AWS

Contents

[IAM (Identity & Access Management) 1](#_Toc851100)

[Virtual Private Cloud (VPC) 2](#_Toc851101)

[Internet Gateway (IGW) 3](#_Toc851102)

[Route Tables (RTs) 3](#_Toc851103)

[Network Access Control List (NACL) 3](#_Toc851104)

[Subnets 4](#_Toc851105)

[Endpoints 4](#_Toc851106)

[NAT (Network Access Translation) 5](#_Toc851107)

[VPC Peering 5](#_Toc851108)

[EC2 5](#_Toc851109)

[Security Groups 5](#_Toc851110)

[Elastic Load Balancer (ELB) 6](#_Toc851111)

[Autoscaling Group (ASG) 7](#_Toc851112)

[Simple Storage Service (S3) 8](#_Toc851113)

[IP Addressing 8](#_Toc851114)

[Direct Connect & VPN 8](#_Toc851115)

[Cloudformation 9](#_Toc851116)

[Sceptre 10](#_Toc851117)

[CIDR blocks 10](#_Toc851118)

[Fundamentals of networking in AWS 10](#_Toc851119)

[RDS 10](#_Toc851120)

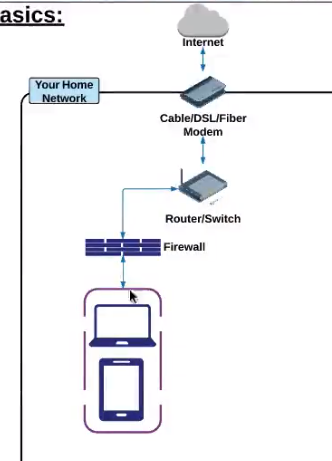
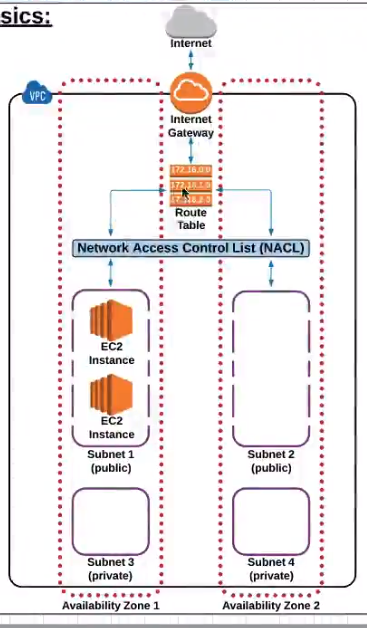
[Steps to build AWS application 11](#_Toc851121)

[Queries 11](#_Toc851122)

# IAM (Identity & Access Management)

* IAM is to manage users and their access to AWS accounts and services.
* The user created when you created AWS account is called the “root” user who has full administrative rights and access. Any new user are created with no access to any AWS services.
* It is better to have users within Groups. Users and Groups can have their own permissions (or policies).
* Permissions has a list of policies which will enable the user or group to access AWS resource.
* Roles will come in place when we want to have one AWS resource trying to connect to another AWS resource (e.g., application in EC2 connecting to S3).

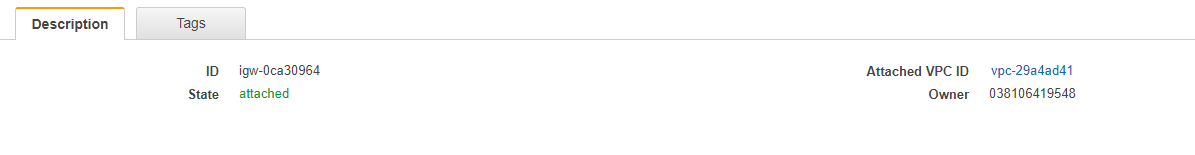
# Virtual Private Cloud (VPC)



* Private sub section of AWS that you control and can place AWS resources for which you have full control over who has access to AWS resources inside VPC. Technically, it is a logical isolated section of AWS cloud where you can launch AWS resources in virtual network whose IP address range, subnet creation, route table and network gateway configuration is controlled by you.
* AWS Infrastructure
  + Regions – grouping of AWS resources in a specific geographical location.
  + Each region consists of availability zones (geographically isolated zones within a region that house AWS resources or AWS data centers). AZ help in providing redundancy for AWS resources.

## Internet Gateway (IGW)

* + We can think of internet gateway as modem in home network.
  + Combination of software and hardware that provides your private network with a route to the world outside. Technically, IGW is a VPC component that allows communication between instances in your VPC and the Internet.
  + AWS resources within same VPC doesn’t need Internet gateway as the connection doesn’t go through internet.
  + Only 1 IGW can be attached to 1 VPC at a time. An IGW cannot be detached from a VPC while there are live/active AWS resources (EC2 or RDS) in the VPC.



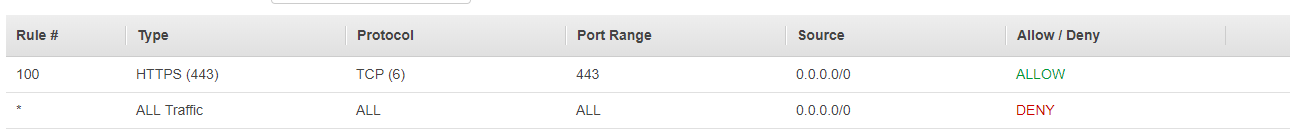
## Route Tables (RTs)

* + We can think of route tables as routers in home network.
  + Route table contains a set of rules called routes that determines where network traffic is directed.
  + Route table has connection to IGW (if you want the connection to internet) and routes to internal parts of VPC in terms of IPs.
  + We can have multiple active route tables in VPC and we can’t delete a route table if it has associated subnets.



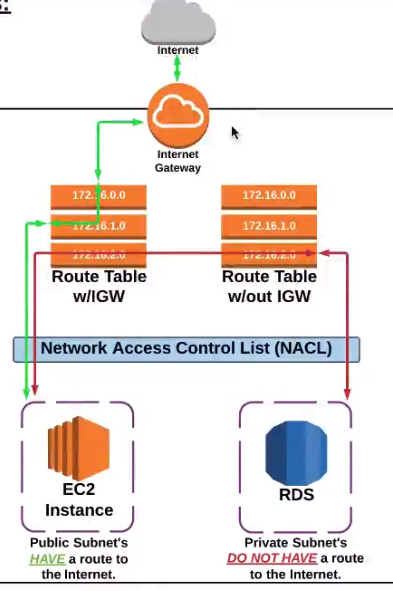
## Network Access Control List (NACL)

* + We can think of firewalls (stateless) in home network.
  + It is an optional layer of security for VPC that acts as a firewall for controlling traffic in and out of one or more subnets.
  + NACL has different inbound and outbound rules, which allows all traffic by default. Rules are evaluated by rule# from lowest to highest.
  + Any new NACL will have all inbound and outbound rules as denied by default.
  + A subnet can be associated with only one NACL.
  + An NACL allows or denies traffic from entering a subnet. Once inside the subnet, other AWS resources may have additional layer of security such as security groups.



## Subnets

* + Subnet is a sub section of network. Technically, when a VPC is created, it extends all of AZs in the region. We can create one or more subnets in each AZs. Each subnet must reside entirely within one AZ and cannot span zones.
  + Public subnet have a route to internet and private subnet have no route to internet. Private subnet is associated with route table which is not associated with IGW.
  + A subnet must be associated with route table.



* + Private Subnet
    - Read about CIDP (VPC and Subnet in AWS documentation)
  + Public Subnet

## Endpoints

* + A VPC endpoint will help us to connect your VPC to supported AWS services in private network (traffic doesn’t go to internet). Instances in your VPC doesn’t need to have public IP address. There are 2 types of endpoints.
    - Interface endpoints
      * It is a network interface that serves as an entry point for traffic destined to supported service (like API Gateway, EC2 API, Key Management service etc). You can add policies (permissions) for Interface endpoints which you want to connect to from your instances.
    - Gateway endpoints
      * It is a gateway that is a target for a specified route in your route table used for traffic destined to supported AWS services like S3 or DynamoDB.

## NAT (Network Access Translation)

* + NAT Gateways enables instances present in private subnet to connect to the internet or other AWS services, but prevent the internet from initiating a connection with those instances.
  + NAT Gateway resides in public subnet and it is recommended to create in every AZ for redundancy.
  + In this case, private subnet will be associated to route table which has CIDR block of VPC and NAT Gateway.

## VPC Peering

* + VPC Peering is a networking connection between two VPCs that enables you to route traffic between them using private IP addresses. They will communicate with each as if they are on the same network. VPC peering connection can happen between VPCs in different accounts and regions and can be used to transfer the data.

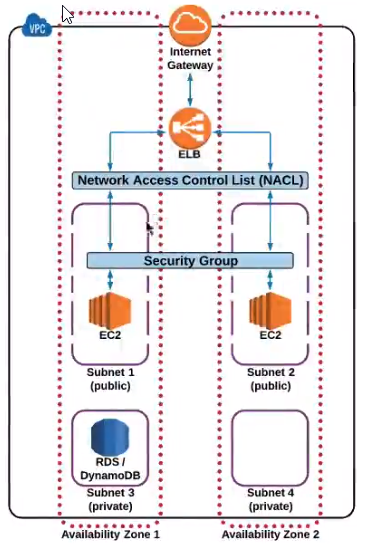
# EC2

## Instance Types

* General purpose (A, T, M)
* Compute Optimized (C)
* Memory Optimized (R, X)
* Accelerated Computing (P, G, F)
* Storage Optimized (H, I, D)

## Security Groups

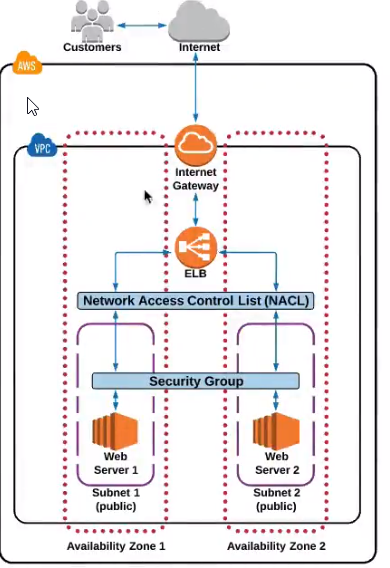
* Security groups are similar to NACL but on instance level. NACL is on subnet level. Security group acts as a virtual firewall that controls the traffic for one or more instances. When you launch an instance, you associate one or more security groups with the instance. You can modify the rules for a security group at any time, new rules are automatically applied to all instances. AWS will evaluate all the rules in all the security groups for the instances.



* For a new security group, all inbound traffic is denied and all outbound traffic is allowed. We need to add inbound rules to allow and outbound rules to restrict.
* Recommendations - Follow the principle of least privilege

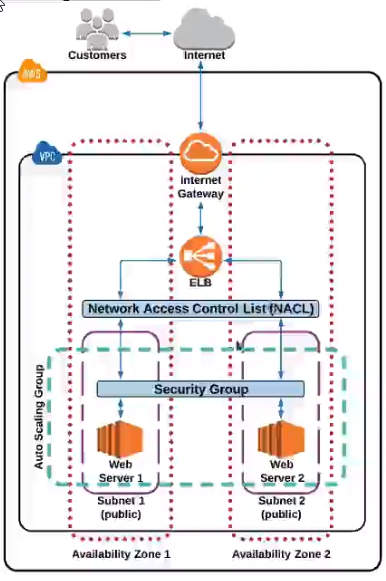
## Elastic Load Balancer (ELB)

* ELB distributes traffic among EC2 instances that are associated with it.
* Health check needs to be configured on ELB which will have ping path (isAlive), ping port, response timeout, interval, healthy threshold, unhealthy threshold.



## Autoscaling Group (ASG)

* Automates the process of adding or removing instances based on traffic demand of the application. ASG is the core component for high availability and fault tolerance in AWS architecture. ASG is a service, not a physical infrastructure which can span across subnets/availability zones. ASG and ELB work hand in hand.
* In ASG, you specify minimum, maximum number of instances, it will launch or terminate based on demand.
* ASG has 2 main components called Launch Configuration which uses EC2 template (AMI, software etc.) when AS adds an additional server to ASG and Auto Scaling Group which contains the rules and settings that govern when an EC2 server is automatically added or removed.
* Using ASG is free, we will be charged for resources that AS provisions.



# Simple Storage Service (S3)

* Bulk, highly scalable, reliable storage service.
* Data structure in S3
  + Buckets – root folder
  + Folder – any subfolder within a bucket
  + Object – files stored in a bucket
* When you create a bucket, we need to select the region where the objects within bucket will be physically stored in the region.
* Storage class – classification assigned to each object in S3. Each storage class has varying attributes like cost, object availability, durability and frequency to access. Check AWS storage class for latest updates.
  + Standard (default)
    - Good object durability and availability.
    - Expensive
  + Unknown or changing access
    - S3 will store the object in infrequent region and will move to frequent access if S3 senses that the object is accessed more.
    - S3 charges extra for monitoring and automation.
  + Reduced Redundancy Storage (RRS)
    - For non-critical, reproducible objects
    - Lesser object durability and availability than Standard.
    - Less expensive than Standard.
  + Infrequent Access
    - As name says, not frequently accessed, but must be immediately available when accessed.
    - Durability same as Standard, but availability lower than Standard.
    - Less expensive than Standard/RRS
  + Glacier
    - For long tem archival storage and may take long hours to retrieve the object.
    - Durability same as Standard
    - Cheapest of S3 storage classes
* Permissions can be set on bucket and object level.

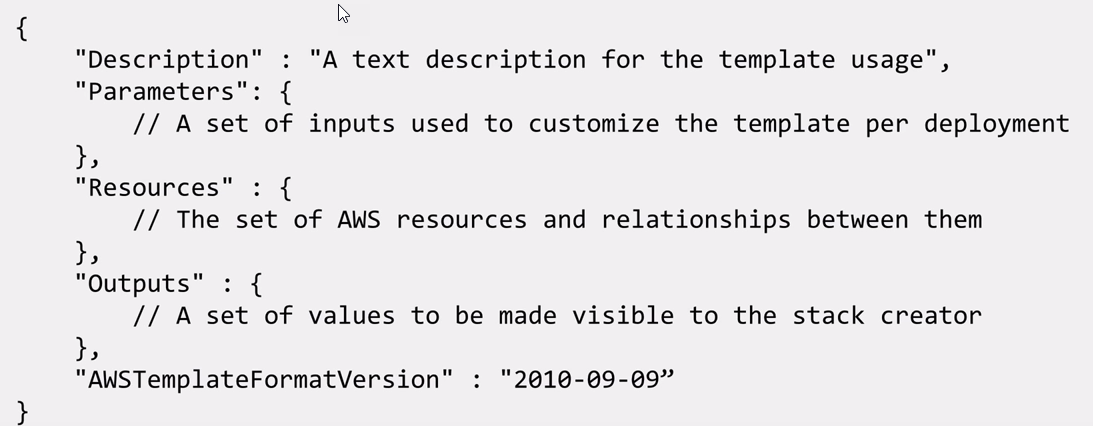
# IP Addressing

* Providing an EC2 instance with public IP address.
* By default, all EC2 instances have private IP address. EC2 instances can be launched with or without public IP address based on VPC/subnet settings. Default settings will have public IP address.
* Public IP addresses are required for the instance to communicate with the internet.

# Direct Connect & VPN

* AWS Direct connect is a cloud service solution to establish a dedicated network connection from on premise to AWS. We can establish private connectivity between data center and AWS which will increase bandwidth throughput, reduce network costs and provide more consistent network experience.
* AWS VPN service lets you establish a secure tunnel between on premise network to AWS global network. AWS VPN consists of: AWS site to site VPN and AWS Client VPN. Site to Site connects data center to your AWS VPC. AWS Client VPN enables you to connect users to AWS or on premise networks.

# Cloudformation

* Easy way to create & manage a collection of AWS resources, allows provisioning and updating of resources, allows to version control AWS infrastructure, deploy and update stacks using console, command line or API (wrapped by variety of SDKs). It’s free, but you pay for the resources you create. It’s parameterized and reusable.
* Cloudformation templates are in JSON formats.
* High level Template Structure
* 
* Resources
  + Resources help us to create AWS resources of type mentioned [here](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html). If you want to create a resource outside of this type, you can define a custom resource. Custom resources require on property ServiceToken which specified where AWS Cloudformation has to send requests to. We can use ARN (Amazon resource names) to identify AWS resources (e.g. for lambda it will be arn:aws:lambda:us-east-1:123456789012:function:ProcessKinesisRecords).
* Template References
  + Resource Type reference (like AWS::EC2:data-type-name)
  + Resource Policy reference
  + Intrinsic reference

# Sceptre

* Sceptre is the tool from Cloudreach to manage Cloudformation. Cloudformation becomes difficult to manage when there are multiple environments, stacks, regions and accounts.
* Before Sceptre, we had environment related configurations inside Cloudformation templates like for ELB, we had security group, VPC and subnet references in Mappings section for every environment. Whenever there was any change, these values need to be changed in Cloudformation template. Another way would have been configuring these references in Jenkins.
* Sceptre enables environment based configuration for Cloudformation.
* With Sceptre, we can parameterize the references. There are 2 folders – config and template. Template folder carries the Cloudformation templates and config folder carries environment related configuration (yaml format). We can define environment specific values in within config folder and common values (like non prod VPCid, subnet id) in environment yaml outside config folder.
* Sceptre has also stack name resolvers through which we can refer to other stack properties.
* In Jenkins, we can run sceptre command as shell script to inject into Jenkins.

# CIDR blocks

# Fundamentals of networking in AWS

* All EC2 instances run within VPC as they need IP address (private or public). EC2 instances run within subnet which are aligned with availability zone.
* Steps to create internet connected VPC
  + Choosing address range
    - AWS uses CIDR notation e.g. 172.31.0.0/16 (private address space) and it is fixed for lifetime of VPC.
  + Create subnets aligned with availability zone
    - Divide the address range for diff subnets (3 subnets if 3 AZ) – e.g. 172.31.0.0/24 for Subnet A, 172.31.1.0/24 for subnet B, 172.31.2.0/24 for subnet C
  + Getting route to internet
  + Authorizing traffic to/from VPC

# RDS

* Aurora maintains 6 copies of database across 3 AZ for high availability. Aurora has Cluster endpoint and Cluster Read Endpoint (it manages to route the multi reads to other instance).
* Aurora Postgres
  + Benefits
    - No operational expertise required as AWS manages underlying infrastructure.
    - We can instantly scale up or out.
    - Devops friendly – POC friendly to provision using AWS CLI or Console and secure it.
    - Cost reduction compared to Oracle licenses.
    - Failover can happen in less than 30s.
  + Shortcomings
    - No cross region replication.
    - No support for multi master – not good for write intensive.
    - No support for table partitioning.

# Steps to build AWS application

* We need to create policies and roles needed for the project (specific to environment). Roles will be like Instance role (for EC2), Service role (for lambda) that will consists of policies based on AWS services it needs.
* We also need to have KMS key for the project that will be used to encrypt and decrypt the instances.
* Next is to take a base AMI and run a Cloudformation template to create EC2 instance out of it. Install the necessary application or middleware components on the instance using Ansible playbooks.
* If possible, scan the instance for compliance and generate a custom encrypted (EBS root and other volumes are encrypted if KMS key present) AMI out of the deployed instance.

# Queries

* How to enable IAM for organization level?