AWS - 1

AWS Fundamentals – Part II

EBS Volumes, Load Balancing and Auto Scaling Groups

1

What's an EBS Volume?

- An EC2 machine loses its root volume (main drive) when it is manually terminated.
- Unexpected terminations might happen from time to time (AWS would email you)
- Sometimes, you need a way to store your instance data somewhere
- An EBS (Elastic Block Store) Volume is a network drive you can attach to your instances while they run
- It allows your instances to persist data

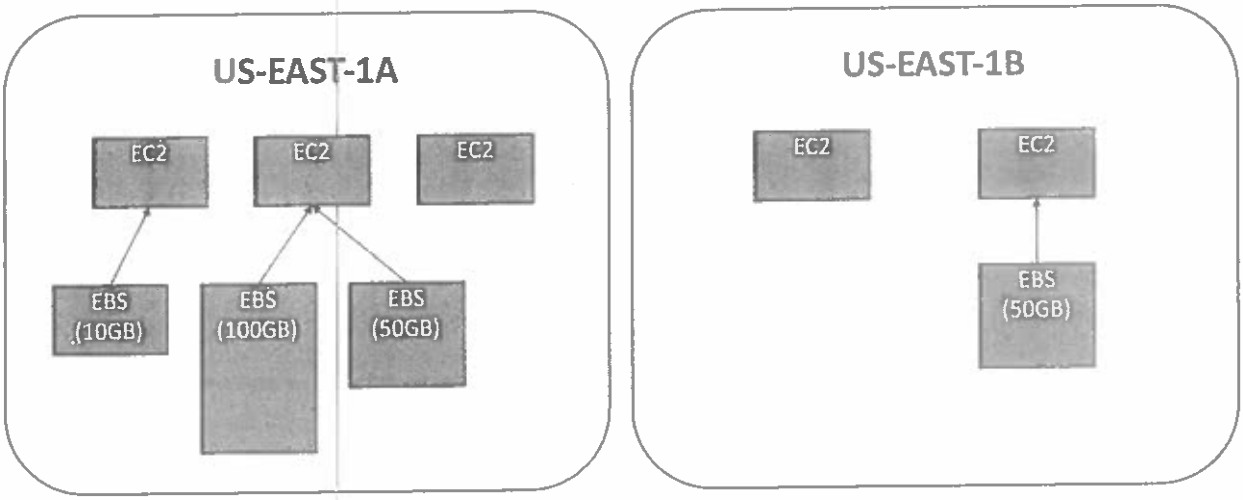
2

EBS Volume

- It's a network drive (i.e. not a physical drive)
  - It uses the network to communicate the instance, which means there might be a bit of latency
  - It can be detached from an EC2 instance and attached to another one quickly
- It's locked to an Availability Zone (AZ)
  - An EBS Volume in us-east-1a cannot be attached to us-east-1b
  - To move a volume across, you first need to snapshot it
- Have a provisioned capacity (size in GBs, and IOPS)
  - You get billed for all the provisioned capacity
  - You can increase the capacity of the drive over time

3

EBS Volume Example



4

EBS Volume Types

- EBS Volumes come in 4 types
  - GP2 (SSD): General purpose SSD volume that balances price and performance for a wide variety of workloads
  - IO1 (SSD): Highest-performance SSD volume for mission-critical low-latency or high- throughput workloads
  - ST1 (HDD): Low cost HDD volume designed for frequently accessed, throughput- intensive workloads
  - SC1 (HDD): Lowest cost HDD volume designed for less frequently accessed workloads
- EBS Volumes are characterized in Size | Throughput | IOPS
- When in doubt always consult the AWS documentation – it's good!

5

EBS Volume Resizing

- You can resize the EBS volumes
- You can only increase the EBS volumes:
  - Size (any volume type)
  - IOPS (only in IO1)
- After resizing an EBS volume, you need to repartition your drive

6

EBS Snapshots

- EBS Volumes can be backed up using "snapshots"
- Snapshots only take the actual space of the blocks on the volume
- If you snapshot a 100GB drive that only has 5 GB of data, then your EBS snapshot will only be 5GB
- Snapshots are used for:
  - Backups: ensuring you can save your data in case of catastrophe
  - Volume migration:
    - Resizing a volume down
    - Changing the volume type
    - Encrypt a volume

7

EBS Encryption

- When you create an encrypted EBS volume, you get the following:
  - Data at rest is encrypted inside the volume
  - All the data in flight moving between the instance and the volume is encrypted
  - All snapshots are encrypted
  - All volumes created from the snapshot
- Encryption and decryption are handled transparently (you have nothing to do)
- Encryption has a minimal impact on latency
- EBS Encryption leverages keys from KMS (AES-256)
- Copying an unencrypted snapshot allows encryption

8

EBS vs Instance Store

- Some instance do not come with Root EBS volumes
- Instead, they come with "Instance Store".
- Instance store is physically attached to the machine
- Pros:
  - Better I/O performance
- Cons:
  - On termination, the instance store is lost
  - You can't resize the instance store
  - Backups must be operated by the user
- Overall, EBS-backed instances should fit most applications workloads

9

EBS Brain Dump

- EBS can be attached to only one instance at a time
- EBS are locked at the AZ level
- Migrating an EBS volume across AZ means first backing it up (snapshot), then recreating it in the other AZ
- EBS backups use IO and you shouldn't run them while your application is handling a lot of traffic
- Root EBS Volumes of instances get terminated by default if the EC2 instance gets terminated. (you can disable that)

10

Scalability & High Availability

- Scalability means that an application / system can handle greater loads by adapting.

- There are two kinds of scalability:
  - Vertical Scalability
  - Horizontal Scalability (= elasticity)

- Scalability is linked but different to High Availability

11

Vertical Scalability

- Vertically scalability means increasing the size of the instance
- For example, your application runs on a t2.micro
- Scaling that application vertically means running it on a t2.large
- Vertical scalability is very common for non distributed systems, such as a database.
- RDS, ElastiCache are services that can scale vertically.
- There's usually a limit to how much you can vertically scale (hardware limit)

12

Horizontal Scalability

- Horizontal Scalability means increasing the number of instances / systems for your application
- Horizontal scaling implies distributed systems.
- This is very common for web applications / modern applications
- It's easy to horizontally scale such as Amazon EC2

13

High Availability

- High Availability usually goes hand in hand with horizontal scaling
- High availability means running your application / system in at least 2 data centers (== Availability Zones)
- The goal of high availability is to survive a data center loss
- The high availability can be passive (for RDS Multi AZ for example)
- The high availability can be active (for horizontal scaling)

14

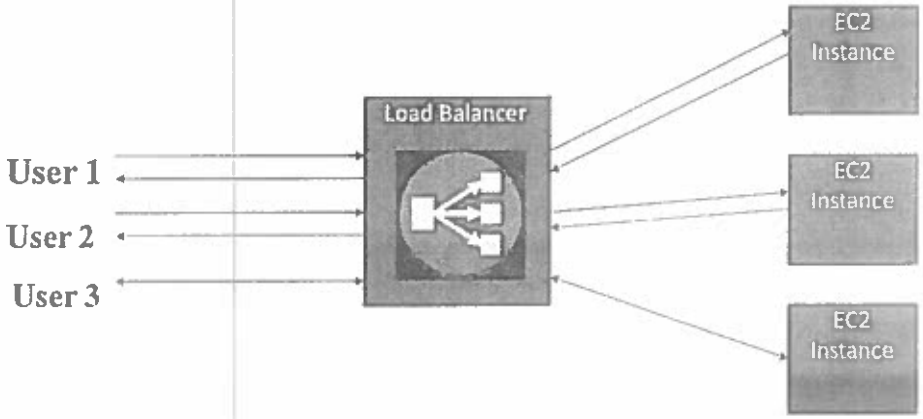
High Availability & Scalability For EC2

- Vertical Scaling: Increase instance size (= scale up / down)
  - From: t2.nano - 0.5G of RAM, 1 vCPU
  - To: u-12tb1.metal – 12.3 TB of RAM, 448 vCPUs
- Horizontal Scaling: Increase number of instances (= scale out / in)
  - Auto Scaling Group
  - Load Balancer
- High Availability: Run instances for the same application across multi AZ
  - Auto Scaling Group multi AZ
  - Load Balancer multi AZ

15

What is load balancing?

- Load balancers are servers that forward internet traffic to multiple servers (EC2 Instances) downstream.



16

Why use a load balancer?

- Spread load across multiple downstream instances
- Expose a single point of access (DNS) to your application
- Seamlessly handle failures of downstream instances
- Do regular health checks to your instances
- Provide SSL termination (HTTPS) for your websites
- Enforce stickiness with cookies
- High availability across zones
- Separate public traffic from private traffic

17

Why use an EC2 Load Balancer?

- An ELB (EC2 Load Balancer) is a managed load balancer
  - AWS guarantees that it will be working
  - AWS takes care of upgrades, maintenance, high availability
  - AWS provides only a few configuration knobs
- It costs less to setup your own load balancer but it will be a lot more effort on your end.
- It is integrated with many AWS offerings / services

18

Types of load balancer on AWS

- AWS has 3 kinds of Load Balancers

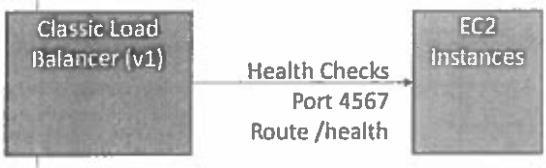
- Classic Load Balancer (v1 - old generation) - 2009
- Application Load Balancer (v2 - new generation) - 2016
- Network Load Balancer (v2 - new generation) - 2017
- Overall, it is recommended to use the newer / v2 generation load balancers as they provide more features

- You can setup internal (private) or external (public) ELBs

19

Health Checks

- Health Checks are crucial for Load Balancers
- They enable the load balancer to know if instances it forwards traffic to are available to reply to requests
- The health check is done on a port and a route (/health is common)
- If the response is not 200 (OK), then the instance is unhealthy



20

Application Load Balancer (v2)

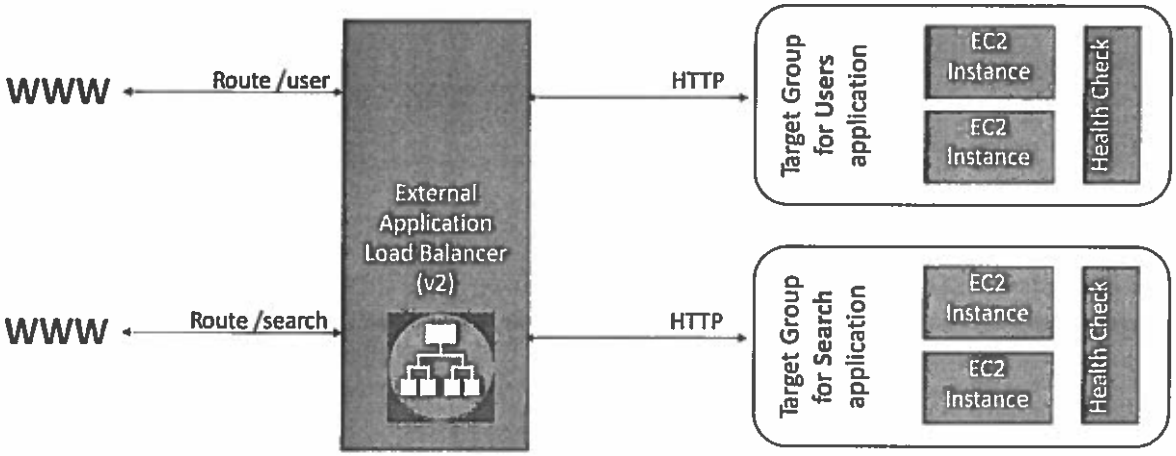
- Application load balancers (Layer 7) allow to do:
  - Load balancing to multiple HTTP applications across machines (target groups)
  - Load balancing to multiple applications on the same machine (ex: containers)
  - Load balancing based on route in URL
  - Load balancing based on hostname in URL

- Basically, they're awesome for micro services & container-based application (example: Docker & Amazon ECS)
- Has a port mapping feature to redirect to a dynamic port

- In comparison, we would need to create one Classic Load Balancer per application before.That was very expensive and inefficient!

21

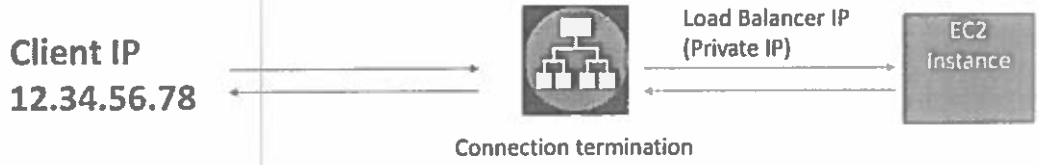
Application Load Balancer (v2) HTTP Based Traffic



22

Application Load Balancer v2 Good to Know

- Stickiness can be enabled at the target group level
  - Same request goes to the same instance
  - Stickiness is directly generated by the ALB (not the application)
- ALB support HTTP/HTTPS & Websockets protocols
- The application servers don't see the IP of the client directly
  - The true IP of the client is inserted in the header X-Forwarded-For
  - We can also get Port (X-Forwarded-Port) and proto (X-Forwarded-Proto)



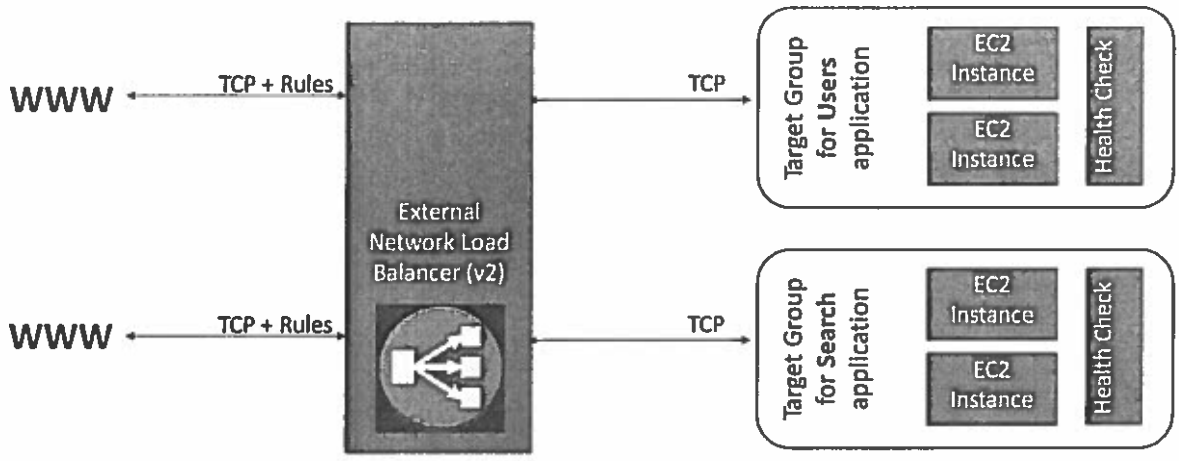
23

Network Load Balancer (v2)

- Network load balancers (Layer 4) allow to do:
  - Forward TCP traffic to your instances
  - Handle millions of request per seconds
  - Support for static IP or elastic IP
  - Less latency ~100 ms (vs 400 ms for ALB)
- Network Load Balancers are mostly used for extreme performance and should not be the default load balancer you choose
- Overall, the creation process is the same as Application Load Balancers

24

Network Load Balancer (v2) TCP Based Traffic



25

Load Balancer Good to Know

- Classic Load Balancers are Deprecated
  - Application Load Balancers for HTTP / HTTPs & Websocket
  - Network Load Balancer for TCP
- CLB and ALB support SSL certificates and provide SSL termination
- All Load Balancers have health check capability
- ALB can route on based on hostname / path
- ALB is a great fit with ECS (Docker)

26

Load Balancer Good to Know

- Any Load Balancer (CLB, ALB, NLB) has a static host name. Do not resolve and use underlying IP
- LBs can scale but not instantaneously – contact AWS for a "warm-up"
- NLB directly see the client IP
- 4xx errors are client induced errors
- 5xx errors are application induced errors
  - Load Balancer Errors 503 means at capacity or no registered target
- If the LB can't connect to your application, check your security groups!

27

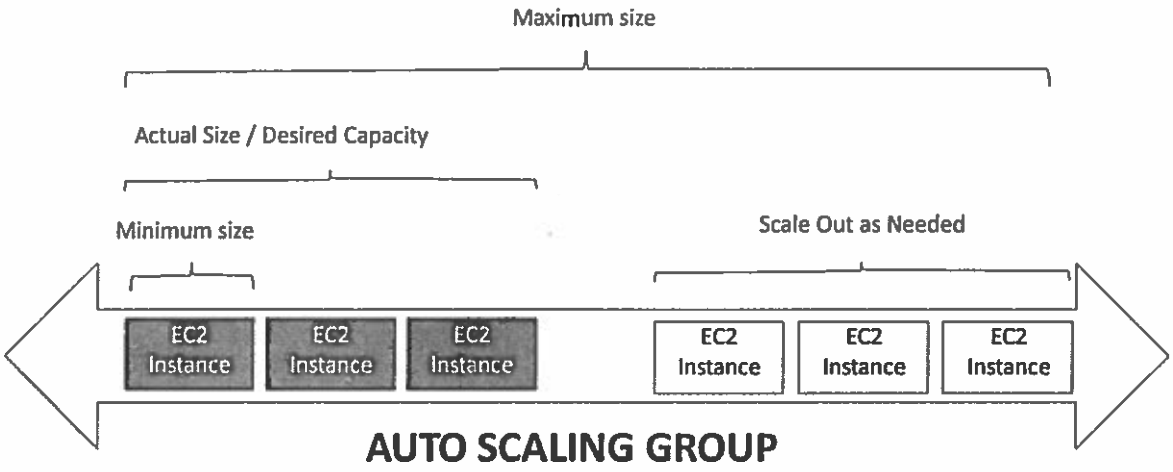
What's an Auto Scaling Group?

- In real-life, the load on your websites and application can change
- In the cloud, you can create and get rid of servers very quickly

- The goal of an Auto Scaling Group (ASG) is to:
  - Scale out (add EC2 instances) to match an increased load
  - Scale in (remove EC2 instances) to match a decreased load
  - Ensure we have a minimum and a maximum number of machines running
  - Automatically Register new instances to a load balancer

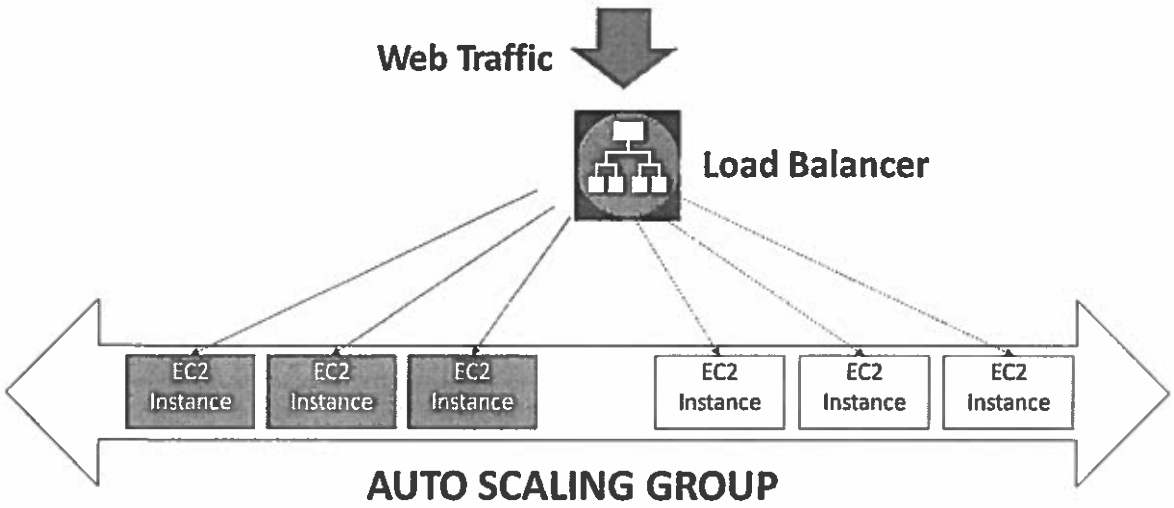
28

Auto Scaling Group in AWS



29

Auto Scaling Group in AWS With Load Balancer



30

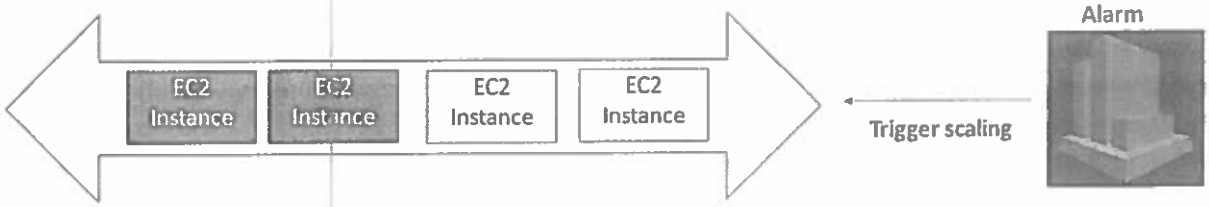
ASGs have the following attributes

- A launch configuration
  - AMI + Instance Type
  - EC2 User Data
  - EBS Volumes
  - Security Groups
  - SSH Key Pair
- Min Size / Max Size / Initial Capacity
- Network + Subnets Information
- Load Balancer Information
- Scaling Policies

31

Auto Scaling Alarms

- It is possible to scale an ASG based on CloudWatch alarms
- An Alarm monitors a metric (such as Average CPU)
- Metrics are computed for the overall ASG instances
- Based on the alarm:
  - We can create scale-out policies (increase the number of instances)
  - We can create scale-in policies (decrease the number of instances)



32

Auto Scaling New Rules

- It is now possible to define "better" auto scaling rules that are directly managed by EC2
  - Target Average CPU Usage
  - Number of requests on the ELB per instance
  - Average Network In
  - Average Network Out
- These rules are easier to set up and can make more sense

33

ASG Brain dump

- Scaling policies can be on CPU, Network… and can even be on custom metrics or based on a schedule (if you know your visitors patterns)
- ASGs use Launch configurations and you update an ASG by providing a new launch configuration
- IAM roles attached to an ASG will get assigned to EC2 instances
- ASG are free. You pay for the underlying resources being launched
- Having instances under an ASG means that if they get terminated for whatever reason, the ASG will restart them. Extra safety!
- ASG can terminate instances marked as unhealthy by an LB (and hence replace them)

34

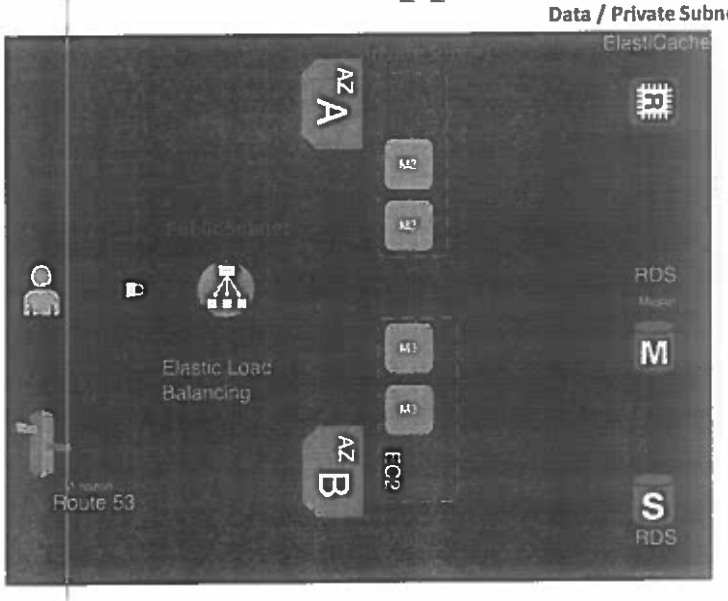
AWS-2

01-12-2019

AWS Fundamentals – Part III
Route 53, RDS, VPC and S3

1

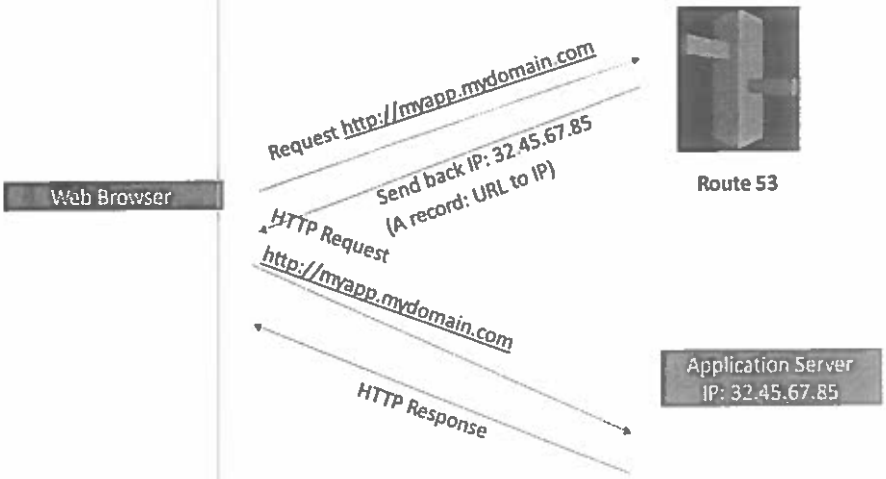
Typical architecture Web App 3-tier



AWS Route 53 Overview

- Route53 is a Managed DNS (Domain Name System)
- DNS is a collection of rules and records which helps clients understand how to reach a server through URLs.
- In AWS, the most common records are:
  - A: URL to IPv4
  - AAAA: URL to IPv6
  - CNAME: URL to URL
  - Alias: URL to AWS resource.

Route 53 – Diagram for A Record



4

AWS Route 53 Overview

- Route53 can use:
  - public domain names you own (or buy) application1.mypublicdomain.com
  - private domain names that can be resolved by your instances in your VPCs. application1.company.internal

- Route53 has advanced features such as:
  - Load balancing (through DNS – also called client load balancing)
  - Health checks (although limited…)
  - Routing policy: simple, failover, geolocation, geoproximity, latency, weighted

- Prefer Alias over CNAME for AWS resources (for performance reasons)

5

Working with Records and Routing Policies

- **Simple routing policy** – Use for a single resource that performs a given function for your domain, for example, a web server that serves content for the example.com website.

- **Failover routing policy** – Use when you want to configure active-passive failover.

- **Geolocation routing policy** – Use when you want to route traffic based on the location of your users.

- **Geoproximity routing policy** – Use when you want to route traffic based on the location of your resources and, optionally, shift traffic from resources in one location to resources in another.

6

Working with Records and Routing Policies

- **Latency routing policy** – Use when you have resources in multiple AWS Regions and you want to route traffic to the region that provides the best latency.

- **Multivalued answer routing policy** – Use when you want Route 53 to respond to DNS queries with up to eight healthy records selected at random.

- **Weighted routing policy** – Use to route traffic to multiple resources in proportions that you specify.

7

01-12-2019

AWS Route 53 Overview

- You should know all the record types:
  - A: URL to IPv4
  - AAAA: URL to IPv6
  - CNAME: URL to URL
  - Alias: URL to AWS resource
- You should know to use Alias records over CNAME for AWS resources

AWS RDS Overview

- RDS stands for Relational Database Service
- It's a managed DB service for DB use SQL as a query language.
- It allows you to create databases in the cloud that are managed by AWS
  - Postgres
  - Oracle
  - MySQL
  - MariaDB
  - Oracle
  - Microsoft SQL Server
  - Aurora (AWS Proprietary database)

9

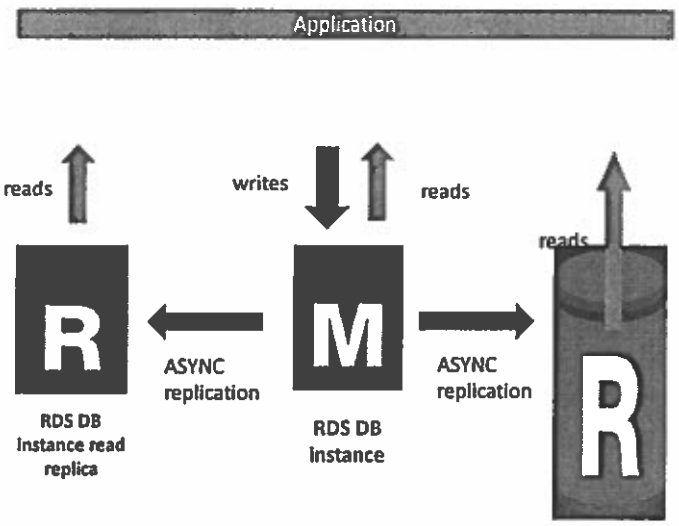
Advantage over using RDS versus deploying DB on EC2

- Managed service:
- OS patching level
- Continuous backups and restore to specific timestamp (Point in Time Restore)!
- Monitoring dashboards
- Read replicas for improved read performance
- Multi AZ setup for DR (Disaster Recovery)
- Maintenance windows for upgrades
- Scaling capability (vertical and horizontal)
- BUT you can't SSH into your instances

10

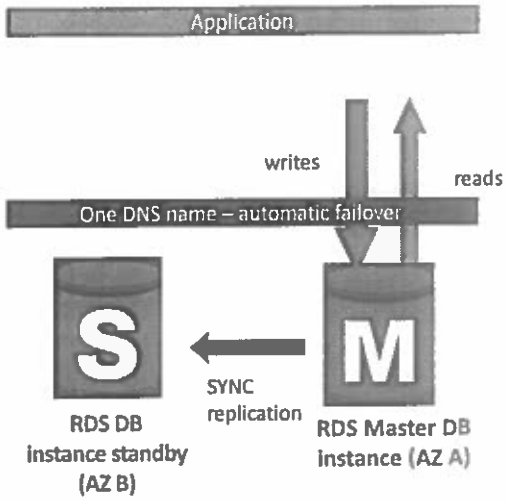
RDS Read Replicas for read scalability

- Up to 5 Read Replicas
- Within AZ, Cross AZ or Cross Region
- Replication is ASYNC, so reads are eventually consistent
- Replicas can be promoted to their own DB
- Applications must update the connection string to leverage read replicas



RDS Multi AZ (Disaster Recovery)

- SYNC replication
- One DNS name – automatic app failover to standby
- Increase availability
- Failover in case of loss of AZ, loss of network, instance or storage failure
- No manual intervention in apps
- Not used for scaling



RDS Backups

- Backups are automatically enabled in RDS

- Automated backups:
  - Daily full snapshot of the database
  - Capture transaction logs in real time
  - => ability to restore to any point in time
  - 7 days retention (can be increased to 35 days)

- DB Snapshots:
  - Manually triggered by the user
  - Retention of backup for as long as you want

13

RDS Security

- RDS databases are usually deployed within a private subnet, not in a public one
- RDS Security works by leveraging security groups (the same concept as for EC2 instances) – it controls who can communicate with RDS
- IAM policies help control who can manage AWS RDS
- Traditional Username and Password can be used to login to the database
- IAM users can now be used too (for MySQL / Aurora – NEW!)

14

RDS vs Aurora

- Aurora is a proprietary technology from AWS (not open sourced)
- Postgres and MySQL are both supported as Aurora DB (that means your drivers will work as if Aurora was a Postgres or MySQL database)
- Aurora is "AWS cloud optimized" and claims 5x performance improvement over MySQL on RDS, over 3x the performance of Postgres on RDS
- Aurora storage automatically grows in increments of 10GB, up to 64 TB.
- Aurora can have 15 replicas while MySQL has 5, and the replication process is faster (sub 10 ms replica lag)
- Failover in Aurora is instantaneous. It's HA native.
- Aurora costs more than RDS (20% more) – but is more efficient

15

01-12-2019

Amazon S3
Another base block of AWS

16

Need more Space? Storage?

- The need for **storage** is increasing every day and knowing the amount of capacity you may need in the future is difficult to predict.
- You may either over-utilize it leading to an application failure because of not having sufficient space.
- you may end up buying stacks of storage which will then be underutilized.
- Keeping all these hassles in mind, Amazon came up with an internet storage service called *AWS S3*.

17

Amazon Simple Storage Service (S3)

- **Amazon S3** (Simple Storage Service) is a scalable, high-speed, low-cost web-based service designed for online backup and archiving of data and application programs.
- It allows to upload, store, and download any type of files up to 5 GB in size.
- This service allows the subscribers to access the same systems that Amazon uses to run its own web sites.
- The subscriber has control over the accessibility of data, i.e. privately/publicly accessible.

18

Amazon S3 Advantages

- **Low cost and Easy to Use** – Using S3, the user can store a large amount of data at very low charges.
- **Secure** – Amazon S3 supports data transfer over SSL and the data gets encrypted automatically once it is uploaded.
- **Scalable** – We can store as much data as we have and access it anytime.
- **Durable-** It regularly verifies the integrity of data stored using checksums e.g. if corruption in data, it is immediately repaired.
- **Higher performance** – Amazon S3 is integrated with Amazon CloudFront, that distributes content to the end users with low latency and provides high data transfer speeds without any minimum usage commitments.
- **Integrated with AWS services**

19

How is data organized in S3?

Data in S3 is organized in the form of buckets.

- A Bucket is a logical unit of storage in S3.
- A Bucket contains objects which contain the data and metadata.

Before adding any data in S3 the user has to create a bucket which will be used to store objects

20

AWS S3 Overview - Buckets

- Amazon S3 allows people to store objects (files) in "buckets" (directories)
- Buckets must have a globally unique name
- Buckets are defined at the region level
- Naming convention
  - No uppercase
  - No underscore
  - 3-63 characters long
  - Not an IP
  - Must start with lowercase letter or number

AWS S3 Overview - Objects

- Objects (files) have a Key. The key is the FULL path:
  - <my_bucket>/my_file.txt
  - <my_bucket>/my_folder1/another_folder/my_file.txt
- There's no concept of "directories" within buckets (although the UI will trick you to think otherwise)
- Just keys with very long names that contain slashes ("/")
- Object Values are the content of the body:
  - Max Size is 5TB
  - If uploading more than 5GB, must use "multi-part upload"
- Metadata (list of text key / value pairs – system or user metadata)
- Tags (Unicode key / value pair – up to 10) – useful for security / lifecycle
- Version ID (if versioning is enabled)

22

AWS S3 - Versioning

- You can version your files in AWS S3
- It is enabled at the bucket level
- Same key overwrite will increment the "version": 1, 2, 3....
- It is best practice to version your buckets
  - Protect against unintended deletes (ability to restore a version)
  - Easy roll back to previous version
- Any file that is not versioned prior to enabling versioning will have version "null"

23

S3 Encryption for Objects

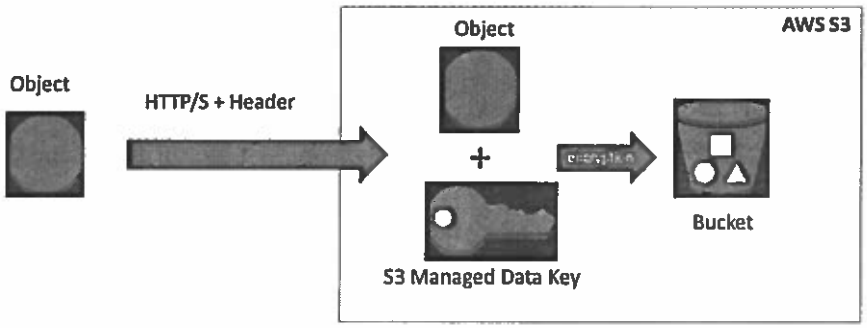
- There are 4 methods of encrypting objects in S3
  - SSE-S3: encrypts S3 objects using keys handled & managed by AWS
  - SSE-KMS: leverage AWS Key Management Service to manage encryption keys
  - SSE-C: when you want to manage your own encryption keys
  - Client Side Encryption

- It's important to understand which ones are adapted to which situation for the exam

24

SSE-S3

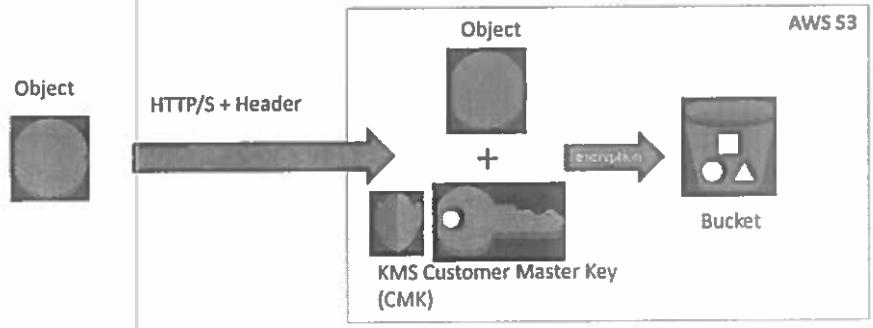
- SSE-S3: encryption using keys handled & managed by AWS S3
- Object is encrypted server side
- AES-256 encryption type
- Must set header: "x-amz-server-side-encryption": "AES256"



25

SSE-KMS

- SSE-KMS: encryption using keys handled & managed by KMS
- KMS Advantages: user control + audit trail
- Object is encrypted server side
- Must set header: "x-amz-server-side-encryption": "aws:kms"

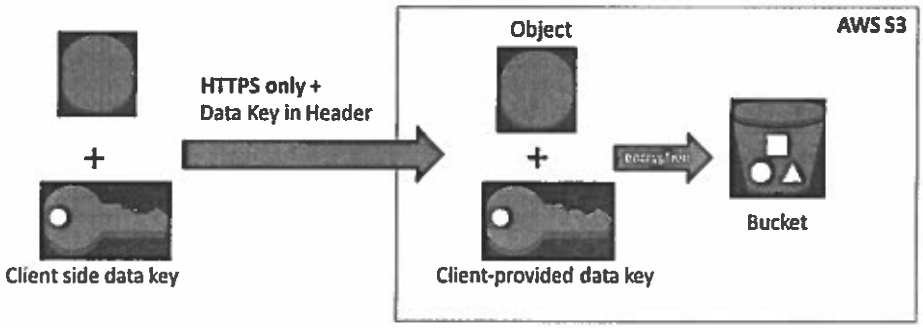


26

- Amazon S3 does not store the encryption key you provide
- HTTPS must be used
- Encryption key must provided in HTTP headers, for every HTTP request made

SSE-C

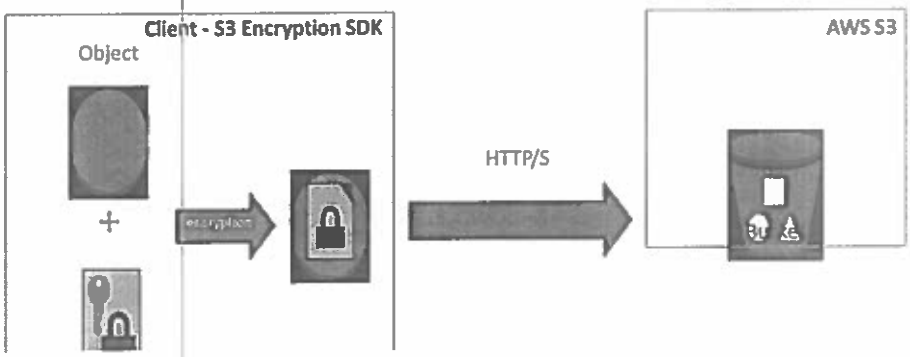
Object



27

- Clients must decrypt data themselves when retrieving from S3
- Customer fully manages the keys and encryption cycle

Client Side Encryption



28

Encryption in transit (SSL)

- AWS S3 exposes:
  - HTTP endpoint: non encrypted
  - HTTPS endpoint: encryption in flight

- You're free to use the endpoint you want, but HTTPS is recommended
- HTTPS is mandatory for SSE-C
- Encryption in flight is also called SSL / TLS

29

AWS S3 Operations

- **Create a Bucket** – Create and name your own bucket in which to store your objects.
- **Write an Object** – Store data by creating or overwriting an object. When you write an object, you specify a unique key in the namespace of your bucket. This is also a good time to specify any access control you want on the object.
- **Read an Object** – Read data back. You can download the data via HTTP or BitTorrent.
- **Deleting an Object** – Delete some of your data.
- **Listing Keys** – List the keys contained in one of your buckets. You can filter the key list based on a prefix.

30

S3 Security

- User based
  - IAM policies - which API calls should be allowed for a specific user from IAM console

- Resource Based
  - Bucket Policies - bucket wide rules from the S3 console - allows cross account
  - Object Access Control List (ACL) – finer grain
  - Bucket Access Control List (ACL) – less common

31

S3 Bucket Policies

- JSON based policies
  - Resources: buckets and objects
  - Actions: Set of API to Allow or Deny
  - Effect: Allow / Deny
  - Principal:The account or user to apply the policy to

- Use S3 bucket for policy to:
  - Grant public access to the bucket
  - Force objects to be encrypted at upload
  - Grant access to another account (Cross Account)

32

S3 Websites

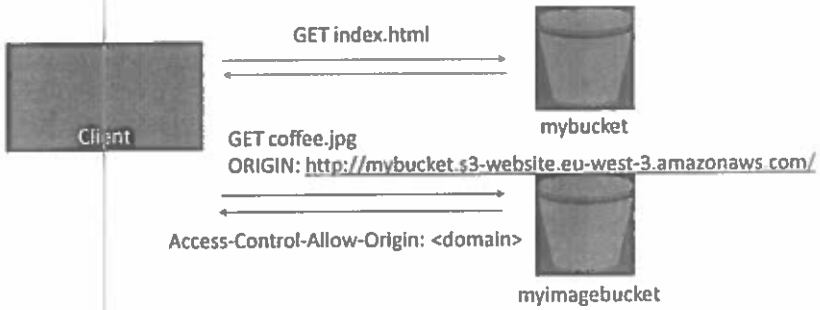
- S3 can host static websites and have them accessible on the www
- The website URL will be:
  - <bucket-name>.s3-website-<AWS-region>.amazonaws.com

    OR
  - <bucket-name>.s3-website.<AWS-region>.amazonaws.com
- If you get a 403 (Forbidden) error, make sure the bucket policy allows public reads!

33

websites that can request your files in S3 (and limit your costs)

S3 CORS



34

AWS S3 - Consistency Model

- Read after write consistency for PUTS of new objects
  - As soon as an object is written, we can retrieve it
    ex: (PUT 200 -> GET 200)
  - This is true, except if we did a GET before to see if the object existed
    ex: (GET 404 -> PUT 200 -> GET 404) – eventually consistent

- Eventual Consistency for DELETES and PUTS of existing objects
  - If we read an object after updating, we might get the older version
    ex: (PUT 200 -> PUT 200 -> GET 200 (might be older version))
  - If we delete an object, we might still be able to retrieve it for a short time ex: (DELETE 200 -> GET 200)

35

S3 storage classes with a "health-care" use case:

• **Amazon S3 Standard for frequent data access**

This is suitable for performance sensitive use cases where the latency should be kept low.

e.g. in a hospital, frequently accessed data will be the data of admitted patients, which should be retrieved quickly.

. Amazon S3 Standard for infrequent data access

This is suitable for use cases where the data is long lived and less frequently accessed, i.e for data archival but still expects high performance.

e.g. in the same hospital, people who have been discharged, their records/data will not be needed on a daily basis, but if they return with any complication, their discharge summary should be retrieved quickly.

36

S3 storage classes with a "health-care" use case:

Amazon Glacier:

Suitable for use cases where the data is to be archived, and high performance is not required, it has a lower cost than the other two services.

e.g. in the hospital, patients' test reports, prescriptions, MRI, X Ray, Scan docs etc. that are older than a year will not be needed in the daily run and even if it is required, lower latency is not needed.

37

Characteristics	Standard	Standard - Infrequent Access	Glacier
Durability	99.99%	99.99%	99.99%
Availability	99.99%	99.90%	N/A
Minimum Object Size	No limit	128KB	No limit
Minimum Storage Duration	No minimum duration	30 Days	90 Days
First Byte Latency	milliseconds	milliseconds	4 hours
Retneval Fee	No Fee	per GB retrieved	per GB retrieved

38

AWS S3 Features

- **Storage Class**

- Amazon S3 offers a range of storage classes designed for different use cases.

- These include Amazon S3 STANDARD for general-purpose storage of frequently accessed data.

- Amazon S3 STANDARD_IA for long-lived, but less frequently accessed data,

- And GLACIER for long-term archive.

01-12-2019

Lets do a small Project

Hosting a Static website on AWS S3

40

40

Project Statement

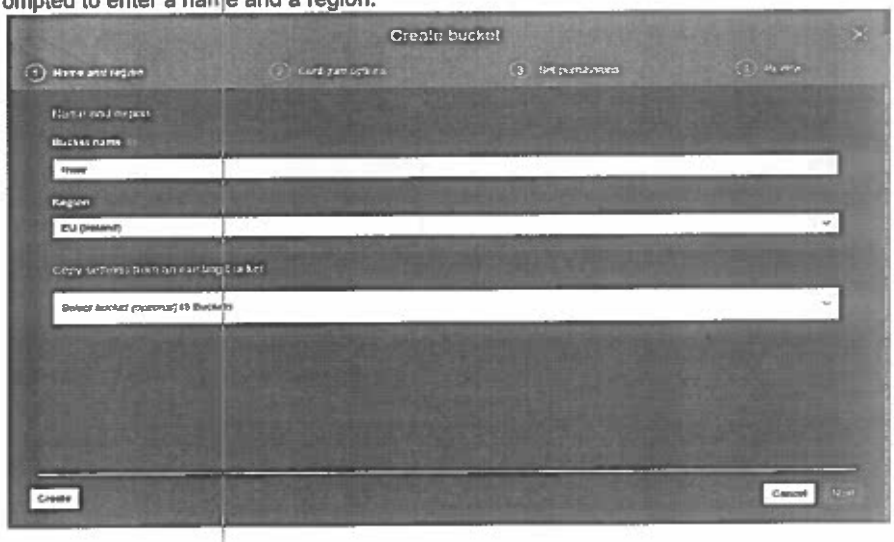
Project Statement – Hosting a Static Website on Amazon S3

- Let's first understand: What is a static website?
- In short, it's a website comprised of only HTML, CSS, and/or JavaScript. That means server-side scripts aren't supported, so if you want to host a Rails or PHP app, you'll need to look elsewhere.
- For simpler purposes, welcome to the wonderful world of hosting websites on AWS S3!

41

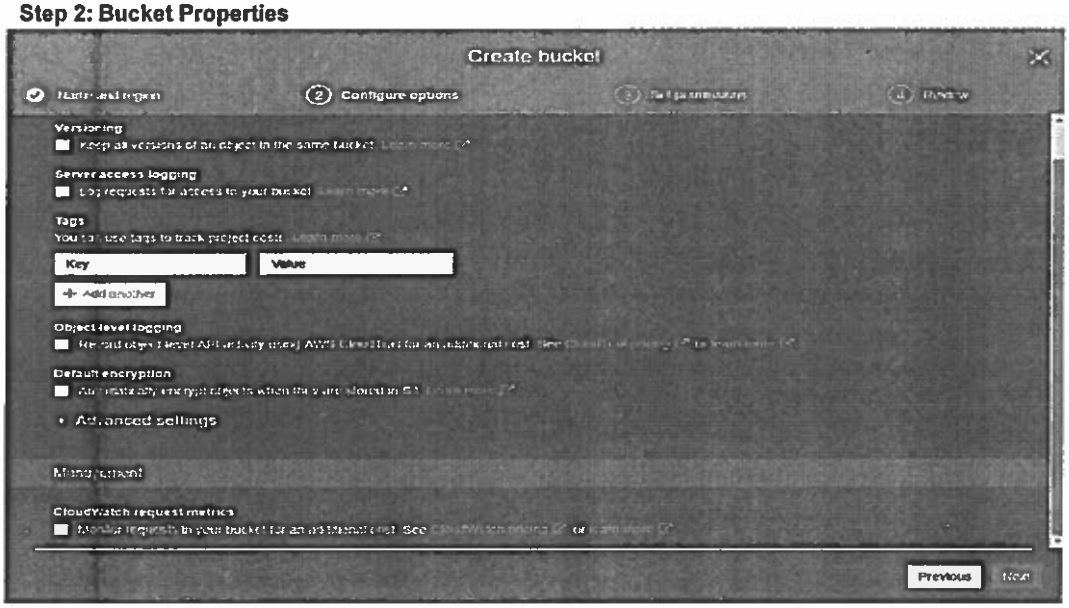
Step 1: Create a bucket

- To create a bucket, navigate to S3 in the AWS Management Console and hit Create Bucket. You'll be prompted to enter a name and a region.

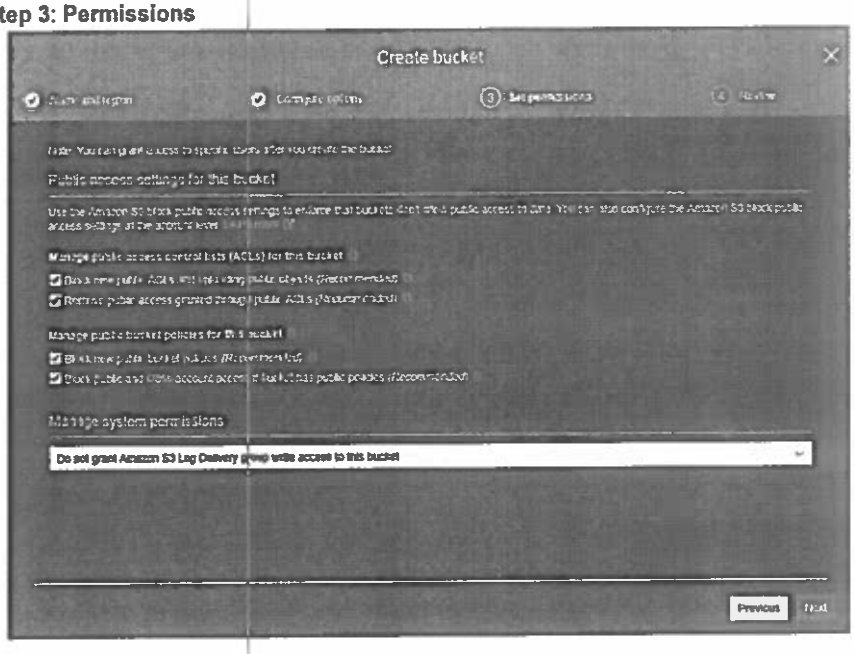


42

Step 2: Bucket Properties

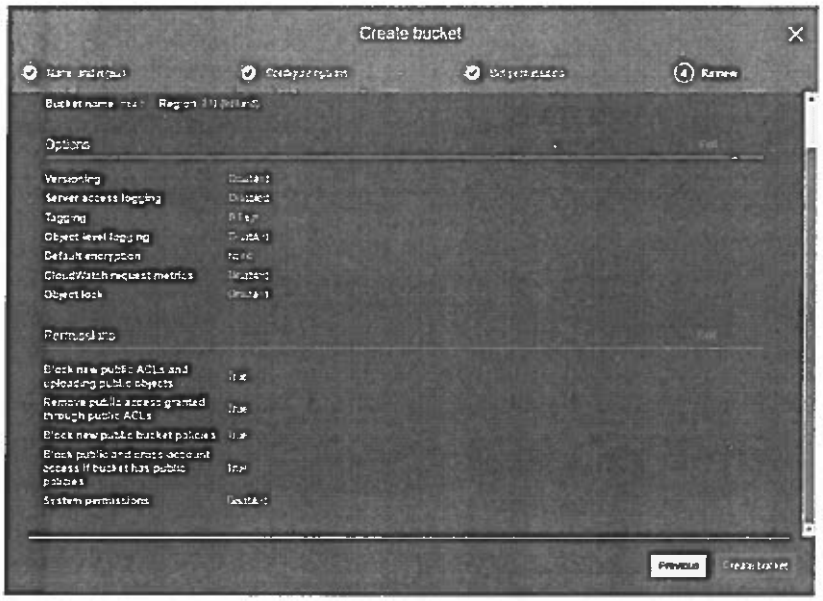


Step 3: Permissions



44

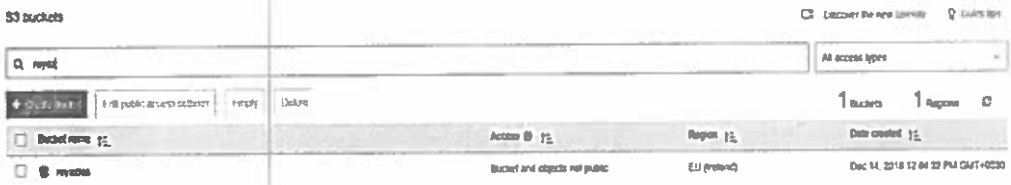
Step 1: Create a bucket



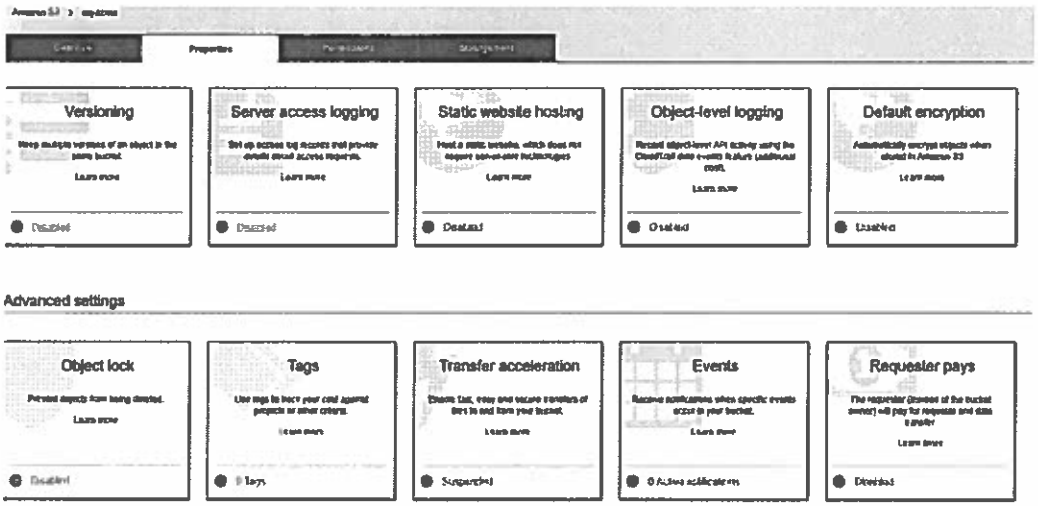
45

01-12-2019

Find your bucket by searching.

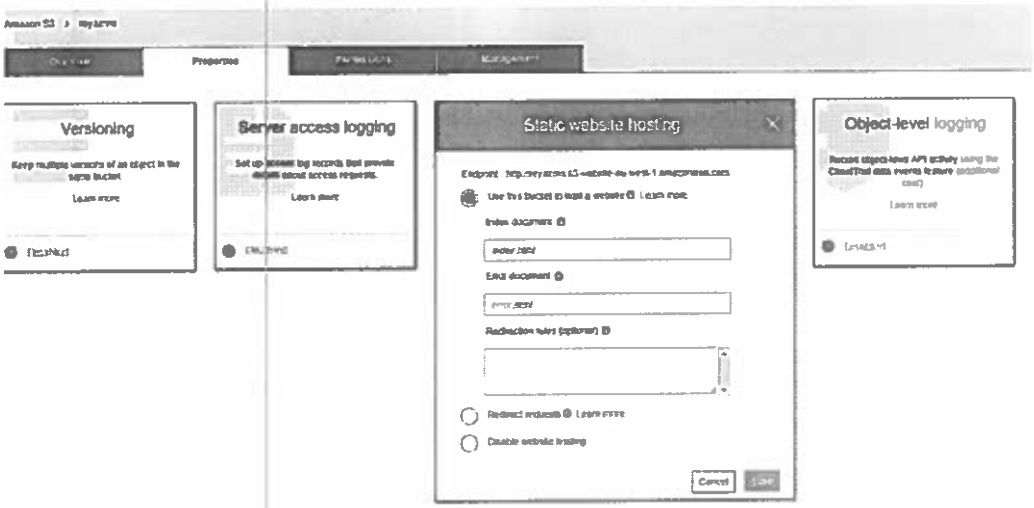


Find your bucket and click on it and you will see the below screen



47

Click on Static website hosting and select "Use this bucket to host a website"



48

You don't have index.html. Create a simple html file with name index.html and upload to the bucket and Make public.
Access the bucket link from the properties

01-12-2019

My first S3 website

I can't believe it was that easy!

50

50

AWS-3

01-12-2019

Networking - VPC

1

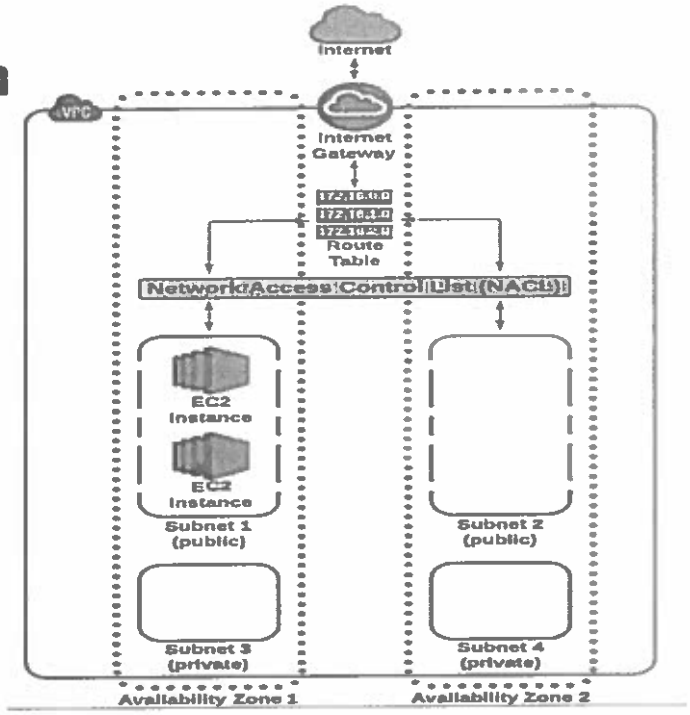
What is a VPC?

- A private sub-section of AWS that we control, in which we can place AWS resources (such as EC2 instances and databases).

- We have FULL control over who has access to the AWS resources that you place inside your VPC.

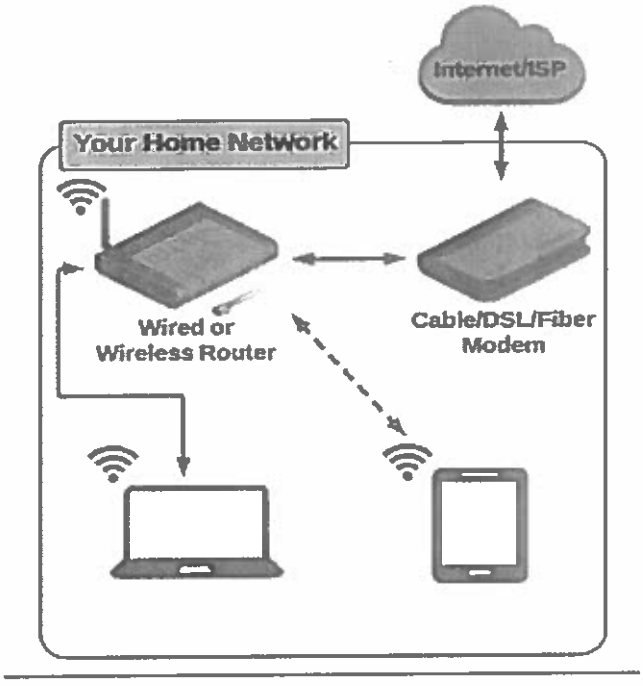
2

VPC Diagran

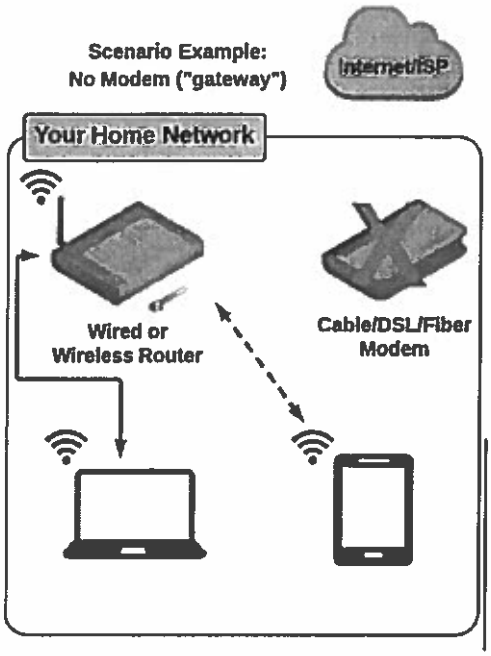


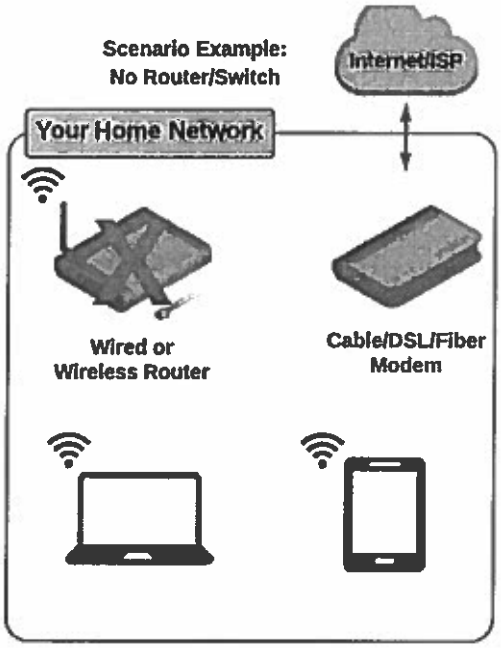
3

Home Network

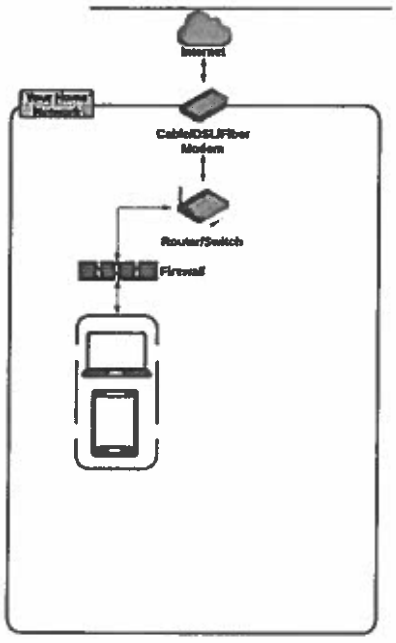


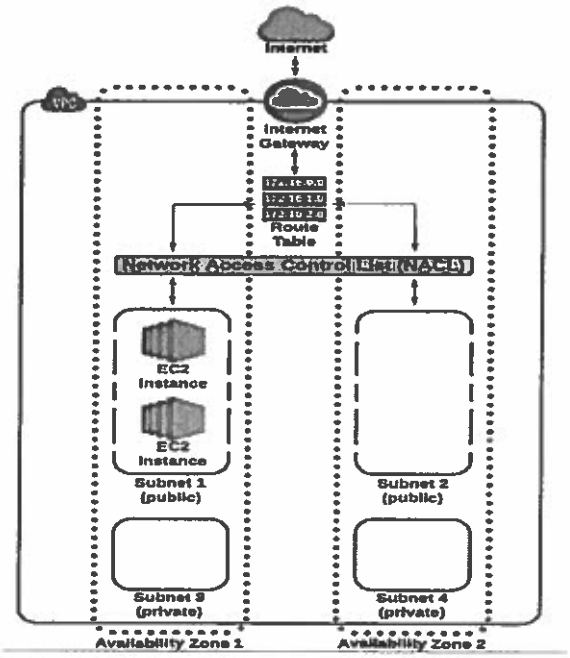
4


Scenario Example:
No Modem ("gateway")
Internet/ISP
Your Home Network
Wired or
Wireless Router
Cable/DSL/Fiber
Modem


Scenario Example:
No Router/Switch
Internet/ISP
Your Home Network
Wired or
Wireless Router
Cable/DSL/Fiber
Modem

5

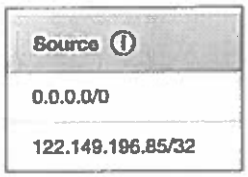

Internet
Cable/DSL/Fiber
Modem
Router/Switch
Firewall


Internet
Internet
Gateway
Route
Table
Network Access Control List (NACL)
EC2
Instance
EC2
Instance
Subnet 1
(public)
Subnet 2
(public)
Subnet 3
(private)
Subnet 4
(private)
Availability Zone 1
Availability Zone 2

6

Understanding CIDR - IPv4 (Classless Inter-Domain Routing)

- CIDR are used for Security Groups rules, or AWS networking in general



- They help to define an IP address range
  - WW.XX.YY.ZZ/32 == one IP
  - 0.0.0.0/0 == all IPs
  - But we can define for ex: 192.168.0.0/26:
    192.168.0.0 – 192.168.0.63 (64 IP)

7

Understanding CIDR

- A CIDR has two components:
  - The base IP (XX.XX.XX.XX)
  - The Subnet Mask (/26)

- The base IP represents an IP contained in the range
- The subnet masks defines how many bits can change in the IP

- The subnet mask can take two forms. Examples:
  - 255.255.255.0 (less common)
  - /24 (more common)

8

Understanding CIDRs Subnet Masks

- The subnet masks basically allows part of the underlying IP to get additional next values from the base IP
  - /32 allows for 1 IP = 2^0
  - /31 allows for 2 IP = 2^1
  - /30 allows for 4 IP = 2^2
  - /29 allows for 8 IP = 2^3
  - /28 allows for 16 IP = 2^4
  - /27 allows for 32 IP = 2^5
  - /26 allows for 64 IP = 2^6
  - /25 allows for 128 IP = 2^7
  - /24 allows for 256 IP = 2^8
  - /16 allows for 65,536 IP = 2^16
  - /0 allows for all IPs = 2^32

- Quick memo:
- /32 – no IP number can change
- /24 - last IP number can change
- /16 – last IP two numbers can change
- /8 – last IP three numbers can change
- /0 – all IP numbers can change

9

Understanding CIDRs Little exercise

- 192.168.0.0/24 = … ?
  - 192.168.0.0 – 192.168.0.255 (256 IP)
- 192.168.0.0/16 = … ?
  - 192.168.0.0 – 192.168.255.255 (65,536 IP)
- 134.56.78.123/32 = … ?
  - Just 134.56.78.123
- 0.0.0.0/0
  - All IP!
- When in doubt, use this website: https://www.ipaddressguide.com/cidr

10

Private vs Public IP (IPv4) Allowed ranges

- The Internet Assigned Numbers Authority (IANA) established certain blocks of IPV4 addresses for the use of private (LAN) and public (Internet) addresses.

- Private IP can only allow certain values
  - 10.0.0.0 – 10.255.255.255 (10.0.0.0/8) <= in big networks
  - 172.16.0.0 – 172.31.255.255 (172.16.0.0/12) <= default AWS one
  - 192.168.0.0 – 192.168.255.255 (192.168.0.0/16) <= example: home networks

- All the rest of the IP on the internet are public IP

11

Default VPC Walkthrough

- All new accounts have a default VPC

- New instances are launched into default VPC if no subnet is specified

- Default VPC have internet connectivity and all instances have public IP

- We also get a public and a private DNS name

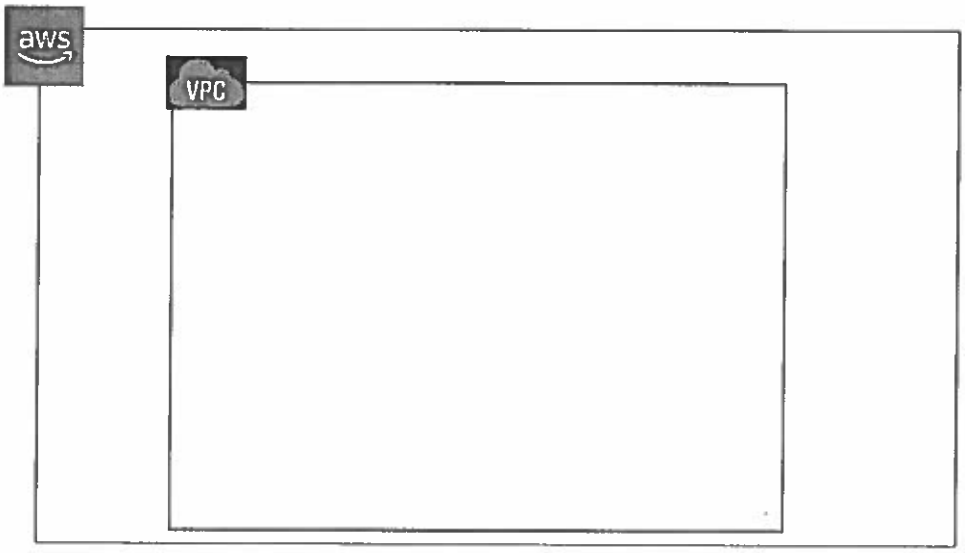
12

VPC in AWS – IPv4

- VPC = Virtual Private Cloud
- You can have multiple VPCs in a region (max 5 per region – soft limit)
- Max CIDR per VPC is 5. For each CIDR:
  - Min size is /28 = 16 IP Addresses
  - Max size is /16 = 65536 IP Addresses
- Because VPC is private, only the Private IP ranges are allowed:
  - 10.0.0.0 – 10.255.255.255 (10.0.0.0/8)
  - 172.16.0.0 – 172.31.255.255 (172.16.0.0/12)
  - 192.168.0.0 – 192.168.255.255 (192.168.0.0/16)
- Your VPC CIDR should not overlap with your other networks (ex: corporate)

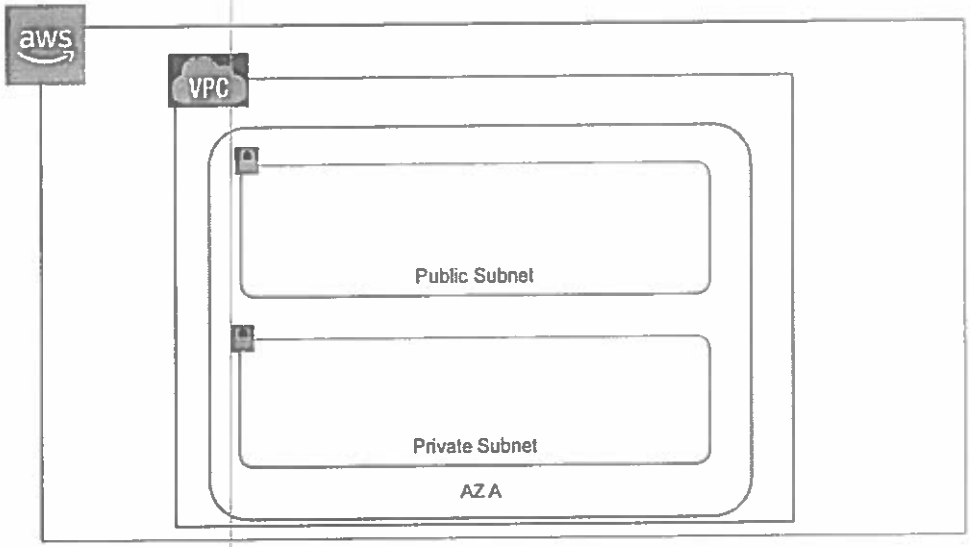
13

State of Hands On



14

Adding Subnets



15

Subnets - IPv4

- AWS reserves 5 IPs address (first 4 and last 1 IP address) in each Subnet
- These 5 IPs are not available for use and cannot be assigned to an instance
- Ex, if CIDR block 10.0.0.0/24, reserved IP are:
  - 10.0.0.0: Network address
  - 10.0.0.1: Reserved by AWS for the VPC router
  - 10.0.0.2: Reserved by AWS for mapping to Amazon-provided DNS
  - 10.0.0.3: Reserved by AWS for future use
  - 10.0.0.255: Network broadcast address. AWS does not support broadcast in a VPC, therefore the address is reserved
- Tip:
  - If you need 29 IP addresses for EC2 instances, you can't choose a Subnet of size /27 (32 IP)

16

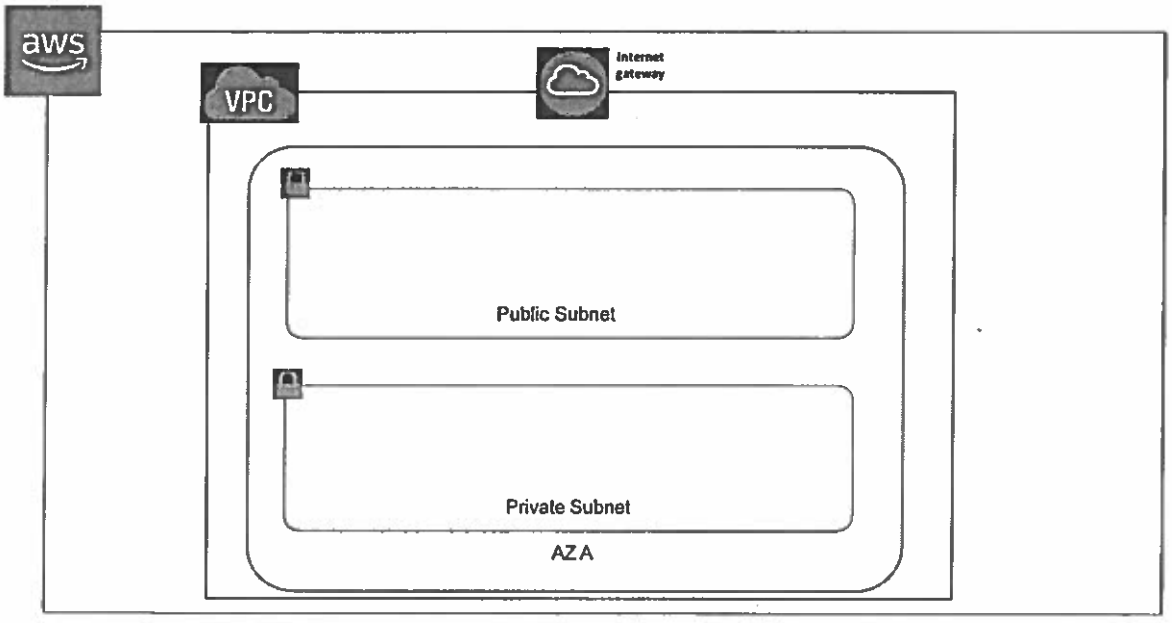
Internet Gateways

- Internet gateways helps our VPC instances connect with the internet
- It scales horizontally and is HA and redundant
- Must be created separately from VPC
- One VPC can only be attached to one IGW and vice versa
- Internet Gateway is also a NAT for the instances that have a public IPv4

- Internet Gateways on their own do not allow internet access…
- Route tables must also be edited!

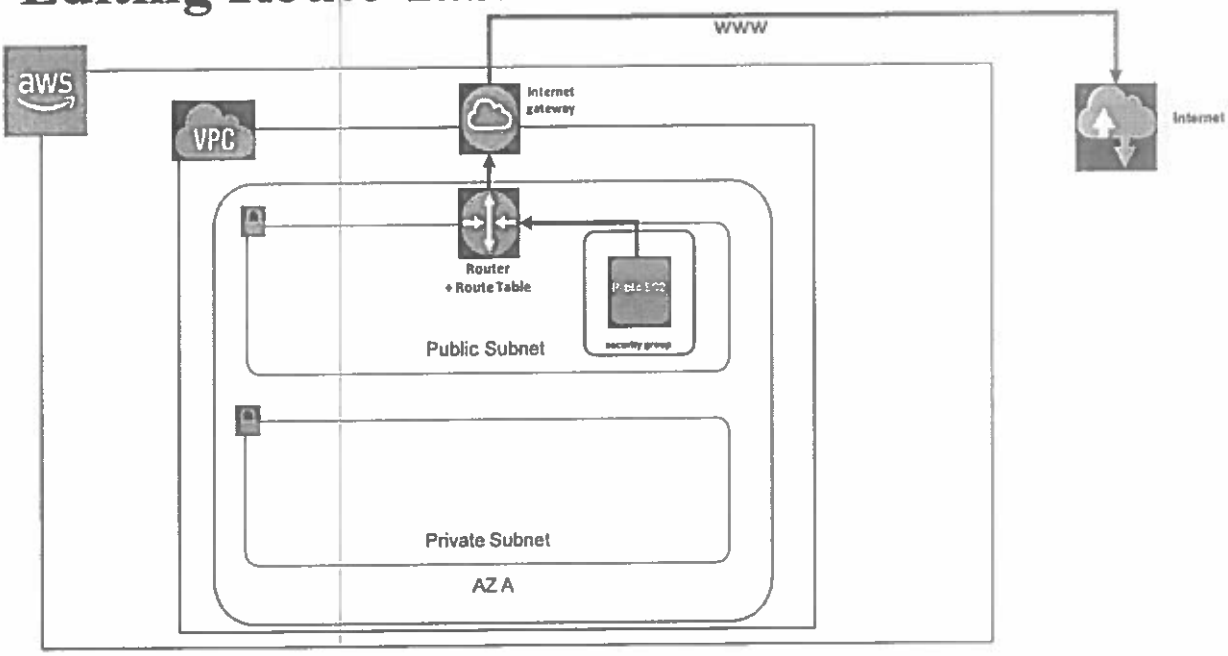
17

Adding IGW



18

Editing Route Tables



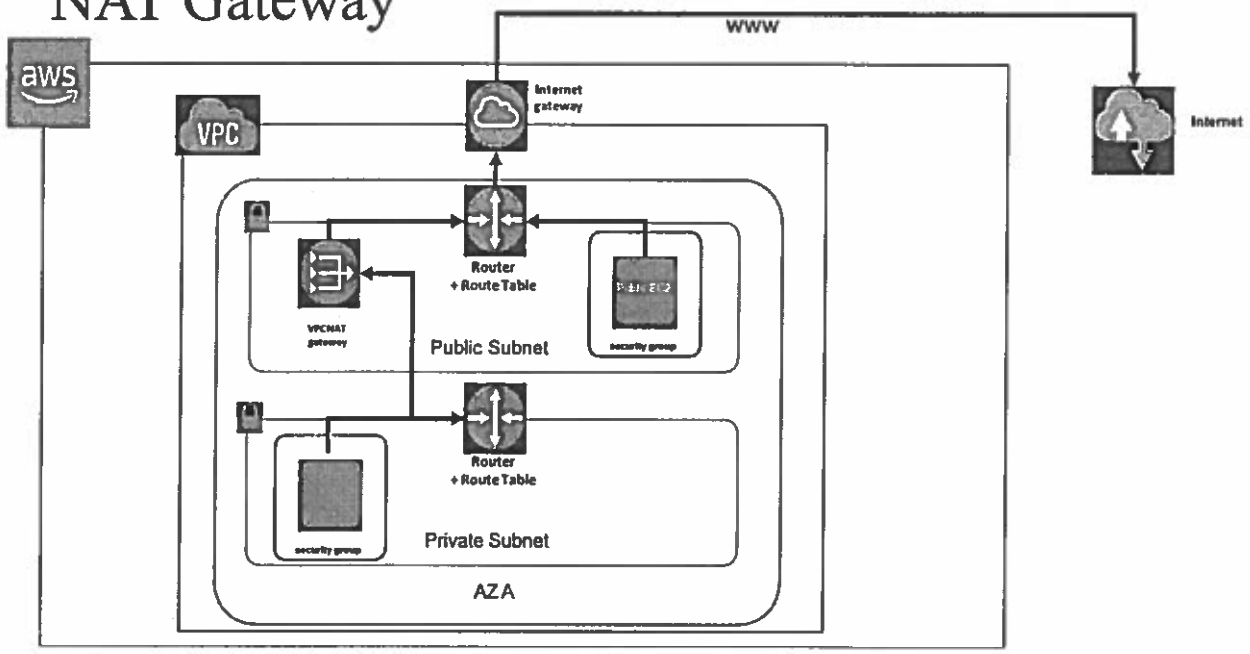
19

NAT Gateway

- AWS managed NAT, higher bandwidth, better availability, no admin
- Pay by the hour for usage and bandwidth
- NAT is created in a specific AZ, uses an EIP
- Cannot be used by an instance in that subnet (only from other subnets)
- Requires an IGW (Private Subnet => NAT => IGW)
- 5 Gbps of bandwidth with automatic scaling up to 45 Gbps

20

NAT Gateway



21

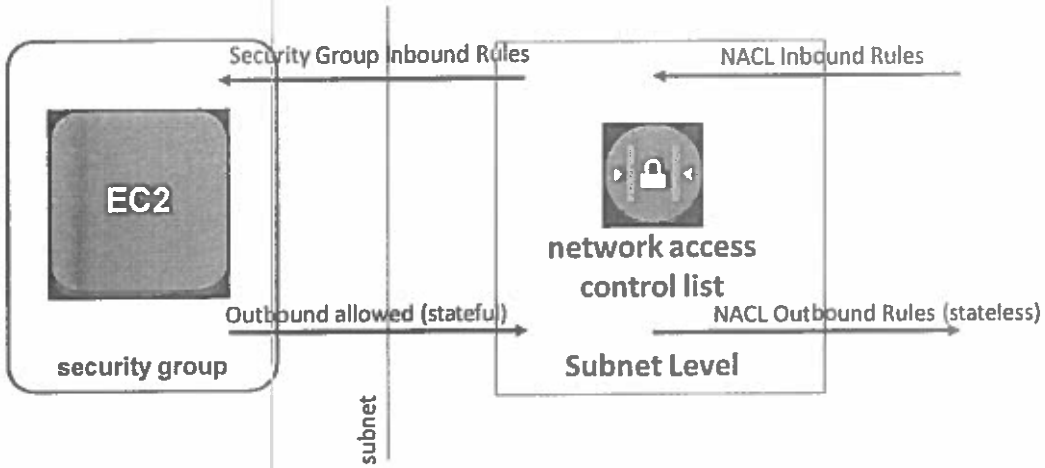
DNS Resolution in VPC

- enableDnsSupport: (= DNS Resolution setting)
  - Default True
  - Helps decide if DNS resolution is supported for the VPC
  - If True, queries the AWS DNS server at 169.254.169.253

- enableDnsHostname: (= DNS Hostname setting)
  - False by default for newly created VPC,True by default for Default VPC
  - Won't do anything unless enableDnsSupport=true
  - If True, Assign public hostname to EC2 instance if it has apublic

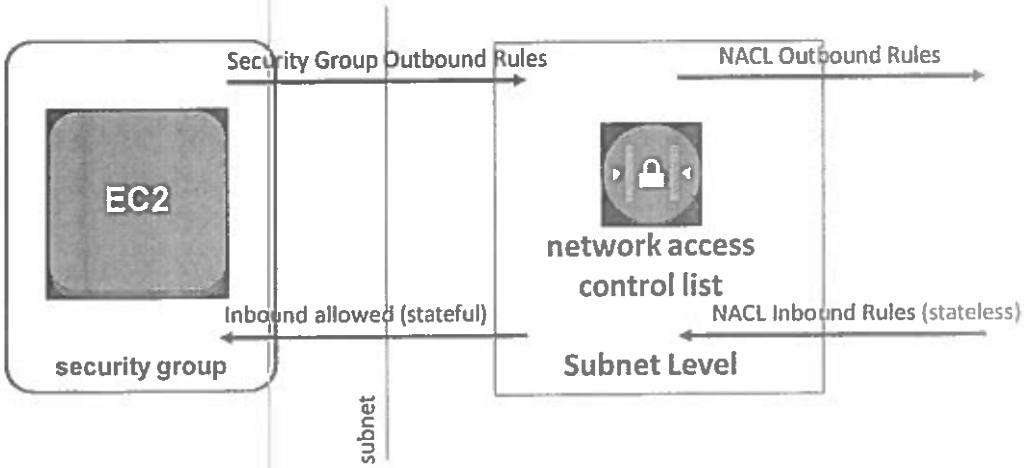
22

Network ACLs & SG Incoming Request



23

Network ACLs & SG Outgoing Request



24

Network ACLs

- NACL are like a firewall which control traffic from and to subnet
- Default NACL allows everything outbound and everything inbound
- One NACL per Subnet, new Subnets are assigned the Default NACL
- Define NACL rules:
  - Rules have a number (1-32766) and higher precedence with a lower number
  - E.g. If you define #100 ALLOW <IP> and #200 DENY <IP> , IP will be allowed
  - Last rule is an asterisk (*) and denies a request in case of no rule match
  - AWS recommends adding rules by increment of 100
- Newly created NACL will deny everything
- NACL are a great way of blocking a specific IP at the subnet level

25

Network ACLs vs Security Groups

Security Group	Network ACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
Is stateful: Return traffic is automatically allowed, regardless of any rules	Is stateless: Return traffic must be explicitly allowed by rules
We evaluate all rules before deciding whether to allow traffic	We process rules in number order when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets it's associated with (therefore, you don't have to rely on users to specify the security group)

26

AWS-4

AWS Monitoring
CloudWatch

1

Why Monitoring is Important

- We know how to deploy applications
  - Safely
  - Automatically
  - Using Infrastructure as Code
  - Leveraging the best AWS components!
- Our applications are deployed, and our users don't care how we did it…
- Our users only care that the application is working!
  - Application latency: will it increase over time?
  - Application outages: customer experience should not be degraded
  - Users contacting the IT department or complaining is not a good outcome
  - Troubleshooting and remediation
- Internal monitoring:
  - Can we prevent issues before they happen?
  - Performance and Cost
  - Trends (scaling patterns)
  - Learning and Improvement

2

Monitoring in AWS

- AWS CloudWatch:
  - Metrics: Collect and track key metrics
  - Logs: Collect, monitor, analyze and store log files
  - Events: Send notifications when certain events happen in your AWS
  - Alarms: React in real-time to metrics / events

3

AWS CloudWatch Metrics

- CloudWatch provides metrics for *every* services in AWS
- Metric is a variable to monitor (CPUUtilization, NetworkIn…)
- Dimension is an attribute of a metric (instance id, environment, etc…).
- Up to 10 dimensions per metric
- Metrics have timestamps
- Can create CloudWatch dashboards of metrics

4

AWS CloudWatch EC2 Detailed monitoring

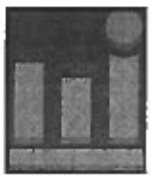
- EC2 instance metrics have metrics "every 5 minutes"
- With detailed monitoring (for a cost), you get data "every 1 minute"
- Use detailed monitoring if you want to more prompt scale your ASG!

- The AWS Free Tier allows us to have 10 detailed monitoring metrics

- Note: EC2 Memory usage is by default not pushed (must be pushed from inside the instance as a custom metric)

5

AWS CloudWatch Alarms



- Alarms are used to trigger notifications for any metric
- Alarms can go to Auto Scaling, EC2 Actions, SNS notifications
- Various options (sampling, %, max, min, etc…)
- Alarm States:
  - OK
  - INSUFFICIENT_DATA
  - ALARM
- Period:
  - Length of time in seconds to evaluate the metric
  - High resolution custom metrics: can only choose 10 sec or 30 sec

6

AWS CloudWatch Events

- Schedule: Cron jobs
- Event Pattern: Event rules to react to a service doing something
  - Ex: CodePipeline state changes!
- Triggers to Lambda functions, SQS/SNS/Kinesis Messages
- CloudWatch Event creates a small JSON document to give information about the change

7

AWS CloudWatch Logs

- Applications can send logs to CloudWatch using the SDK
- CloudWatch can collect log from:
  - Elastic Beanstalk: collection of logs from application
  - ECS: collection from containers
  - AWS Lambda: collection from function logs
  - VPC Flow Logs: VPC specific logs
  - API Gateway
  - CloudTrail based on filter
  - CloudWatch log agents: for example on EC2 machines
  - Route53: Log DNS queries
- CloudWatch Logs can go to:
  - Batch exporter to S3 for archival
  - Stream to ElasticSearch cluster for further analytics

8

AWS CloudWatch Logs

- CloudWatch Logs can use filter expressions
- Logs storage architecture:
  - Log groups: arbitrary name, usually representing an application
  - Log stream: instances within application / log files / containers
- Can define log expiration policies (never expire, 30 days, etc..)
- Using the AWS CLI we can tail CloudWatch logs
- To send logs to CloudWatch, make sure IAM permissions are correct!
- Security: encryption of logs using KMS at the Group Level

Aws-5

Cloud Trail

1

AWS CloudTrail

- Provides governance, compliance and audit for your AWS Account
- CloudTrail is enabled by default!
- Get an history of events / API calls made within your AWS Account by:
  - Console
  - SDK
  - CLI
  - AWS Services
- Can put logs from CloudTrail into CloudWatch Logs
- If a resource is deleted in AWS, look into CloudTrail first!

2

CloudTrail continued…

- CloudTrail shows the past 90 days of activity
- The default UI only shows "Create","Modify" or "Delete" events
- CloudTrail Trail:
  - Get a detailed list of all the events you choose
  - Ability to store these events in S3 for further analysis
  - Can be region specific or global
- CloudTrail Logs have SSE-S3 encryption when placed into S3
- Control access to S3 using IAM, Bucket Policy, etc…

AWS-6

AWS CLI

1

Introduction

- So far, we've interacts with services manually and they exposed standard information for clients:
  - EC2 exposes a standard Linux machine we can use any way we want
  - RDS exposes a standard database we can connect to using a URL
  - ASG / ELB are automated and we don't have to program against them
  - Route53 and S3 was setup manual

- Developing against AWS has two components:
  - How to perform interactions with AWS without using the Online Console?
  - How to interact with AWS Proprietary services? (S3, DynamoDB, etc…)

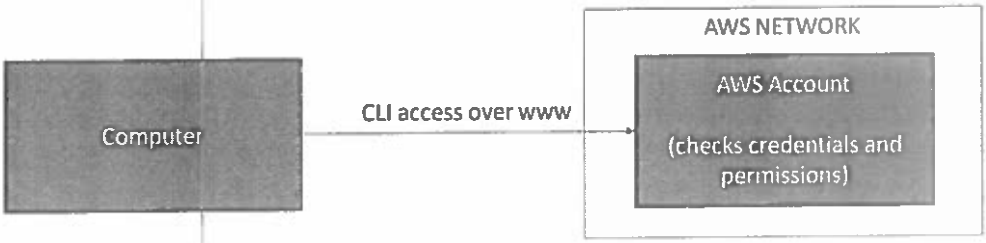
2

Introduction

- Developing and performing AWS tasks against AWS can be done in several ways
  - Using the AWS CLI on our local computer
  - Using the AWS CLI on our EC2 machines
  - Using the AWS Instance Metadata Service for EC2

3

AWS CLI Setup on Linux



- We'll learn how to get our access credentials and protect them
- Do not share your AWS Access Key and Secret key with anyone!

4

AWS CLI ON EC2... THE BAD WAY

- We could run `aws configure` on EC2 just like we did (and it'll work)
- But... it's SUPER INSECURE
- NEVER EVER EVER PUT YOUR PERSONAL CREDENTIALS ON AN EC2
- Your PERSONAL credentials are PERSONAL and only belong on your PERSONAL computer

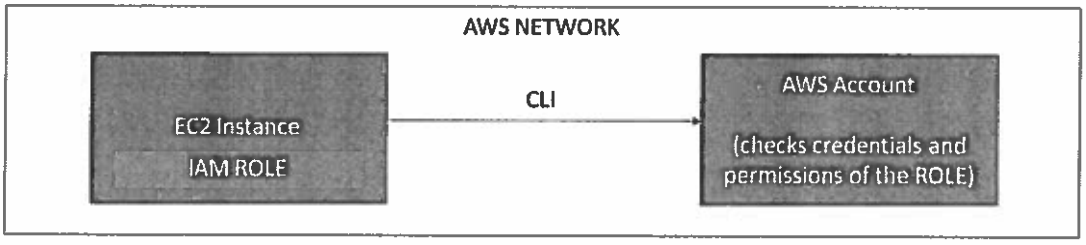
- If the EC2 is compromised, so is your personal account
- If the EC2 is shared, other people may perform AWS actions while impersonating you

- For EC2, there's a better way... it's called AWS IAM Roles

5

AWS CLI ON EC2... THE RIGHT WAY

- IAM Roles can be attached to EC2 instances
- IAM Roles can come with a policy authorizing exactly what the EC2 instance should be able to do

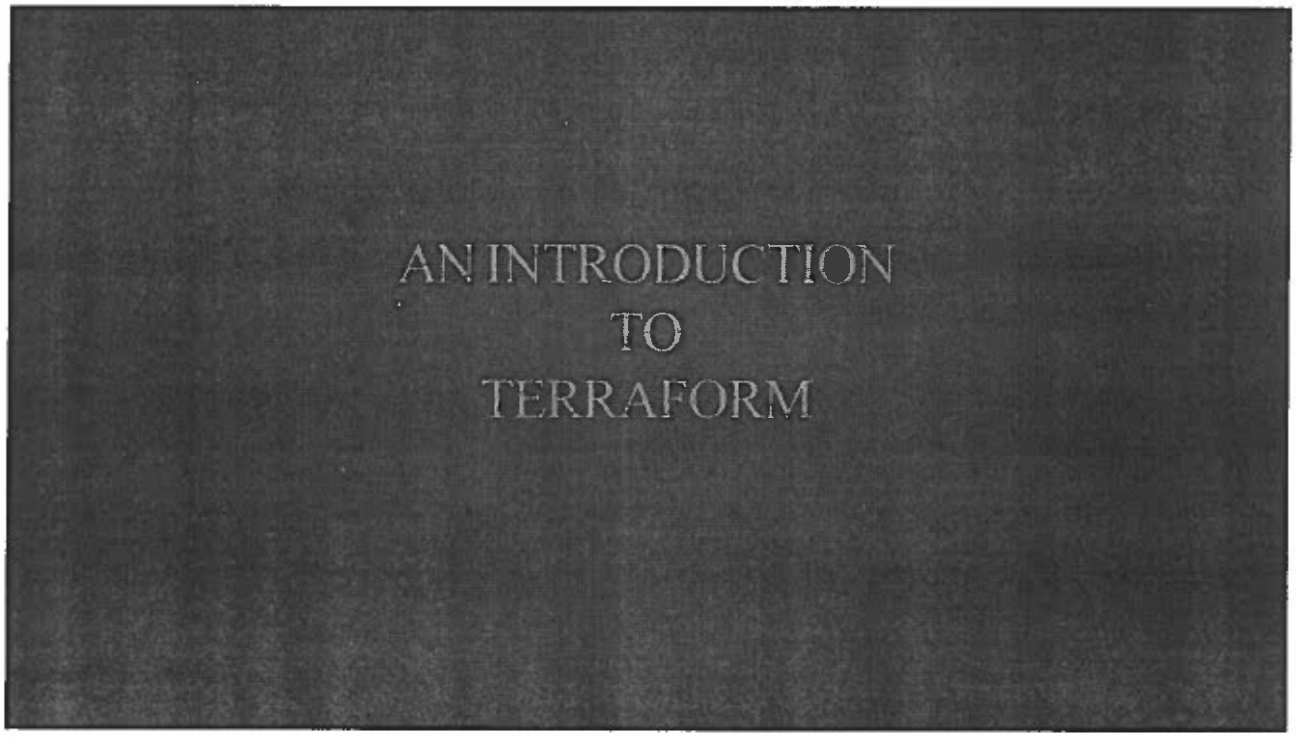


- EC2 Instances can then use these profiles automatically without any additional configurations
- This is the best practice on AWS and you should 100% do this.

6

AWS-7

01-12-2019



1

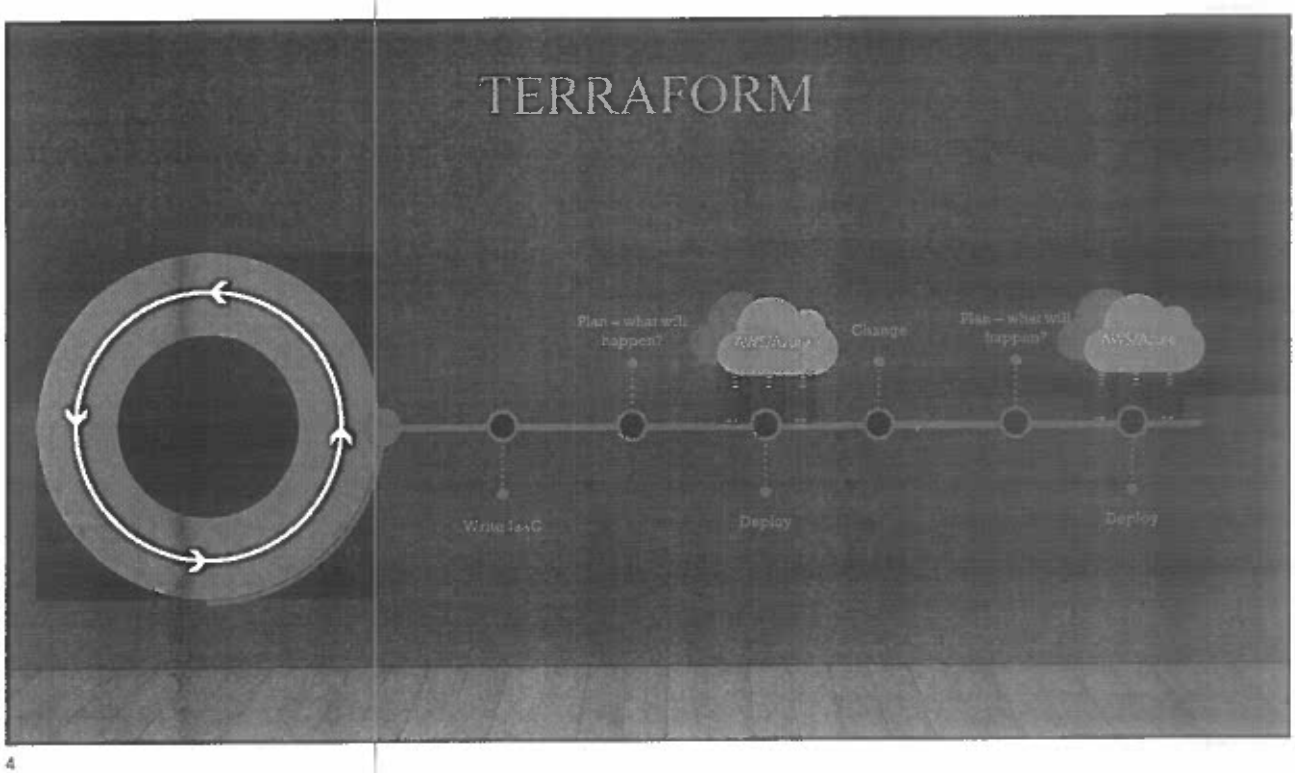
TERRAFORM

- Open Source
- Created by Hashicorp (vagrant, consul, packer, vault)
- Started in 2014
- Written in Go Pluggable

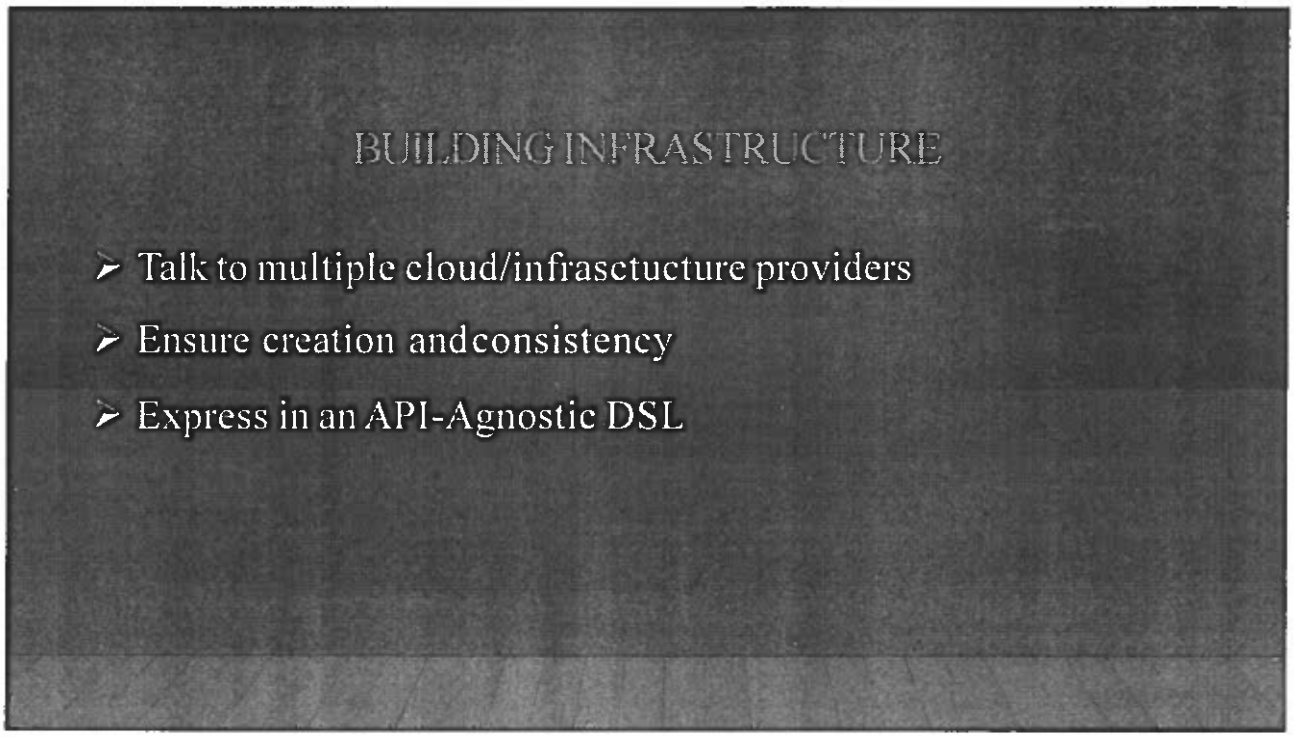
Terraform original goal

- Terraform is a tool to *Build*, *Change*, and *Version Control* your infrastructure.
- Write, *plan* and create infrastructure as code
- Same workflow for all deployment scenarios

3


TERRAFORM
Plan – what will happen?
Change
Plan – what will happen?
Write IaaC
Deploy
Deploy

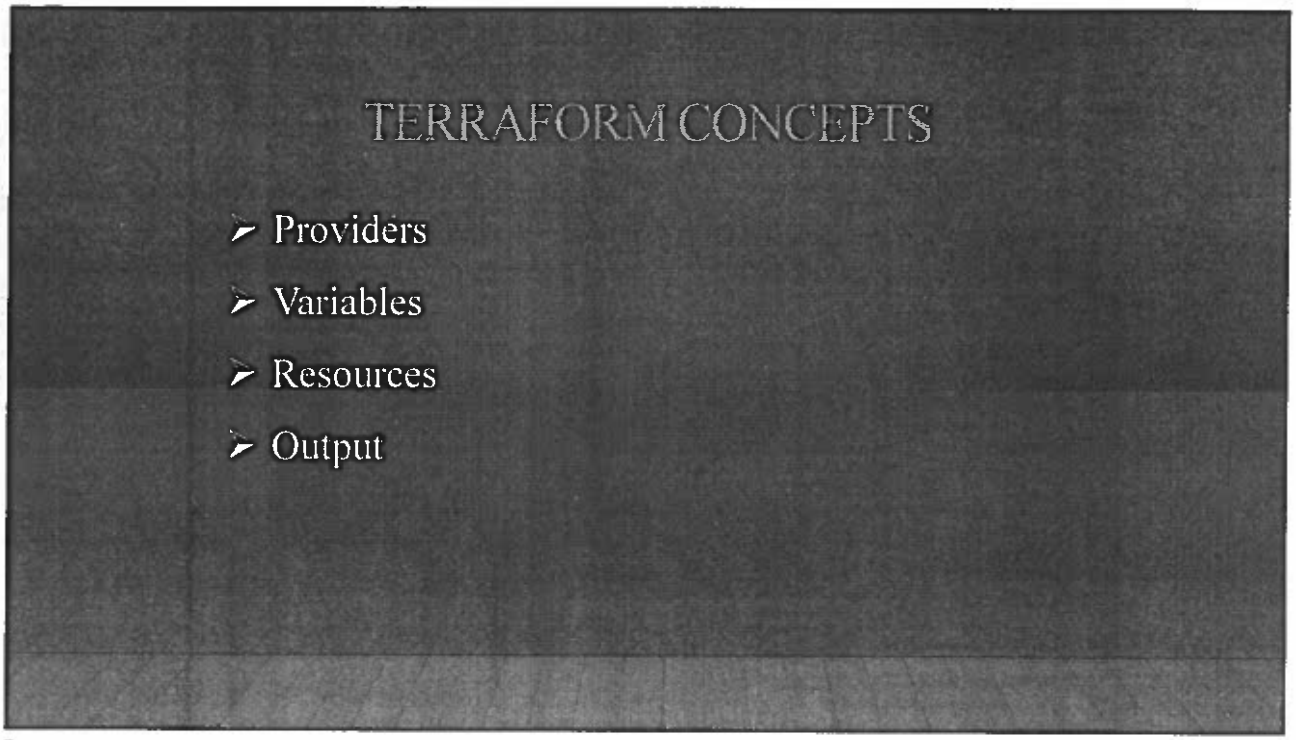
4


BUILDING INFRASTRUCTURE
Talk to multiple cloud/infrasctucture providers
Ensure creation andconsistency
Express in an API-Agnostic DSL

5

CHANGE INFRASTRUCTURE

- Apply incremental changes
- Destroy when needed
- Preview changes
- Scale easily


TERRAFORM CONCEPTS
Providers
Variables
Resources
Output
7

PROVIDERS

- Configuration of a resource provider
- e.g.: cloud (AWS/Azure), dns provider, ...
- Often require URL, API Key, ...

8

TERRAFORM - VARIABLES

- Terraform takes variables as input
- Typed variables
- JSON, HCL or CLI

RESOURCES

- The most essential components of configuration files are 'resources'.
- We declare the type of resource and all of the resource specific settings

The general syntax for a Terraform resource is:

```
resource "<PROVIDER>_<TYPE>" "<NAME>" {
        [CONFIG ...]
}
```

Terraform - Output

- Output specific text, ip addresses, ...
- Console or JSON

PLAN AND APPLY

- terraform plan
  - Creates a plan of changes required
  - Does nothing to the infra
- terraform apply
  - Applies a plan, make all the changes
  - Can make the plan before if needed

12

Terraform - Statefile

- Current know state of the infra
- Stored in file or externally
- Locking

TERRAFORM - DESTROY

- terraform destroy
- Delete all the resources
- resources can be "protected" in config

LET'S SEE TERRAFORM IN ACTION

- Download Terraform and Extract it
- $ unzip terraform_0.11.1_linux_amd64.zip -d /usr/bin
- $ terraform –v
- $ mkdir terraform-templates && cd terraform-templates
- $ touch template.tf
- $ terraform apply

15

INITIALISE A TERRAFORM

- $ terraform init
- A typical Terraform module will have the following structure:

```
my-terraform-files
|
└──my-terraform-module
|   |   main.tf
|   |   variables.tf
|   |   outputs.tf
|
└──my-other-terraform-module
|   |   main.tf
|   |   variables.tf
|   |   outputs.tf
```


16

PROVIDER

- We must specify which IaaS provider we are using
- It downloads the plugins necessary to read from and write to the hosting service.
- all configuration files (.tf) in the directory are loaded when run running commands

main.tf

```
provider "aws" {
 region = "eu-west-1"
 version = "~> 1.19"
 access_key = "${var.aws_access_key}"
 secret_key = "${var.aws_secret_key}"
}
```

17

VARIABLES

- Declare values essential to our resource provisioning such as instance sizings, autoscale desired capacities …
- The possible variable types are string (default type), list and map

```
variable "aws_access_key" {
 description = "The AWS access key."
 default     = ""
}

variable "aws_secret_key" {
 description = "The AWS secret key."
 default     = ""
}
```


18

RESOURCES

- The most essential components of configuration files are 'resources'.
- We declare the type of resource and all of the resource specific settings

```
resource "aws_volume_attachment" "this_ec2" {
  device_name = "/dev/sdh"
  volume_id   = "${aws_ebs_volume.this.id}"
  instance_id = "${module.ec2.id[0]}"
}

resource "aws_ebs_volume" "this" {
  availability_zone = "${module.ec2.availability_zone[0]}"
  size              = 1
}
```

19

DATA SOURCES

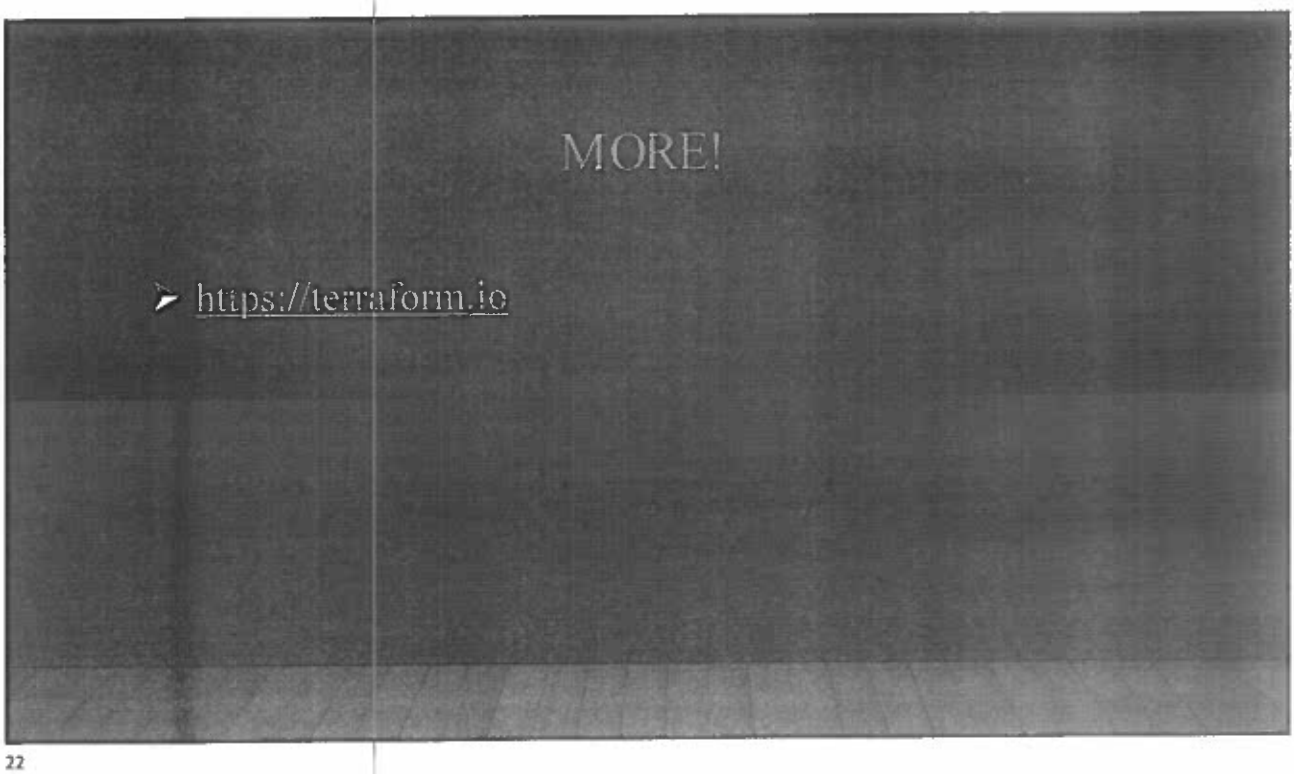
- It enables us to reference resources that should already exist
- allowing us to extract information from them to feed into new resources etc

```
data "aws_ami" "amazon_linux" {
  most_recent = true

  filter {
    name = "name"

    values = [
      "amzn-ami-hvm-*-x86_64-gp2"
    ]
  }

  filter {
    name = "owner-alias"

    values = [
      "amazon",
    ]
  }
}
```

20

PLAN AND APPLY

- $ terraform plan
- $ terraform apply
- Terraform store the state of the deployed resources (.tfstate) file
- $ terraform destroy



22