIONS (Monitoring, Logging, Backup and Systems Manager)

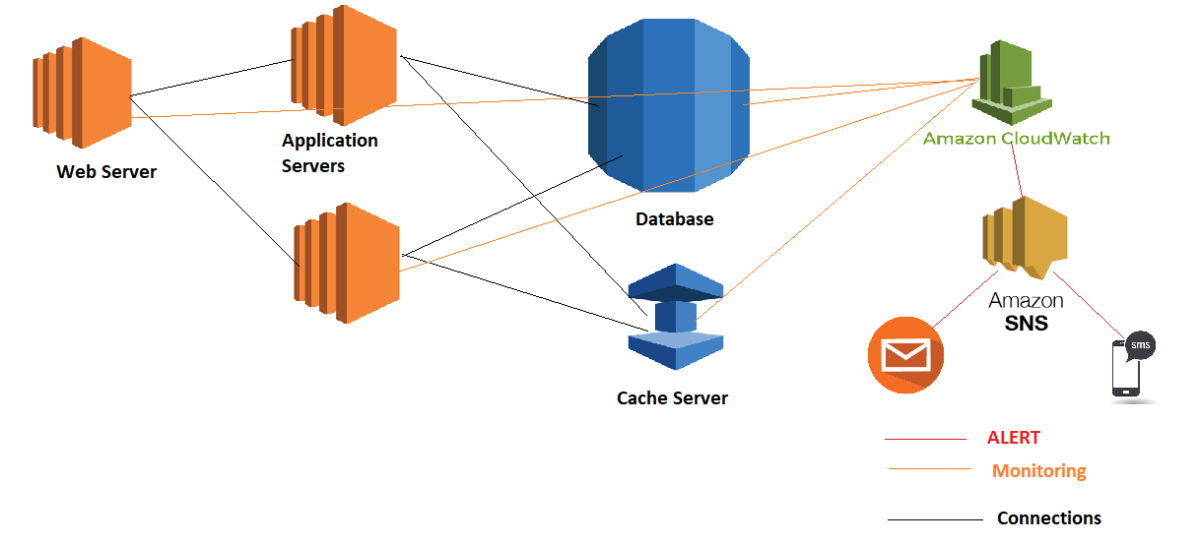
**DECEMBER 10, 2019**

## AWS Classroom Series – 10/Dec/2019

### Sample Architecture

* Consider this Sample Architecture 

### Monitoring & Logging

* We need an alerting System to notify the errors as well as warnings 
* For Logs also Cloudwatch is used.
* Cloud Trail can be used for Audit Logs.
* Prereqs:
  + Individual Resource Creation and Default Monitoring

### Responsibilities

* Creating an Alerting Mechanism for the failures defined
* Creating a dashboard which shows all the important metrics
* Integration of Alerts with External Systems.
* Create Log Queries for anamoly detection.

### Batch

* Batch Processing Using AWS Batch

### Backup & Recovery

* Create a Backup & Recovery Mechanism for
  + EC2 Instances (Virtual Machines)
  + RDS Instances (Databases)
  + EBS Volumes (Disks)
* Prereqs:
  + How to create EC2, RDS, EBS
  + Where the backup has to be stored (Region/AZ)

### Systems Management

* OS Patching
* Software installations on EC2 Instances
* Configuration Management on EC2 Intances
* Service: EC2 Systems Manager
* Prereqs:
  + EC2 Instance Creation
  + Usecases

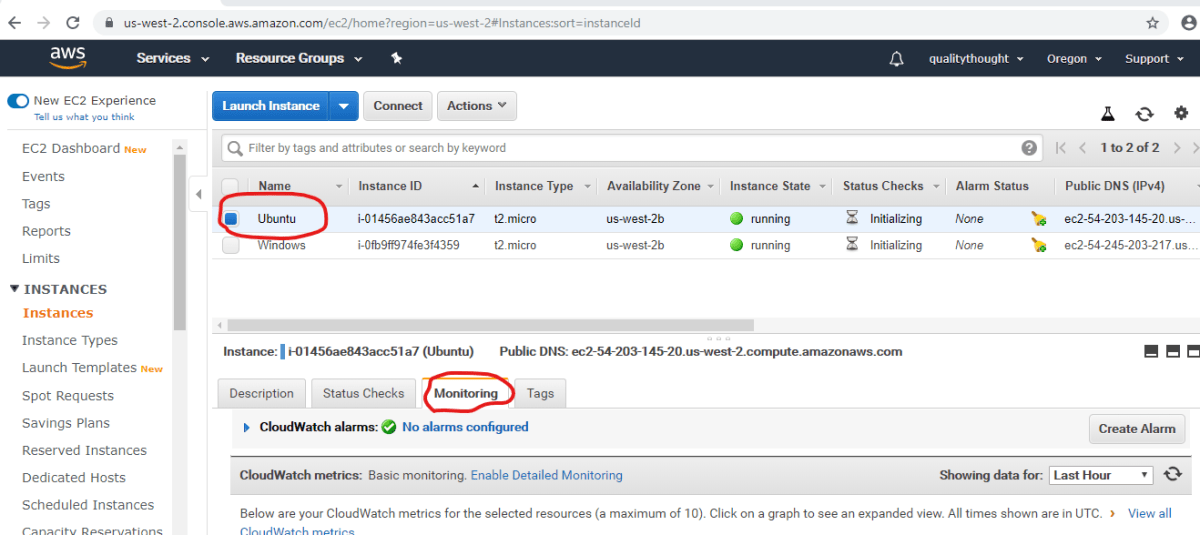
### Templating (IAC)

* Create CloudFormation Templates for Architecture/Application Deployments
* Prereqs:
  + JSON
  + Manually how to deploy the application architecture.

**DECEMBER 11, 2019**

## AWS Classroom Series – 11/Dec/2019

### Monitoring single instances

* Create an EC2 instance
  + Ubuntu 18.04 AMI
  + Windows 2016 Server AMI
* Inbuilt graphs are rendered by AWS once the instance is created. 
* Now navigate to Cloudwatch Service and Click on Metrics

### Creating a Graph for Monitoring CPU, DISK and Network on Linux Instance

* Navigate to Per-Instance Metrics, Select the Ubuntu Machine with following Metrics
  + CPU Utilization
  + Disk WriteOps
  + Disk ReadOps
  + Network PacketsIn
  + Network PacketsOut
* Change the graph type to Number
* Add this graph to Dashboard

### Creating a Graph for Monitoring CPU, DISK and Network on Windows Instance

* Navigate to Per-Instance Metrics, Select the Windows Machine with following Metrics
  + CPU Utilization
  + Disk WriteOps
  + Disk ReadOps
  + Network PacketsIn
  + Network PacketsOut
* Change the graph type to Number
* Add this graph to Dashboard

### Creating an Area Graph Comparing CPU Utilization of Windows and Ubuntu Instances

* Navigate to Per-Instance Metrics, Select the Ubuntu Machine and Windows Machine with following Metric
  + CPU Utilization
* Change the graph type to Stacked
* Add this graph to Dashboard

### Exercise

* Create a Line Graph Comparing Network Packets In for Windows and Ubuntu Instances
* Create a Number Comparing Network Package Out for Windows and Ubuntu Instances.

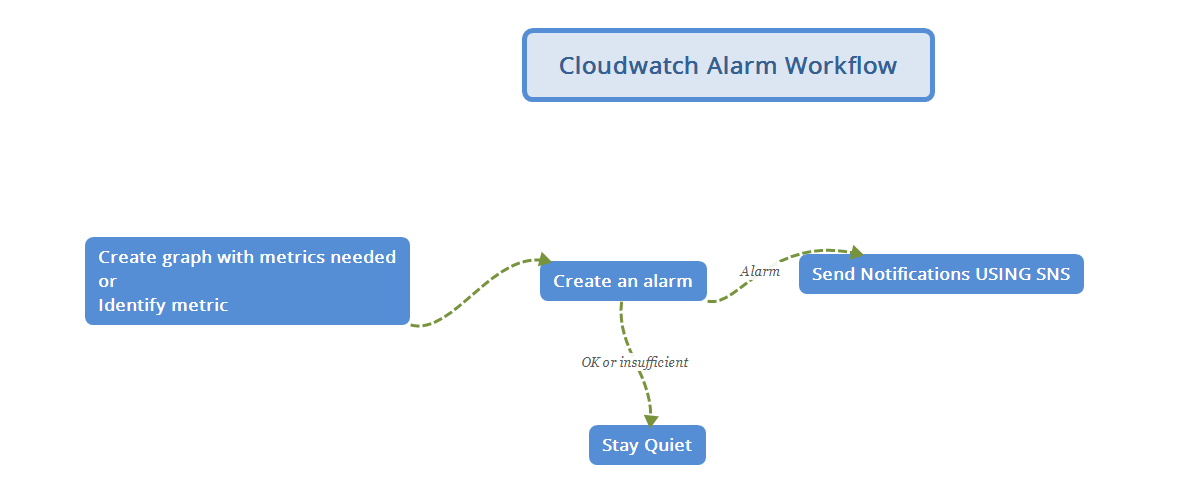
**DECEMBER 12, 2019**

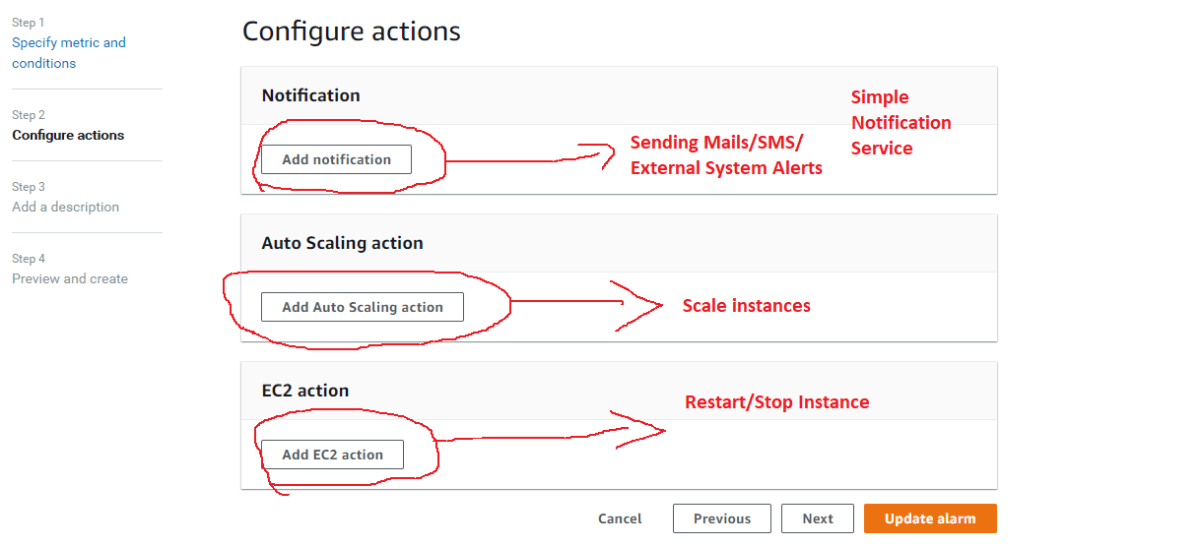
## AWS Classroom Series 12/Dec/2019

### Cloudwatch Terms

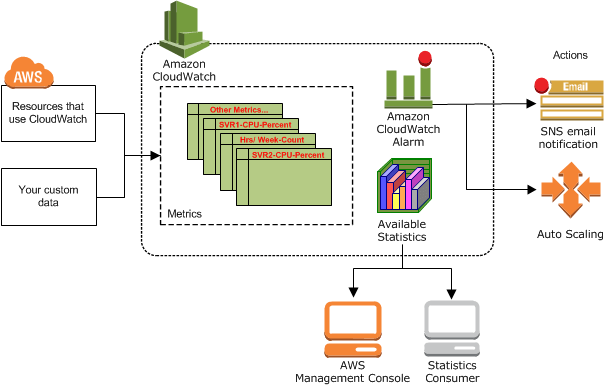
* Metric: Any measurable unit of the Infrastructure/Application/Platform. Eg CPU Utilization, Disk Read Bytes, Network Packets Out
* Unit: measurement unit generally something like Bytes/Sec, Percentages
* TimePeriod: Duration of the metric
* Aggregations: Min, Max, Sum, Count, Average, Percentile
* Alarm: Is state when something measurable is wrong.
  + When CPU Utilzation is > 85% for 10 mins
  + When free disk space < 1GB
  + When Network Packets IN == 0 for 10 mins
* Alarm States:
  + Alarm: When the Alarm condition is met for eg CPU Utilization is > 85% for over 10 mins
  + OK: When Alarm condition is not met for eg CPU Utilization is == 74% for over 10 mins
  + Insufficient: Cloudwatch doesnot have data of timeperiod. eg Instance has been create 2 mins ago. Data is not sufficent to recognize the alarm or ok state.

### Basic Cloudwatch Alarming

* Workflow
* 
* Alarms can do the following action when state is Alarm
  + Send Notifications using SNS
  + EC2 AutoScaling Action
  + EC2 Actions



### AWS CloudWatch Overview



note: From official AWS docs

### Next Steps

* Cloudwatch and EC2
* Cloudwatch and RDS
* Cloudwatch and ELB
* Other Service Integrations

**DECEMBER 13, 2019**

## AWS Classroom Series – 13/Dec/2019

### Ec2 Actions

### Exercise-1 Create a Cloud Watch Alarm to Stop Ec2 Machine when CPU Utilization < 5

* Create an Ec2 Instance
* Create an alarm for CPU Utilization < 5 for 5 minutes
* Add an EC2 action to stop(shutdown) the Ec2 instance
* Wait till the alarm is raised and observe ec2. Ec2 instance should be stopped

### Exercise-2 Create a Cloud Watch Alarm which reboots the EC2 instance when CPU Utilization = 100 for 5 minutes

* Create an Ubuntu Ec2 Instance.
* SSH into Ubuntu Instance and execute the following commands

sudo apt-get update

sudo apt-get install stress -y

stress --cpu 8 --timeout 100m -v

* Create an alarm with above spec to reboot ec2 instance

### Exercise-3: Create a Cloud Watch Alarm which terminates the EC2 instance when CPU Utilizations is < 20 for 120 minutes

### Simple Notification Service

* Email Notifications using SNS
* Create SNS Topic
* Each Topic has subscriptions. Create an Email Subscription.
* Confirm the Subscription.
* For all the above exercise send the notification to topic.

### Note

* [Refer Here](https://directdevops.blog/2019/09/12/track-your-aws-free-tier-usage/) (<https://directdevops.blog/2019/09/12/track-your-aws-free-tier-usage/>)  to track your free tier usage.

**DECEMBER 17, 2019**

## AWS Classroom Series – 17/Dec/2019

### AWS Logging and Audit Trail

* Cloudwatch logs
* Cloud Trail

### Cloudwatch Logs

* Create a log group
* Using Insights create queries/visualizations for logs
* [Refer Here](https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/CWL_QuerySyntax.html) (<https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/CWL_QuerySyntax.html>)  for Query Language for Searching Logs

### Installing Agent

* Install Agent From [here](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/download-cloudwatch-agent-commandline.html) (<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/download-cloudwatch-agent-commandline.html>)
* Create necessary roles from [here](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/create-iam-roles-for-cloudwatch-agent-commandline.html) (<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/create-iam-roles-for-cloudwatch-agent-commandline.html>)
* Execute the agent on Ec2 Server from [here](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/install-CloudWatch-Agent-commandline-fleet.html) (<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/install-CloudWatch-Agent-commandline-fleet.html>)

**DECEMBER 18, 2019**

## AWS Classroom Series 18/Dec/2019

### CloudWatch Events

* As a response to Cloud Watch Events, Actions can be created.
* Event Rules can be created. Event Rules have
  + Event Source: Any Resource of AWS
  + Target(s):
    - SNS
    - Lambda functions
    - SSM run-commands

### Billing Alerts

* [Refer Here](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/monitor_estimated_charges_with_cloudwatch.html) (<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/monitor_estimated_charges_with_cloudwatch.html>)

### Scenarios

* I want to install a necessary software in all the ubuntu ec2 instances running in my account.
* Your manager wants to patch all the windows servers
* Create an Image(AMI) of all the machines which are running for over 1 year and delete them if they are in Dev/Test Environments

### Solution

* AWS Offers a Service specially designed for EC2 instances which can help in simplifying complex tasks called EC2 Systems Manager.

**DECEMBER 19, 2019**

## AWS Classroom Notes – 19/Dec/2018 – Systems Manager

### Sample using Systems Manager

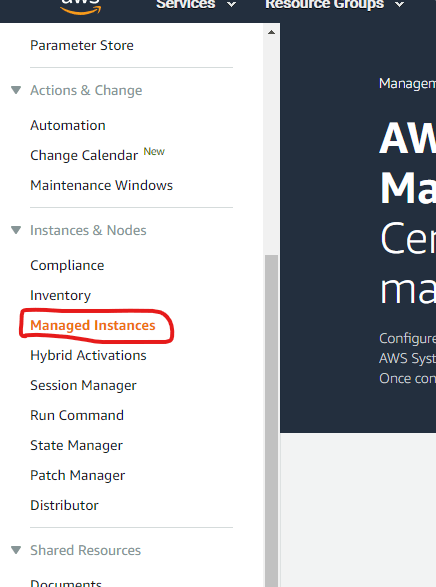
* Create two Linux Ec2 Instances (Ubuntu). Tag one machine as Dev and other as Test
* Navigate to Systems Manager.

### Managed Instance

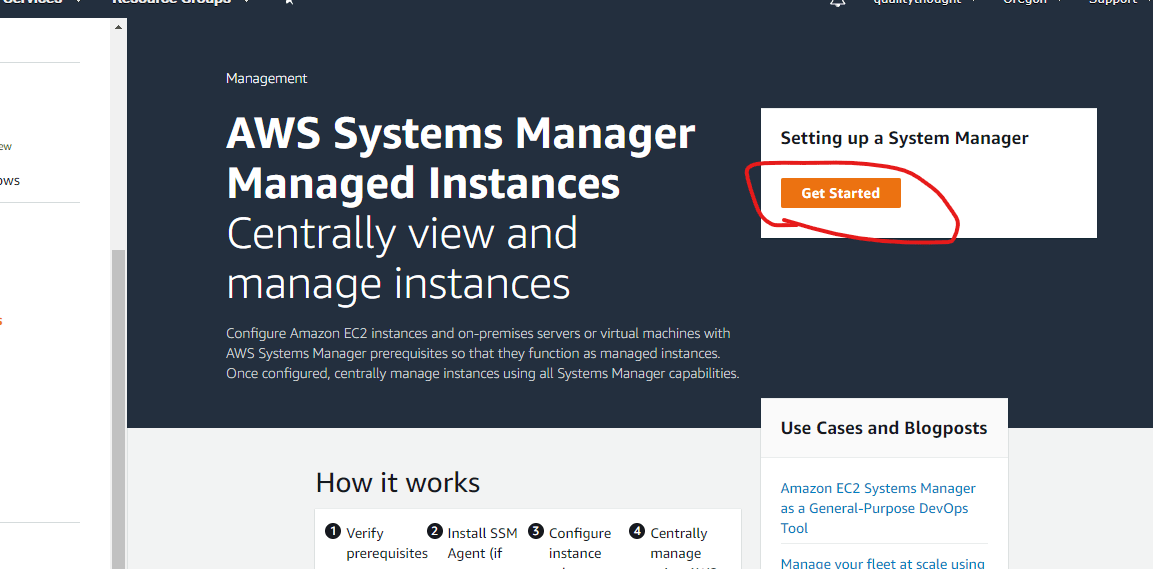
* Any Ec2 Instance that can be managed by Systems Manager is Managed Instance

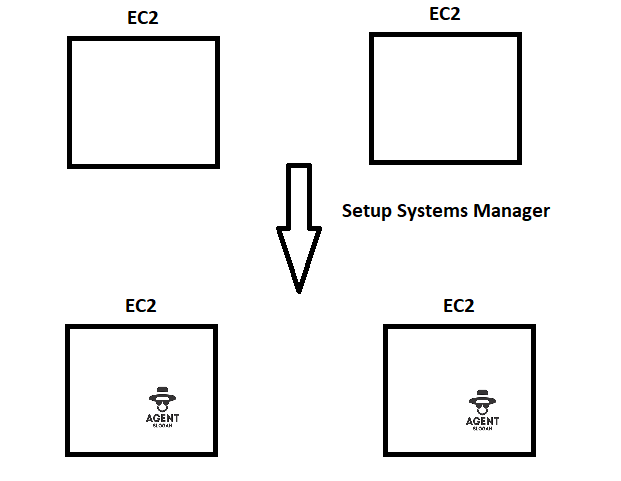
### Activities:

* Add Existing Ec2 instances to be managed by Systems Manager
* Navigate to Systems Manager and Select Managed
* Instances



* Select Get Started



* [Refer Here](https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-quick-setup.html) (<https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-quick-setup.html>) for setup
* Select the two instances and setup Systems Manager, agents will be installed on the instances which can be managed by Systems Manager 
* Lets update the Packages in both the machines using **Run Command**

sudo apt-get update

### Exercises

* Install apache server on Dev

sudo apt-get update

sudo apt-get install apache2 -y

* Install JDK8 server on Test

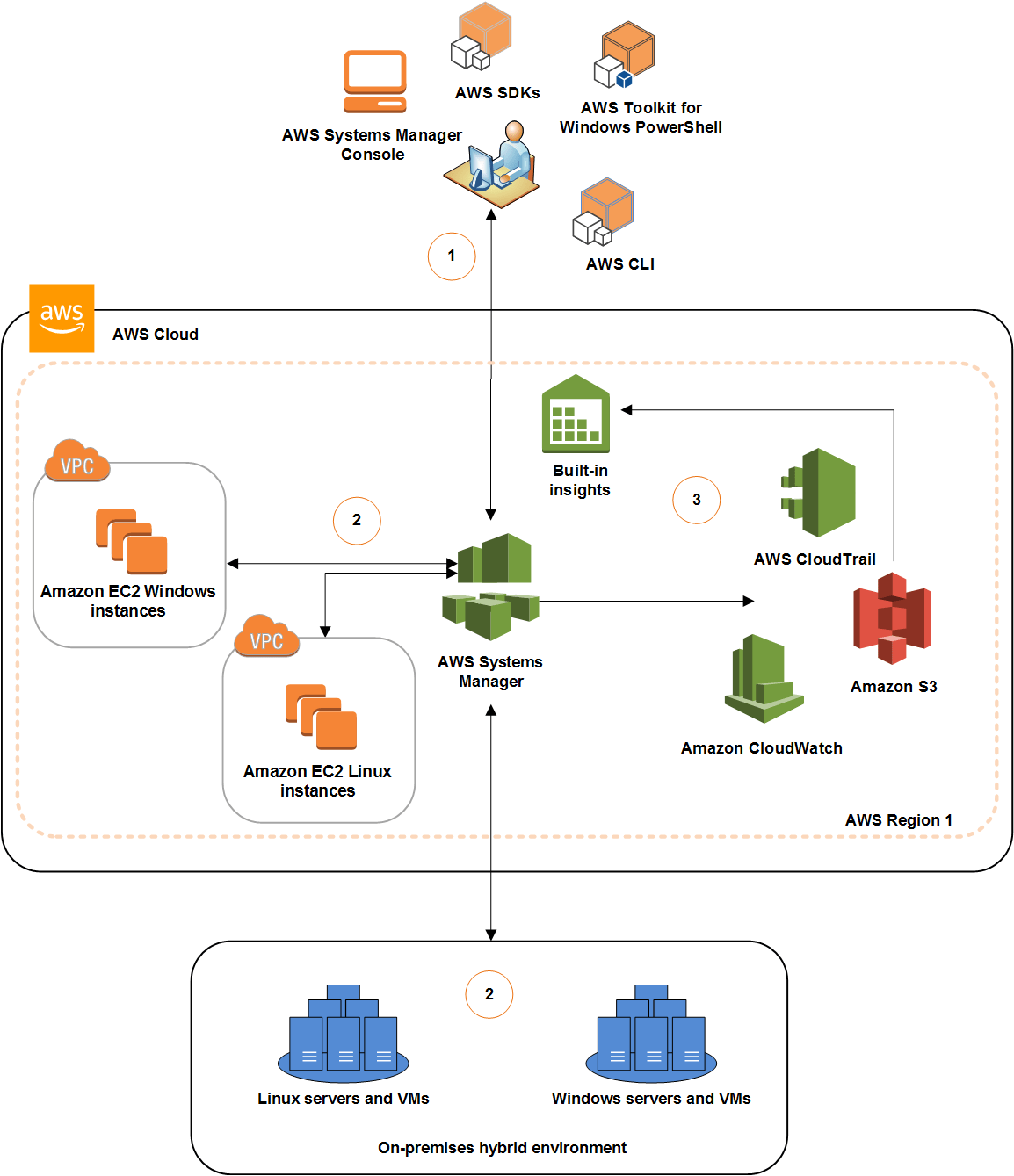
sudo apt-get update

sudo apt-get install openjdk-8-jdk -y

**DECEMBER 20, 2019**

## AWS Classroom Series – 20/Dec/2019

### Systems Manager Architecture



### How it Works

* Using AWS CLI/Console/SDK users can navigate to Systems Manager.
* System Manager Performs operations of Ec2 Linux or Windows Instances.
* During this process it interacts with other AWS Services such as
  + S3
  + Cloudwatch
  + CloudTrail
* So, to give permissions for one AWS service to interact/access other AWS Services , we need to create a role.

### Systems Manager Capabilities

* [Refer Here](https://docs.aws.amazon.com/systems-manager/latest/userguide/features.html#w384aab5c35c11b5b3) (<https://docs.aws.amazon.com/systems-manager/latest/userguide/features.html#w384aab5c35c11b5b3>)

### Run-Command

* Run-Command Executes the SSM Documents and the execution is done by SSM agent

### SSM-Document

* Reusable set of actions are shared as SSM-Documents
* SSM-Documents can be created by the user using JSON formats
* For Syntax [Refer Here](https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-doc-syntax.html) (<https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-doc-syntax.html>)
* For Actions [Refer Here](https://docs.aws.amazon.com/systems-manager/latest/userguide/automation-actions.html) (<https://docs.aws.amazon.com/systems-manager/latest/userguide/automation-actions.html>)
* SSM Documents are stored in Shared Section

### Patch Manager

* Used for Updating Windows & Linux OS Patches
* System Manager Agent can execute Patches

### Exercise:

* Download Execute Shell Script SSM Document
* add one step extra to manual steps which displays environment variables

action:

set

**DECEMBER 23, 2019**

## AWS Classroom Series – 23/Dec/2019

### Other Capabilities of Systems Manger

* Patch Manger:
  + We create Patch Baselines
  + Schedules for applying patches are created
  + Works for both Linux and windows
  + Patching can be completely done from cli.
  + [Refer Here](https://docs.aws.amazon.com/systems-manager/latest/userguide/patch-manager-cli-commands.html) (<https://docs.aws.amazon.com/systems-manager/latest/userguide/patch-manager-cli-commands.html>)
* Session Manager:
  + Manages connectivity to Ec2 instances without impacting security group rules.
* State Manager:
  + Deploy the configurations/applications using the state manager playbooks (yaml).

### Backup Service

* AWS Backup is a managed Service to perform Backups.
* Backup can be created for storages such as:
  + EBS
  + EFS
  + RDS
  + Dynamo DB
  + Storage Gateway
* Workflow
  + Create a Backup Vault
  + Create a Backup Plan. Mention number of days the backup has to be retained, transfer to cold storage and time to create backup

# Cloud Formation

**DECEMBER 24, 2019**

## AWS Classroom Notes – Cloudformation – 24/Dec/2019

### AWS Batch

* Managed Service to Perform Batch Operations
* To use create a docker image or use existing docker images
* AWS helps in calling docker container with the command.
* This container should execute the Batch job

### JSON

* It is Data Representation Format
* Data is organized as name value pairs.
* Syntax is

name: value

value => number/text/boolean/list/object

* Text name value. Text is represented by single or double quotes

"name": "QT"

* Number name value

"age": 10

* boolean value

"isonline": True

* list

"courses": ["AWS", "DevOps", "Azure"]

* Object: Multiple name value pairs

"address": {

"Flat no": "208-B",

"Building": "Nilgiri",

"LandMark": "Metro Station"

}

### YAML

* It is Data Representation Format
* Data is organized as name value pairs.
* Syntax is

name: value

value => number/text/boolean/list/object

* Text name value. Text is represented by single or double quotes

"name": "QT"

* Number name value

"age": 10

* boolean value

"isonline": yes

* list

"courses":

- AWS

- AZURE

- DevOps

* Object: Multiple name value pairs

"address":

"Flat no": "208-B"

"Building": "Nilgiri"

"LandMark": "Metro Station"

### Example:

1. Write a JSON and YAML Representing Electronics Store:

{

"Name": "Bajaj",

"Address": {

"Flat no": "208-B",

"Building": "Nilgiri",

"LandMark": "Metro Station"

},

"Categories": {

"Consumer": {

"Samsung UHD 49": {

"price": 89327

}

}

}

}

YAML

---

Name: Bajaj

Address:

Flat no: 208-B

Building: Nilgiri

LandMark: Metro Station

Categories:

Consumer:

Samsung UHD 49:

price: 89327

1. Write a Json Representing Bank Accounts:

JSON

{

"Name": "Khaja",

"PAN": "klskdlfjs",

"AADHAR": "xxxx-xxxx-xxxx-xxxx",

"Savings": [

{

"Accountno": 3982749832,

"Balance": 8932473892743982732989234

}

],

"Current": [],

"FD": [],

"PPF": []

}

YAML

---

Name: Khaja

PAN: klskdlfjs

AADHAR: xxxx-xxxx-xxxx-xxxx

Savings:

- Accountno: 3982749832

Balance: 8.932473892743983e+24

Current: []

FD: []

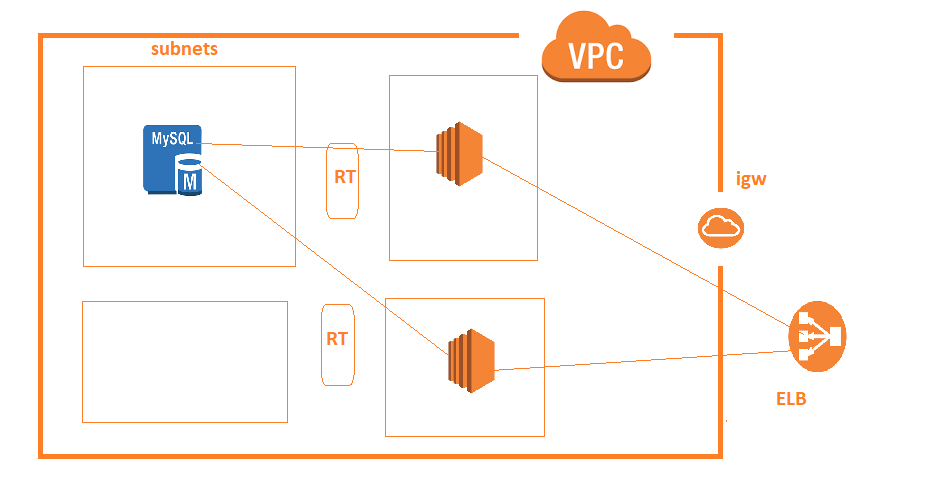
PPF: []

**DECEMBER 25, 2019**

## AWS Classroom Notes – Cloudformation – 25/Dec/2019

### Cloud Formation Template

* CF Template gives easier way to create aws application/infrastructure creational/update templates.
* It allows us to define parameters
* Template can be shared with others to create the application/infra in any account.
* CF Templates can be called from CI/CD Engines like Jenkins
* CF Template can be written for any service to create any resource.
* CF Templates can be written in
  + JSON
  + YAML
* Example Scenario: Create the following in AWS



* Evaluate the Scenario:
  + Console: Creation is simple but repetition can be difficult
  + CLI: Scripting might require some prior knowledge, but changes will be difficult to handle in repetition
  + CF Template: Create is time consuming but repetition is effective and easy.

### Cloud Formation Basics

* Template: JSON/YAML file which has resources, parameters etc
* Resource: Anything which can be created in AWS
* Parameter: Input from user
* Output: result of execution of Template
* Stack: Stack is required for execution of Template.

### Cloud Formation Syntax

* [Refer Here](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-reference.html)(<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-reference.html>)  for official docs
* [Refer Here](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-anatomy.html) (<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-anatomy.html>) for template structure
* [Refer Here](https://directdevops.blog/2019/01/30/aws-cloudformation-introuction/)(<https://directdevops.blog/2019/01/30/aws-cloudformation-introuction/>)  for directdevops.blog reference.
* JSON Syntax

{

"AWSTemplateFormatVersion" : "version date",

"Description" : "JSON string",

"Metadata" : {

template metadata

},

"Parameters" : {

set of parameters

},

"Mappings" : {

set of mappings

},

"Conditions" : {

set of conditions

},

"Transform" : {

set of transforms

},

"Resources" : {

set of resources

},

"Outputs" : {

set of outputs

}

}

* Sample Minimal Template

{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "learning cf",

"Resources": {

}

}

### Sample AWS CF Template for Creating S3 bucket

* Manually create a S3 bucket to understand the approach.
* Create a directory and create a json file in this with following content

{

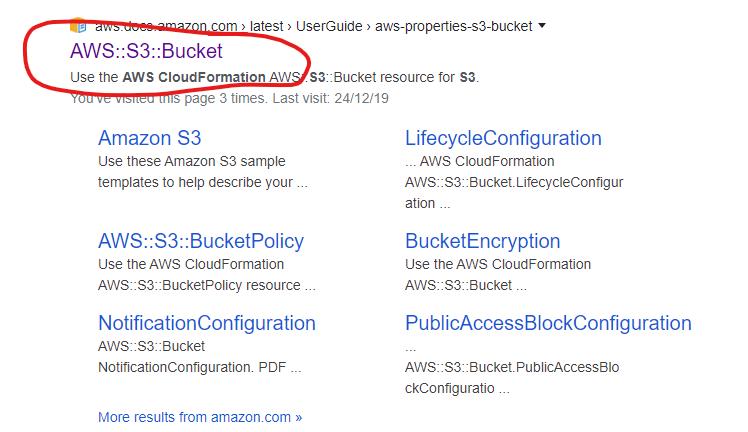
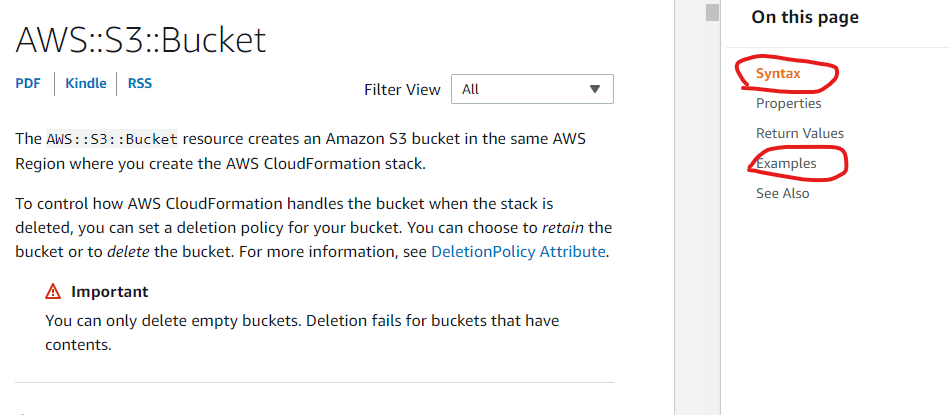
"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "learning cf",

"Resources": {

}

}

* Now since we are supposed to create a s3 resource, google aws cloudformation s3 bucket 
* Now click and open the aws docs which has the resource structure/syntax defined. 
* The following content will be enough to create s3 bucket

{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "learning cf",

"Resources": {

"mys3": {

"Type" : "AWS::S3::Bucket",

"Properties": {

"BucketName": "qt-s3-learning-forcf.com"

}

}

}

}

* Now navigate to CLoudformation Service and Create stack by uploading the above template.

### Dev Environment Setup

* From [here](https://directdevops.blog/2019/01/30/aws-cloudformation-introuction/)(<https://directdevops.blog/2019/01/30/aws-cloudformation-introuction/>)  Refer Setting up your machine for AWS CloudFormation section

**DECEMBER 26, 2019**

## AWS Classroom Series – Cloudformation – 26/Dec/2019

### Getting Started with Cloud Formation

* Creating a VPC

Create a new directory with some json file

* Now start the template

{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "trying to create sample arch",

"Resources": {

}

}

* Now lets add the basic resource syntax

{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "trying to create sample arch",

"Resources": {

"myVpc" : {

"Type": "",

"Description": ""

Properties: {

}

}

}

}

* Now open the cf resource for VPC [from here](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-vpc.html#aws-resource-ec2-vpc-syntax) and update the above JSON

{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "trying to create sample arch",

"Resources": {

"myVPC": {

"Description": "This is VPC",

"Type": "AWS::EC2::VPC",

"Properties": {

"CidrBlock" : "10.100.0.0/16",

"EnableDnsHostnames": true,

"Tags": [

{

"Key": "Name",

"Value": "From CF"

}

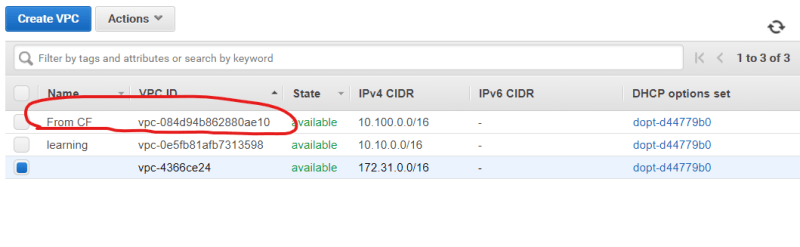
]

}

}

}

}

* Now create a CF stack by using the above JSON and check the vpc page after success 
* Create a subnet. In the properties section of subnet VPC id has to be passed. In Cloud formation whenever a resource is created it will also give return values, to use VPC id we would leverage "Reference". To use Ref

{"Ref": "<name of the resource>"}

* Name of the resource for VPC is "myVPC" and that would be used as shown below

{"Ref": "myVPC"}

* The whole file after adding one subnet appears as shown below

{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "trying to create sample arch",

"Resources": {

"myVPC": {

"Description": "This is VPC",

"Type": "AWS::EC2::VPC",

"Properties": {

"CidrBlock" : "10.100.0.0/16",

"EnableDnsHostnames": true,

"Tags": [

{

"Key": "Name",

"Value": "From CF"

}

]

}

},

"subnet1": {

"Description": "first subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2a",

"CidrBlock" : "10.100.0.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet1"

}

],

"VpcId": { "Ref": "myVPC" }

}

}

}

}

* Add three more subnets and update the stack with the following template

{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "trying to create sample arch",

"Resources": {

"myVPC": {

"Description": "This is VPC",

"Type": "AWS::EC2::VPC",

"Properties": {

"CidrBlock" : "10.100.0.0/16",

"EnableDnsHostnames": true,

"Tags": [

{

"Key": "Name",

"Value": "From CF"

}

]

}

},

"subnet1": {

"Description": "first subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2a",

"CidrBlock" : "10.100.0.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet1"

}

],

"VpcId": { "Ref": "myVPC" }

}

},

"subnet2": {

"Description": "second subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2b",

"CidrBlock" : "10.100.1.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet2"

}

],

"VpcId": { "Ref": "myVPC" }

}

},

"subnet3": {

"Description": "third subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2c",

"CidrBlock" : "10.100.2.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet3"

}

],

"VpcId": { "Ref": "myVPC" }

}

},

"subnet4": {

"Description": "fourth subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2d",

"CidrBlock" : "10.100.3.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet4"

}

],

"VpcId": { "Ref": "myVPC" }

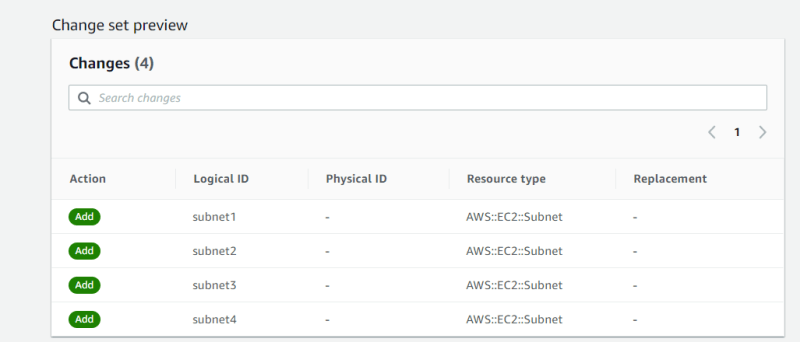
}

}

}

}

* If you preview the change-set



**DECEMBER 28, 2019**

## AWS Classroom Series – 28/Dec/2019

### Adding other resources

### Internet Gateway

* As of now our template looks like

{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "trying to create sample arch",

"Resources": {

"myVPC": {

"Description": "This is VPC",

"Type": "AWS::EC2::VPC",

"Properties": {

"CidrBlock" : "10.100.0.0/16",

"EnableDnsHostnames": true,

"Tags": [

{

"Key": "Name",

"Value": "From CF"

}

]

}

},

"subnet1": {

"Description": "first subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2a",

"CidrBlock" : "10.100.0.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet1"

}

],

"VpcId": { "Ref": "myVPC" }

}

},

"subnet2": {

"Description": "first subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2b",

"CidrBlock" : "10.100.1.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet2"

}

],

"VpcId": { "Ref": "myVPC" }

}

},

"subnet3": {

"Description": "first subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2c",

"CidrBlock" : "10.100.2.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet3"

}

],

"VpcId": { "Ref": "myVPC" }

}

},

"subnet4": {

"Description": "first subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2a",

"CidrBlock" : "10.100.3.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet4"

}

],

"VpcId": { "Ref": "myVPC" }

}

}

}

}

* To this template, lets add Internet Gateway, the template is as shown below

{

"AWSTemplateFormatVersion": "2010-09-09",

"Description": "trying to create sample arch",

"Resources": {

"myVPC": {

"Description": "This is VPC",

"Type": "AWS::EC2::VPC",

"Properties": {

"CidrBlock": "10.100.0.0/16",

"EnableDnsHostnames": true,

"Tags": [

{

"Key": "Name",

"Value": "From CF"

}

]

}

},

"subnet1": {

"Description": "first subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2a",

"CidrBlock": "10.100.0.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet1"

}

],

"VpcId": {

"Ref": "myVPC"

}

}

},

"subnet2": {

"Description": "first subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2b",

"CidrBlock": "10.100.1.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet2"

}

],

"VpcId": {

"Ref": "myVPC"

}

}

},

"subnet3": {

"Description": "first subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2c",

"CidrBlock": "10.100.2.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet3"

}

],

"VpcId": {

"Ref": "myVPC"

}

}

},

"subnet4": {

"Description": "first subnet",

"Type": "AWS::EC2::Subnet",

"Properties": {

"AvailabilityZone": "us-west-2a",

"CidrBlock": "10.100.3.0/24",

"Tags": [

{

"Key": "Name",

"Value": "Subnet4"

}

],

"VpcId": {

"Ref": "myVPC"

}

}

},

"learningigw": {

"Type": "AWS::EC2::InternetGateway",

"Properties": {

"Tags": [

{

"Key": "Name",

"Value": "From CF"

}

]

}

},

"AttachGateway": {

"Type": "AWS::EC2::VPCGatewayAttachment",

"Properties": {

"VpcId": {

"Ref": "myVPC"

},

"InternetGatewayId": {

"Ref": "learningigw"

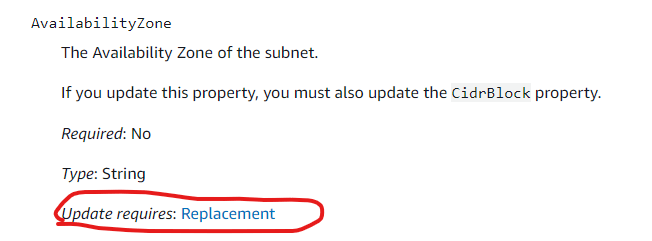
}

}

}

}

}

* Now update the stack.
* Now lets try to change some fields, subnet1 is in us-west-2a lets change that to us-west-2b. Before we do this we need to understand impact. Impact is documented by section Update requires, which has two values
  + replacement: recreate the resource (probably downtime)
  + No interruption: no impact ( zero downtime) 

# Developer Concepts

**DECEMBER 31, 2019**

## AWS Classroom Series – 31/Dec/2019- Developer

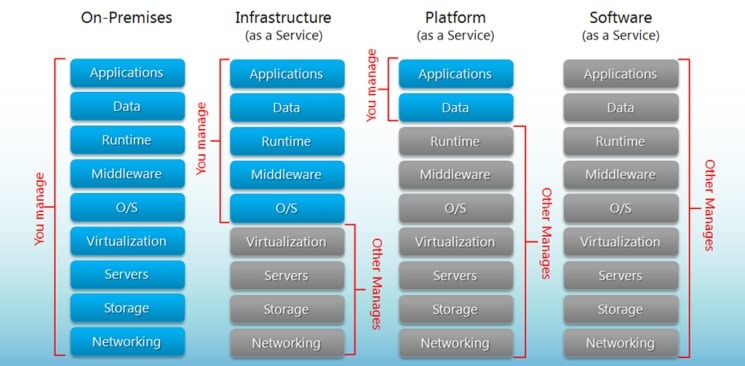
### AWS Development

* General Applications:
  + Run anywhere
  + Examples:
    - Book my show
  + Tools:
    - Logging and Monitoring
  + Services:
    - Elastic Beanstalk
    - API Gateway
* AWS Applications:
  + Run on AWS
  + Examples:
    - Amazon Prime Video
    - DropBox (Was a aws specific application until 2018)
  + Tools:
    - AWS SDK (Boto3)
  + Services:
    - AWS Lambda (Server less)

### Cloud Terms

* Service: Anything AWS is offering
* Resource: Anything User Creates using Services
* Region and Availability Zones: Where Amazon hosts datacenters
* High Availability:

### IAAS vs PAAS vs SAAS



### Elastic BeanStalk

* Provides Runtimes

**JANUARY 2, 2020**

## AWS Classroom Series – Developer- 2/Jan/2020

### Application Categories

* Web Applications: Applications which serve Web Pages and interact over http(s)
* Web API: Application which serves REST API and interacts over http(s)
* Batch: Background jobs which run at a scheduled time
* Desktop Applications: Applications which require installation on Client.

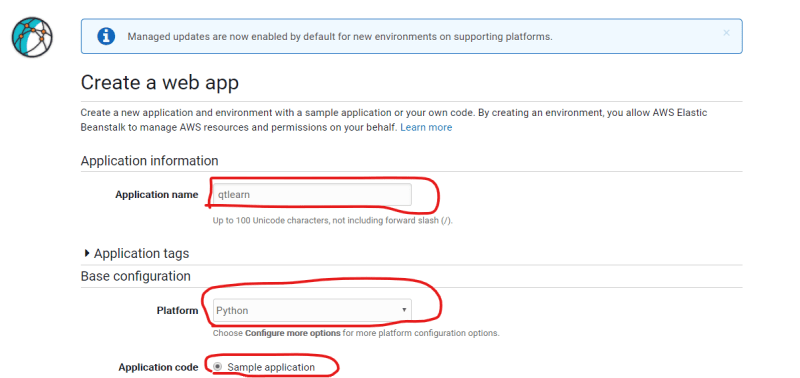
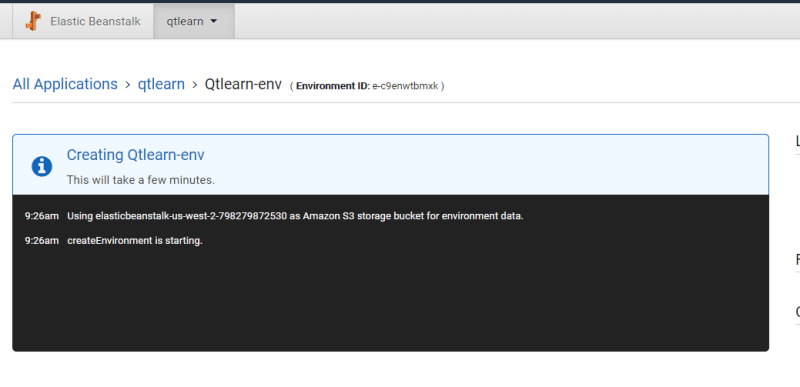
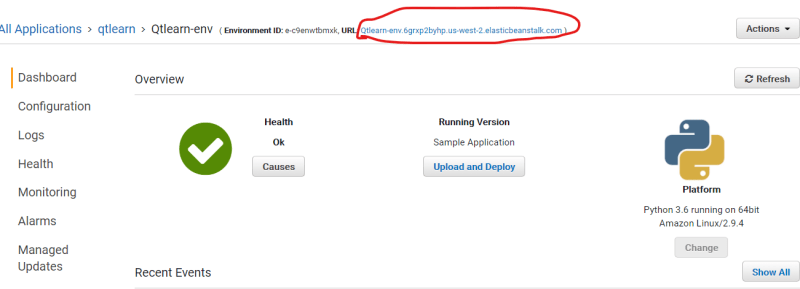
### Elastic Beanstalk

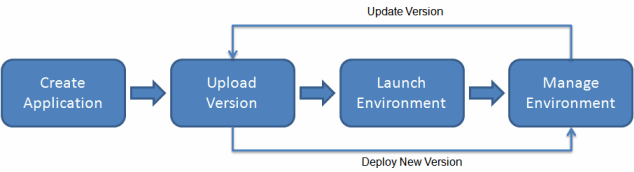
* PAAS Platform of AWS, which allows users to create Web Applications or Batch Job (Worker)

### Terms

* Platform: Techonology used to Develop Application. AWS supports the following Technology Platforms
  + Java
  + .net
  + .net core
  + Python
  + Ruby
  + Php
  + Nodejs
  + Go
  + Docker
  + Custom Platforms can be created using Packer.
* Application Version: Version of the application running
* Environment:

### Deploy Application using Elastic Beanstalk

* Navigate to Compute => Elastic Beanstalk
* Click on Get Started
* Enter your application details 
* Click on Create Application 
* Click on the Generated URL and you should see the sample app running 
* Now navigate to Services => EC2 and you should see an EC2 instance running with ENV name.
* Elastic Beanstalk is free and billing is applicable to what Beanstalk creates (Ec2, S3, EBS, EIP)



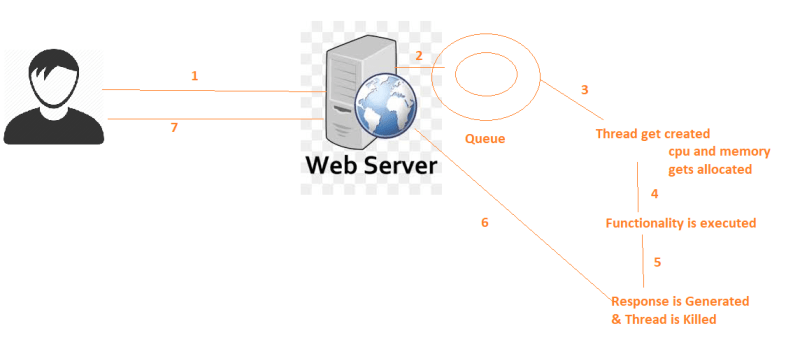
* Note: Terminate Environments and Delete Applications

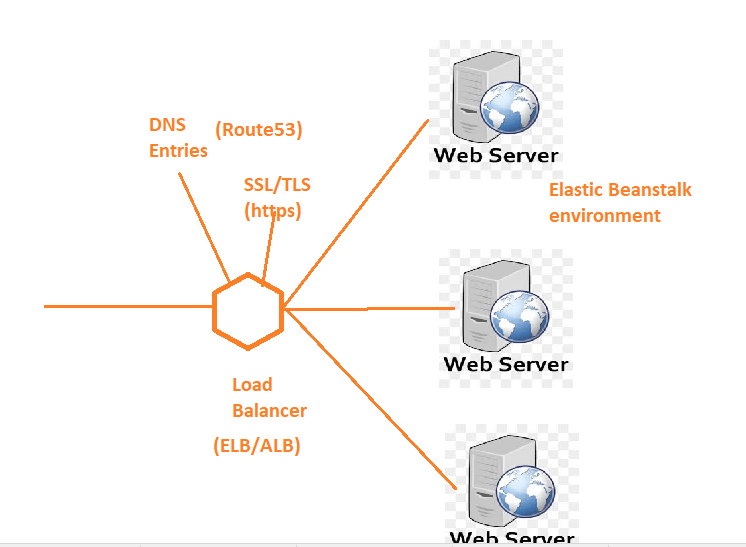
**JANUARY 3, 2020**

## AWS Classroom Series – 03/Jan/2020 Elastic Beanstalk

### Web Application Consideration

* Web Application Request Workflow



* When user requests increase, the need for cpu and memory increases. Possible solutions are
  + Vertical Scaling: Increasing H/W capacity of the server
  + Horizontal Scaling: Increasing Servers
* Since we are hosting web applications on Elastic BeanStalk, As the requests increase we should be able to scale and set the load balancer.
* Configure Route53 to point your domain name to Load balancer/Elastic Beanstalk
* Also SSL/TLS Certificates are to be configured. 

### Making Web App Ready for Scaling.

* Persistent Data: Always store/retrieve any information from Databases/Other Storages.
* Sessions: When any information is processed based out of client, Ensure Stick Sessions are enabled on Load balancer. Try to store even session information in other storage/db
* Use Cache Servers for Quick informations.

### Tools For Development

* [Eclipse](https://aws.amazon.com/eclipse/)
* [Visual Studio](https://docs.aws.amazon.com/toolkit-for-visual-studio/latest/user-guide/welcome.html)
* [PyCharm](https://aws.amazon.com/pycharm/)
* [IntelliJ](https://aws.amazon.com/intellij/)

**JANUARY 4, 2020**

## AWS Classroom Series – 04/Jan/2020

### Elastic Beanstalk Developer Environments

* Install toolkits
* Create an IAM User with necessary permissions and configure the profile

aws configure

# Enter Access key, Secret Key, Region

* Profiles can be configured from UI (AWS Toolkit)
* For Coding
  + Create Elastic Beanstalk Environment
  + Redeploy changes to existing environment.

### For non-IDE

* Source Bundle:
  + Zip the Source Code
* EB cli:
  + [Installation](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-install.html) (<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-install.html>)

### Exercise:

* Create a sample web application and deploy that application to elastic beanstalk using IDE tools.
* Make Changes and redeploy

**JANUARY 5, 2020**

## AWS Classroom Series – 05/Jan/2020

### Serverless Technology

* AWS Lambda is Implementation of Serverless by AWS.
* Is also called as Functions as a Service

### Backend as a Service

* [Refer Here](https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/)(<https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/>)  for BAAS
* [Refer Here](https://blog.back4app.com/2019/09/25/baas-open-source/)(<https://blog.back4app.com/2019/09/25/baas-open-source/>)  for open source BAAS technologies

### Function as a Service

* Small functionalites can be exposed as a Service.
* [Refer Here](https://www.cloudflare.com/learning/serverless/glossary/function-as-a-service-faas/)(<https://www.cloudflare.com/learning/serverless/glossary/function-as-a-service-faas/>)  for function as a service.

### Serverless

* Application Deployments requires to create Server first and then do the deployment.
* In Serverless, directly deploy the code. Whenever code is supposed to be executed, Servers will be created automatically by the vendor.
* Charges will be applied to users only when their code is executed.
* Amazon implemented Serverless functionality using AWS Lambda

**JANUARY 7, 2020**

## AWS Classroom Series – 07/Jan/2020

### Creating a Simple Lambda Function in Python

* Navigate to AWS Lambda and create a simple hello world from scratch with python 3.8
* Create the test function (configure test event)
* Run the test

### Creating a Simple Lambda Function in JavaScript

* Navigate to AWS Lambda and create a simple hello world from scratch with node
* Create the test function (configure test event)
* Run the test

**JANUARY 8, 2020**

## AWS Classroom Series – 8/Jan/2020

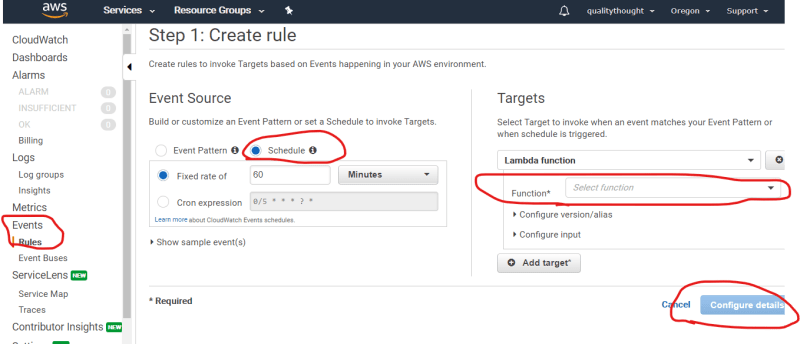
### AWS Lambda Pricing

* Charged for
  + Number of requests
  + Execution Time
* [Refer Here](https://aws.amazon.com/lambda/pricing/) (<https://aws.amazon.com/lambda/pricing/>)

### AWS Lambda Application

* An AWS Lambda application is a combination of Lambda functions, event sources, and other resources that work together to perform tasks.

### How to trigger lambda function on schedule

* Create a Lambda function
* Cloud Watch Event => Schedule 

### Developer Guide

* [Refer Here](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html) (<https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>)
* [Refer Here for Boto3](https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html) (<https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html>)

**JANUARY 9, 2020**

## AWS Classroom Series – 09/Jan/2020

### Cloud9

* Cloud9 is an IDE
* Internally an AWS EC2 machine is created with all the developer needs
* Environments created can be shared with other users for parallel development.

### Code Commit

* Code Commit is used as a Git Repo

**JANUARY 10, 2020**

## AWS Classroom Series – 10/Jan/2020

### Code Commit

* A Git Repository from AWS
* Can be authenticated for an IAM user using following protocols
  + https
  + git (ssh)

### Configuring SSH authentication to Code Commit Repository on linux machine

* Create an ubuntu ec2 instance and execute the following commands

sudo apt-get update

sudo apt-get install git -y

* Create an AWS Code Commit Repository
* Now Configure the git for linux machines [from here](https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-ssh-unixes.html)
* Create and configure key in linux

ssh-keygen

# enter path as /home/ubuntu/.ssh/codecommit\_rsa

* Create a file with name config in ~/.ssh

touch config

* Navigate to IAM user with codecommit permissions and upload ssh key (/home/ubuntu/.ssh/codecommit\_rsa.pub)
* add the following content to config

Host git-codecommit.\*.amazonaws.com

User <YOUR SSH KEY ID>

IdentityFile ~/.ssh/codecommit\_rsa

* Execute the following commands

ssh git-codecommit.us-west-2.amazonaws.com

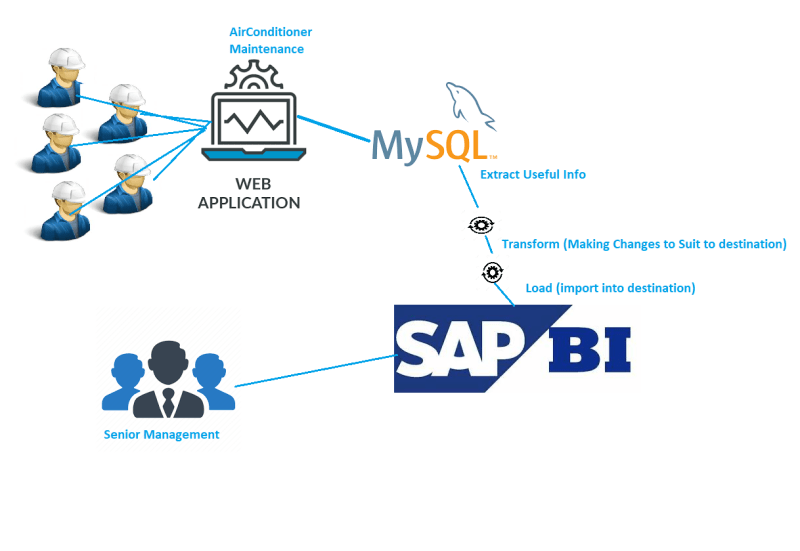
ssh -v git-codecommit.us-west-2.amazonaws.com

* Now navigate to Code commit and get the clone url for ssh and

git clone <ssh url from code commit>

### Extract Transform And Load (ETL)

* Consider the following Scenario



* When the data from one system / format has to be imported into other system, in many cases it is not straight forward so
  + Extract the needed information
  + Transform i.e. making necessary changes to suit the needs
  + Load i.e. import into destination
* Whenever you are working with data, to support ETL on AWS, a Service called AWS Glue is provided.

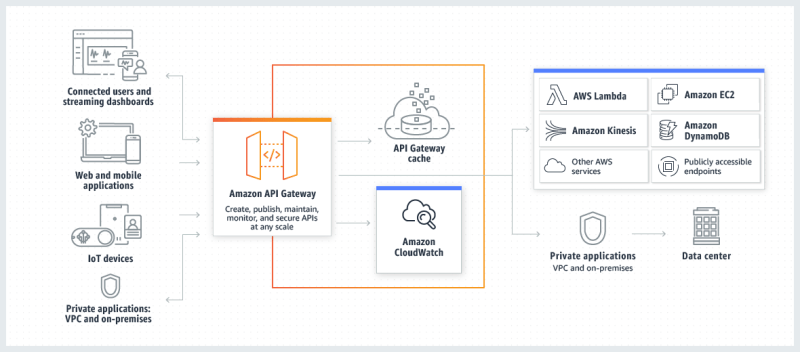
### API Gateway

* API Gateway is an interface on top of the apis developed
* API Gateways can provide
  + Logging
  + Authentication
  + Traffic Control
  + Tracing
  + Operations

**JANUARY 20, 2020**

## AWS Classroom Series 20/Jan/2020

### API Gateway



* Services (Http(s)) running on Lambda or Ec2 or Elastic Bean stalk or any other AWS Service can be redirected using AWS API Gateway.
* Log records and tracing records will be stored in Cloud Watch
* Cache can be added to Gateway.

### Web Application Firewall

* To Stop Common attacks and vulnerabilities to the services.
* AWS has 3 product lines
  + WAF
  + Firewall Manager
  + Shield

**JANUARY 21, 2020**

## AWS Classroom Series – 21/Jan/2020

### Key Management Service

* KMS is used to store keys for encryption and decryption
* KMS supports
  + symmetric
  + asymetric
* KMS is used widely
  + to encrypt data at rest
  + to encrypt the sensitive contents of the applications/configurations
* Services widely used in association with KMS
  + S3
  + Storesync
* KMS can be used with Language SDKS.

### AWS Glue

* Helps in Extracting information from Datasources.
* Data can be in the forms shown below for crawlers to work with
  + text files => regular expressions to parse text
  + xml files => XPATH
  + json files => JSON Path
  + CSV => delimeters
* [Refer Here](https://docs.aws.amazon.com/glue/latest/dg/aws-glue-programming-python-samples-legislators.html) (<https://docs.aws.amazon.com/glue/latest/dg/aws-glue-programming-python-samples-legislators.html>)
* [Refer Here](https://boto3.amazonaws.com/v1/documentation/api/latest/guide/kms-example-encrypt-decrypt-file.html) (<https://boto3.amazonaws.com/v1/documentation/api/latest/guide/kms-example-encrypt-decrypt-file.html>)

# Migrations

**JANUARY 23, 2020**

## AWS Classroom Series – 23/Jan/2020

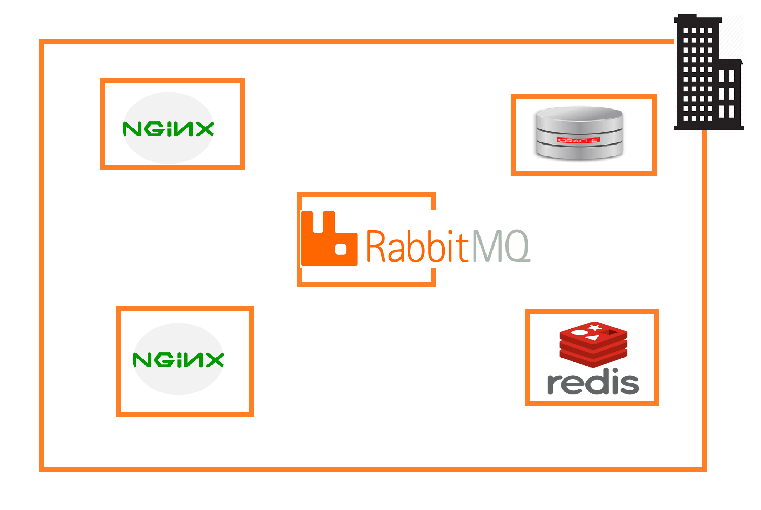
### AWS Free Tier Account Creation

* [Refer Here](https://directdevops.blog/2020/01/22/aws-free-tier-account-creation-video/) (<https://directdevops.blog/2020/01/22/aws-free-tier-account-creation-video/>)

### Migrations

* Goal: Ensuring Existing infrastructure on-premise to work on cloud.

### Scenario

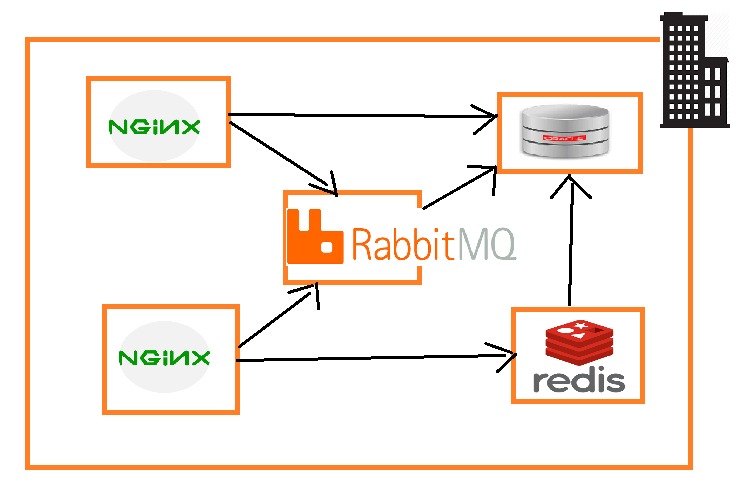
* An Application has following infrastructure 

### Possibilities

* Infrastructure: Physical/Virtual
* Virtualization Platform: VM Ware ESXI/Hyper-V

### Approach for Solution

1. Create a Depenendency Graph to decide on order of migration



1. Migration Approches for Operating Systems
   * Physical to Virtual Migration (P2V)
   * Virtual to Virtual Migration (V2V)
2. Database Migrations
   * Check for possibility to migrate to Service (RDS)
3. Possibility of PAAS/Docker kind of Platforms
4. Look at support matrix.

### Migrations to be done by all of us

* Physical to Amazon EC2
* VMs on Hyper-V to EC2
* VMs on VmWare ESXI to EC2
* Database Migration

**JANUARY 24, 2020**

## AWS Classroom Series – 24/Jan/2020

### Design Approach

* [6 R’s of Migration](https://directdevops.blog/2019/10/10/6-rs-of-migration/) (<https://directdevops.blog/2019/10/10/6-rs-of-migration/>)
* ReHost
* RePlatform
* RePurchase
* Refactor/Re-architect
* Retain
* Retire

### Data Migration Strategy

### Aws Services For Migration

* Import/Export Image
* AWS Server Migration Service
* AWS Migration Hub
* AWS Application Discovery
* Database Migration Service

### Other useful Services for Migration

* AWS S3
* AWS Snowball
* StorSimple
* CloudWatch
* VPC and VPN

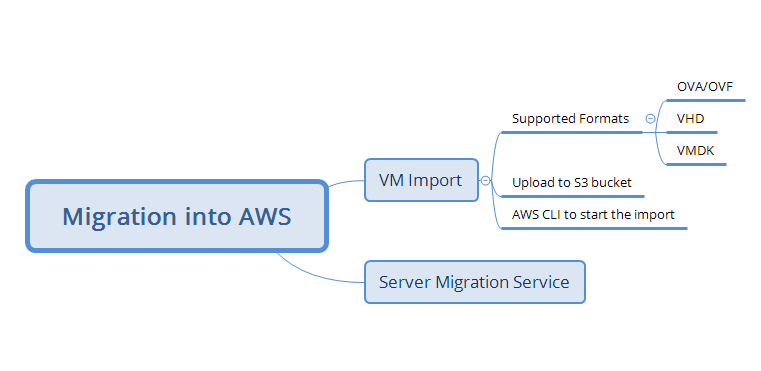
### Highlevel overview of Migration

1. Import/Export Image:
   * One VM in VMWare
   * Export that to suitable format
   * Upload this format to AWS S3
   * Import to AWS AMI
2. Server Migration Service: (V2V)
   * Download an VM image from AWS
   * Create a VM with this image on your hypervisor (VMWare ESXI/Hyper-V)
   * Use this VM to Select & uplod other vms on Hypervisors

**JANUARY 25, 2020**

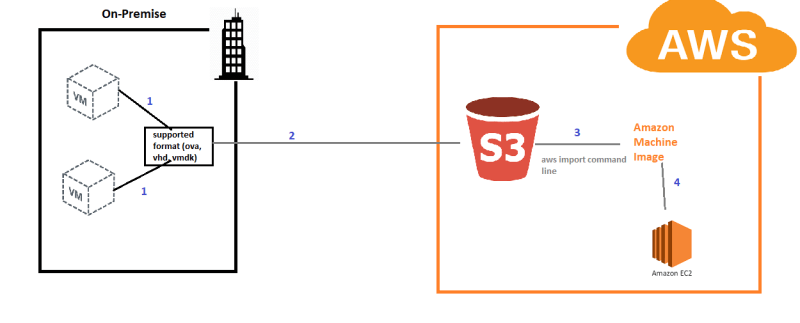
## AWS Classroom Series – 25/Jan/2020

### VM Import into AWS



* [Refer](https://docs.aws.amazon.com/vm-import/latest/userguide/vmimport-image-import.html) (<https://docs.aws.amazon.com/vm-import/latest/userguide/vmimport-image-import.html>)

### Steps



1. Convert the vms into supported formats
   * OVA/OVF (Open Virtualization Appliance)
   * VHD (Hyper-V)
   * VMDK (VMWare)
2. Upload the exported format to the s3 bucket
3. Execute aws cli to create the AMI from the exported format
4. Create an ec2 instance with the AMI created.

### Lab Environment for VM Import

* Download the OVA files from [here](https://cloud-images.ubuntu.com/bionic/current/) (<https://cloud-images.ubuntu.com/bionic/current/>)
* Create a S3 bucket and upload the ova file
* Install AWS CLI
  + Windows:
    - Download 32-bit/64-bit msi [from here](https://docs.aws.amazon.com/cli/latest/userguide/install-windows.html#install-msi-on-windows) (<https://docs.aws.amazon.com/cli/latest/userguide/install-windows.html#install-msi-on-windows>)
    - Install the AWS CLI from the downloaded MSI.
    - Open Powershell and execute aws --version
    - Create an IAM user [Refer Here](https://serverless-stack.com/chapters/create-an-iam-user.html) (<https://serverless-stack.com/chapters/create-an-iam-user.html>)
    - Configure IAM user to aws cli

aws configure

# pass accesskey , secret key and region (us-west-2)

aws s3 ls

# check whether the s3 bucket which you have created is present or not

**JANUARY 27, 2020**

## AWS Classroom Series – 27/Jan/2020

### Import the Image uploaded to S3 bucket

* [Refer Here](https://docs.aws.amazon.com/vm-import/latest/userguide/vmimport-image-import.html) (<https://docs.aws.amazon.com/vm-import/latest/userguide/vmimport-image-import.html>)  for official docs
* Create a json file called as contaienr.json with the following content

[

{

"Description": "Ubuntu 18 imported",

"Format": "ova",

"UserBucket": {

"S3Bucket": "qtvmimportdemo",

"S3Key": "bionic-server-cloudimg-amd64.ova"

}

}]

* Execute the following command to import the ova into ami

aws ec2 import-image --description "Ubuntu 18 imported" --disk-containers "file://C:\Users\QT\Downloads\containers.json"

#### output #########

{

"Status": "active",

"Description": "Ubuntu 18 imported",

"SnapshotDetails": [

{

"UserBucket": {

"S3Bucket": "qtvmimportdemo",

"S3Key": "bionic-server-cloudimg-amd64.ova"

},

"DiskImageSize": 0.0,

"Format": "OVA"

}

],

"Progress": "2",

"StatusMessage": "pending",

"ImportTaskId": "import-ami-09cef8ab9a75fa7fa"

}

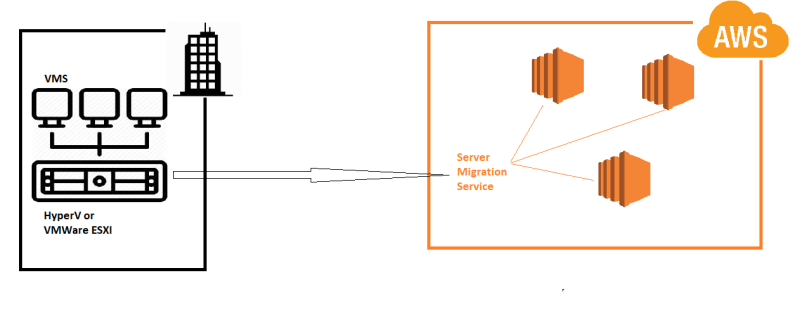
* Now monitor this import task by executing

aws ec2 describe-import-image-tasks --import-task-ids import-ami-09cef8ab9a75fa7fa

### Other Import/Export Options

* Importing an EC2 instance from VM
* Exporting an EC2 instance to VM
* Exporting an EC2 AMI to VM image.

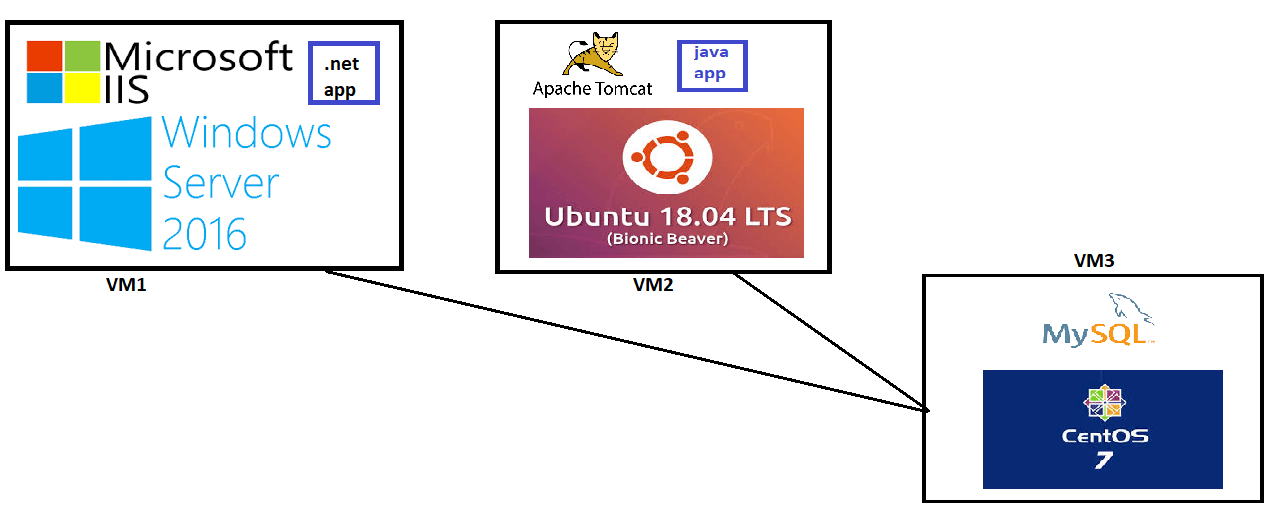
### Server Migration Service



**JANUARY 28, 2020**

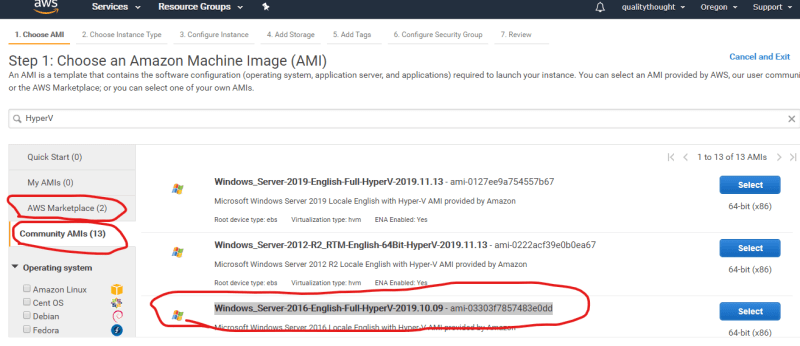
## AWS Classroom Series – 28/Jan/2020

### Scenario For Migration

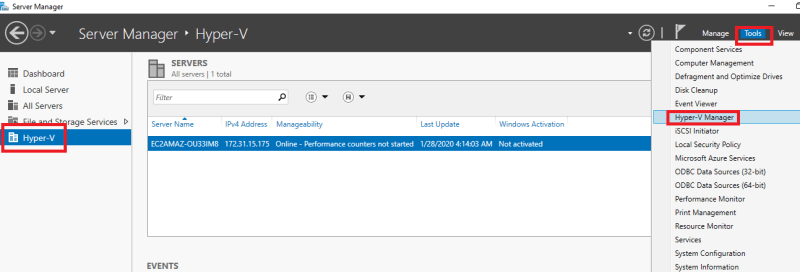


### Step-1: Create a Hyper-V supported VM in AWS

* Create an Ec2 Instance with Windows Server 2016 and HyperV image and atleast a 4 core instance type



* Login into the machine and open Server Manager



* Connect to the VM using RDP and Perform the following steps
* Install Chocolatey

# Open Powershell as Admin and execute

Set-ExecutionPolicy Bypass -Scope Process -Force; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))

* Restart VM
* Connect to RDP again and install chrome

choco install googlechrome -y

* Download Windows Server 2016, Ubuntu 18.04 LTS iso and Centos 7 dvd iso

**JANUARY 29, 2020**

## AWS Classroom Series – 29/Jan/2020

### Creating Network and DHCP Manager

* Reference from [here](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/nested-virtualization#set-up-internet-connectivity-for-the-guest-virtual-machine) (<https://docs.microsoft.com/en-us/azure/virtual-machines/windows/nested-virtualization#set-up-internet-connectivity-for-the-guest-virtual-machine>)

New-VMSwitch -Name "InternalNAT" -SwitchType Internal

Get-NetAdapter

# make a note of iflindex

New-NetIPAddress -IPAddress 192.168.0.1 -PrefixLength 24 -InterfaceIndex 13

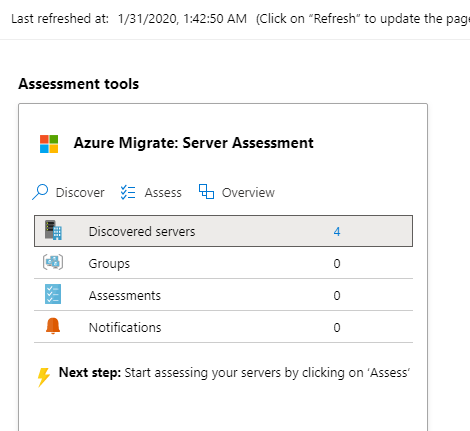
* Assign IP Addresses to the VMs using option 1 from [here](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/nested-virtualization#assign-an-ip-address-to-the-guest-virtual-machine)

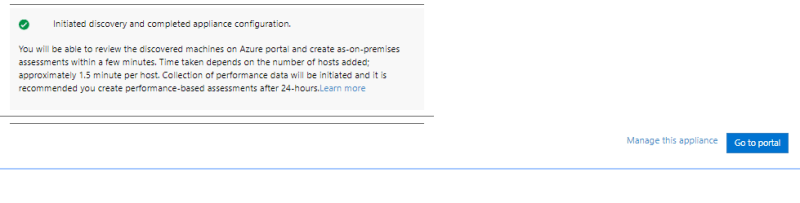
**JANUARY 31, 2020**

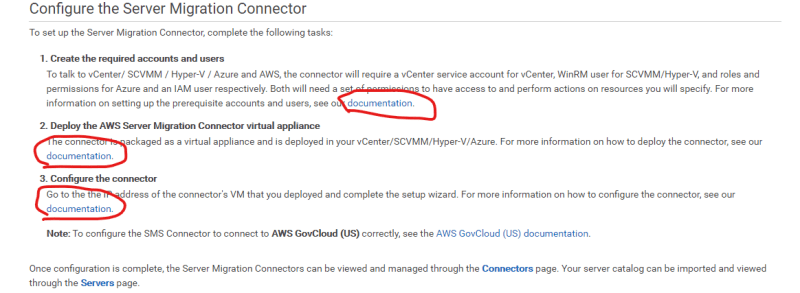
## AWS Classroom Series – 31/01/2020

### Server Migration Service

* Navigate to Server Migration Service



* Download the HyperV Connector in to the HyperV Host 
* For configuring the connector refer documentations from console



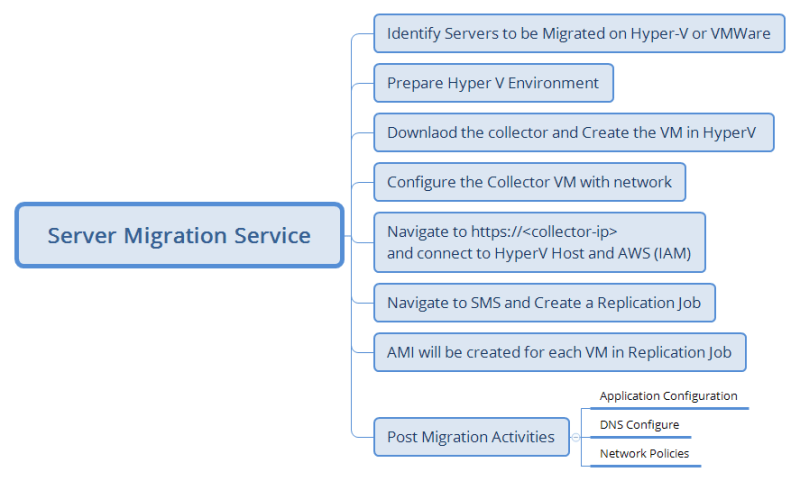
**FEBRUARY 4, 2020**

## AWS Classroom Series – 04/Feb/2020

### Replicate VMs

* After Connecter is Shown on AWS Server Migration Service, Select the VMs and create a replication job
* If the Replication job is succesful, the AMIs will be created.
* Use the AMIs to create EC2 instances in the network (VPC)

### Steps For Migration

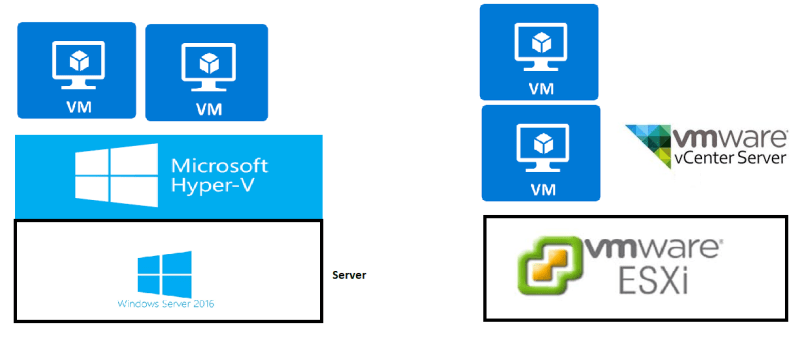


* For Hyper V Collector configuration [Refer Here](https://docs.aws.amazon.com/server-migration-service/latest/userguide/HyperV.html) (<https://docs.aws.amazon.com/server-migration-service/latest/userguide/HyperV.html>)
* Replicating VMS [Refer Here](https://docs.aws.amazon.com/server-migration-service/latest/userguide/console_workflow.html) (<https://docs.aws.amazon.com/server-migration-service/latest/userguide/console_workflow.html>)

**FEBRUARY 5, 2020**

## AWS Classroom Series – 05/Feb/2020

### VMWare ESXi vs HyperV

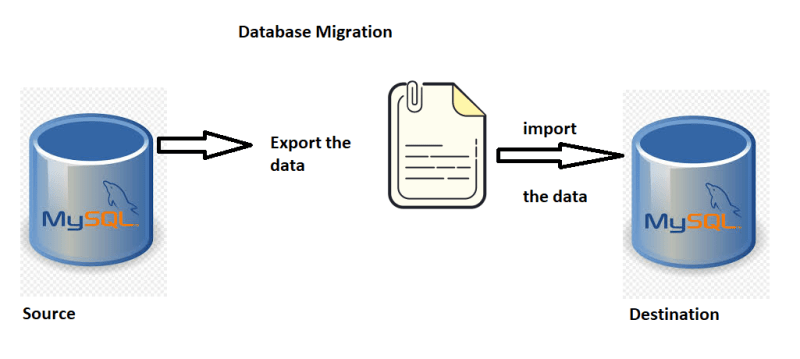
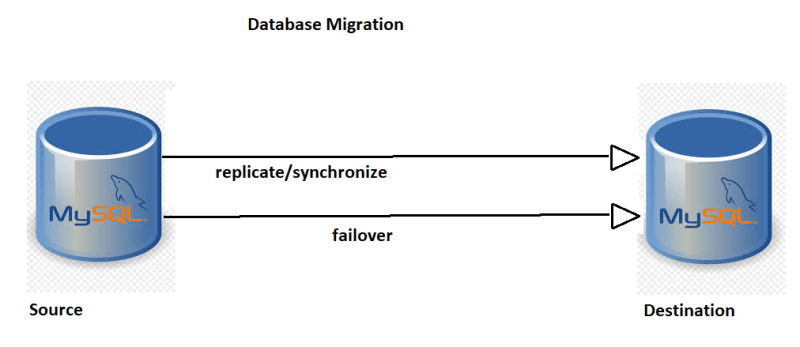
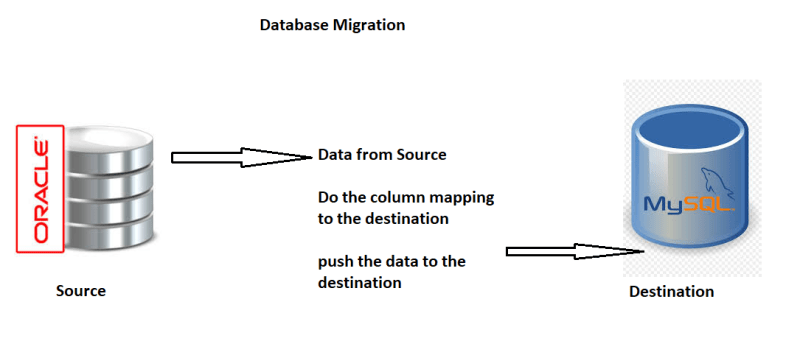


* In the Case of HyperV
  + Downloaded Connector
  + Created a VM on Hyper V
  + Ensured there is connection b/w
    - Connector and HyperV Host
    - Connector and AWS
  + Once the connector is setup
    - Import Server Catalog
    - Create Replication Job
* In the Case of VmWare
  + Download Connector
  + Create a VM from OVA file
  + Ensure there is connection b/w
    - Connector & VCenter
    - Connector & AWS
  + Once the connector is setup
    - Import Server Catalog
    - Create Replication Job

**FEBRUARY 6, 2020**

## AWS Classroom Series – 06/Feb/2020

### Database migration approaches

1. Backup and Restore:
   * Take a backup/export of source database to a file/folder
   * Move the backup to destination
   * Restore/import the backup into destination db
   * Advantages:
     + Simple
   * Disadvantage:
     + Downtime will be required if the source database is changing 
2. Replicate or Synchronize Databases:
   * Create a Replica of the source database
   * Once the replication is complete
   * Failover to the destination database
   * Advantage:
     + Less Downtime (in minutes)
   * Disadvantage:
     + DBA skills to perform
     + Replication adds performance overhead to the source 
3. Migration from different database engine 

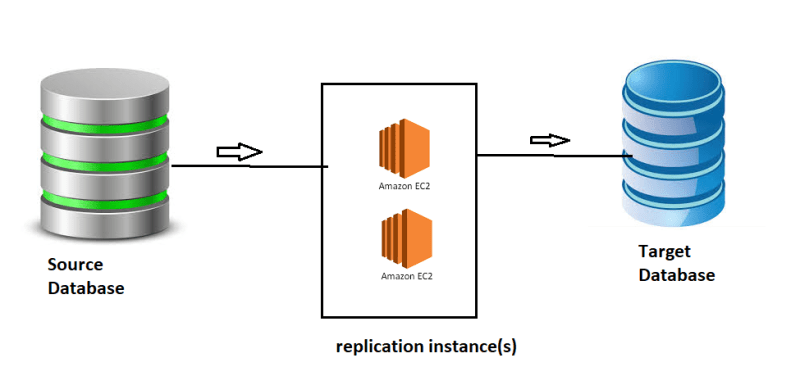
### Database Migrations in AWS

* AWS has a service DMS which is designed to migrate
  + From any source (onpremise/cloud) to any destination (onpremise/cloud)
  + from some database engines to some other databases

**FEBRUARY 7, 2020**

## AWS Classroom series – 07/Feb/2020

### Database Migration Service



### Terms

* Endpoints
  + Source Endpoint: Connection Parameters of Source Db
  + Target Endpoint: Connection Parameters of Destination Db
* Replication Instance
  + An ec2 instance managed by AWS to perform migration activities
* Replication Task
  + A task initiated to perform database migration

### Scenario for Migration

* Source:
  + HR Application which use employee database [refer here](https://dev.mysql.com/doc/employee/en/) (<https://dev.mysql.com/doc/employee/en/>)
  + Location:
    - Azure or AWS
    - If you have static ip address then create a database on that machine.
* Destination:
  + AWS RDS
* Replication Instance:
  + Size: t2.micro
  + IP => public ip

### Setup

1. Target Endpoint:
   * Create RDS instance with mysql

# AWS Systems Manager

**SEPTEMBER 19, 2019**

## AWS Systems Manager Classroom Series – 19/Sep/2019

### Managed Instances

* Instance which is in cloud or on premise with ssm agent installed and connected to AWS Systems Manger is managed instance

### Run-Command

* Command which can be executed on the managed instance

### Document

* We have used a Document called as **AWS-Runshellscript**
* Two kinds of Documents
  + Predefined Documents:
    - Create by AWS
  + Custom Documents:
    - Create by users
    - To Create Custom Documents we need to know JSON & Document Syntax. [Refer Here](https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-doc-syntax.html) (<https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-doc-syntax.html>)

### State Manager

### Use-Cases:

* Ensure SSH Access to all linux machine is production environment is denied
  + Execute stop command periodically to maintain state is stopped sudo service sshd stop
  + Ensure apache server is enabled
  + Ensure free space on your C drive is greater than 5 GB

**SEPTEMBER 20, 2019**

## AWS Systems Manager – Classroom Series 20/Sep/2019

### Systems Manager Setup

1. Create a IAM role with policy **AmazonEC2RoleforSSM** for ec2 machine to connect to Systems manager.
2. Create a IAM user with policy **AmazonSSMFullAccess**(also look into other ssm policies).
3. In the onpremise or ec2 machines ssm agent has to be installed. To install ssm agent refer
   * Windows Systems [Refer](https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-install-ssm-win.html) (<https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-install-ssm-win.html>)
   * Linux Systems [Refer](https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-manual-agent-install.html) (<https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-manual-agent-install.html>)
4. Only when you install ssm agent and have necessary IAM Permissions(roles/user), then the ec2 machine/virtual machine will be managed instance.

### Other Capabilities

* Sessions Manager: No need for bastion host. You can connect to machines in vpc with only private ip’s and execute your configuration.
* Patch Manager: From Patch manager you create or manage os patching
* Shared Resources: Build Reusable Document (like Run-AWSShellScript) & store common parameters in Parameters store.

# AWS Backup and Recovery

**SEPTEMBER 14, 2019**

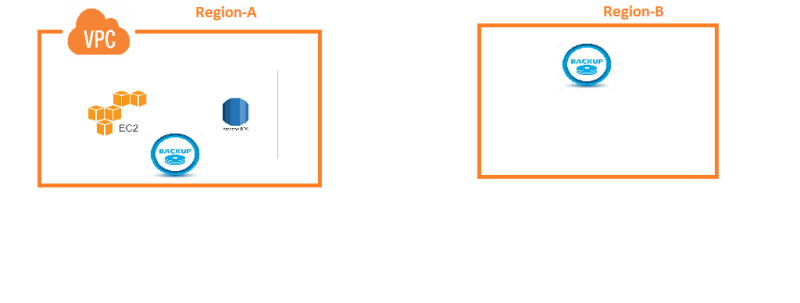
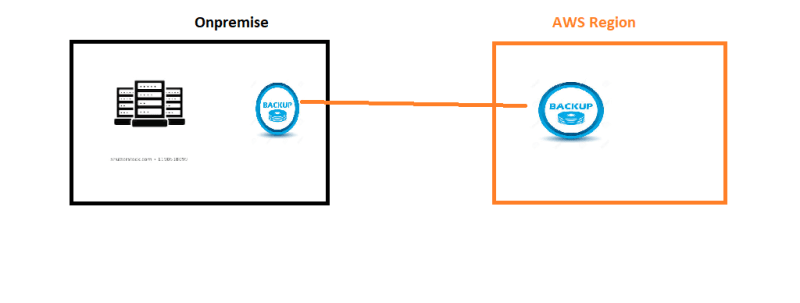
## AWS Backup & Recovery Series – 14/Sep/19

### Business Continuity and Disaster Recovery

### Types

1. Backup:
   * Purpose: Quick Recovery
   * Location: Closer to failure point
   * Frequency: Regular
   * Cost: Spend more on storage and speed
2. Archive
   * Purpose: Disaster Recovery
   * Location: Safe place far away
   * Frequency: Ocassional (once in a week/month)
   * Cost: Spend less on storage

### Usecase

1. Cloud: 
2. Hybrid: 

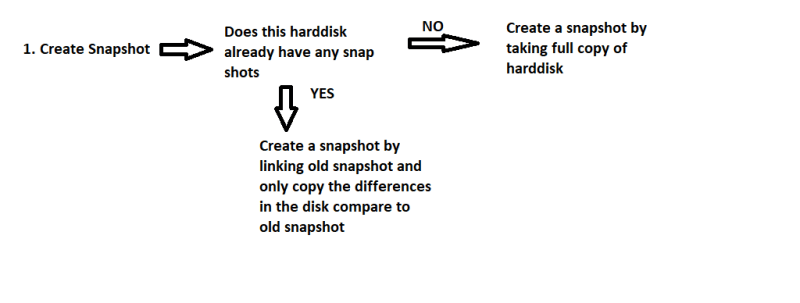
### Main Backup Sources

1. Hard disk
2. Databases/Application

### Lab

1. Create a backup of Ec2 machine. steps:
   * Create a ec2 machine
   * Backup of Hardisk (Volume) is called as Snapshot
   * From snapshot create a volume
   * Copy snapshot to other region
   * Dont forget to delete ec2 machine, snapshots and volumes (To avoid billing)

### Snapshot Way of Working



**SEPTEMBER 16, 2019**

## AWS Backup and Recovery Classroom Series – 16/Sep/2019

### AWS Backup Service Exploration

### Backup Policy

* when to take backup
* what has to be backed up
* AWS Backup service allows following bakcups
  + EBS
  + RDS
  + EFS
  + Dynamo Db
  + Storage Gateway
* Cron Schedules

### Permissions to be setup

* Need a role for AWS Backup Service

### Exercise today

1. Backup EBS now
2. Configre backup at 12:30 PM for RDS

**SEPTEMBER 17, 2019**

## AWS Backup and Recovery Classroom Series – 17/Sep/2019

### How AWS Backup Service Works

1. In AWS All the five services EBS, EFS, RDS, Dynamodb, Storage Gateway have their own mechanism for taking backups populary known as snapshots
2. AWS Backup service internally creates the same.
3. Eventhough the approach looks same, Backup Service has Backup policy which makes it usable.

### What is Backup Policy

1. What has to backed up is already decided by AWS Backup Service
2. Backup policy allows you to choose
   * When to take backup
   * How many days you want to hold backups (retention)
   * Lifecycle of Backup can be created

### Who takes the Backup in the case AWS Backup Services

* AWS Backup Service takes the backup
* So you need to give permissions.
* Permissions to AWS Service is called as IAM Role

### Where are these Backup stored?

* Backups are store in Vault

### What can be done with Backups

* Restore

### Steps

1. Create an IAM role for AWS Backup Service with necessary permissions
2. Create a vault
3. Create a backup plan
   * Select Timings
   * Select resources which you want to be backed up
4. Wait for Restore Points to be created.
5. In the cases of failure restore from restore points.

# AWS Managerial Administration

**SEPTEMBER 10, 2019**

## AWS Administration Classroom notes – 10/Sep/2019

### Major Activities

1. Monitoring AWS Resources for System Failures
2. Cost Management and Cost Optimizations
3. Backup Services
4. Disaster Recovery (BCDR)
5. Resource Specific Monitioring
6. Log Analysis and Auditing
7. Patching Servers

### Minor Activities

1. Notifications
2. Team Specific Billing

### Lab Setup

1. Execute Cloud Formation Template provided to create some servers
2. AWS Free tier account

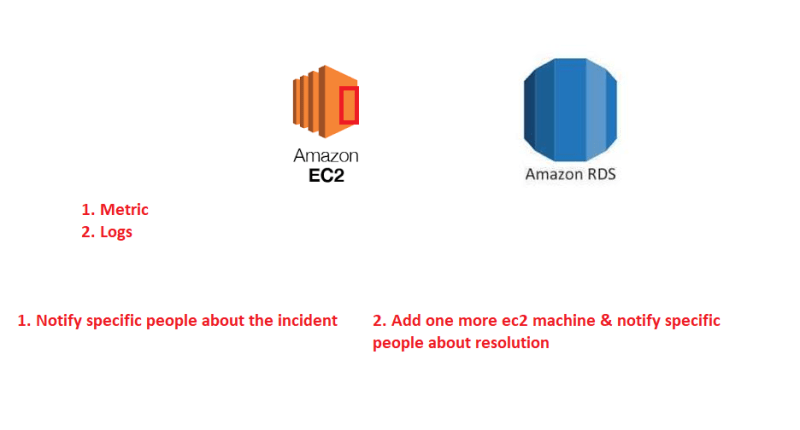
### Services

1. CloudWatch
2. CloudTrail
3. SNS
4. AWS Backup Services
5. AWS Systems Manager
6. Batch Processing
7. Billing Widgets

**SEPTEMBER 11, 2019**

## AWS Monitoring Classroom Series – 11/Sep/2019

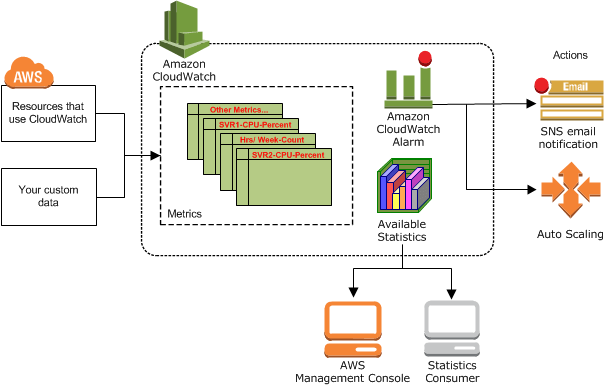
### Basics of Monitoring



### How can AWS Help ??

1. Cloudwatch
2. Simple Notification Service

### How Cloudwatch Works



### Cloud Watch Alarm States

1. Insufficient
2. Alarm
3. OK

### SNS

1. Create a topic and we create one email listener

### Lab

1. Create a Ec2 machine and when cpu utilization is > 50 % send email to your id

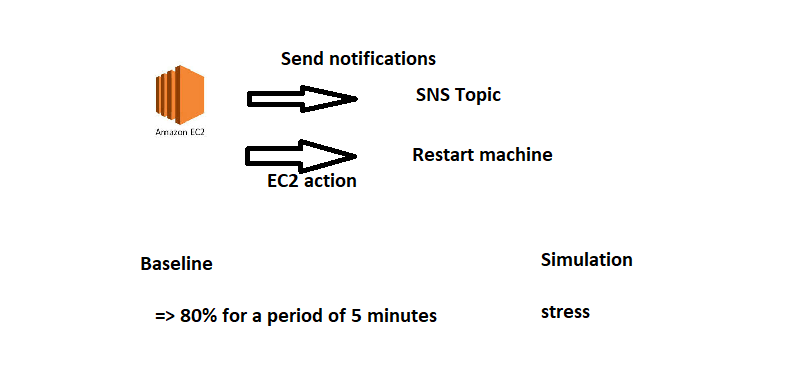
**SEPTEMBER 12, 2019**

## AWS Monitoring Classroom Series – 12/Sep/2019

### AWS Monitoring Exercise

1. Download ec2 zip file from [here](https://github.com/QT-DevOps/AWSIssues/issues/38) (<https://github.com/QT-DevOps/AWSIssues/issues/38>)
2. Unzip ec2
3. Launch Cloudformation from your aws account (us-west-2/oregon)
4. Create a stack by passing necessary parameters to create two virtual machines in two subnets

### Workflow



### References

For complete list of metrics per service refer [here](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/aws-services-cloudwatch-metrics.html) (<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/aws-services-cloudwatch-metrics.html>)

**SEPTEMBER 13, 2019**

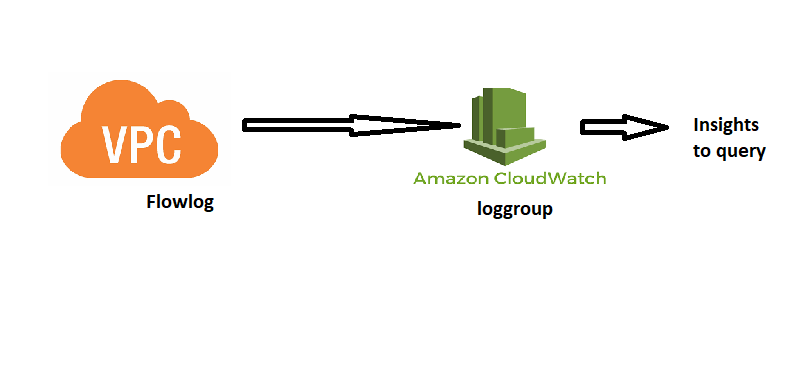
## AWS Monitoring Series – 13/Sep/2019

### AWS Log Monitoring workflow

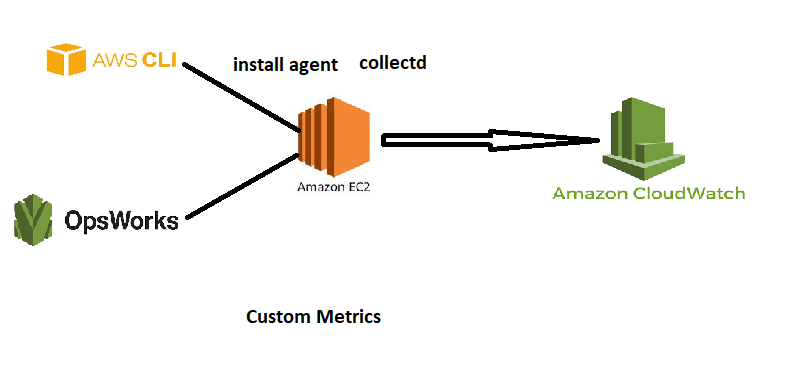
### AWS Log Basics

* Cloudwatch to store logs
* Create a log group
* In Loggroup we have logstreams
* Each logstream will have individual logs

### Simple Workflow



### Collectd Workflow



### For Querying Logs

* Insights Refer [Here](https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AnalyzingLogData.html) (<https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AnalyzingLogData.html>)
* Query Syntax Refer [Here](https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/CWL_QuerySyntax.html) (<https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/CWL_QuerySyntax.html>)

### Reference

* <https://aws.amazon.com/blogs/aws/new-cloudwatch-plugin-for-collectd/>