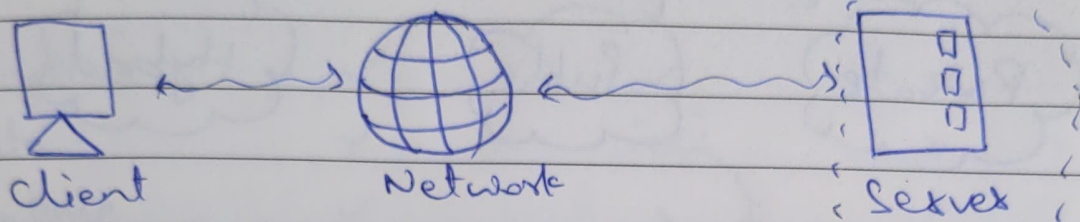
Cloud Practitioner

Date:

# Basics




IP.

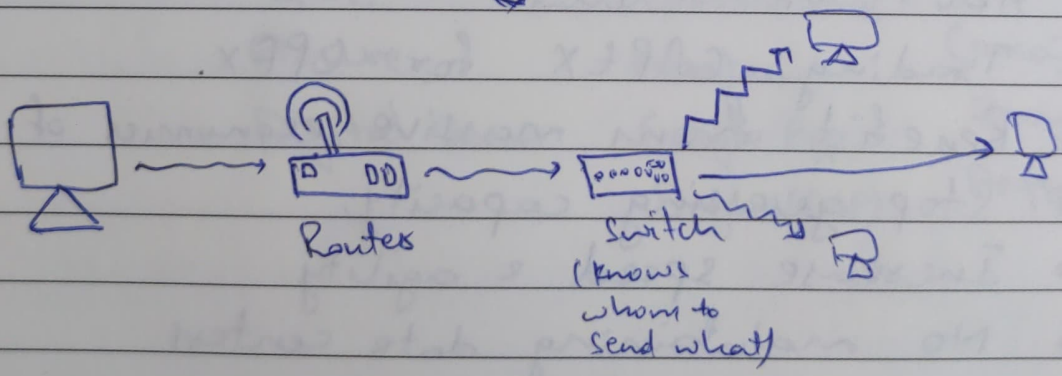
IP

Compute: CPU
Memory: RAM
=
Brain

Storage
data
↓
files

Database
↓
Data in structured way

Network
↓
Routers, switches, PNS




Traditionally setting up own servers was messy & lot of effort

⇓

Saviour
Cloud
Computing

Date:

## Deployment Models in Cloud

- Private
- Public
- Hybrid

## Characteristics of Cloud.

* On-demand Self-Service
* Broad Network Access
* Multi-tenancy & Resource Pooling
* Rapid elasticity & scalability
* Measured Service.

## Adv's of Cloud

* Trading CAPEX for OPEX
* Benefit from massive economies of scale
* Stop guessing capacity.
* Increase speed & agility
* No maintaining data centers
* Go global in minutes.

Date:

# Types of Cloud Computing

| On-Prem | IAAS | PAAS | SAAS |
|---|---|---|---|
| Applications | ✓ | ✓ | ✗ |
| Data | ✓ | ✓ | ✗ |
| Runtime | ✓ | ✗ | ✗ |
| Middleware | ✓ | ✗ | ✗ |
| O/S | ✓ | ✗ | ✗ |
| Virtualization | ✗ | ✗ | ✗ |
| Servers | ✗ | ✗ | ✗ |
| Storage | ✗ | ✗ | ✗ |
| Networking | ✗ | ✗ | ✗ |
| | ⇓ | ⇓ | ⇓ |
| | Amazon EC2, GCP, Azure, Digital Ocean | Elastic Beanstalk (AWS), Heroku, Google App Engine (GCP) | Rekognition (AWS), Gmail, Zoom, Dropbox. |

are to be managed by us

Managed by AWS.

Date:

## Pricing in AWS

Compute → Pay for compute time

Storage → Pay for data stored in cloud

Data Transfer → Pay for data going OUT of cloud.

Data transfer IN is free

## AWS Region

* Cluster of data centers
* Ex:- US-east-1, eu-west-3, ..
* Most AWS services are region scoped

## How to choose?

* Compliance
* Proximity
* Available services
* Pricing

Date:

## Availability Zones (AZ)

* Each region has many AZ's (Usually 3)
Ex: ap-southeast-2a, 2b, 2c.

* Each AZ is one or more discrete data centers with redundant power, networking & connectivity.
* Isolated from disasters
* Connected with high bandwidth, ultra-low latency networking.

## Points of Presence (Edge location)

→ 216 PoP → 84 cities → 42 countries
→ Content is delivered to end users with low latency.

## Shared Responsibility Model

| Customer | AWS |
| --- | --- |
| ⇓ | ⇓ |
| Responsible for security in the cloud | Responsible for security of the cloud. |

Date:

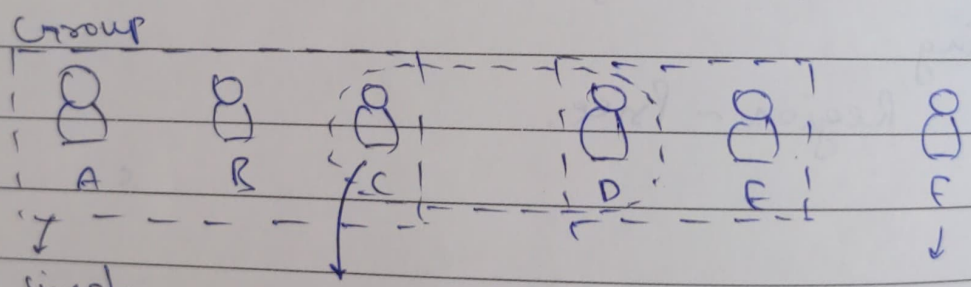
Date:

## Points of Presence (Edge locations)

* 216 POP → 84 cities → 42 countries
* lower latency.

## IAM → Global Service

* Identity & Access Management
* Root Account is created by default, but it shouldn't be used or shared
* We create users, they can be grouped.
* Groups cannot contain groups.




not best practice, but doable.

Date:

Users and Groups can be given permissions using JSON documents called Policies.

Ex:-

```
{
    "version" : ~
    "Statement" : [
        {
            "Effect":
            "Action":
            "Resource":
        }
        :
        ]
    }
}
```

We use Least Previlige Principle.

Giving users or groups the permissions which are required only.

Date:

# In IAM Console

* We create a admin user and do everything with it → It is best practice not to use root user.

* If we give a group some IAM Policy, all the users inherit that.

* For a user who is not in any group, we have inline policy.

* IAM Policy Structure:-

version, Id (optional), Statement

statement id → Sid (optional)
Effect : Allow / Deny
Principal : account / user / role
Action : List of actions
Resource : List of resources
Condition (optional) : when this policy takes effect.
↓
if not given
→ always.

Date:

# Protecting Users and Groups

## IAM - Password Policy

* Strong Password = high security
* We can setup our own password policy:
  - → min length
  - → Require specific chars
    - Upper case
    - lower case
    - number
    - non alpha-numeric
  - → Allowing users to change their passwords
  - → Password expiration.
  - → Prevent Password reuse.

## Multi-factor Authentication (MFA)

||

Password You Know + Security Device You Own

- Virtual MFA Device
  - Google Authenticator
  - Authy *
- U2F Security key
  - Yubikey
- Hardware key fob
  - Gemalto
- Hardware key fob for AWS Gov Cloud
  - SurePass-ID.

Date:

## We can access AWS by

(i) Management Console (Pwd + MFA)
(ii) CLI (access key)
(iii) SDK (access key)

* Access keys are generated through AWS console
* Users manage their own access keys
* Access keys are secret & No sharing

Access key ID ~ Username
Secret Access Key ~ Password.

Note:- Never use root account to create access keys.

In the terminal do → AWS configure.

AWS CloudShell → Not available in all regions.

terminal in cloud
persistence these
Download & Upload files
By default region is what we are using.

Date:

# IAM Roles for Services

* Some AWS services will perform actions on our behalf
* Hence they need permissions

⇓

Given by IAM Roles

Not to be used by physical users

* Common roles
  - → EC2 instance roles
  - → Lambda function roles
  - → Roles for Cloud formation.

Roles get the credentials for a short amount of time to get the work done.

Date:

# IAM Security Tools

* IAM Credential Report (account-level)
  - → report containing all users and status of their credentials.

Direct download button in the option.

* IAM Access Advisor (user-level)
  - → permissions granted to user and when they were last used.
  - → can be used to apply least previlge principal.

Available in Users → Access Advisor

## Shared Responsibility Model for IAM

| AWS | (Us) |
| --- | --- |
| ⇓ | ⇓ |
| * Infrastructure (Global network security) | * Users, Groups, Roles, Policies mgmt & monitoring |
| * Config & vulnerability analysis | * Enable MFA, rotate keys |
| * Compliance validation | * Use IAM Tools to apply appr. permissions & analyze and review them |

Date:

# Bills

* We can setup billing constraints on root user & IAM users seperately.

* All our monthly bills can be viewed in bills section.

* The free-tier window will show our status with our account.

* Creating a budget
  ↓
  Cost Budget
  ↓
  Some limit (10$ for the course)
  ↓
  Setup Budget alerts.
  (Two types
  ↙ ↘
  Actual forecasted)

By default we get mail, but we can add actions too.

Date:

# EC2 (More like virtual machines)

* Elastic Compute Cloud
* Infrastructure as a service.
* Capability of
  - Renting virtual machines (EC2)
  - Storing data on virtual drives (EBS)
  - Distributing load across machines (ELB)
  - Scaling services (ASG)

## Config options.

→ OS : Linux, Windows, MacOS
→ Compute power & cores (CPU)
→ How much RAM.
→ How much Storage space
  → Network Attached (EBS & EFS)
  → Hardware (EC2 instance store)
→ Network Card : Speed, public IP.
→ Firewall rules : security gp
→ Bootstrap Script (configure on first launch) : EC2 user data :

- used to automate boot tasks like installing s/w's, downloading some files etc.
- ran only once, with pseudo root rights.

Date:

While creating an EC2

↓

Name,

OS,

Instance type (t2-micro)

Key-Pair (Used for SSH)

↳ (.pem for macOS, Windows 10

.ppk for windows 7, 8 ..)

Network settings

↳ Delete on termination is by default true.

Advance details

↳ user data. → remember?

of course you don't

&

script to run at start of instance

⇓

Summary

When we stop & restart an instance

it's Public IPv4 might change

while

Private IPv4 stays intact

Date:

# EC2 Instance Types

* There are 7 types → some link.
* Naming Convention.

[Instance class] [generation]. [size]

Ex:- m5 . 2xlarge
m6 . 3x large
(gen increases as hardware gets better)

## (i) General Purpose

→ Great for diversity of workloads
- web servers, code repos.
→ Balance between
* Compute
* Memory
* Networking.

Date:

## (ii) Compute Optimized.

→ Great for compute intensive tasks that require high performant processors

- Batch processing
- Media transcoding
- High Performance web servers
- High Performance Computing (HPC)
- Scientific modeling & ML
- Dedicated Gaming servers.

(Currently all c name servers)

## (iii) Memory Optimized.

→ fast performance for workloads that process large data sets in memory

- High perf. rel/non-rel dbs
- Distributed web scale cache stores
- In memory dbs optimized for BI (Business Intelligence)
- Appln performing real-time processing of big unstructured data.

(R stuff, or & for RAM)

Date:

(iv) Storage optimized instances.

* Great for storage intensive tasks that require high, sequential read & write access to large data sets on local storage.
  - High frequency OLTP systems.
  - Relational & NoSQL db's
  - Cache, for in-memory db's (Redis)
  - Data Warehouse apps
  - Distributed file systems.

(I, D, H)

# SECURITY GROUPS

* fundamentals of Network Security in AWS.
* Controls traffic (both in & out)

* Only contain "allow" rules
* Can be referenced by IP or by other security gps

⇓

Nested are allowed.

Date:

So Basically,

they are ~~like~~ firewall to our EC2 instance
They regulate
- Access to ports
- Authorized IP Ranges - $V_4$ & $V_6$
- control of inbound & outbound network.

Good to knows:-

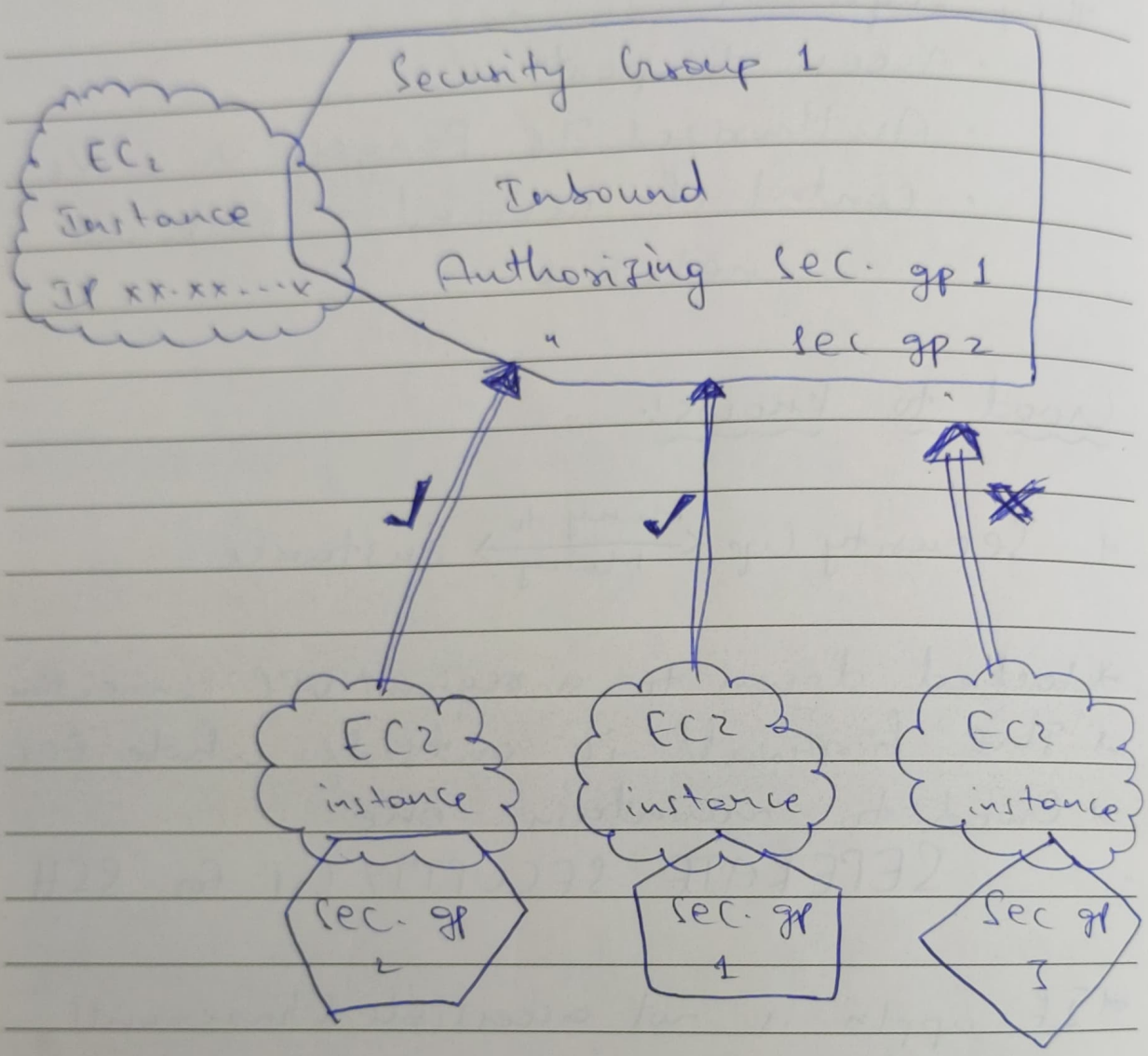
→ Security Grp $\xleftrightarrow[\text{Many}]{\text{Many to}}$ Instance

→ locked down to a region/ VPC connection
→ The firewall is outside whole EC2
→ Good to maintain one
SEPERATE SECURITY GP for SSH.

→ If appln is not accessible (time-out) then it's security gp issue.
→ If appln give "connection refused" then it's internal appln error.
→ All inbound traffic is blocked by default
→ All outbound traffic is authorized by default.

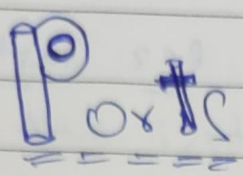
Date:

One cute feature (which allows us to forget about IP's)




So if we authorize sec. gp's instead of IP's then all the instances who are attached with those sec. gp's can send traffic

Date:




22 → SSH (Secure Shell) - log into Linux instance

21 → FTP (File Transfer Protocol) - upload files into a file share.

22 → SFTP (Secure File Transfer Protocol) - upload files using SSH.

80 → HTTP - access unsecured websites

443 → HTTPS - access secured websites

3389 → RDP (Remote Desktop Protocol) - log into a windows instance.

0.0.0.0/0 means anywhere

If it's a timeout then the problem is 100% security

Date:

# How to SSH into EC2 instance.

1) Have a .pem file
2) Get Public IPv4 address of the instance
(Also ensure 22 port is enabled in Sec. gp)
3) Go to where .pem file is and do

```
ssh -i ~.pem ec2-user@ public IPv4
```

⇓

might give unprotected private key, just change permissions.

⇓

```
chmod 0400 ~.pem
```

4) "exit" to disconnect

## Alternative

## EC2 - Instance Connect

* Browser Based SSH connect.
* Does all the above steps for us and directly opens SSH in new tab.

Date:

Steps

1. Click on the instance.
2. You'll find an option "connect".
3. In that "EC2 Instance connect"
4. Just click "connect"

(Sometimes just IPv4 inbound traffic allow may not be enough, we need to give IPV6 inbound allow too from anywhere)

{Never Ever Ever enter secret keys into EC2 Instance Console as other who can connect using our account can retreive it.}

↓

Cannot do aws configure!!!

↓

Then How?

↓

Attach an IAM Role to the EC2 Instance with sec. permissions.

↓

Voila!!!!!

All AWS commands work!!!

Date:

# EC2 Instance Purchasing Options

① On Demand Instances
- short workload
- Predictable Pricing
- Pay by second.

② Reserved (1 & 3 Years)
- Reserved Instances
  - long workloads
- Convertible Reserved Instances
  - long workloads with flexible instances.

③ Savings Plans (1 & 3 Years)
- commitment to an amount of usage
- long workload.

④ Spot Instances
- short workload
- cheap
- can lose instances (less reliable)

⑤ Dedicated Hosts.
- books an entire physical server
- controls instance placement

Date:

⑥ Dedicated Instances
- no other customer will share your hardware.

⑦ Capacity Reservations
- reserve capacity in a specific AZ for any duration.

for more details look into slides.
(Hotel Analogy helps).

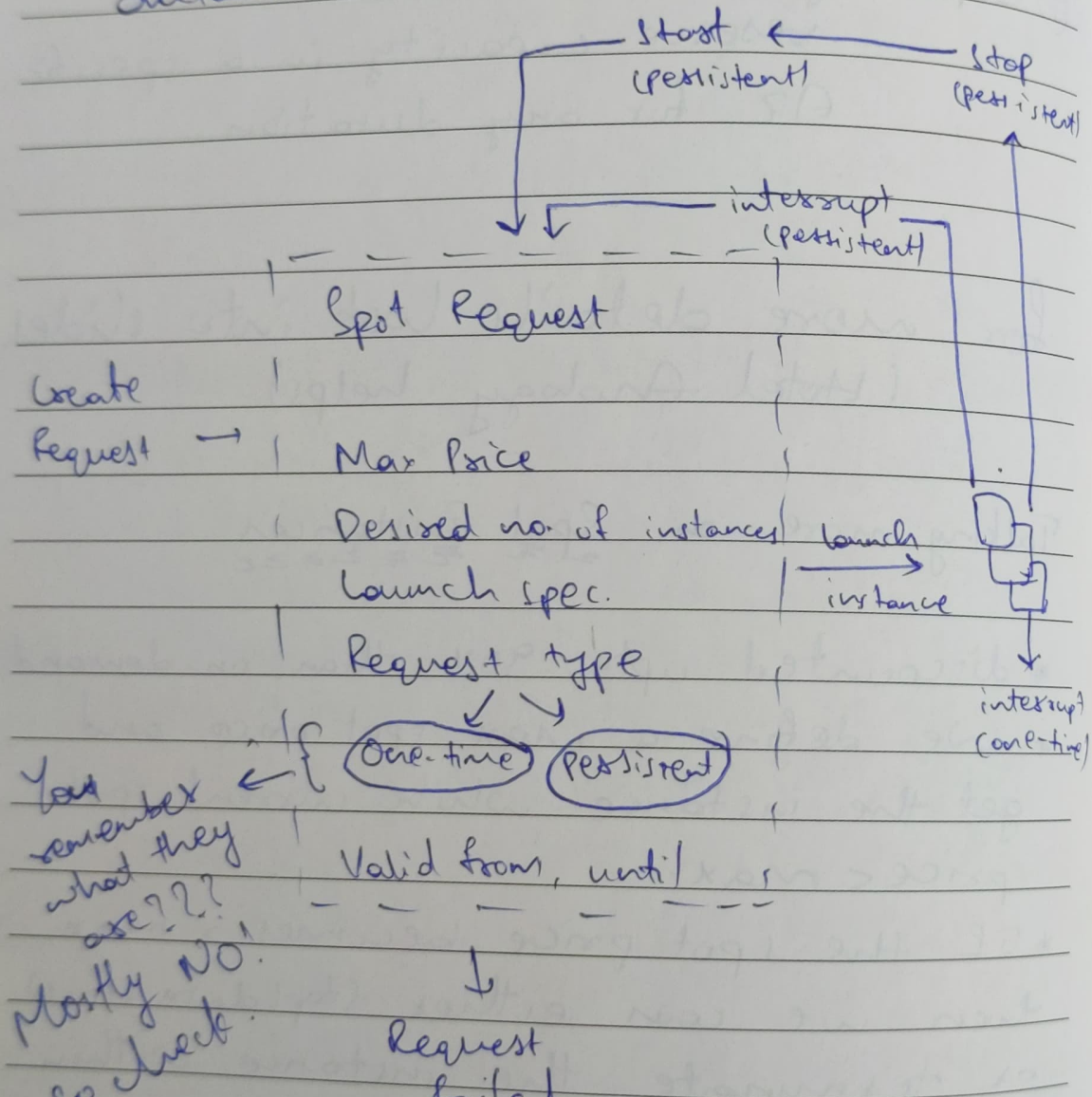
Talking more on Spot instances

* discounted upto 90% than on-demand
* We define a max spot price and get the instance while current spot price < max.
* If the spot price becomes > max, then we can either stop (data stays) or terminate the instance within 2 min grace period.

Spot Block (Stopping from Dec 2022)
- we can block a spot for certain time
- rare reclaims occur.

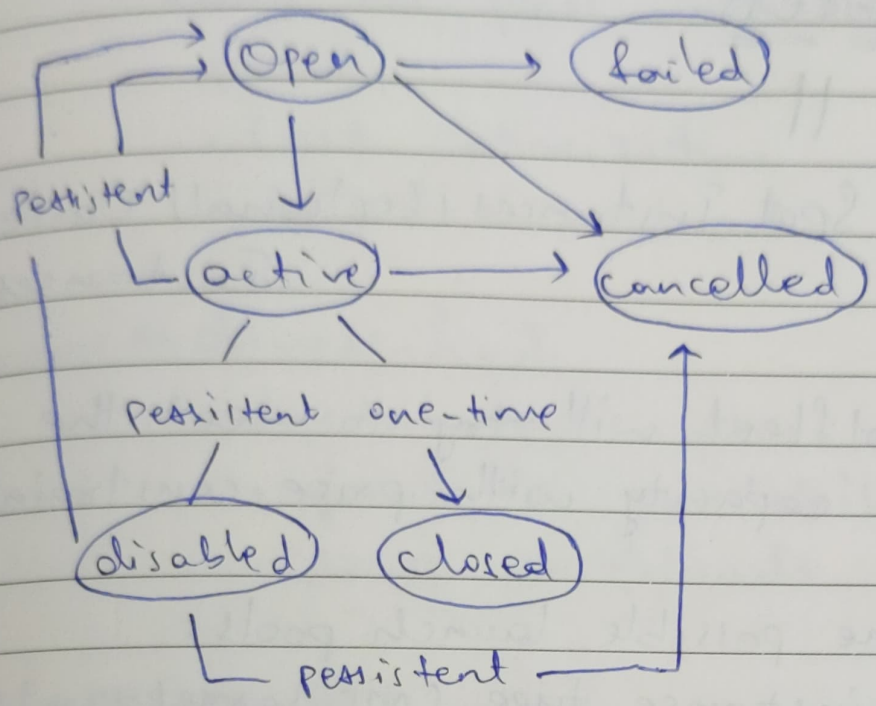
Date:

* Better for batch jobs, data analysis or workloads that are resilient to failures.
* Not great for critical jobs or databases.



Date:

Create Request




## How to terminate Spot Instances?

* You can only cancel when they are in open, active or disabled.
* Cancelling a spot request does not terminate instances

* We must first stop the Spot Request, and then terminate the associated Spot instances.

(If we don not stop the request and terminate, then they will again be instantiated due to the request).

Date:

# Spot fleets.

||

Set of Spot Instances + (optional) On-Demand Instances

* The Spot fleet will try to meet the target capacity with price constraints

- Define possible launch pools:
  - instance type (m5-large)
  - OS
  - Availability Zone

- Can have multiple launch pools, so that the fleet can choose

- Spot fleet stops launching instances when reaching capacity or max cost.

The Spot fleet intelligently decides and follows the following strategies to allocate Spot instances

Date:

- lowest Price
  - from the pool with the lowest price
    - Cost optimization
    - Short workload.

→ diversified
  - distributed across all pools
    - great for availability
    - long workloads

- Capacity Optimized
  - pool with optimal capacity for no. of instances.

It automatically requests instances with the lowest price.

## Shared Responsibility Model for EC2

| AWS | (CUS) |
| --- | --- |
| ⇓ | ⇓ |
| * Infra (Global network security) | * Security Group Rules |
| * Isolation on physical hosts | * OS patches & updates |
| * Replacing faulty h/w | * S/w installed on OS |
| * Compliance validation. | * IAM Roles assigned to EC2 & IAM user access mgmt |
| | * Data security on our instance |

Date:

# Private vs Public IP

* Networking has two sorts of IP

**IPV4**

[0-255].[0-255].[0-255].[0-255]

Most common format

**IPV6**

Hexadecimal stuff

7 numbers sep by :

Solves problems for IOT

**Public IP**

* machine can be identified on internet (www)
* Unique across web
* can be geo-located

**Private IP**

* identifiable only in private network
* unique across the private network.
* Two diff. private net. can have same IP's
* Connect to WWW using NAT + Internet gate way
* Only specified IP's can be used as private.

# EBS (Elastic Block Store)

* EBS Volume → network drive we can attach to an instance while it runs.

* Instances can persist data even after termination.
* One instance ↔ one EBS volume (at CP level)
* Bound to specific AZ.

{{

Network USB Stick.

So,

* It's a network drive (i.e. not physical drive)
  → there might be latency as network.
  → detached & attached easily.
* It's locked to AZ
  → EBS created in us-east-1a cant be attached to us-east-1b
  → can be done using snapshot though.

Date:

* Have a provisioned capacity (GBS, IOPS)
  - Billed for provisioned capacity.
  - increase capacity over time.

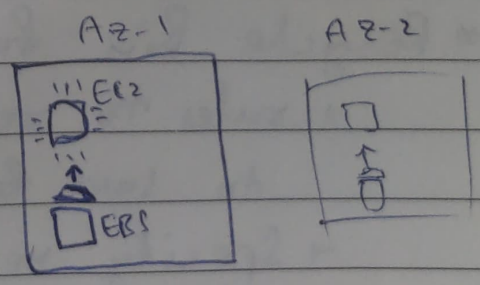
## Delete on Termination

* Controls EBS behaviour when EC2 instance terminates.
  - By default,
    - Root EBS volume is deleted
    - other attached EBS volumes are not

can be controlled by AWS Console/CLI

* Use case :- Preserve the root volume when instance is terminated.

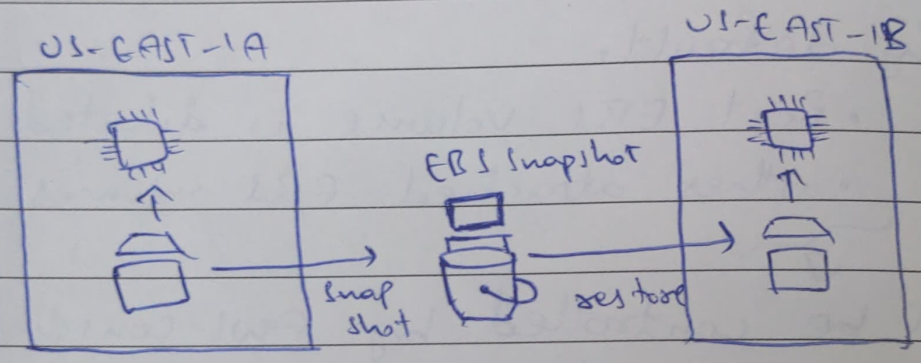
There is a feature called EBS Multi-Attach, not in scope of CP though.




Date:

# EBS Snapshots

* Make a backup (snapshot) of our EBS volume at a point in time.
* Not necessary to detach before snapshotting but recommended.
* Can copy snapshots across AZ or Region




## features

* EBS Snapshot Archive
  - Move snapshot to "archive tier" that is 75% cheaper.
  - Takes within 24-72 hours to retrieve from archive.

* Recycle Bin for EBS Snapshots
  - rules to retain deleted snapshots to save from accidental deletion
  - Specify retention (from 1 day to 1 year)

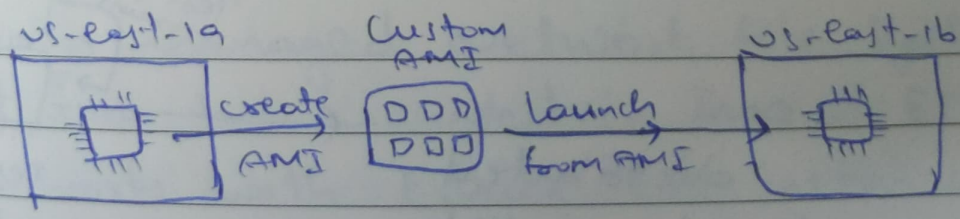
Date:

# AMI

* Amazon Machine Image
* Basically customization of EC2 instance
* Built for specific region (can be copied across regions)
* We can launch EC2 from
  - Public AMI → AWS Provided
  - Own AMI → we make them & maintain
  - AWS Marketplace AMI → an AMI someone else made & we bought

## Custom AMI Process

* Start & customize an EC2 instance
* Stop the instance (for data integrity)
* Build an AMI (creates EBS snapshots also)
* Launch instances using this AMI.




Date:

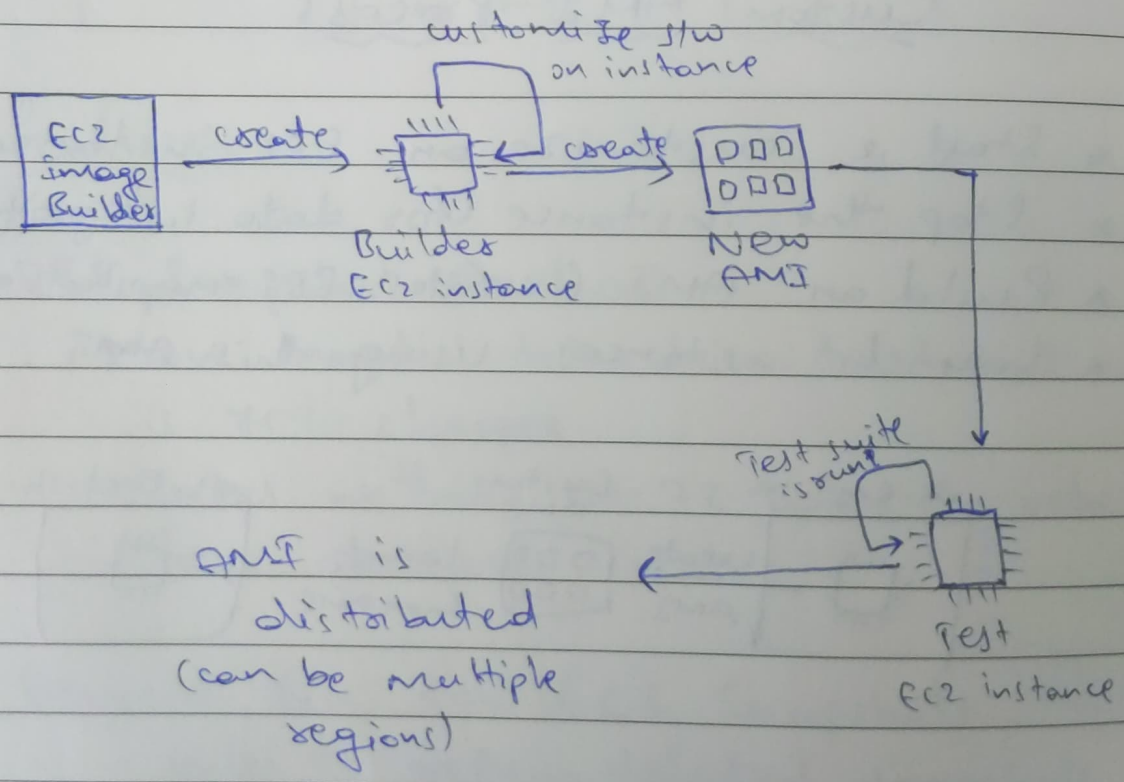
# EC2 Image Builder

* Used to automate the creation of virtual machines or containers images.

⇓

Automate the creation, maintain, validate & test EC2 AMI's.

* Can be run on a schedule (weekly, whenever packages are updated, etc..)
* free service (only pay for underlying resource usage)




Date:

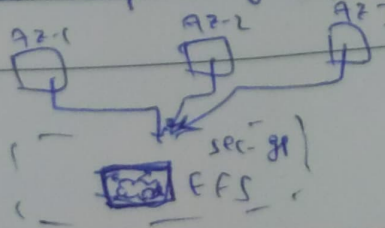
# EC2 Instance Store

* EBS volumes have good but "limited" perf.
* If we want high perf hardware disk, that is directly (physically) attached to our instances hardware, choose EC2 Instance Store.

* Better I/O Performance
* It's ephemeral storage, as it is directly linked to our instance, if instance stops data goes "poof!"
* Good for buffer, cache, scratch data, temp content
* Risk of dataloss if hardware fails.
* Backups & replications are our resp.

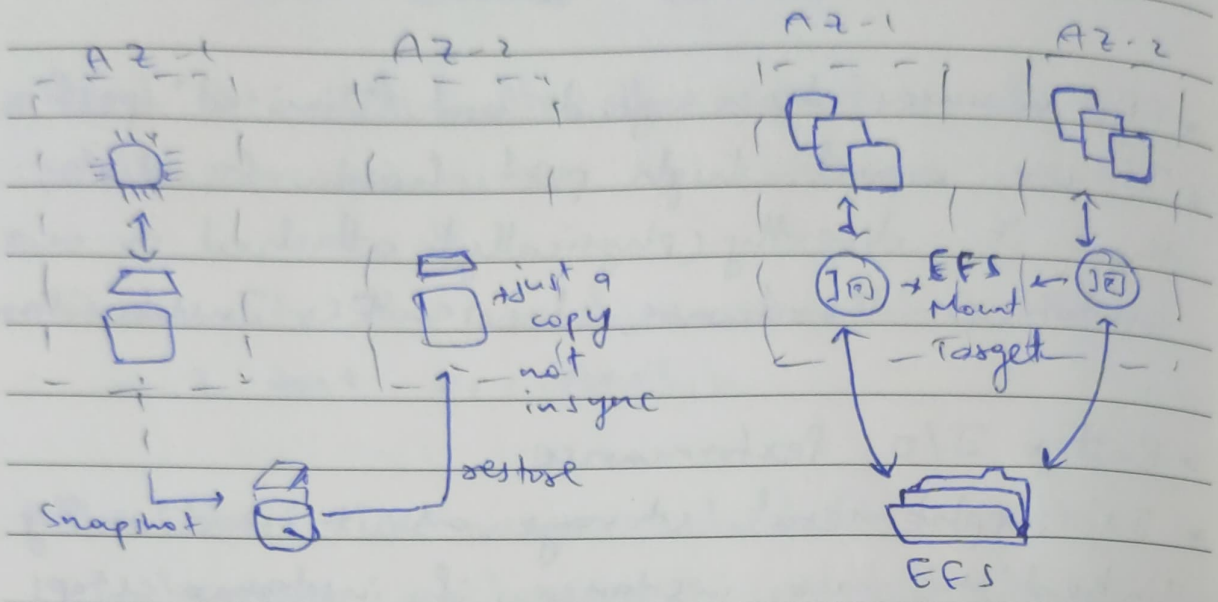
# EFS - Elastic File System

* It is a managed Network file System that can be mounted on 100s of EC2's
* EFS works with Linux EC2 instances in multiple AZ's
* High availability, scalable, 3 times expensive, pay per use, no capacity planning.




Date:

EBS vs EFS




## EFS-IA (Infrequent Access)

* Storage class that is cost optimized for files not accessed every day.
* Upto 92% lower cost compared to standard EFS.
* We can write a lifecycle policy, and EFS automatically moves the files to EFS-IA based on that
* Ex:- move my files to EFS-IA after 60-days non-usage.
* No drawbacks, transparent to applications, good.

Date:

# Shared Responsibility Model

| AWS | (US) |
| --- | --- |
| ⇓ | ⇓ |
| * Infra | * Setting up backup/ snapshot procedures |
| * Replication of data for EBS & EFS | * Setting up encryption |
| * Replacing faulty hardware | * Data being written into drives |
| * Ensuring their employees cannot access your data | * Understanding risk of EC2 instance store |

## Amazon FSx

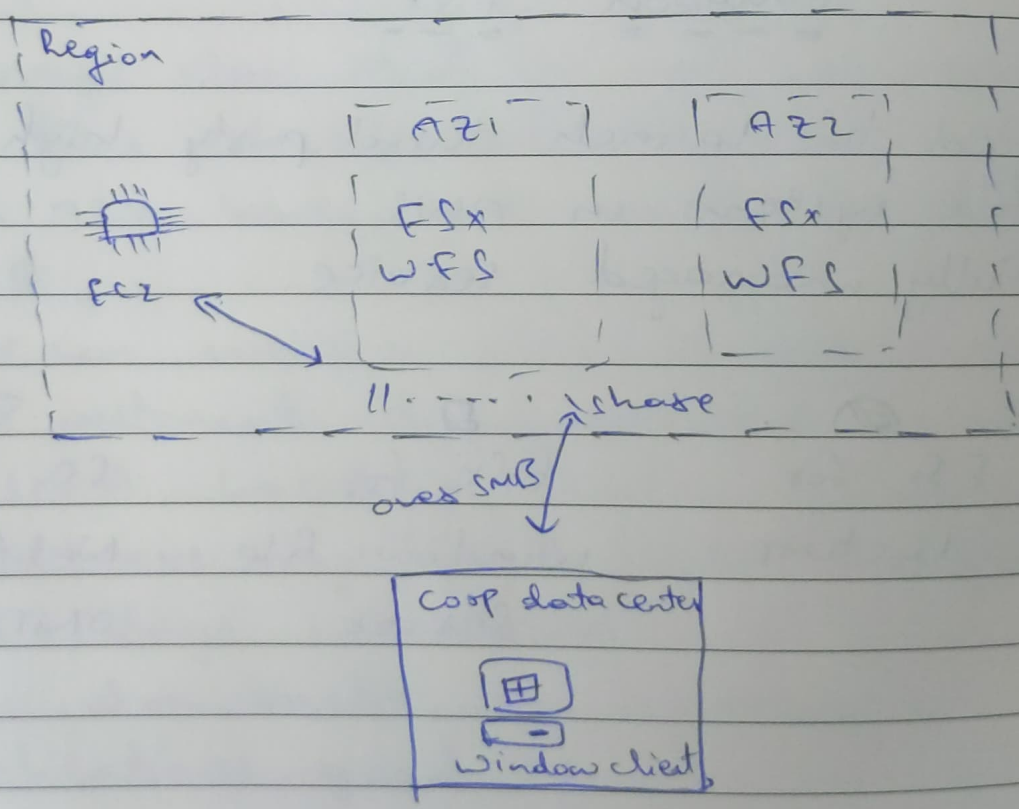
* Used to launch third party high-perf file systems on AWS.
* Fully managed service.

| | | |
| --- | --- | --- |
| FSx for Lustre | FSx for Windows file Servers | FSx for NetApp ONTAP |

Date:

# Amazon FSx for Windows File Server (WFS)

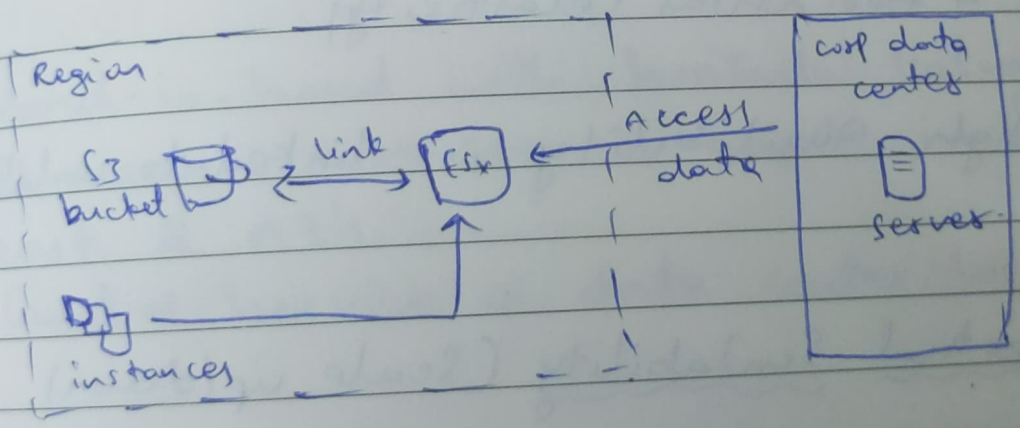
* A fully managed, highly reliable, and scalable Windows Native shared file system.
* Built on Windows File Servers.
* Supports all native protocols like SMB & NTFS. (mounting)
* Integrated with Microsoft Active Directory
* Can be accessed from AWS or our on-prem infra.




Date:

# AMAZON FSx for Lustre

- A fully managed, high performance, scalable file storage for **High** Performance Computing (HPC).
- Lustre = Linux + Clustre.
- AI high compute ⇒ ML, Analytics, Video Processing, Financial Modeling, etc..
- Scales upto 100's GB/s, millions of IOPS, sub ms latensies.



# Elastic Load Balancing
# &
# Auto Scaling Groups

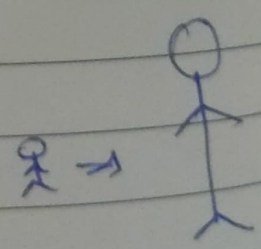
## Scalability & High Availability

* Appln or system can handle loads by adapting.
* Two types
  - → Vertical
  - → Horizontal (elasticity)

* High Availability is linked to scalability

## Vertical Scalability (Scale up/down)

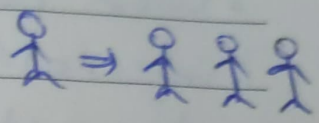
* Increasing the size of instance
* Ex:- t2.micro → t2.large
* Very common for non-distributed systems, like DB.
* There is usually a limit on how much we can vertically scale (hardware limit)



Date:

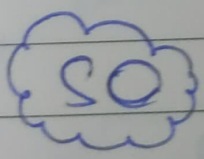
## Horizontal Scalability (Scale in/out)
(ASG & LB)

- Increasing the no. of instances or systems for our appln.
- This implies distributed systems
- Very common for web apps/ modern appl.
- Easy to do with EC2



## High Availability (ASG & LB in multi AZ)

- Hand-in-hand with horizontal scaling
- It means running our appln in at least 2 AZ's.
- Goal ⇒ Survice a data center loss.




Scalability ⇒ ability to accomodate larger load by (scale up/out).

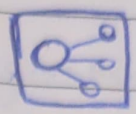
Elasticity ⇒ Some "auto-scaling" ⇒ cloud friendly, optimize costs.

Agility ⇒ resources are click away, we are agile, weeks to minutes.

it's a distraction

Date:

# ELB (Elastic Load Balancing)



* Load Balancers are servers that forward internet traffic to multiple servers downstream.

## Why LB ?

* Spread load
* Expose single point of access (DNS)
* Handle failures
* Do regular health checks.
* Provide SSL termination.
* High availability across zones

## Why ELB?

* Managed load Balancer
  AWS
  - gaurentees it works
  - takes care of upgrades, maintance & highavailability
  - we need to set only a few config stuff.

Date:

* If we setup our own LB, it will cost less but more effort.

* 3 types of LB's by AWS:

* Application Load Balancer → Layer 7 (HTTP/HTTPS only)

* Network Load Balancer → Layer 4 (ultra-high perf, allows for TCP)

* Classic Load Balancer (slowly retiring) → Layer 4 & 7.

There is Gateway Load Balancer but advanced

Hands-On

Internet-facing / Internal, IPV4/blah, Sec. gp, Network Mapping (AZ's), Target group
↓
(Group of instances)

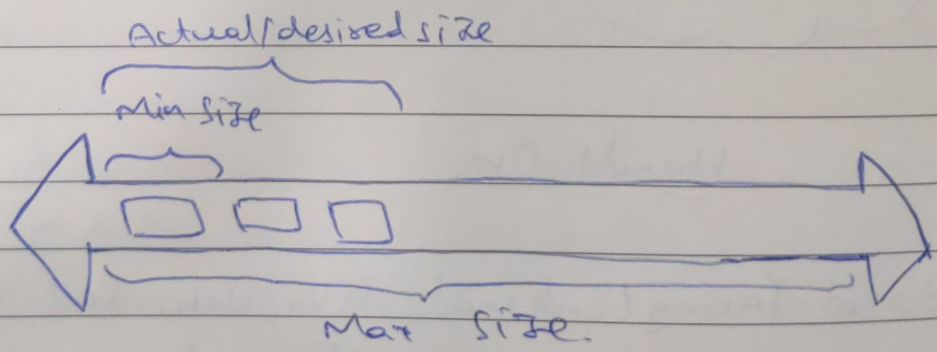
Date:

# Auto Scaling Group

* In real life, load on our websites keep changing.
* In cloud we can create & get rid of instances easily.
* Goal of ASG:
  - scale out when load ⇑
  - scale in when load ⇓
  - Ensure min & max no. of machines running.
  - Automatically register new instances to Load Balancer
  - Replace unhealthy instances.

* Cost Saving: as it runs on optimal capacity.

In AWS.




PS:- Load Balancer registers and unregisters instances accordingly.

Date:

Hands-on

launch template (like a AMI), VPC, AZ's & subnets, Load Balancer, health check, size (min, max, desired), Policy

## Scaling Strategies

* Manual Scaling :- Updating size manually
* Dynamic Scaling :- Respond to demand.

- Simple / Step Scaling
  → give a condn & what to do.
  Ex:- when CPU > 70%, add 2 units
  when CPU < 30%, remove 1 unit.

- Target tracking scaling
  → easy policy.
  Ex:- average CPU usage must be 40%.
  (ASG automatically does stuff).

- Schedule scaling
  → give time & what to do, gen dep on patterns.
  Date:
  Ex:- inc. min cap to 10 at 5 pm friday.

* Predictive Scaling
  - Uses ML to predict future traffic
  - Automatically provisions right no. of instances in advance.

Useful when the load has predictable time-based patterns.

# S3 (Simple Storage Service)

* One of the main building blocks of AWS.
* It is "infinitely scaling" storage.

* Many websites & AWS services use Amazon S3.

## Use Cases

Backup & Storage, Disaster Recovery, Archive, Hybrid cloud storage, Application hosting, Media hosting, Data lakes & Big data analytics, Software Delivery, Static websites

## Amazon S3 - Buckets

* We store objects (files) in buckets (directories)
* Buckets
  - → must have globally unique name
  - → are defined at region level.
  - → Naming convention
    - No uppercase/underscore, 3-63 long, not an IP, must start with lowercase letter or number, must not prefix xn-- suffix -s3alias.

Date:

# Amazon S3 - Objects

* Objects have key
* key is the FULL path.

Ex:- s3://my-bucket/$\underbrace{\text{my-folder}}_{\text{prefix}}$/$\underbrace{\text{my-file.txt}}_{\text{object name}}$.

key =

* NO concept of "directories" in buckets
  Just keys with very long names containing slashes.

* Object values are the content of the body.
  → Max obj size is 5TB.
  → If uploading > 5GB, must use "multi-part upload".

* Metadata (list of text key/value pairs - system/user metadata)
* Tags (Unicode key-value pair - upto 10)
  - useful for security/lifecycle.

* Version ID (if versioning is enabled)

Date:

Hands-on.

Unique name → create with default.

we cannot access with public URL but can with "pre-signed url."

AMAZon S3 - Security.

* User based
  - IAM Policies - which API calls should be allowed for a specific user from IAM.

* Resource based
  - Bucket Policies
    - → Bucket wide rules from S3 console.
    - → Allows cross-account.
  - Object Access Control List (ACL)
    - → finer grain (can be disabled)
  - Bucket Access Control List (ACL)
    - → less common (can be disabled)

Note:- an IAM principal can access S3 obj if
* the user IAM ALLOW it OR resource policy ALLOW it
* AND there is no explicit DENY.

* Encryption: Encrypt obj in Amazon S3 using keys.

How do S3 Bucket Policies look?
(same as IAM Policies)

* JSON based policies
  → Resources: buckets & objects
  → Effect: Allow/Deny
  → Actions: Set of API to Allow/Deny
  → Principal: Account/User to apply policy to

Uses
→ Grant public access for the bucket

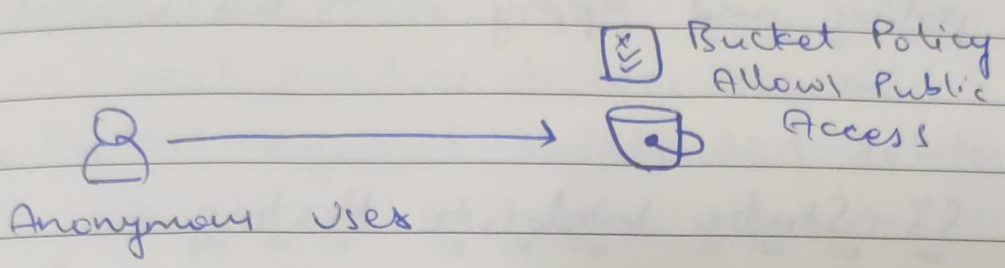
⇓

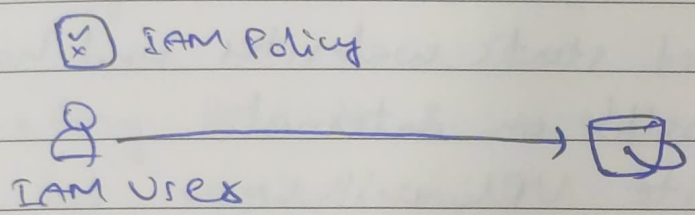
```
{
    "version": "—",
    "statement": [
        {
            "Sid": "PublicRead",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "S3:GetObject"
            ],
            "Resource": [
                "arn:aws:S3:::exampleBuck/*"
            ]
        }
    ]
}
```

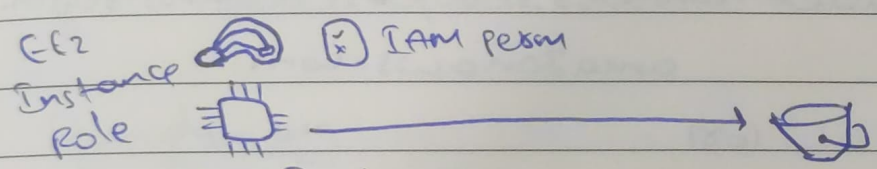
Date:

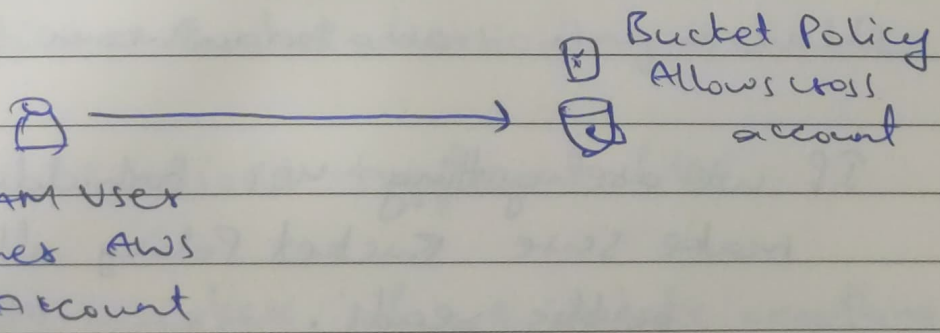
→ force objects to be encrypted at upload
→ Grant access to another account (cross account).

Examples:




* Bucket settings for Block Public Access
→ created by AWS to prevent data leaks
→ If we don't want our S3 to be public, leave these on, even if by accident policy allows, it doesn't gain access.

Date:

## Hands on

* Change public access to enable & use AWS Policy Generator to create bucket policy and apply it to our bucket.

## S3 - Static Website Hosting

* S3 can host static websites and have them accessible on internet
* The website URL will be

http://bucket-name.s3-website-aws-region.amazonaws.com.

(or)

http://bucket-name.s3-website.aws-region.amazonaws.com.

If we are getting 403 forbidden error, make sure Bucket Policy allows public reads.

Hand-on → Properties → Static Website Hosting → enable

# Amazon S3 - Versioning

* We can version our files
* Enabled at bucket level
* Same key overwrite will change the "version" : 1, 2, 3, ...
* It is best practice to version as
  - → Protect against unintended deletes (ability to restore a version)
  - → Easy roll backs to prev. version.

Note:- * Any file that is not versioned is of version "null"
* Suspending versioning does not delete prev version ⇒ Safe operation.

Handy-on

* Properties → Bucket Versioning → enable
* Show versions toggle al files.
* Delete markers if we delete anything can be restored by deleting delete marker.

Date:

# Amazon S3 - Replication (CRR & SRR)

* Must enable versioning in SRC & DEST buckets.
* CRR → CROSS Region Replication
* SRR → Same Region Replication
* Buckets can be in diff AWS accounts
* Copying is asynchronous
* Must give proper IAM perms to S3.

Use Cases:

* CRR - compliance, lower latency access, replication across accounts.
* SRR - log aggregation, live replication between prod & test accounts.

Hands-on

On Source Bucket → Management
↓
create new rule ← Replication Rules
↓ save

asks if existing objects to be replicated by batch process

{versionID's are also replicated}

Date:

# S3 - Storage Classes

* S3 Standard - General Purpose
* S3 Standard - Infrequent Access (IA)
* S3 One-Zone Infrequent Access
* S3 Glacier Instant Retrieval
* S3 Glacier Flexible Retrieval
* S3 Glacier Deep Archive
* S3 Intelligent Tiering

We can move between classes manually
(or)
using S3 Lifecycle Configurations

* Durability
  - High durability (11 9's) of objects across multiple AZ's.
  - If we store 10,000,000 objects in S3, we can on average expect a loss of single object once every 10,000 years.
  - Same for all storage classes.

* Availability
  - How readily a service is available.
  - Varies depending on storage class.
  - Ex:- standard → 99.99% availability
    = 53 min. downtime a year.

Date:

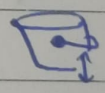
## S3 Standard - General Purpose

* 99.99 % Availability
* Used for frequently accessed data.
* Low latency & high throughput
* Sustains 2 concurrent facility failures
* Uses :- Big data analytics, mobile & gaming apps, content distribution, etc.

## S3 Storage - Infrequent Access

* for data that is less frequently accessed, but requires rapid access when needed.
* lower cost than S3 Standard.

→ S3 - Standard IA



- 99.9 % Availability
- Use Cases : Disaster Recovery, backups.

→ S3 - One Zone - IA

- High durability in single AZ (data lost if AZ destroyed)
- 99.5 % Availability
- Uses :- Secondary backups of on-prem data & data we can re create.

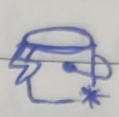
Date:

# Glacier Storage Classes

* low-cost storage meant for archiving
* Pricing: price for storage + object retrieval cost.

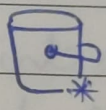
→ S3 Glacier Instant Retrieval

- millisecond retrieval
- great for data accessed once in quarter
- Min Storage duration: 90 days



→ S3 Glacier Flexible Retrieval

- Expedited (1 to 5 min)
- Standard (3 to 4 hrs)
- Bulk (5 to 12 hrs) - free
- Min. Storage : 90 days



→ S3 Glacier Deep Archive

- for long term storage
- Standard (12 hours)
- Bulk (48 hours)
- Min. storage = 180 days

Date:

# S3 Intelligent Tiering

* Small monthly monitoring & auto-tiering fee.
* Moves objects automatically between Access Tiers based on usage.
* No retreival charges

## Tiers

* Frequent Access tier (automatic) : default tier
* Infrequent " " " : objs not accessed for 30 days
* Archive instant Access tier (auto) : 90 days
* Archive access tier (optional) : 90 - 700+ days
* Deep Archive Access tier (optional) : 180 to 700+ days.

## Handy-on

Bucket → Properties → Storage class.

Management → Create lifecycle rule
↓
Write days → tiers.
↓
Create rule.

Date:

# S3 Encryption

| No Encryption | Server Side Encryption | Client Side Encryption |
| --- | --- | --- |
| ↓ | ↓ | ↓ |
| Blah | Server encrypts after receiving the file | User encrypts before uploading |

# Shared Responsibility Model

| AWS | Us |
| --- | --- |
| ⇓ | ⇓ |
| * Infra (security, durability, availability, sustaining loss of data centers) | * S3 Versioning |
| * Config & vulnerability Analysis | * Bucket Policies |
| * Compliance validation | * Replication |
| | * Logging & Monitoring |
| | * Storage class |
| | * Data encryption |

Date:

# AWS Snow family

* Highly secure, portable devices to collect and process data at the edge and migrate data into & out of AWS

Data Migration: Snow Cone, Snowball Edge, Snowmobile

Edge Computing: Snow Cone, Snowball Edge

## Data Migration with AWS Snow family

Without it:- 1 PB at 10GBPS ⇒ 12 days

&

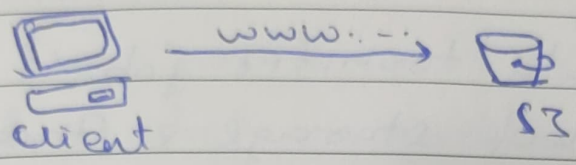
limited connectivity, bandwidth,
high network cost
shared bandwidth
connection instability.

If it takes more than a week to transfer over network use Snowball devices

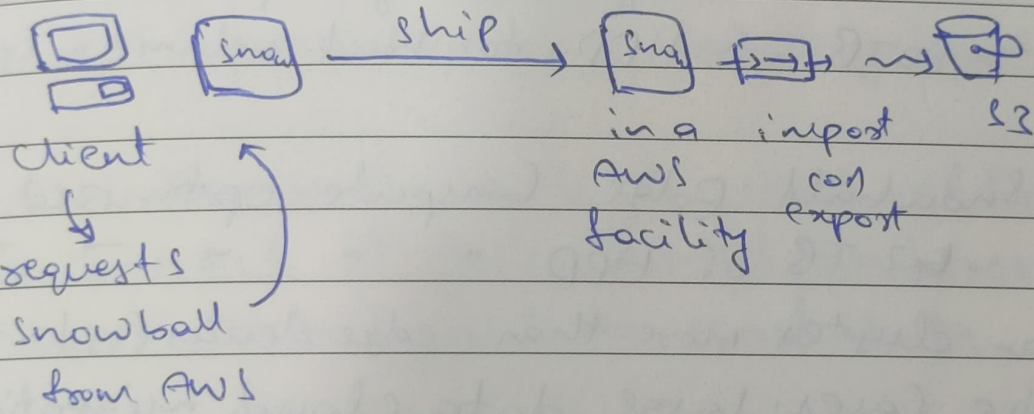
Date:

Snow family is basically offline devices to perform data migrations.

Direct upload:




With Snow family




P.T.O

Date:

# Snowball Edge (for data transfer)

* Physical data transport : moves TBs or PBs of data in or out of AWS

* Pay per data transfer job.
* Provide block storage & Amazon S3-compatible object storage.

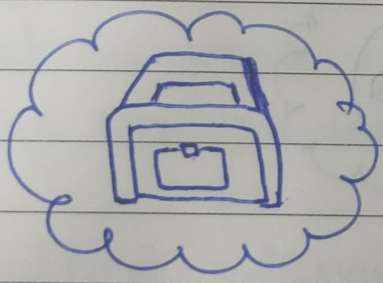
* Snowball Edge Storage optimized
  - 80TB of HDD for block volume & S3 storage

* Snowball Edge Compute optimized
  - 42 TB of HDD " " " " "
* Can cluster more than 1 edge devices (15)
* Use Cases: large data cloud migrations, DC decomission, disaster recovery.

# SnowCone

* Small, portable, computing, anywhere, rugged & secure, withstands harsh environments.
* light (4.5 pounds, 2.1 kg)
* Used for edge computing & data transfer
* 8TB of usable storage.
* Use SnowCone where Snowball does not fit (space constrained)
* Must provide own battery/cables.

- Can be sent back to AWS offline, or connect it to internet and use "AWS DataSync" to send data.

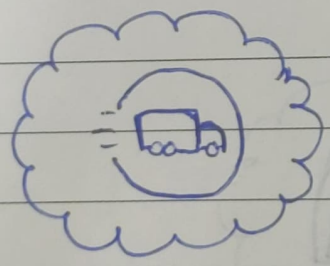


# Snowmobile

* literally uses a truck to transfer data.
* Transfer Exabytes of data (1 EB = 1000 PB = 1000000 TB)

* Each Snowmobile has 100 PB capacity (use multiple in parallel)

* High security, temp. controlled, GPS, 24/7 video survillance.
* Better than Snowball if you transfer more than 10 PB.



Date:

## Usage Process of Snow family

1. Request Snowball Device
2. Install snowball client / AWS OpsHub
3. Connect to your server and copy files.
4. Ship back to AWS facility.
5. Data will be loaded into S3 bucket
6. Snowball is completely wiped.

---

## Edge Computing

* Processing data while it is being created on edge location.

⇓

A truck on road, a ship in sea, a mining station underground...

⇓

limited / no internet access
limited / no access to compute power

* We setup Snowball Edge / Snowcone to do edge computing

⇓

Pre-process data, ML at edge, Transcoding media streams.

Date:

* Eventually if we need, we can ship the device & transfer the data.

Options to use

* Snowcone (smallest)
  - 2 CPUs, 4 GB of memory, wired or wireless access.
  - USB-C power using cord or the optional battery.

* Snowball Edge - Compute Optimized
  - 52 vCPUs, 208 GiB of RAM
  - Optional GPU (ML, video processing)
  - 42 TB usable storage

* Snowball Edge - Storage optimized
  - Upto 40 VCPU's, 80 GiB of RAM
  - Object storage cluster available

All: can run EC2 instances & AWS Lambda functions (Using AWS IoT Greengrass)

* Long term deployment options: 1 and 3 years, discounted pricing.

Date:

# AWS OpsHub

* Historically we needed CLI for Snow family devices, but it was painful

⇓

Graphical thing → OpsHub.

- Configuring stuff
- Transferring files
- Monitor device metrics
- Launch services.

---

# Storage Gateway

* AWS is pushing for "hybrid cloud"

↓

on-prem + cloud

* Can be due to migrations, security, compliance, etc..

it is proprietory unlike EFS/NFS

To expose (S3) data to on-prem we use AWS Storage Gateway

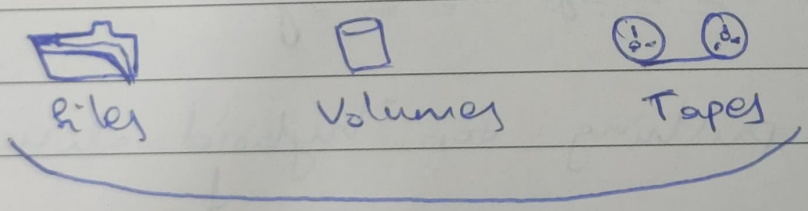
* Bridge between on-prem data & cloud data in S3.

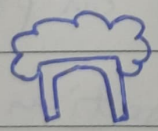
* Allows on-prem to seamlessly access AWS cloud data.

* Uses :- disaster recovery, backup & restore, tiered storage

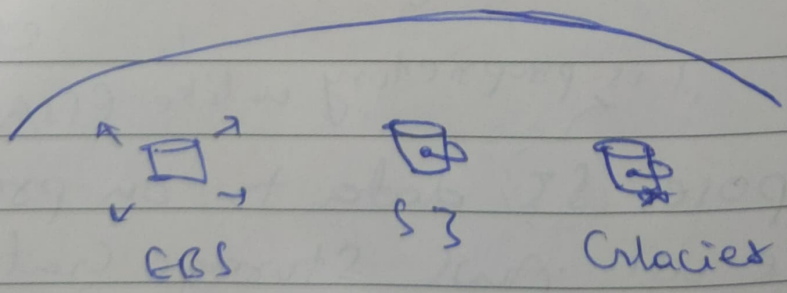
* Types.

File  Volume  Tape



AWS Storage gateway




Date:

# Databases

- On disk we can store on EBS, EFS, EC2 Instance Store, S3) → but they have limits.

- In database
  - → we can structure data.
  - → index them & search/query effectively.
  - → define relnships b/w databases.

Every database is optimized for a specific purpose.

## Relational DB

→ Tables, rows, columns, SQL

## NO SQL DB

→ non SQL ⇒ non relational dbs.
→ They are purpose built for specific data models and have flexible schemas.
→ Benefits:-
  flexible, scalable, high perf, highly func.
→ Ex:- key-value, document, graph, in-memory, search db's

Date:

JSON.

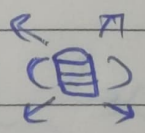
In AWS Managed DB's we get:-

* Quick Provisioning, high availability, Vertical & Horizontal scaling.
* Automated Backup & Restore, Operations, Upgrades.
* OS patching.
* Monitoring, Alerting.

Note:- We can run our own db's on EC2 instances but then all the above become our resp's.

---

AWS RDS



* Relational Database Service.
  ↳ SQL.
* We can create
  - Postgres
  - MySQL
  - MariaDB
  - Oracle
  - Microsoft SQL Server
  - Aurora (AWS Proprietary)

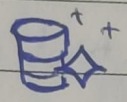
Date:

We get all adv's of managed DB &

* Multi AZ setup (disaster recovery)
* Storage backed by EBS (gp2 or io1)

But we can't SSH into our instances.

## Amazon Aurora



* Owned by AWS (not open source)
* Has PostgreSQL & MySQL.
* It is "AWS cloud optimized" & claims 5X perf impr. on MySQL (RDS)
  3X " " " PostgreSQL (RDS)

* The storage automatically grows in increments of 10 GB, up to 64TB.
* It costs 20% more than RDS, but is more efficient.
* Not included in free tier.

Date:

Hands-on RDS

Standard create / Easy create

Engin options (MySQL / Aurora / etc..)

Templates (Prod / test / freetier)

credential stuff for admin.

DB Instance class.

Storage (size & auto scaling)

Connectivity (VPC, public access)
Database port

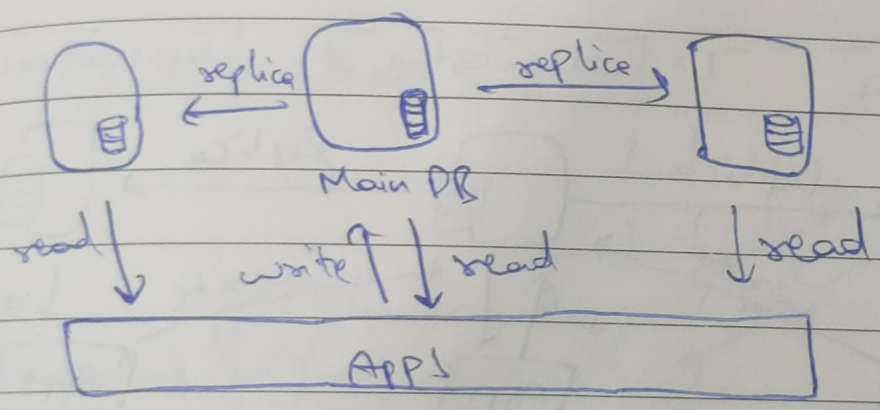
Database Authentication.

#Snapshot can also be created

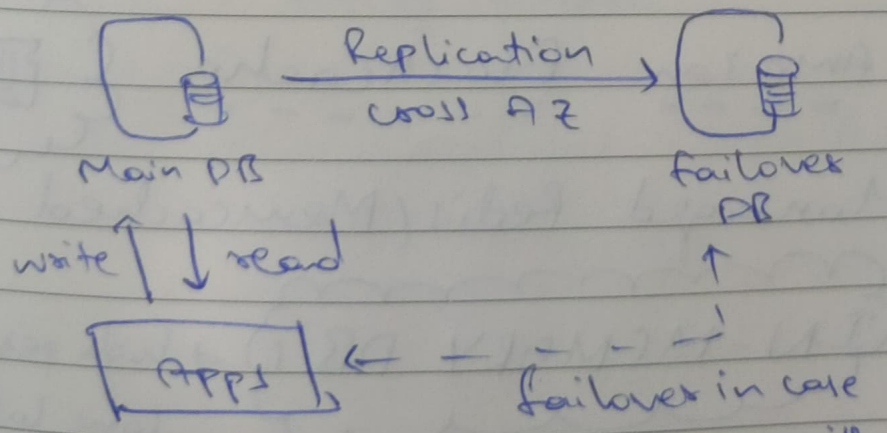
Date:

# RDS Deployment Options

* Read Replicas:
  - Scale the read workload
  - Can create upto 5 read replicas
  - Data is written only to main DB.

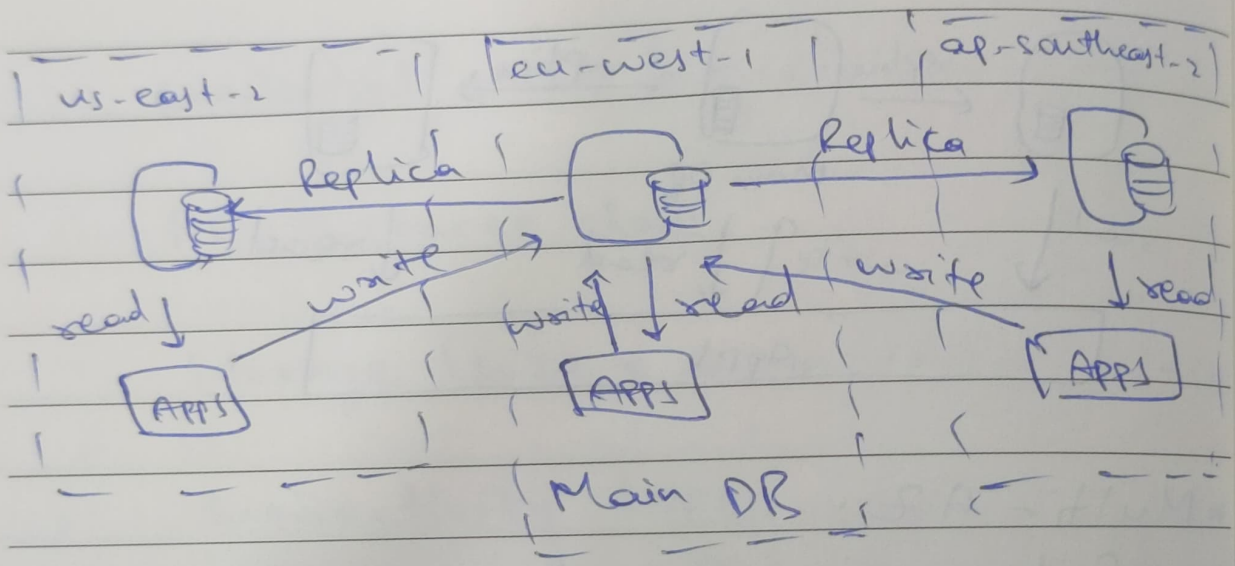



* Multi-AZ:
  - failover in case of AZ outage
  - Data is read/write into main DB
  - Can only have one other AZ as failover

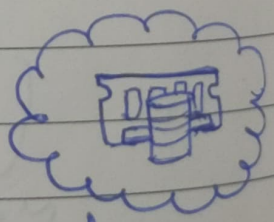



Date:

* Multi Region (Read Replicas)
  - → Disaster recovery in case of region issue.
  - → local perf for global reads (less latency)
  - → Replication cost.




Amazon ElastiCache



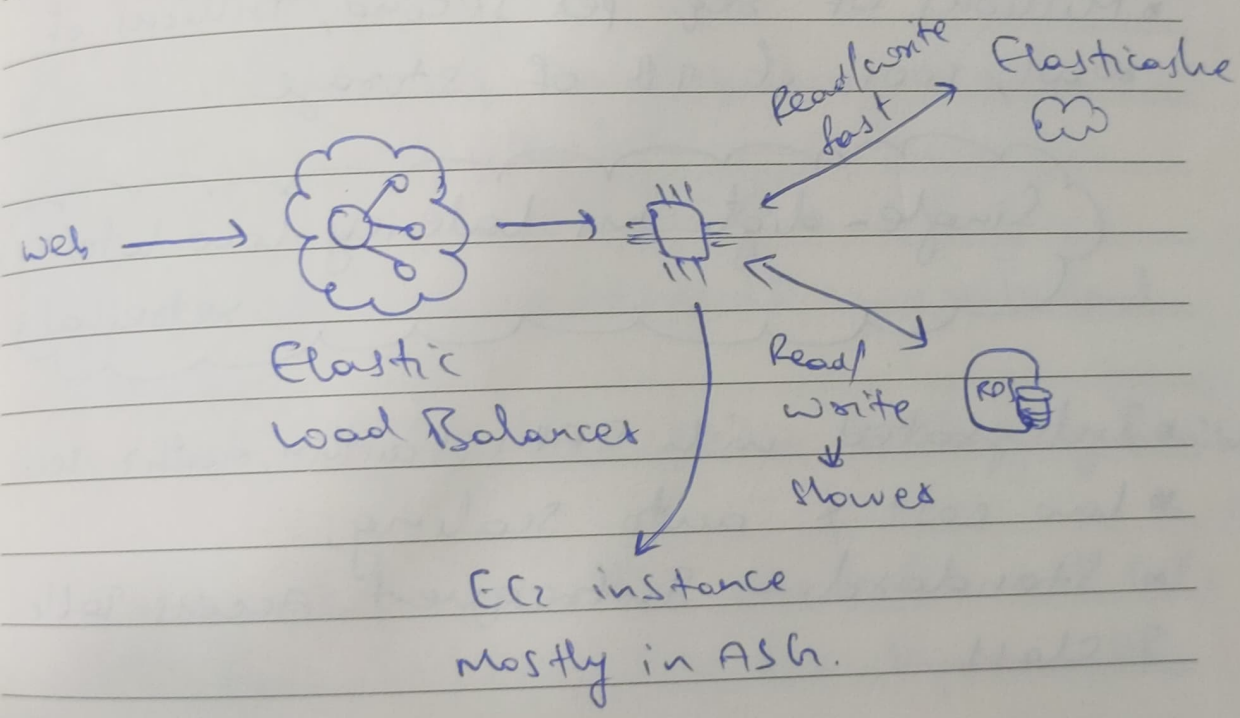
* Managed Redis / Memcached
* IN-MEMORY DB's → high perf, low latency.

Date:

* Helps reduce load for read intensive tasks.

AWS takes care of OS maintanance, optimizations, setup, config, monitoring, failure, recovery & backup.

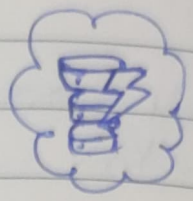
Architecture.




P.T.O

Date:

# Dynamo DB



* fully Managed, Highly available with 3 AZ replication.
* No SQL DB.
* Scales to massive workloads

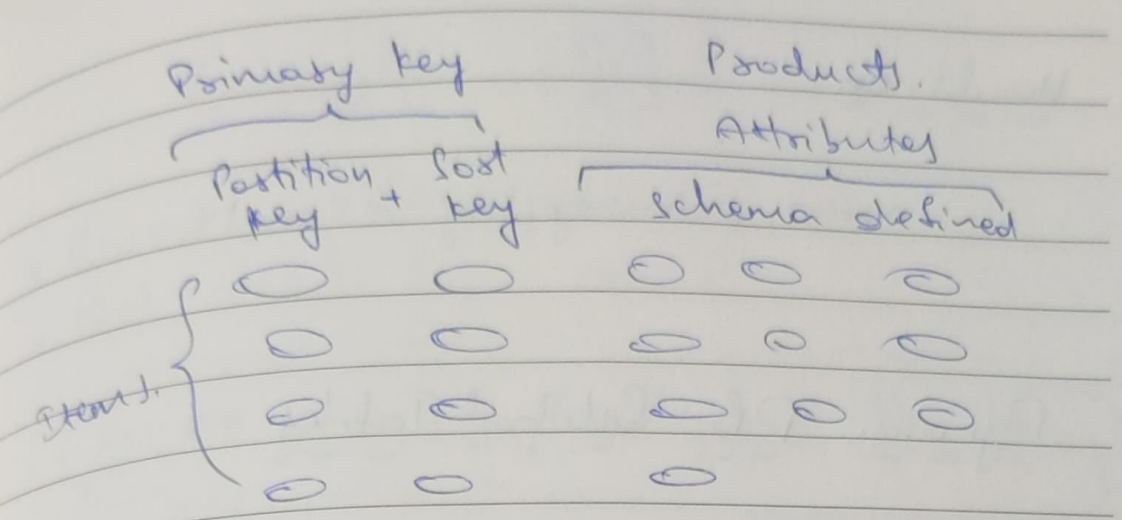
distributed (serverless) database

* Millions of req. per second, trillions of rows, 100s of TB of storage

(Single-digit ms latency - low latency retrieval)

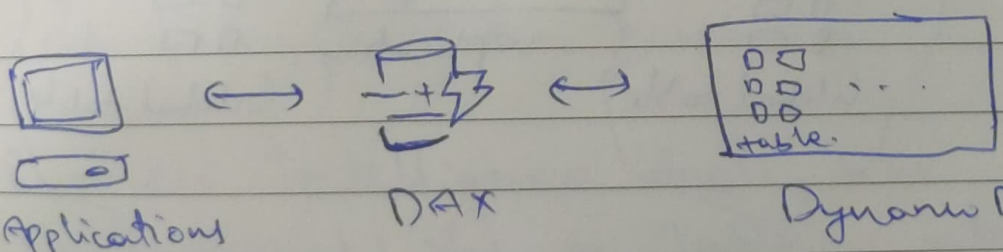
* Integrated with IAM (authn, authz, sec.)
* low cost & auto scaling.
* Standard & Infrequent Access Table class.

* It is a key/value DB




## Dynamo DB Accelerator - DAX

- Fully Managed in-memory cache for Dynamo DB.
- 10x perf improvement - microsecond latency.
- Secure, highly scalable & highly available
- Elasticache can be used for all DB's but this is specific for Dynamo DB.



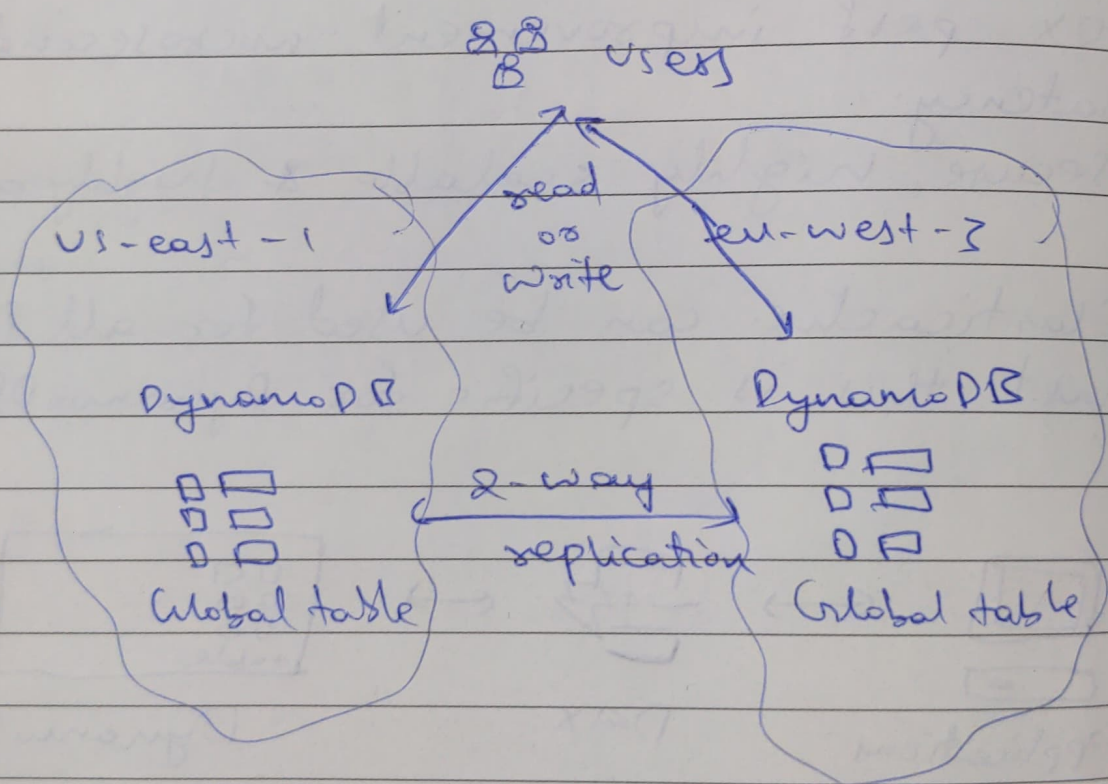

Date:

Hands-on

name, primary key & create.

## DynamoDB - Global Tables

* Make a DynamoDB table accessible with low latency in multiple regions
* Active - Active replication (read/write to any AWS region)




Date:

# Redshift

- Based on PostgreSQL
- Not used for OLTP

OLAP : Online Analytical Processing

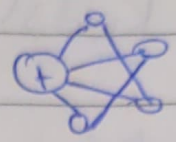
analytics

Data warehousing

- Load data every hour.
- 10x better perf than other warehouses
- Columnar storage of data (instead of rows)
- Massively Parallel Query Execution (MPP)
  Highly available.
- Pay as you go.
- SQL interface for querying.
- Business Intelligence tools like AWS Quicksight or Tableau integrate with it.

Date:

# Amazon EMR



* "Elastic Map Reduce"

* Creates Hadoop cluster (Big data) to analyse and process vast data.

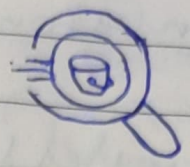
* Clusters can be made of hundreds of EC2 instance.
* Also supports Apache Spark, HBase, Presto, Flink, ...

* EMR takes care of all provisioning & configuration.
* Auto-scaling & integrated with spot instances.

* Use cases:- data processing, machine learning, web indexing, big data, ..

Date:

# Amazon Athena

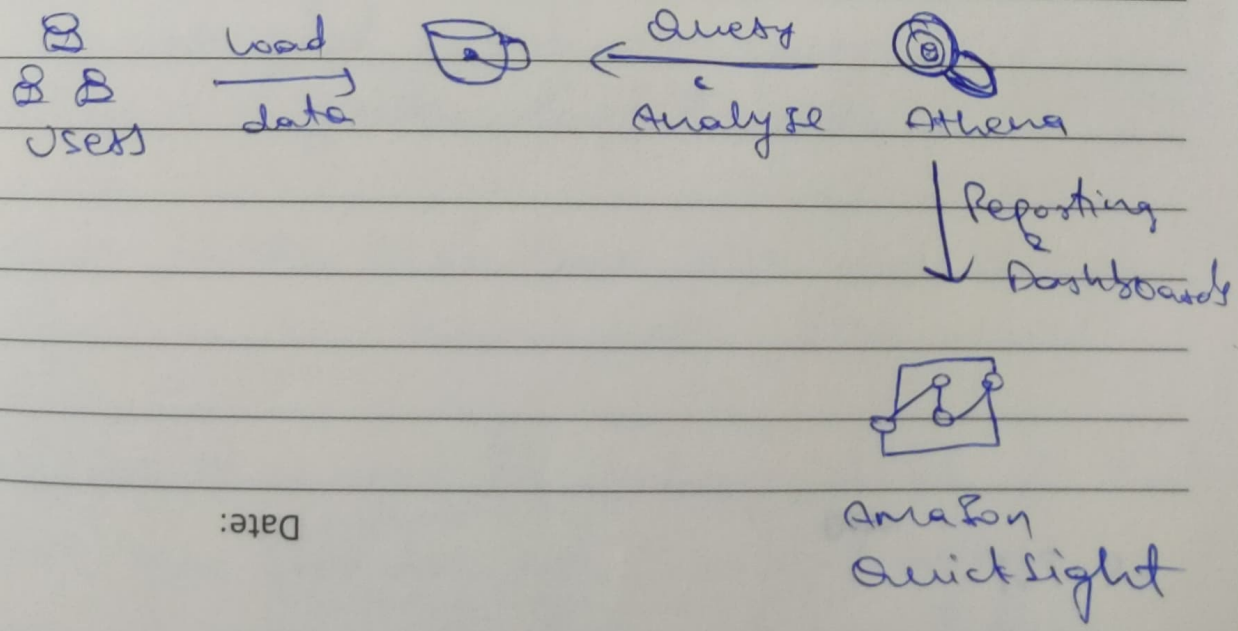


* Serverless query service to perform analytics against S3 objects

* Uses standard SQL
* Supports CSV, JSON, ORC, Avro, Parquet (built on Presto)

* Pay per scan of data (compressed / columnar data = less scan)

Use cases :- BI / analytics / reporting, analyze & query VPC flow logs, ELB logs, CloudTrails




Date:

# Amazon QuickSight

* Serverless, ML powered BI services to create interactive dashboards
* fast, autoscale, embeddable, per-session pricing
* Use cases:-
  - → Business Analytics
  - → Building visualizations
  - → Perform ad-hoc analysis
  - → Get business insights using data.
* Integrated with RDS, Aurora, Athena, Redshift, S3..

# Document DB

* AWS implementation of Mongo DB
* NoSQL DB

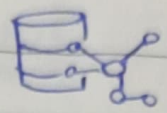
* Store, query & index json data.
* Similar deployments as Aurora fully Managed, highly available, replication in 3 AZ's, storage auto grows in 10 GB upto 64 TB.

* Automatically scales to workloads with millions of requests per second.

Date:

P.T.O

# Amazon Neptune



* Fully Managed {Graph DB}

Ex:- Social Network:-
- → Users have friends
- → Posts have comments
- → comments have likes
- → User share & like posts.

* Highly Available across 3 AZ, 15 read replic

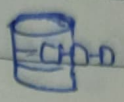
* Optimized for hard & complex queries.

* Can store upto billions of relations and query the graph with ms latency.

* Uses:- knowledge Graphs (Wikipedia)
  fraud detection
  recommendation engines
  social networking

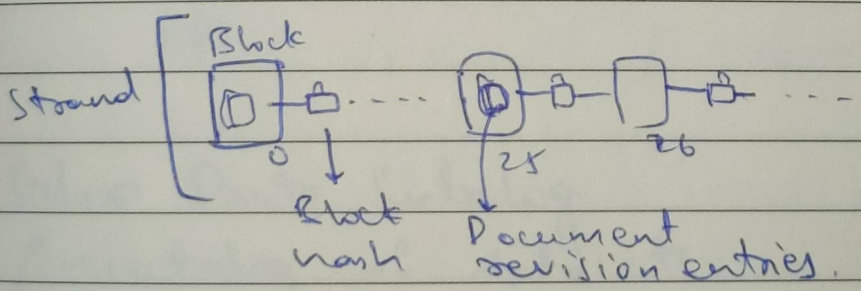
Date:

# Amazon QLDB



* Quantum Ledger Database.
* useful for (financial transactions)
* fully managed, serverless, highly available, Replication across 3 AZ.
* Review the history of all the changes made to our appln.
* Immutable System: no entry can be removed (or) modified, cryptographically verifiable.

QLDB Journal




* 2-3x better perf than common ledger blockchain frameworks, SQL usagl.

Amazon Managed Blockchain diff is that this has (NO decentralization) concept, in accordance with financial regulation

Date:

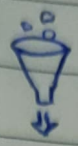
# Amazon Managed Blockchain

* Blockchain makes it possible to build applications where multiple parties can execute transactions without the need for a trusted, central authority.

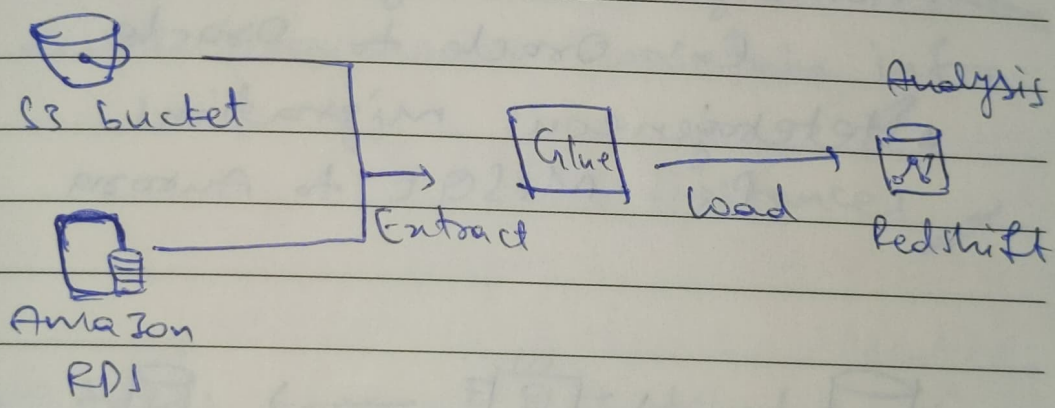
* It is a managed service to
  - → Join public blockchain networks
  - → Create own scalable private network.

* Compatible with Hyperledger fabric, ethereum.

# AWS Glue



* Managed ETL service
  - Extract
  - Transform
  - Load
* Useful to prepare data for analytics
* Fully serverless service




* Glue Data Catalog

  - catalog of datasets
  - can be used by Athena, Redshift, EMR.

has metadata about our data.

Date:

# DMS.

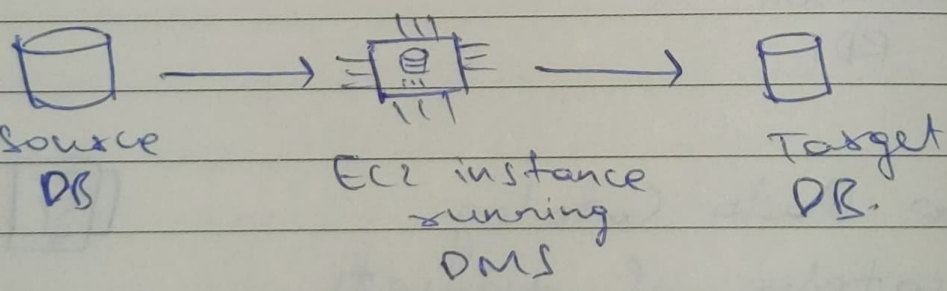
* Database Migration Service

* Quickly and securely migrate DB's to AWS, resilient & self healing.

* The source database remains available during migration.

Supports :-

* Homogeneous migrations
  Ex :- Oracle to Oracle
* Heterogeneous migrations
  Ex :- MySQL to Aurora.




look at overview for all DB's

Date:

# ECS

We know Docker.

↓

containers, scaling, Docker Repo, ECR.
Docker vs VM's.

## ECS

* Elastic Container Service
* launch Docker containers on AWS
* We must provision & maintain infra
  ↙
  EC2 instances

* AWS takes care of starting/stopping containers
* Has integration with Application Load Balancer. → it knows where to put the docker container (in which EC2 instance)

Date:

# Fargate

* launch Docker containers on AWS
* We need not provision the infra

no EC2 instances to manage

Serverless

* AWS runs containers for us based on RAM & CPU we need.

# ECR

* Elastic container Registry
* Private Docker Registry for AWS
* This is where we store our Docker images so they can be run by ECS (or) Fargate.

Date:

# What's Serverless?

* It is a new paradigm in which we don't have to manage servers
* They just deploy code

⇓

functions

⇓

Serverless == Faas (Function as a Service)

* Initially only lambda was there but now :- database (S3), messaging, storage (DynamoDB) Fargate, ... etc.

Date:

P.T.O

# AWS Lambda

EC2 :-

- Virtual servers in cloud
- limited by RAM & CPU
- Continuosly running
- Scaling means intervention to add/remove servers.

Lambda :-

- Virtual functions → no servers to manage
- limited by time → short executions
- Run on-demand
- Scaling is automated!

Benefits :-

* Easy Pricing: → calls & duration.
  - Pay per request & compute time
  - Free tier of 1,000,000 lambda requests & 400,000 GBs of compute time.

* Integrated with whole AWS services

Date:

* Event-Driven: functions get invoked when needed.
* Easy monitoring through cloudwatch
* Easy to get more resources per fn (up to 10 GB of RAM)
* ↑ RAM ⇒ ↑ CPU + ↑ Network.

Language support
⇓

Node, Python, Java, C#, Golang, Powershell, Ruby, Custom Runtime API (Ex:- Rust)

⇒ Lambda Container Image
→ the container image must implement the lambda runtime API.
→ ECS/fargate is preferred for running arbitrary docker images.

Ex Use case:-
* Thumbnail creation.
* CRON Jobs.

* It is usually very cheap ⇒ very popular

Date:

## Hands-on.

Create function → choose → repo.

- Scratch
- Blue print
- Container image

⇓

name & function definition.

⇓

Test function.

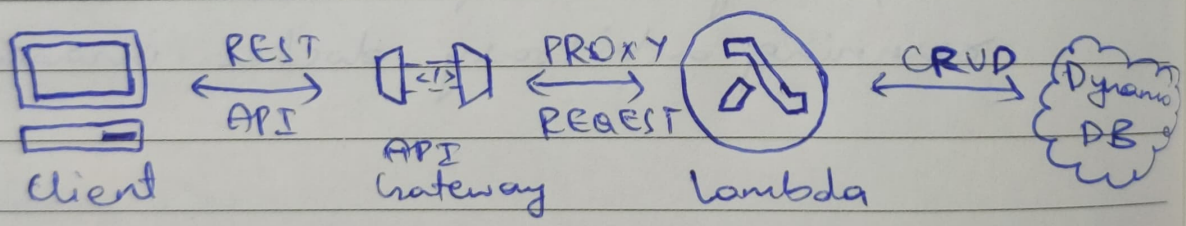
⇓

config, monitor, trigger.

---

## Amazon API Gateway

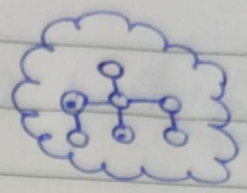
* Ex:- Building a serverless API




* fully managed service for developers, to easily create, publish, maintain, monitor & secure API's.
* Serverless, scalable, Restful API's, WebSockets
* Support for security, authn, API throttling, API keys, monitoring, ...

Date:

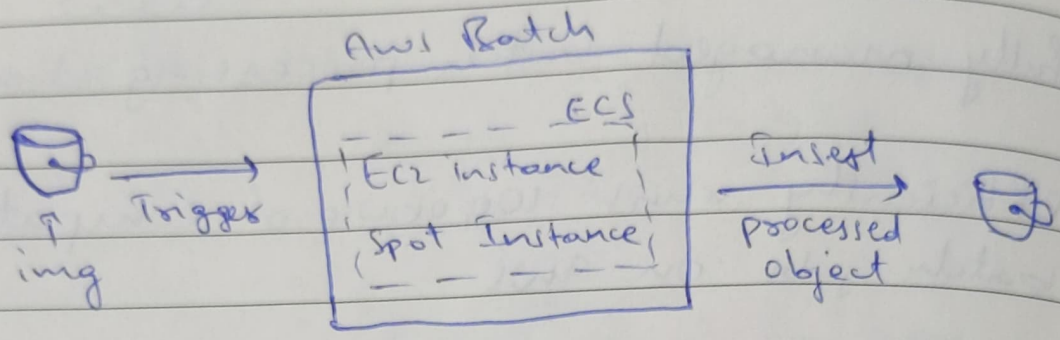
# AWS Batch



* fully managed batch processing at any scale
* Efficiently runs 100,000s of computing batch jobs on AWS.
* A batch job is a job with a start and an end
  Ex:- starts at 3:am, ends at 5 am.
* Batch will dynamically launch EC2 instances or Spot Instances.
* AWS Batch provisions the right amount of compute/memory.
* We just submit or schedule batch jobs and AWS Batch does the rest!
* Batch jobs are defined as Docker Images and run on ECS
* Helpful for cost optimizations & focusing less on the infrastructure.

Date:

## Simplified Example




| Batch | vs | Lambda |
|---|---|---|
| * No time limit | | * Time limit (~15 min) |
| * Any run time as long as it's packaged as a Docker Image | | * limited runtimes<br>* limited temporary disk space |
| * Rely on EBS/instance store for disk space | | |
| * Relies on EC2 (can be managed by AWS) | | * Serverless. |

# Amazon Lightsail

- Virtual servers, storage, databases, and networking.
- Low & Predictable pricing
- Simple alternative to using EC2, RDS, ELB, EBS, Route 53, ...
- Great for people with little cloud exp.
- Can setup notifications and monitoring of your lightsail resources.
- Use cases:-
  - → Simple web applns (LAMP, Nginx, MEAN, node)
  - → Websites (WordPress, Magneto, Plesk, Joomla)
  - → Dev/Test Env.
- Has high availability but no autoscaling limited AWS integrations.

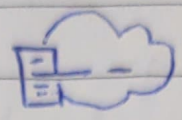
## Hands-on.

It's very easy & intutive and cute.

Date:

# Deploying & Managing Infra at Scale

## Cloud formation



* Declarative way of outlining our AWS infra, for all resources.

Ex :- I want a sec. gp.
I want two EC2 instances with this
I want S3 bucket
I want ELB in front of these

⇓

created in right order with the exact configuration

## Benefits

* Infra as code
  → No resources are manually created (excellent for control)
  → Changes to infra are reviewed through code

Date:

* Cost
  - we can easily see how much each stack costs using the identifier
  - estimate cost using our template
  - Saving strategy: In Dev, we could automation delete of templates at 5 PM and recreated at 8 AM, safely.

* Productivity
  - Ability to destroy & re-create an infra on cloud on the fly
  - Automated generation of diagram for our template.
  - Declarative Programming

* Don't re-invent the wheel.
  - leverage existing template on web!
  - leverage documentation.

* Supports all AWS Resources
  - Everything is supported
  - we can use "custom resources" for resources that are not supported.

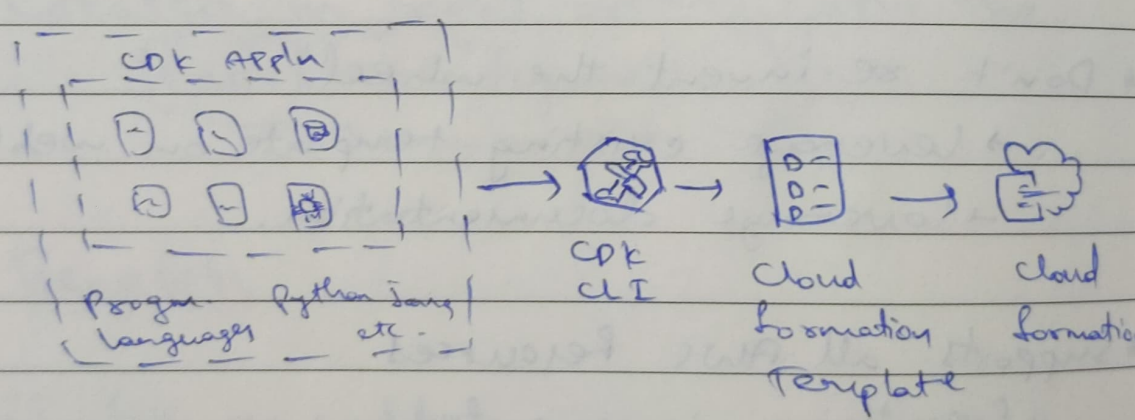
Date:

# AWS Cloud Development Kit (CDK)

* We can define our cloud infra using a familiar language
  - → JS, Python, Java, .NET

* The code is compiled into a cloudformation template. (JSON / YAML)

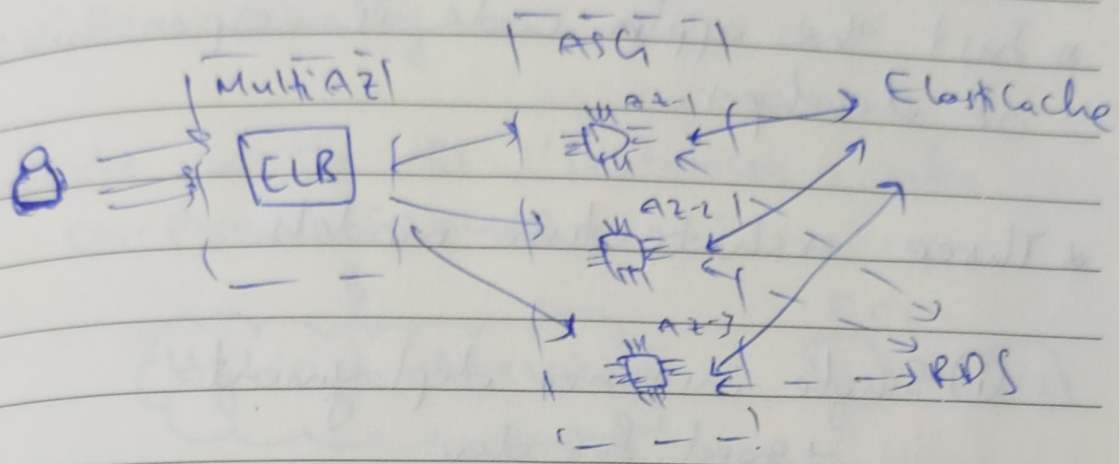
* We can therefore deploy infra and appln runtime together.
  - → Great for lambda fns
  - → Great for Docker containers in ECS / EKS.




Date:

# Beanstalk

The gen three-tier architecture is




This can be done easily using Beanstalk.

⇓

So we as developers can just run code everywhere in the same way.

* It's a developer centric view of deploying an appln in AWS.
* It uses all the components we have seen before.
* But it's a all in one view.
* We have full control over config.
* It is a PaaS
* It's free, we pay for underlying instances.

Date:

* It is a managed service, so we get all the benefits.

* Just the appln code is responsibility of developer.

* Three architecture models.

(i) Single instance deployment
→ good for dev.

(ii) LB + ASG
→ great for prodn & pre-prodn web applications

(iii) ASG only.
→ great for non-web apps in prodn (workers, etc..)

* Supports many platforms.

## Health Monitoring

* Health agent pushes metrics to cloudwatch
* Checks for app health, publishes health events.

Date:

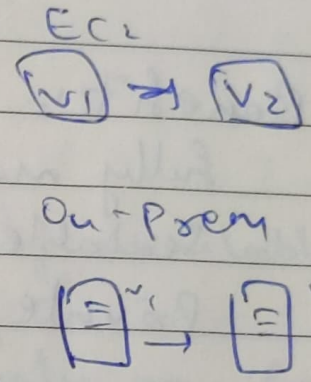
# AWS CodeDeploy

* We want to deploy our appln automatically.

* Works with EC2 instances & also On-Prem servers.

⇓

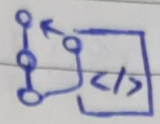
Hybrid service




* Servers/Instances must be deployed ahead of time with the CodeDeploy Agent

Date:

# AWS CodeCommit



- Basically GitHub of AWS.

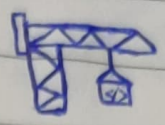
⇓

Git Based Repo
Collaborate with others
versioned.

Benefits:-
fully managed,
Scalable & highly available,
Private, secured & Integrated with AWS.

Date:

# AWS Code Build



* Code Building Service.

↓

compiles, runs tests, produces package that can be deployed

## Benefits

* fully managed, serverless.
* continuously scalable, highly available.
* Secure
* Pay as-you-go pricing - only pay for build time.

# AWS CodePipeline

- Orchestrate different steps to have the code automatically pushed to deployment Production.

- Code → Build → Test → Provision → Deploy.
- Basis for CICD.

- Benefits.
  - fully managed.
  - compatible with CodeCommit, CodeBuild, CodeDeploy, Beanstalk, CloudFormation, Github, 3rd party & custom plugins.

Ex. Pipeline

Code Commit → Code Build → Code Deploy → Beanstalk.

Date:

# AWS CodeArtifact

* S/w packages have dependencies.
* Storing & retreiving is called

artifact management

* Traditionally, we need to setup our own artifact management system.

⇓ Tantata

CodeArtifact is a secure, scalable, & cost-effective artifact management for s/w dev.

* Works with common dep. tools like Maven, Gradle, npm, yarn, twine, pip & NuGet

* Developers / CodeBuild can retreive directly from CodeArtifact.

Date:

## AWS CodeStar

* Unified UI to easily manage s/w dev. activities at one place.

* "Quick way" to get started to correctly setup CodeCommit, CodePipeline, CodeBuild, CodeDeploy, Beanstalk, EC2, etc.

* can edit code "in the cloud" using AWS Cloud9

* UI stuff.

## AWS Cloud9

* It is a cloud IDE for writing, running & debugging code.
* Insted of installing anything, it runs in a browser

⇓

can work from anywhere by anyone

⇓

allows code collaboration in real time.

Date:

# AWS System Manager (SSM)

* Helps us manage your EC2 & On-Prem systems at scale.

* Hybrid AWS service

* We can operational insights about the state of our infra.
* Suite of 10+ products.
* Most imp features are:-
  - → Patching automation for enhanced compliance.
  - → Run commands across an entire fleet of servers.
  - → Store parameters config with the SSM Parameter Store.

* Works for both Windows & Linux OS.

P.T.O

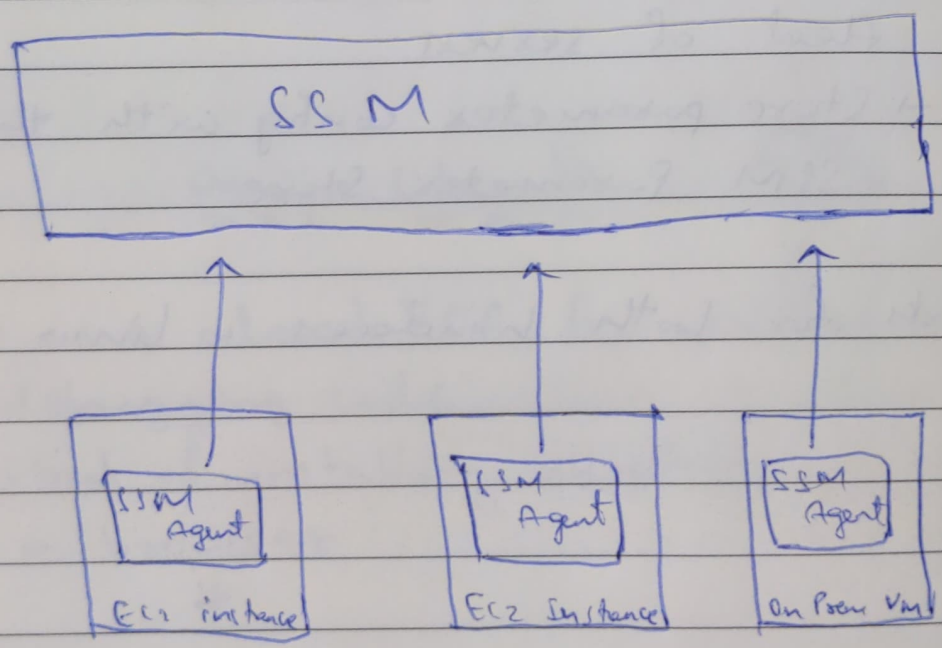
Date:

## How System Manager works.

* Install SSM Agent on the EC2 instance or on-prem server.
* It is by default installed in Amazon Linux AMI & some Ubuntu AMI.

* If an instance can't be controlled with SSM, it's probably an issue with SSM agent!

* Using the SSM agent we can run command patch & configure our servers.




Date:

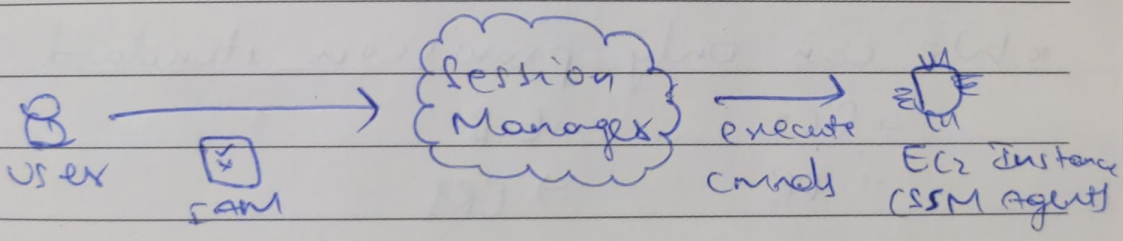
# System Manager - SSM Session Manager

* Allows us to start a secure shell on our EC2 & on-prem server.

* No SSH Access, bastion hosts or SSH keys needed.
* No port 22 needed (better security)

* Supports linux, macos & windows.

* Send session log data to S3 or cloudwatch




Hands-on

Create instance (with necessary IAM)

⇓

Fleet Manager in SSM will show all eligible EC2's

⇓

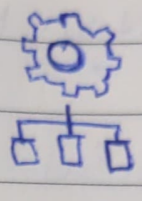
Session Manager can open SSH

⇓

Session history is stored.

Date:

# AWS OpsWorks



* firstly, chef & puppet help us perform servers config automatically, or repititive actions.

* they work with EC2 & On prem VM.

* AWS OpsWork = Managed chef & Puppet

* It's an alternative to AWS SSM.

* We can only provision standard AWS sta
  - → EC2 → LB
  - → DB → EBS

Same architecture

# Global Infrastructure

## Why global appln's

- global appln = appln deployed in multiple geographic
  ⇓
  Regions / Edge locations.

Uses:-

- → Decreased latency
- → Disaster Recovery
- → Attack Protection.

Good Map View.

Region
AZ
Edge location.

P.T.O

Date:

# Amazon Route 53

* It is a managed DNS.

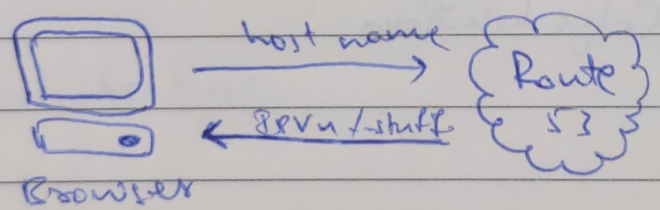
* The most common records are.

host name ⇒ IPV4 • A record
host name ⇒ IPV6 • AAAA record
host name ⇒ new host name • CNAME "
host name ⇒ AWS Resource • Alias.

## Routing Policies

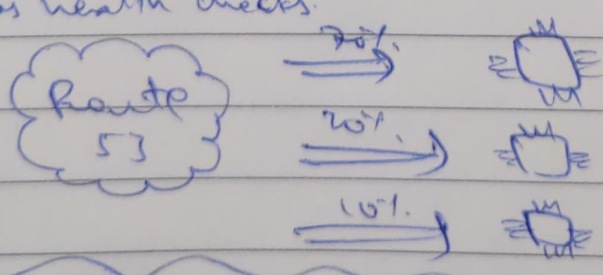
### Simple Routing Policy

No health checks




### Weighted Routing Policy (sort of load balancing)

has health checks.

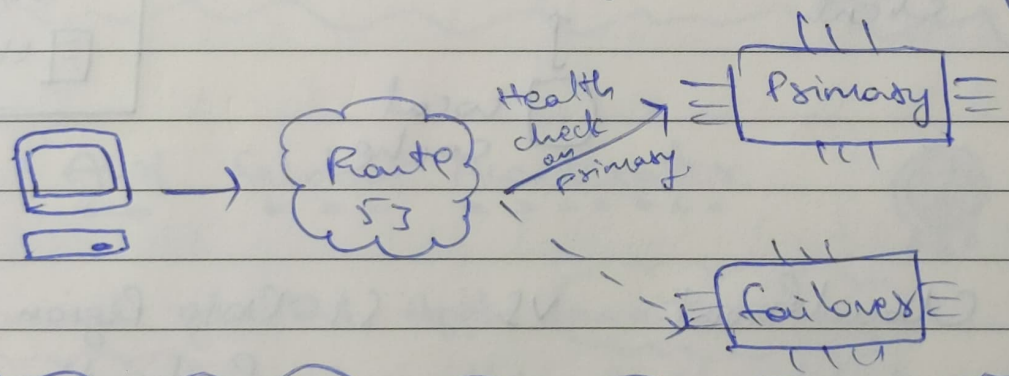



Date:

Latency Routing Policy
↓ (does health checks)

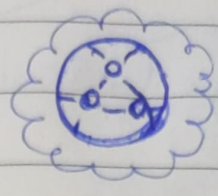
If we have two instances in diff parts of world, based on what area the request is coming from, the least latency instance is chosen.

Failover Routing
(Disaster Recovery)

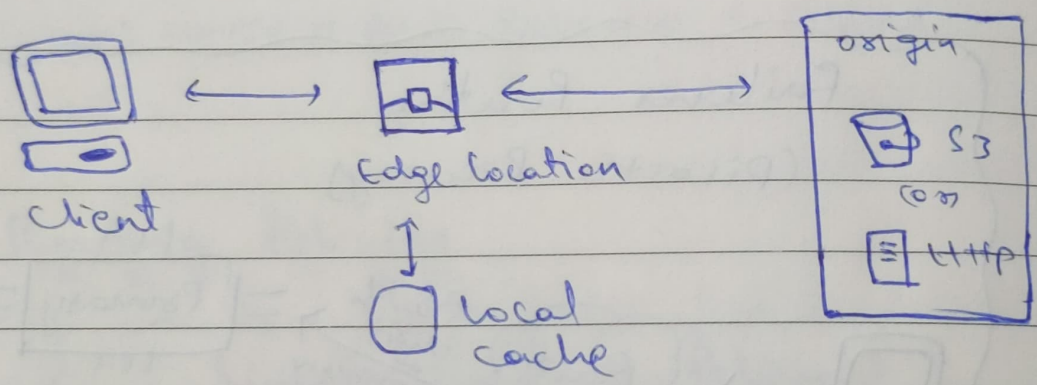



Date:

# AWS Cloudfront



* Content Delivery Network.
* Improves read performance, content is cached at the edge location.
* 216 PoP (edge locations)
* DDoS Protection (on worldwide), integration with shield, AWS Web Appln Firewall.




| Cloudfront | vs | S3 Cross Region Replication |
|---|---|---|
| * Global Edge network<br>* Files are cached for a TTL (maybe a day)<br>* Great for static content | | * Must be setup for each region.<br>* Files are updated in near real-time.<br>* Read only<br>* Great for dynamic content at low-latency |

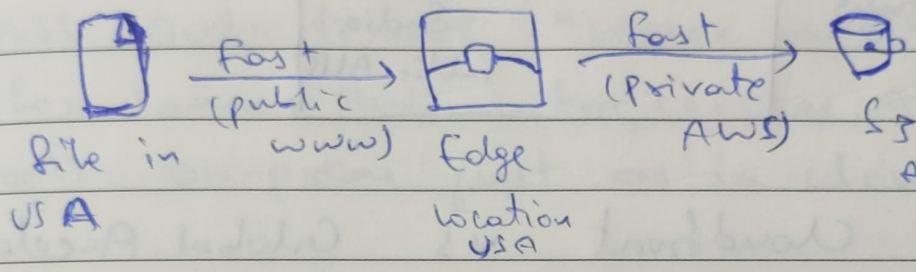
Hands-on:

OAI (Origin Access Identity), Web Access Control List.

Date:

## S3 Transfer Acceleration

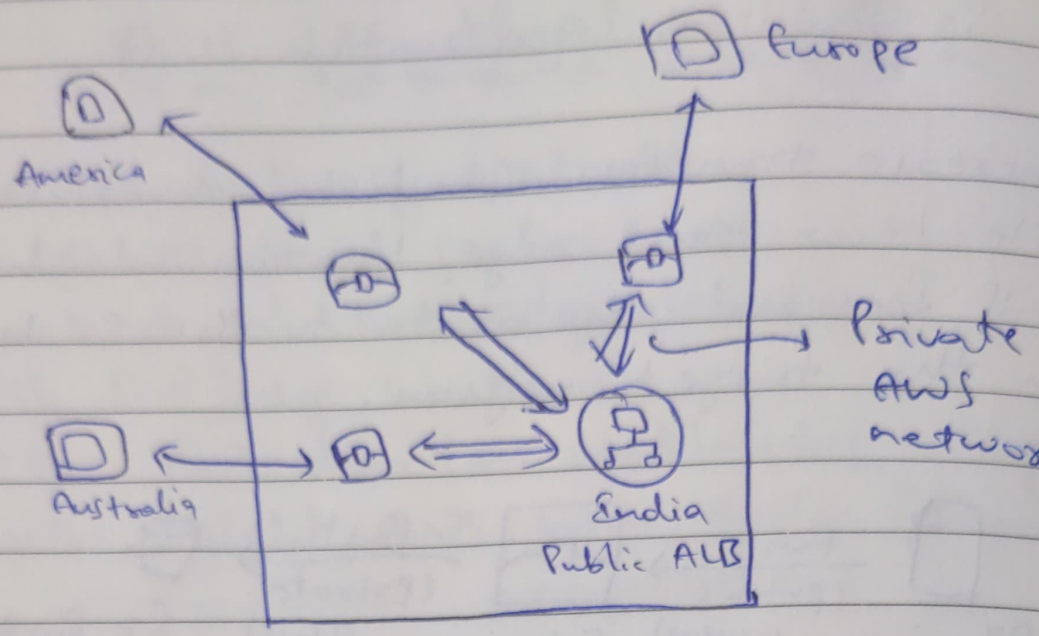
* Increase transfer speed by transferring file to an AWS edge location which will forward the data to the S3 bucket in the target region.




## AWS Global Accelerator

* Improved global Appln availability & performance using the AWS global network.
* It leverages the AWS Internal network to optimize the route to our application (60% improvement)
* 2 Anycast IP are created for our appln and traffic is sent through Edge locations.
* The Edge locations send traffic to the appln.

Date:




AWS Cloudfront vs Global Accelerator

* Both use global network & Edge locations
* Both have AWS Shield for DDoS protection

* Cloudfront - Content Delivery Network
  → caches content (image/video)
  → Content is served at the edge.

* Global Accelerator
  → No caching, proxies packets at the edge to applications running
  → Improves performance (TCP & UDP)
  → Good for HTTP use case
    - that require static IP addresses
    - deterministic, fast failover.

Date:

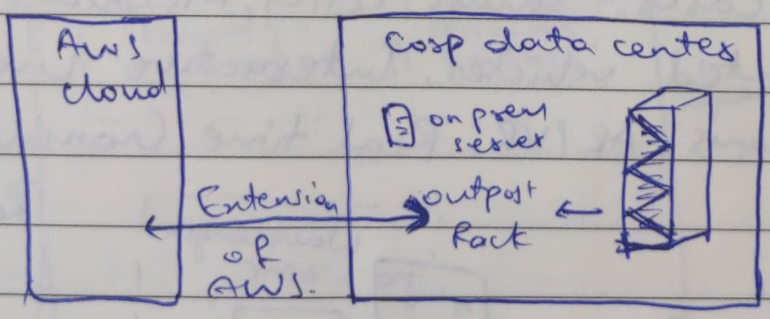
# AWS Outposts

* Most companies run Hybrid cloud.
* Therefore, they deal stuff in 2 ways
  (i) One for AWS cloud
  (ii) One for on-prem infra.

* AWS Outposts are "server racks" that offer same AWS infra, services, API & tools on-prem just as in cloud.

* AWS will setup and manage the "Outpost Racks" within our on-prem.

* Here, we are responsible for the Outpost Rack Physical security.




Benefits:-
low latency, local data processing, data residency, easier migration from on-prem to cloud, fully managed service.

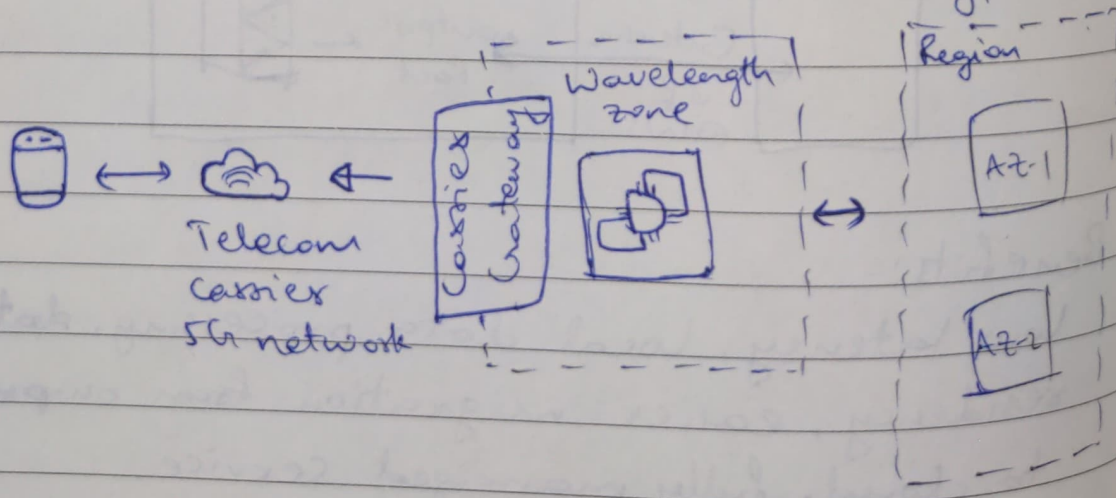
Date:

# AWS Wavelength

* Wavelength zones are infra deployments embedded within the telecommunication providers' datacenters at the edge of the 5G networks.

* Ex:- EC2, EBS, VPC.

* Ultra-low latency applications through 5G networks.
* Traffic doesn't leave the Communication Service Provider's (CSP) network.
* High-bandwidth and secure connection to the parent AWS Region.
* No additional charges or service agreement
* Use cases:- Smart cities, ML diagnostics, Connected vehicles, Interactive Live Video Streams, AR/VR, Real time Gaming, ...

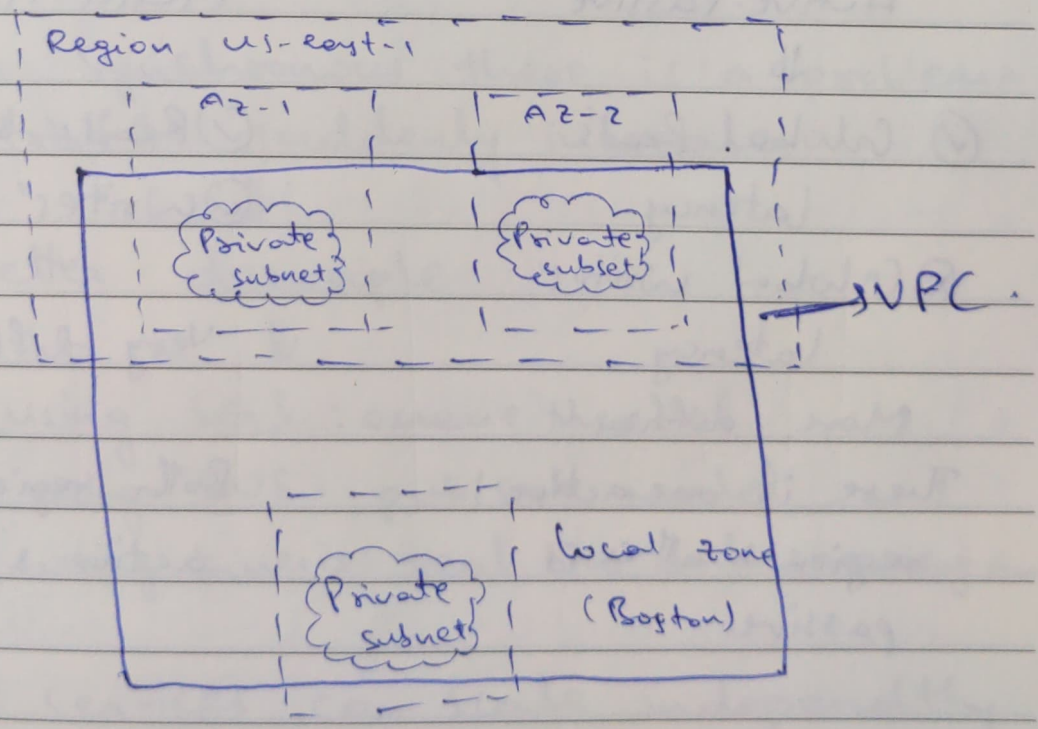



Date:

# AWS Local Zones

* Places AWS compute, storage, database & other selected AWS services closer to end users to run latency-sensitive appln's.

* Basically an "extension of an AWS Region"
* Compatible with EC2, RDS, ECS, EBS, Elasticache, Direct Connect, ...
* Example:-
  - → AWS Region: N. Virginia (us-east-1)
  - → AWS Local Zones: Boston, Chicago, Dallas, Houston, Miami, ...




Date:

# Global Application Architecture

Single Region
Single AZ
⇓
✗ High Availability
✗ Global latency
very less difficult
to set up.

Single Region
Multi AZ
⇓
✓ High Avai.
✗ Glob. latency
a bit harder
to setup.

Multi Region
Active-Passive
⇓
✓ Global Read's
latency
✗ Global writes'
latency
More difficult
There is one active
region & other is
passive

Multi Region
Active-Active

✓ Reads' latency
✓ Writes' latency

Very difficult.

Both regions are
active & available.

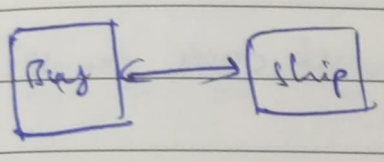
Date:

# Cloud Integration

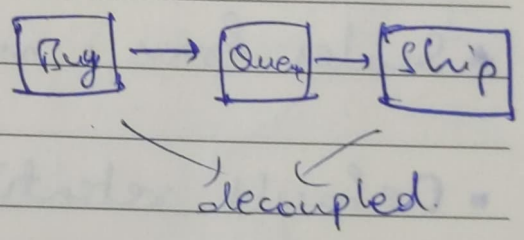
* When we deploy multiple applications, they will need to communicate with each other.

* Two patterns for appl'n communication

(i) Synchronous




(ii) Asynchronous / Event based




With Synchronous there is a problem if traffic suddenly increases.

⇓

Better decouple

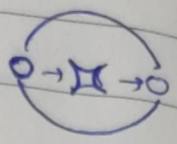
⇓

using SQS : queue model
using SNS : pub/sub model
using Kineis : real time data streaming model

* These services can scale independently from our application.

Date:

# Amazon SQS



* Simple Queue Service.
* Oldest AWS offering (10 years)
* fully managed service
  ↓
  ~serverless

* Used to decouple applications.

* Scales from 1 msg/sec to 10,000s per sec.

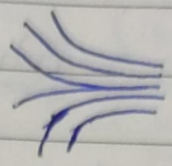
* Default retention of msgs: 4 days
  max: 14 days
* No limit on how many msgs can be in Queue.
* Messages are deleted after they are read by consumers.

* low latency (<10ms on publish & recieve)

* Consumers share the work to read messages and scale horizontally

Date:

# Amazon Kinesis

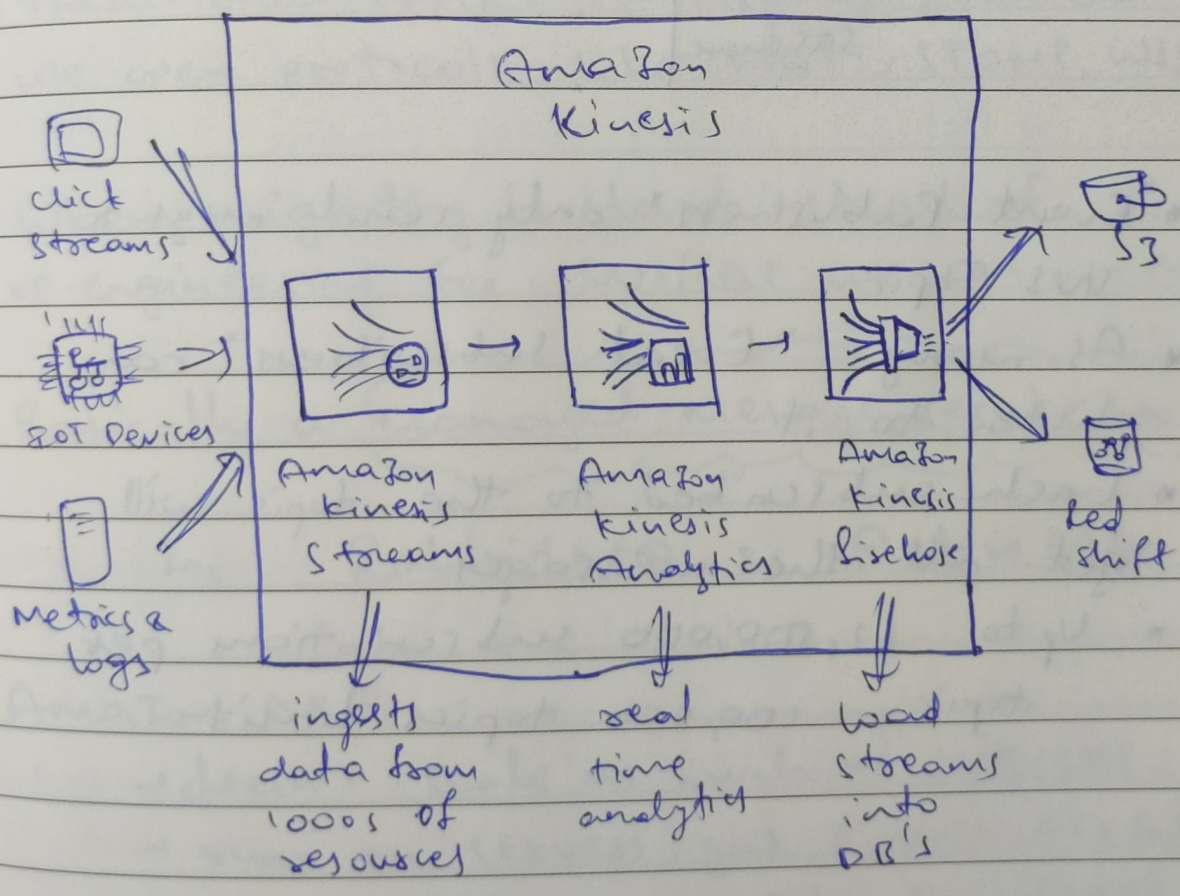


* for exam
  ⇓

Kinesis = real time data streaming

* Managed service to collect, process and analyze real-time streaming data at any scale.

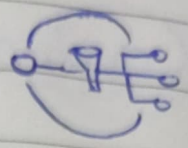
## Good to knows:-



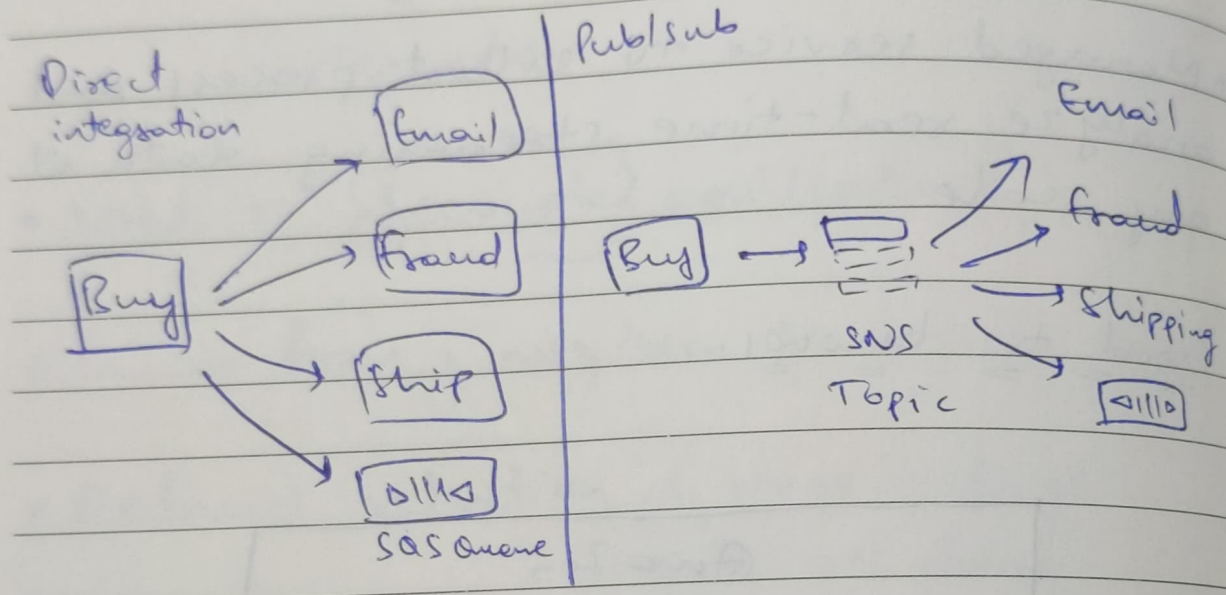

Kinesis Video Streams ⇒ monitor real-time video streams for analytics or ML.

Date:

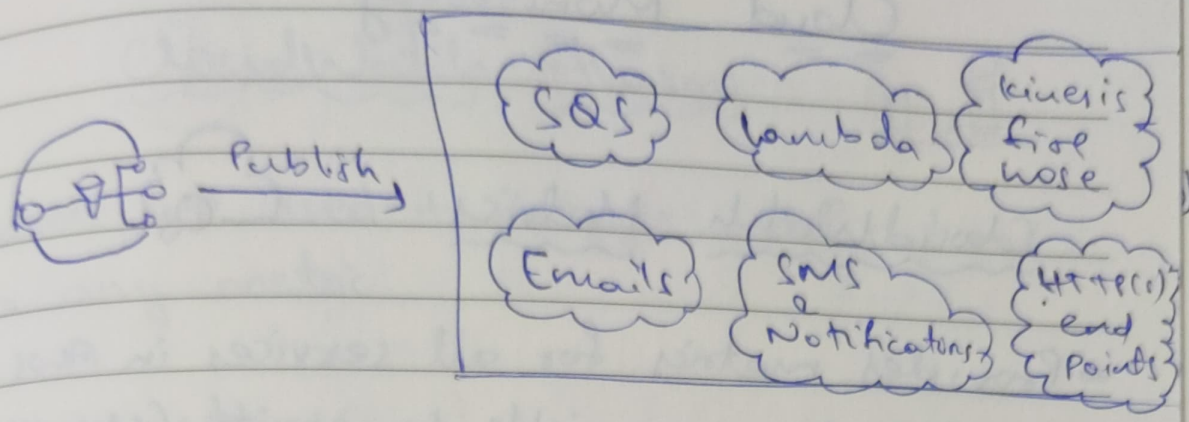
# Amazon SNS



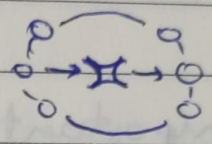
* Simple Notification Service.
* What if we want to send one msg to many receivers?




* "Event Publishers" only send msgs to SNS Topic.
* As many "Event Subscribers" can listen to it.
* Each subscribed to the topic will get all the messages
* Up to 12,500,000 subscriptions per topic, 100,000 topics limit




## Amazon MQ



* SNS & SQS are cloud native AWS proprietories.

* Traditional appln's running on-prem can use open protocols like MQTT, STOMP, WSS, etc

* When migrating to cloud, instead of re-engineering for SNS/SQS we can use MQ.

* Basically a managed message service broker for RabbitMQ & ActiveMQ

Amazon MQ
- → doesn't scale as much as SNS/SQS
- → runs on servers, good if multi-AZ & failover
- → has both Queue (~SQS) & topic (~SNS) features.

# Cloud Monitoring

## CloudWatch Metrics

* Provides metrics for all services in AWS
* Metric is a variable to monitor (CPU utilization)
* Metrics have timestamps
* Can create CloudWatch dashboards of metrics

## Important Metrics

* EC2 Instances:
  - → CPU Utilization, Status Checks, Network (Not RAM)
  - → Default metrics every 5 mins.
  - → Option for detailed monitoring ($$$) → metrics every 1 min.

* EBS Volumes: Disk Read/Writes.

* S3 Buckets: BucketSizeBytes, AllRequests, Number Of Objects.

* Billing: Total estimated charge (only us-east-1)

* Service Limits: usage of service API

Date:

* Custom metrics: push our own metrics.

# CloudWatch Alarms

* Alarms are used to trigger notifications for any metric.

* Alarm actions:
  - → Auto Scaling: ↑ or ↓ EC2 instance count
  - → EC2 Actions: stop, terminate, reboot or recover
  - → SNS notifications.

* Various options (sampling, %, max, min, etc.)
* Can choose the period to evaluate the metric.
* Ex:- Billing alarm.

Alarm states

OK | INSUFFICIENT DATA | ALARM

Handy on

Monitoring dashboard
(or)
Any specific service → Monitoring

US-East-1 → Billing Alarm.

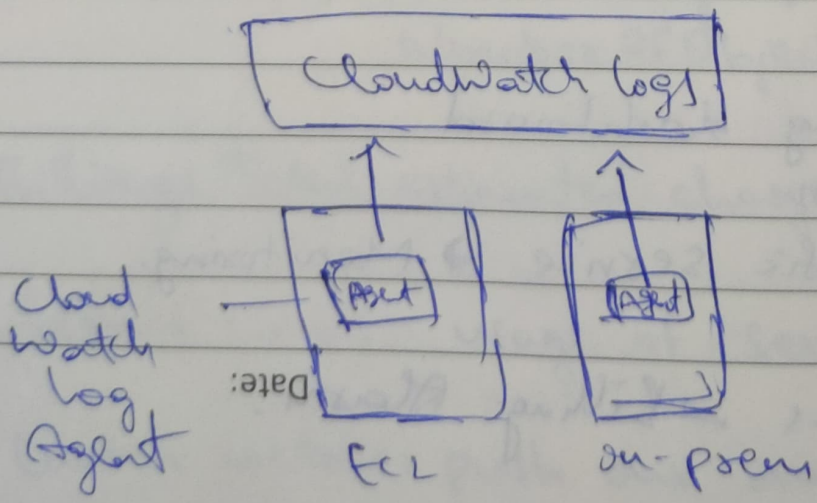
# CloudWatch logs

It can collect logs from
- → Elastic Beanstalk : logs from applⁿ
- → ECS : from containers
- → Lambda : from functional logs
- → CloudTrail based on filter
- → CloudWatch log agents installed on EC2 instances / on-prem servers
- → Route 53 : DNS Queries.

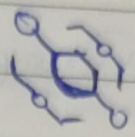
* Enable real-time monitoring of logs
* Adjustable retention period of logs.

## CloudWatch log Agent

* Can be setup on EC2 / on-prem server to send logs to CloudWatch logs
* Needs proper IAM permissions.



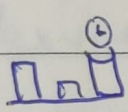
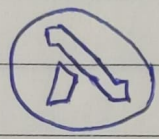
# Event Bridge



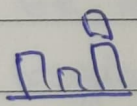
- formerly called CloudWatch Events.
- Based on something we can do some other thing

Ex:-

(i) Schedule : CRON Jobs

every hour  →  some lambda fn.

(ii) Event Pattern: based on something

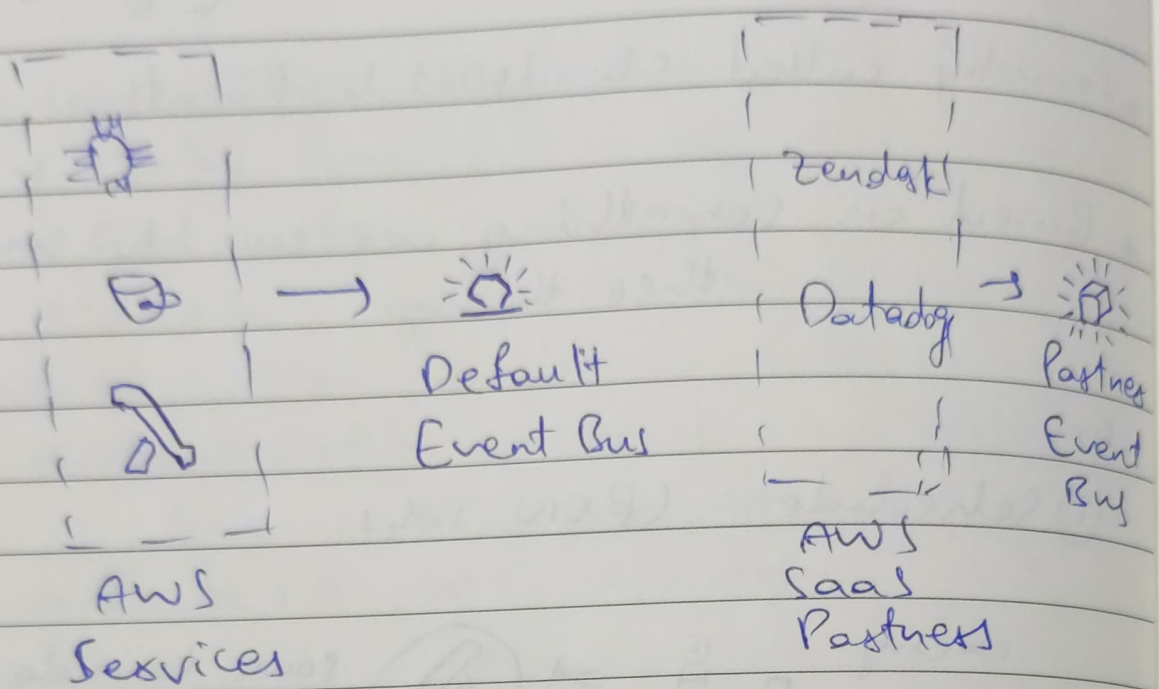
IAM Root User sign in  → SNS Topic with Email

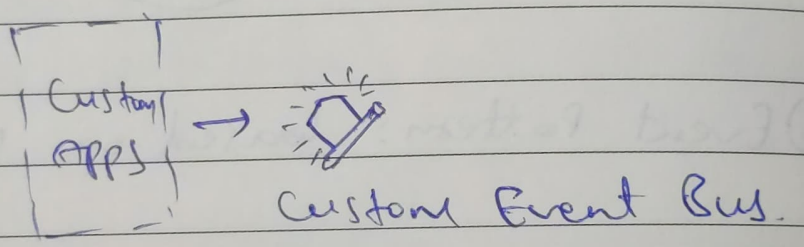
Many Sources & Many Destinations

↓

Compute, Integration, Orchestration & Maintanance.

# More capabilities.

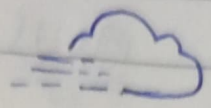



* Schema Registry: model event schema.
* We can archive events (all/filtered) and send to an event bus.
* Ability to replay archived events.

Date:

# CloudTrail



* Provides governance, compliance and audit for our AWS account
* Enabled by default.

* A history of events/API calls made from:
  - Console
  - SDK
  - CLI
  - AWS services

* for retention we can put logs from CloudTrail to CloudWatch Logs or S3.

* A trail can be applied to All Regions (default) or a single region.

* If a Resource is deleted in AWS, investigate CloudTrail first!!!

Any API call to be looked up

CloudTrail Insights:- detect unusual activity with API calls by continuously analysing CloudTrail mgmt events.

Date:

# AWS X-Ray

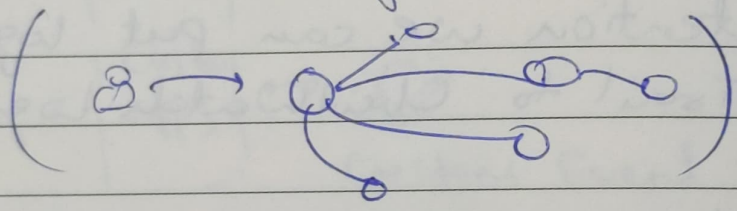
* Debugging one big monolith is easy but distributed services is hard, as we don't exactly know whats logging what.
* No common view for entire architecture

⇓

Enter AWS X-Ray

⇓

Visual Analysis of appln.



Adv's

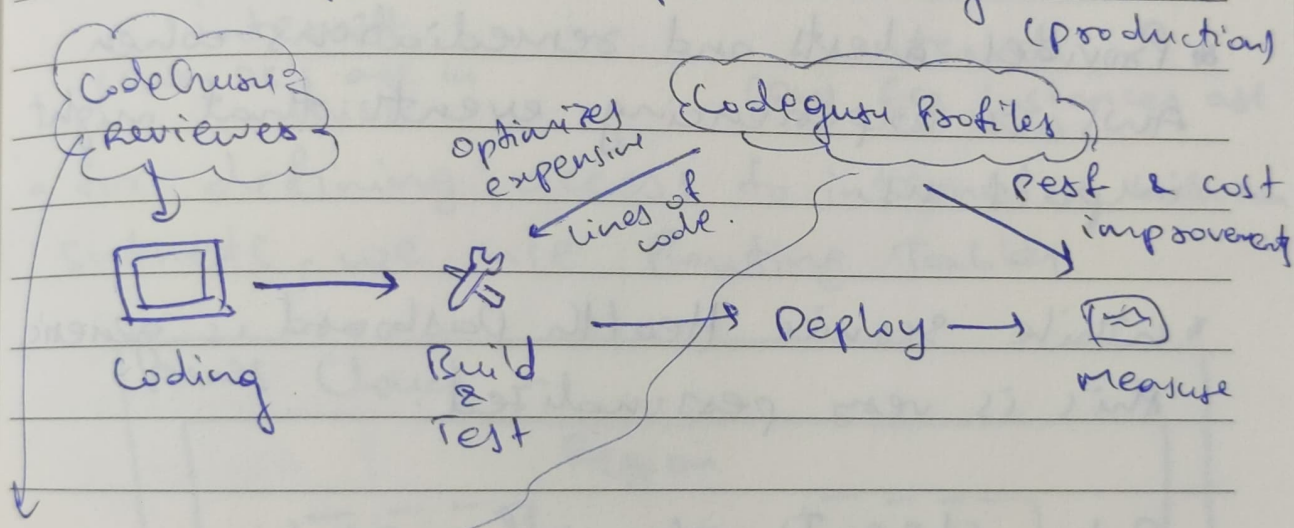
→ Troubleshooting perf. (bottlenecks)
→ Understanding deps in microservices
→ Pinpoint service issues
→ Review request behaviour
→ find errors & exceptions
→ Are we meeting SLA?
→ Where am I throttled?
→ Identify users that are impacted.

# CodeGuru

- ML Powered service
- Two functionalities

(i) CodeGuru Reviewer
→ automated reviews for static code analysis.

(ii) CodeGuru Profiler
→ visibility / recommendations about application perf during runtime (production)




- Currently supports Java and Python.
- Integrates with Github, BitBucket & AWS Code Commit.

Both AWS & on-Prem.
Minimal overhead on appln.

Date:

## AWS Status - Service Health Dashboard

* Shows all regions, all services health
* Historical info each day
* Has an RSS feed we can subscribe to.

status.aws.amazon.com.

## AWS Personal Health Dashboard

* Provides alerts and remediations when AWS is experiencing events that might impact us.

* While service Health Dashboard is generic, this is very personalized.

* Relevant & Timely info
* Proactive Notification
* help us plan scheduled activities.

phd.aws.amazon.com.

Date:

# VPC

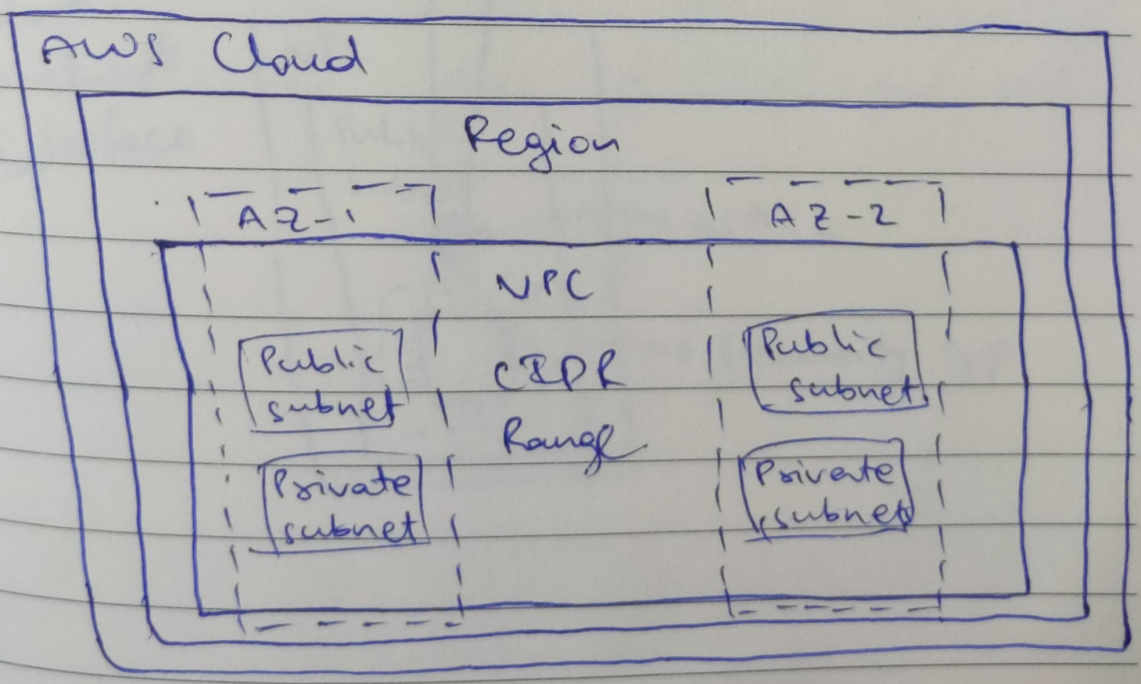
* Virtual Private Cloud.
  * private network to deploy our resources
  * Regional resource.

* Subnet
  * Partition in a VPC (in an AZ)

Private subnet
↓
not accessible in internet
(our DB's are in this)

Public subnet
↓
Accessible in internet.
(our EC2 instances are in this)

* for defining access to internet & between subnets, we use Routing Tables.




Date:

* Internet Gateway
  - helps our VPC instances connect with the internet
  - Public subnets have a route to internet gateway.

* NAT Gateway (AWS-Managed) & NAT Instances (Self-Managed)

  - allow our instances in private subnet to access the internet while remaining private.

P.T.O

Date:

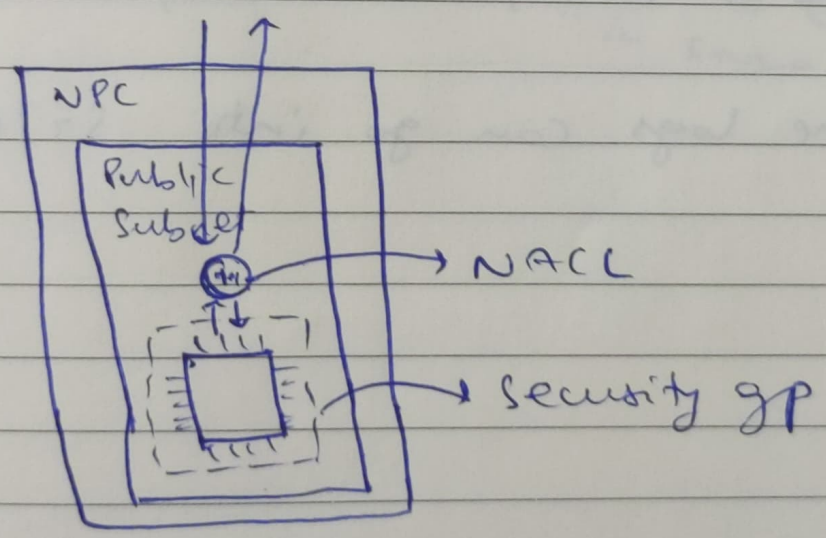
Lines of Defence

Network ACL & Security Groups

NACL
- firewall controlling traffic from and to subnet.
- can have ALLOW & DENY rules
- Are attached at subnet level
- Rules only include IP addresses

Security Groups
- firewall that controls traffic to & from an ENI / EC2 instance.
- can have only ALLOW rules
- Rules include IP's & othe sec. gp's.

Elastic Network Interface




Date:

# VPC flow logs

* Capture info about IP traffic going into your interfaces:
  - → VPC flow logs
  - → Subnet flow logs
  - → Elastic Network Interface flow logs.

* Helps to monitor & troubleshoot connectivity issues.

  Ex:- Subnet to Internet
  Subnet to subnet
  Internet to Subnet

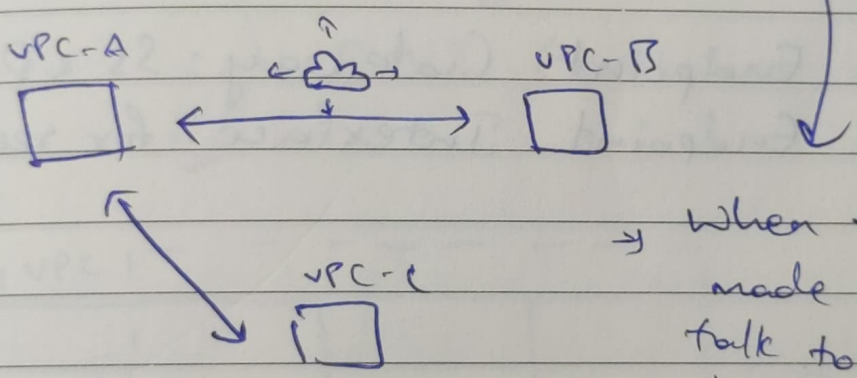
* Also get infor from AWS Managed services too ⇒ ELB, Elasticache, Aurora, etc..

* The logs can go into S3/CloudWatch logs

# VPC Peering

* Connect two VPC, privately using AWS network.
* Make them behave as if they were in same network.
* Must not have overlapping CIDR (IP Addresses).

* It is not transitive




→ When this is made B cannot talk to C as in same network

P.T.O

Date:

# VPC Endpoints

* Generally we are connected to all AWS services using public internet.
* Endpoints allow us to connect to the using a Private network.

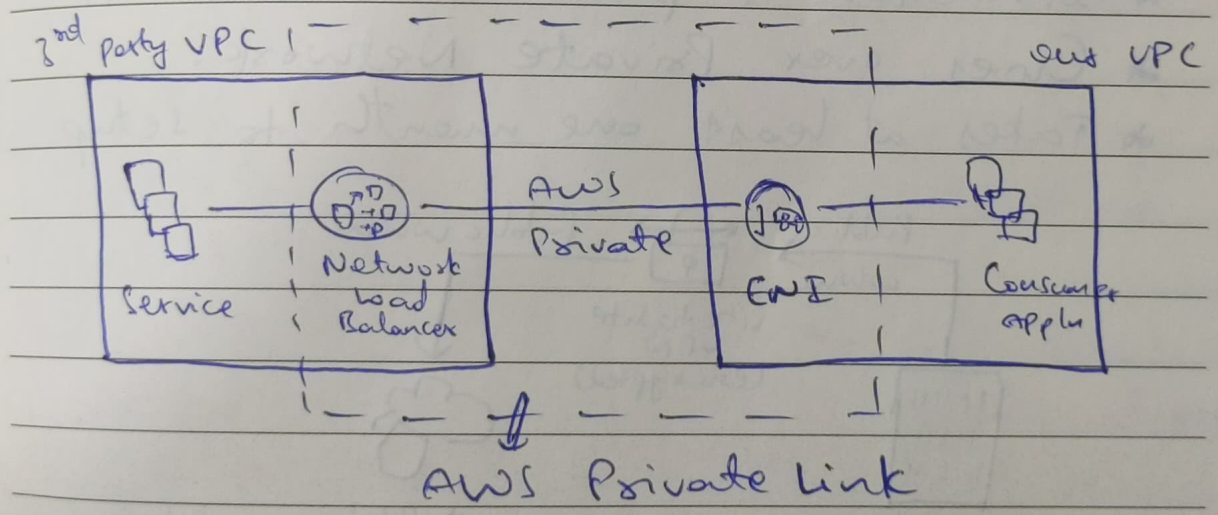
⇓

Enhanced Security,
lower latency.

VPC Endpoint Gateway: S3 & Dynamo
VPC Endpoint Interface: for rest.

# AWS Private Link

- Most secure and scalable way to expose a service to 1000's of VPCs
- Does not require VPC Peering, internet gateway, NAT, route tables, ...
- Requires a Network load balancer (service VPC) & ENI (Customer VPC)




Date:

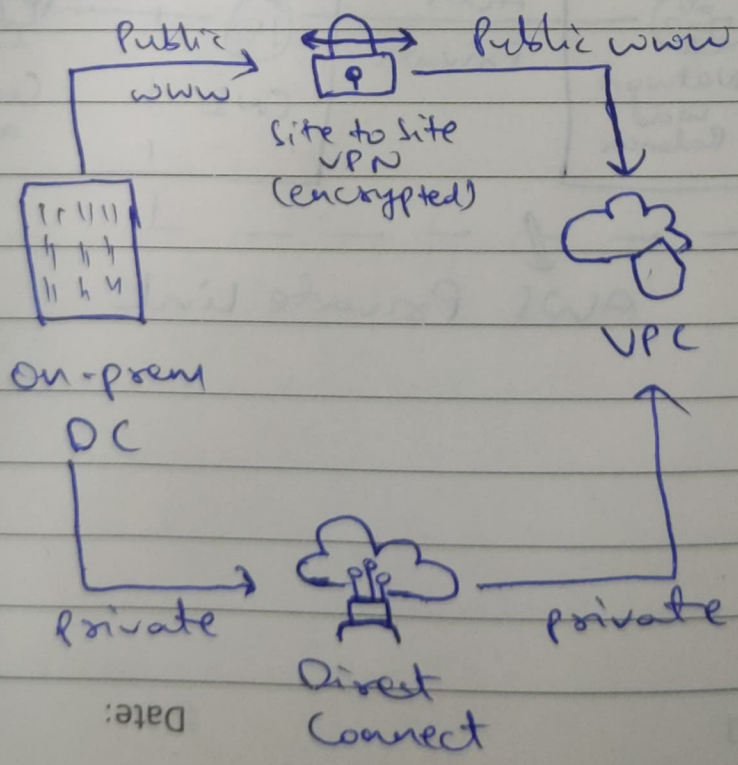
# Site to Site VPN & Direct Connect

## Site to Site VPN

* Connect an on-prem VPN to AWS
* Connection is automatically encrypted
* Goes over public internet
* Can be set up in like 5 mins

## Direct Connect (DX)

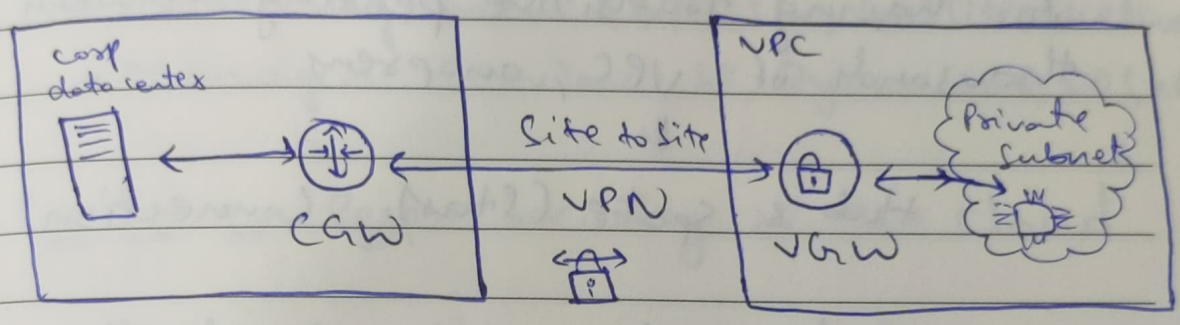
* Establish a physical connection between on-prem & AWS
* Connection is private, secure & fast
* Goes over Private Network
* Takes at least one month to setup




Date:

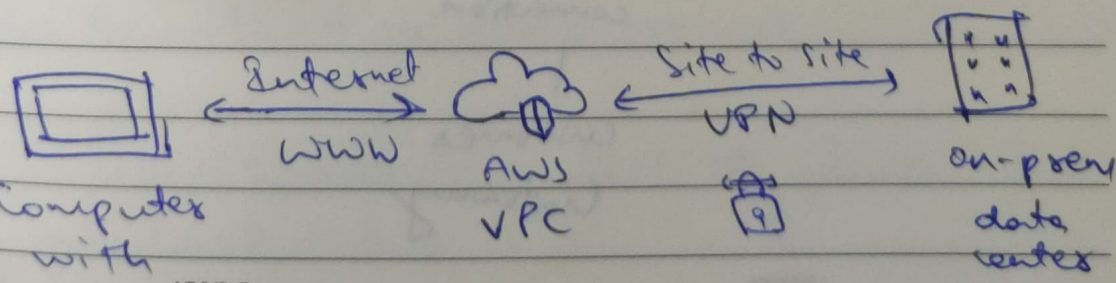
On-prem must have Customer Gateway (CGW)

AWS must have Virtual Private Gateway (VGW)



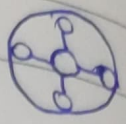

## AWS Client VPN

- Connect from your computer using Open VPN to our private network in AWS & on-prem.
- Allows us to connect to our EC2 instance over private IP (just as if we were in the private VPC network)
- Goes over Public internet




Date:

# Transit Gateway



* Using all the previous stuff, the network can become quite complex with routings

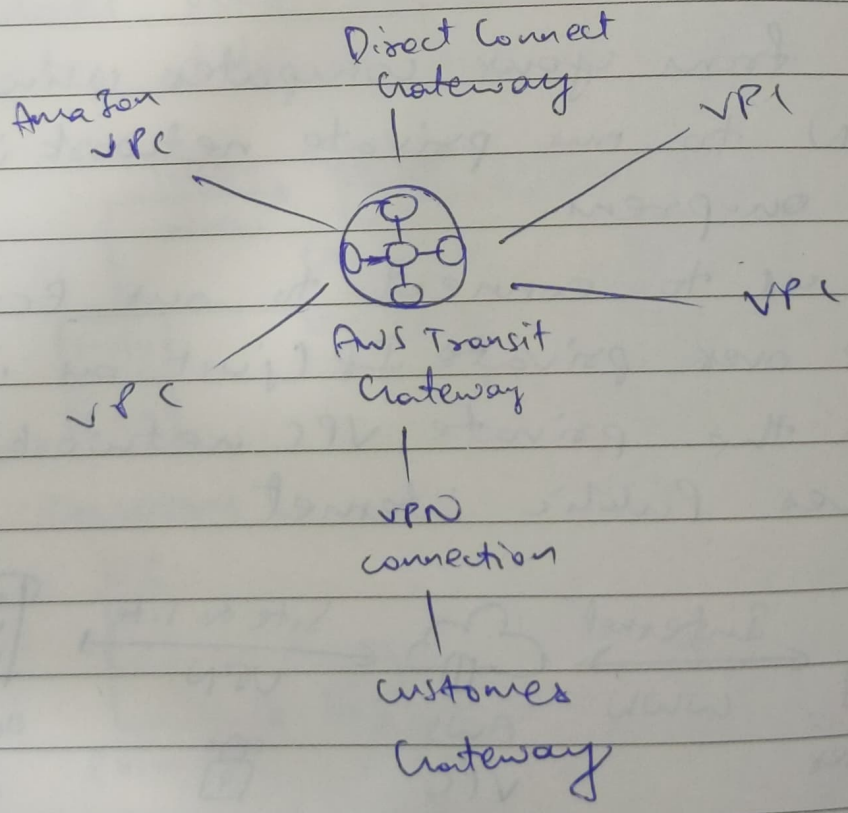
⇓

for having transitive peering between thousands of VPC, on-prem

⇓

Hub & Spoke (Star) Connection

* One single gateway to provide this.
* Work with Direct Connect Gateway VPN Connection




Date:

# Security & Compliance

## AWS Shared Responsibility Model

**AWS Resp** - Security of the cloud

* Protecting infra (h/w, s/w, facilities, network)
* Managed services: S3, DynamoDB, RDS, etc.

**Customer Resp** - Security in the cloud

* IAM, firewall, network config, OS patches in EC2 instances
* Encryption appl data.

**Shared controls**

* Patch Management
* Config Management
* Awareness & Training

Date:

# DDOS Protection: WAF & Shield

* Distributed Denial of Service.

hacker → master servers → bots → many many requests → Boom!

* AWS Shield Standard
  → Protects against DDOS attack for our website, applications, for all customers at no additional cost.

* AWS Shield Advanced
  → 24/7 premium DDOS protection

* AWS WAF: filter specific requests based on rules
  web Apple firewall

* Cloudfront & Route 53:
  → Available protection using global edges
  → Combined with AWS shield
  ↓
  Provides attack mitigation at edge.

* Be ready to scale - leverage AWS Auto Scaling

Date:

# AWS Shield

## AWS Shield Standard

* free service for all customers
* protection from SYN/UDP Floods, Reflection attacks and other layer 3/layer 4 attacks.

## AWS Shield Advanced

* $3000 per month per org.
* Protection against more sophisticated attacks on EC2, ELB, Cloudfront, S3.
* 24/7 Access to DDOS Response Team (DRP)
* Protect against higher fees during usage spikes due to DDOS.

Date:

P.T.O

# AWS WAF - Web Appln Firewall

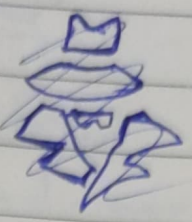
* Protects our web appln from common web exploits (layer 7) → HTTP.

* Deploy on Application Load Balancer, API Gateway, Cloudfront

* Define web ACL (Access Control List):
  - → Rules can include IP addresses, HTTP headers, body or URI strings
  - → Protects from common attacks
    - SQL injection
    - Cross Site Scripting (XSS)
  - → Size constraints, geo-match (block countries)
  - → Rate-based rules (to count occurences of events) - for DDos protection.

Date:

# Penetration Testing



* It is when we test our own appln.
* Some services such as EC2, RDS, Cloudfront, Aurora need no permissions, we can test them as we want

* Some are prohibited
  - → DNS Zone Walking
  - → DOS, DDOS, Simulated DoS or DDOS
  - → Post flooding
  - → Protocol flooding
  - → Request flooding

⇓

Basically whatever seems an attack on AWS.

⇓

we need to contact aws-security-simulated-event@amazon.com

# Encryption

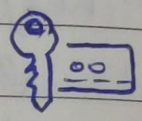
Two types

At rest: in storage

At transit: while moving from one location to other.

* We want to encrypt in both states to protect it & we use encryption key

## AWS KMS (Key Management Service)



* You hear encryption

⇓

Its KMS.

⇓

AWS Manages encryption keys for us.

⇓

We can say who have access.

* Encryption opt-in

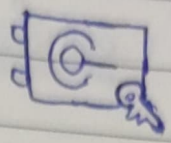
EBS, S3, Redshift, RDS, EFS

* Encryption Auto-enabled: CloudTrail, S3 Glacier

Storage gateway

Date:

## Cloud HSM



* AWS provides encryption hardware
* Dedicated hardware (HSM = Hardware Security Model)
* We manage our keys entirely
* The hardware is tamper resistant.

## Types of Customer Master keys : CMK

* Customer Managed CMK.
  - → Create, manage & used by customer
  - → rotation policy can be set
  - → bring-your-own-key possible

* AWS Managed CMK
  - → created, managed & used by AWS
  - → Used by AWS services

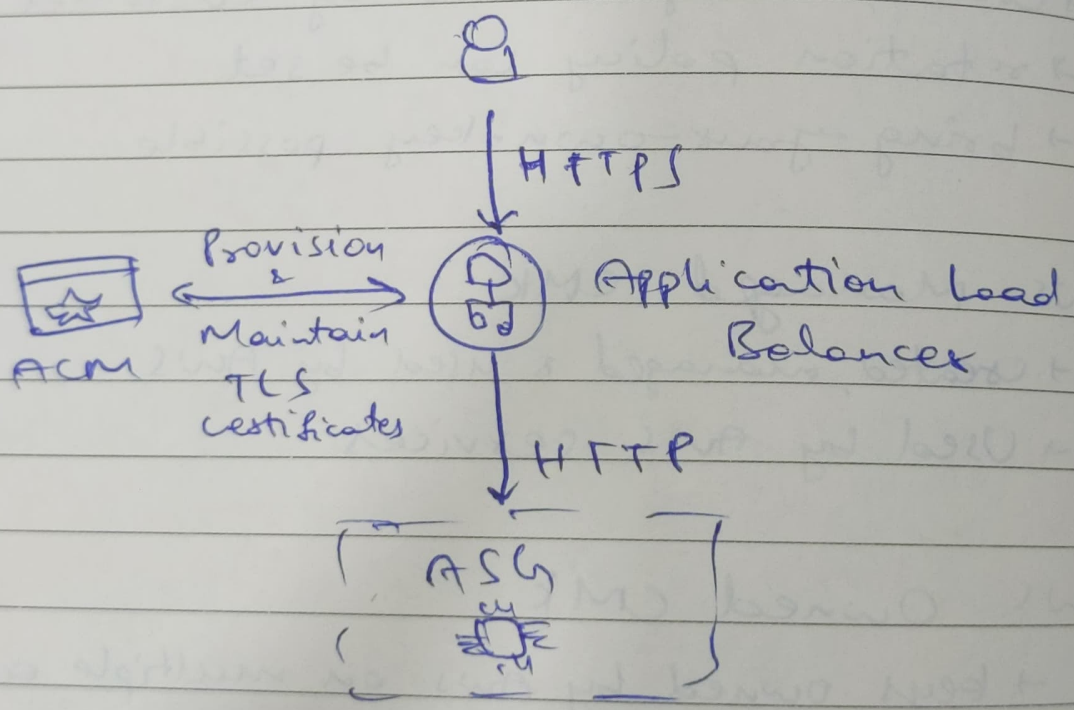
* AWS Owned CMK
  - → keys owned by AWS on multiple accounts
  - → we can't even view the keys

* CloudHSM keys (custom keystores)
  - → keys generated from our own cloudHSM.
  - → Cryptographic opn's are performed in our own cloudHSM clusters.

Date:

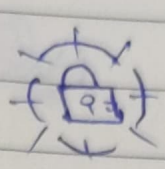
# AWS Certificate Manager (ACM)

* Lets us easily provision, manage and deploy SSL/TLS certificates
* Used to provide in-flight encryption for websites (HTTPS)
* Supports both public and private TLS certificates
* Free of charge
* Automatic certification renewal
* Integrations with
    * ELB, Cloudfront, API Gateway




Date:

## AWS Secrets Manager

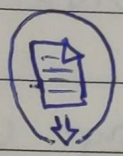


* Meant for storing secrets
* Capability to force rotation of secrets every X days.
* Automate gen. of secrets on rotation (✓)
* Integration with

  RDS MySQL, Postgres, Aurora

* Secrets are encrypted using KMS.

## AWS Artifact



* Portal that provides customers with on-demand access to AWS compliance documentation & AWS agreements
* Artifact Reports & Artifact Agreements
* Can be used to support internal audit and/or compliance.

Date:

# Amazon GuardDuty

* Intelligent threat discovery using ML
* One click enable (30 day trail), no s/w req
* Input data includes
  - Cloudtrail Event logs
    - unusual API calls
    - unauthorized deployments
  - VPC flow logs
    - unusual traffic
    - unusual IP addresses
  - DNS logs, Kubernetes Audit logs

* Can setup CloudWatch Event rules to be notified for any findings.

  ⇓

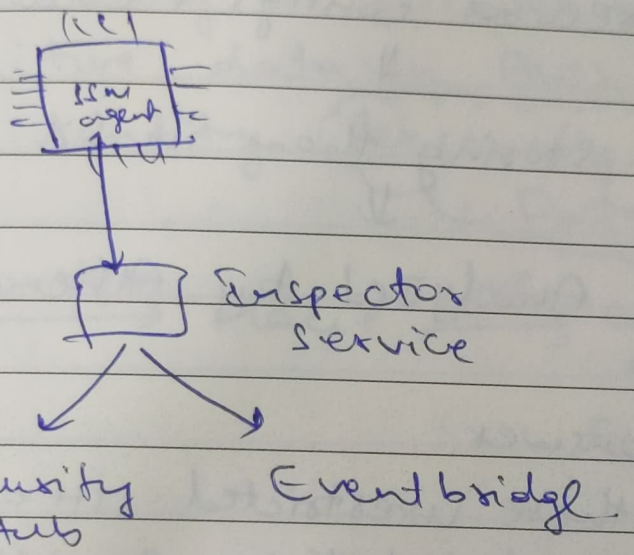
  can target lambda / SNS

* Can protect against Crypto Currency attacks (has dedicated finding for it)

Date:

# Amazon Inspector

* Automated Security Assessments
* It's for EC2 instances & ECR.




* It evaluates
  - Continuous scanning of infra when needed
  - Package vulnerabilities - database of CVE
  - Network Reachability (EC2)

* A risk score is associated with all vulnerabilities for prioritization.

Date:

# AWS Config

* Helps with auditing & recording compliance of our AWS resources.
* Helps record configs & changes over time.

↓

Storing them in S3

↓

Analyzed by Athena

* It can answer
  - → Is there unrestricted SSH access.
  - → Do my buckets have public access
  - → ALB changes over time.

↓

can receive alerts with SNS.

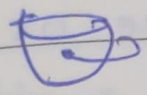
* It's a per region service but can be aggregated across regions and accounts.

* We can see CloudTrail logs and see who made what changes.

Date:

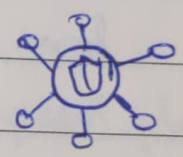
# Amazon Macie

* fully managed data security and data privacy service that uses ML and pattern matching to discover and protect our sensitive data in AWS.
* Used to identity (Personally Identifiable Data)




$\xrightarrow{\text{analyze}}$ Macie $\xrightarrow{\text{notifies}}$ Cloudwatch events (or) Event Bridge → integration

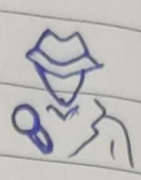
# AWS Security Hub



* Central Security Tool to manage security across several AWS accounts and automate security checks.
* Integrated dashboard showing stuff and actions from
  - GuardDuty, Inspector, Macie.
  - IAM Access Analyzer
  - AWS System/ Firewall Manager
  - AWS Partner Network Soln's
* Must first enable the AWS Config Service.

Date:

## Amazon Detective



* All the prev. tools can identify issues
* Sometimes finding root cause is complex

⇓

Enter Detective

↓

Analyzes, investigates and quickly identifies the root cause of security issue or suspicious activity (using ML & graph)

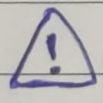
⇓

Automatically collects and processes events from VPC flow logs, CloudTrail, GuardDuty & create Unified view.

⇓

Will get to the root cause.

## AWS Abuse



* We can report suspected AWS resources used for abusive or illegal purposes
* Ex:- Spam, Port Scanning, DoS or DDOS, Intrusion, Hosting objectionable or copyrighted content, distributing malware

* Contact: AWS Abuse team or abuse@amazonaws.com.

# Root User Privileges

Root

- Root User = Account Owner
  ↓
  Has complete access

- lock away our AWS account root user access keys!
- Do not use root account for everyday tasks, even admin tasks.

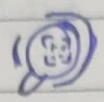
- Actions that can be performed by root user only are
  - → Change account settings
  - → Close our AWS account
  - → Change or Cancel AWS support plan
  - → Register as a seller in Reserved Instance Marketplace

& view invoices, restore IAM perms, enable MFA for S3, edit/delete S3 policy, sign up for GovCloud.

Date:

# Machine Learning

## Rekognition:-

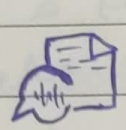


* find objects, people, text, scenes in images and videos using ML.
* Facial Analysis & facial Search.
* Create a DB of familiar faces or compare against celebs.

* Use Cases:-

labeling, Content Moderation, Text Detection, face detection & analysis, face search and verification, celebrity recogn. Pathing (sports game analysis).

## Transcribe



* Automatically converts speech to text
* Uses Deep learning process called Automatic Speech Recognition (ASR)
* Automatically removes PII using Redaction.
* Supports Automatic Language Identification for multi-lingual audio.

* Use cases:-

customer service calls, CC & subtitles, metadata for media assets for searching

Date:

## Polly

* Turns text to speech using DL

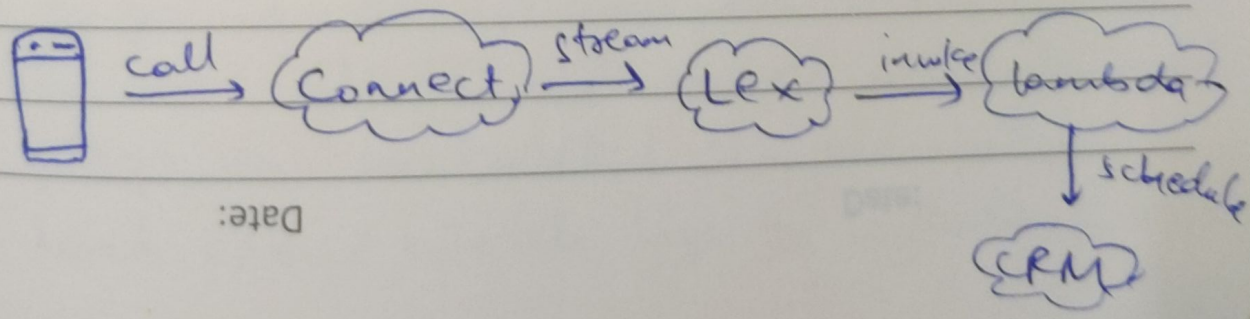
## Translate

* Natural and accurate language translation
* Internationalization can be done

## Lex & Connect

* Lex → Same tech that powers Alexa
  → ASR to convert speech to text
  → NLP & understands intent of text, call
  → Helps build chatbots, call center bots.

* Connect
  → Receive calls, create contact flows, cloud-based virtual contact center
  → Can integrate with CRM systems (AWS
  → No upfront payments, 80% cheaper than traditional call center soln's




Date:

## Comprehend

* for NLP
* fully managed and serverless.
* Uses ML to find
  - language of text
  - Extracts key phrases, people, brand, event
  - Positive/Negative.
  - tokenizations
  - topic classification
* Use cases
  - Analyze Customer Interactions
  - Group Articles.

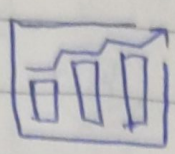
## SageMaker

* fully managed service for developers/ data scientists to build ML models.
* Helps us to do all the processes in one place + server provision
* Blah.

Basic ML stuff.

Date:

## Forecast



* fully managed service that uses ML to delivers highly accurate forecasts.
* 10% more accurate than looking at data.
* forecasting time from months to hours
* Use cases
  - Product Demand planning
  - Financial planning
  - Resource planning

## Kendra

* fully managed document search service (ML)
* Extracts answers from within a document
* Natural Language Search Capabilities
* learn from users interactions (Incremental Learning)
* Ability to manually fine-tune search results (importance, freshness, et..

Date:

## Personalize

* fully managed ML-Service to build apps with real time personalized recommendations
* Same tech used by Amazon.com.
* Integrates into
  websites, apps, SMS, mail, etc.
* Implementable in days
* Use cases : retail stores, media, entertainment.

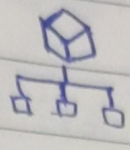
## Textract

* Automatically extracts text, handwriting, and data from any scanned documents using AI/ML
* Extract data from form/table.
* Use cases:-
  financial, Healthcare, Public sector.

Date:

# Account Mgmt, Billing & Support

## AWS Organisations



* Global Service
* Allows to manage multiple AWS Accounts
* Main Acct → Master Acct, other are child
* Cost Benefits
  (i) Consolidated Billing
  (ii) Aggregated Usage
  (iii) Pooling of Reserved instances.

* AWS Account Creation can be automated with API.
* Can Restrict account privileges using Service Control Policies (SCP).

## Multi Account Strategies

* Create accts per dept, cost center, dev/test/prod, regulatory restrictions, resource isolation (VPC), logging.

* Multi Acct vs One Acct Multi VPC
* tagging standards for billing purposes
* Enable CloudTrail on all accounts, send logs to centralized S3 account
* Send CloudWatch logs to centralised acct

Date:

# Service Control Policies (SCP)

* Whitelist or Blacklist IAM actions.
* Applied at Org. Unit / Account level.
* Does not apply to Master Account
* SCP is applied to all the Users and Roles of the account, including Root.
* SCP does not restrict service linked roles
  ↳ these help use AWS services.

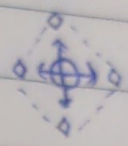
* SCP must have an explicit allow (does not allow anything by default).
* Use cases:
  → Restrict access to certain services
  → Enforce PCI compliance by explicitly disabling services.

Deny > Allow.

# Consolidated Billing

- Combined Usage and One Bill.
- Mgmt account can turn off Reserved Instances discount sharing for any account in org including itself.

## Control Tower



- Easy way to setup and govern a secure and compliant multi-account AWS env. based on best practices.
- Benefits
  - → Automated setup with a few clicks
  - → Automate ongoing policy mgmt using guardrails
  - → Detect policy violations and remediate them
  - → Monitor compliance through interactive dashboard.
- AWS Cloud Tower runs on top of AWS organizations:
  - → It automatically sets up AWS orgs to organize accounts and implement SCP's.

Date:

# Pricing Models in AWS

(i) Pay as you go

(ii) Save when you reserve

(iii) Pay less by using more

(iv) Pay less as AWS grows.

* There are many free tier stuff

## Compute Pricing - EC2

* Pay for what we use
* No. of instances
* Config of them - Region, OS, type, size
* ELB running time & amt of data processed.
* Detailed Monitoring.

On-demand Instances
Reserved Instances
Spot Instances
Dedicated Hosts
Savings Plan.

Lambda → Pay per call & duration
ECS → AWS resources allocated underneath
Fargate → for vCPU & Memory resource allocated (instances)

Date:

# Storage Pricing

S3 - Standard, IA, One Zone IA, Intelligent Tiering
Glacier, & Glacier Deep Archive

- No. & size of objects.
- No. & type of requests.
- Data transfer OUT of S3.
- Transfer acceleration
- Lifecycle transitions.

Similar → EFS → pay per use, IA, lifecycle rules.

EBS -

- Volume type
- Storage volume in GB per month provisioned
- IOPS
  - Gen. purpose SSD
  - Provisioned IOPS SSD
  - Magnetic.
- Snapshots
  - Added data cost per GB per month
- Data transfer
  - Outbound data transfer are tiered for volume discounts.

Date:

# Database Pricing

## RDS :-

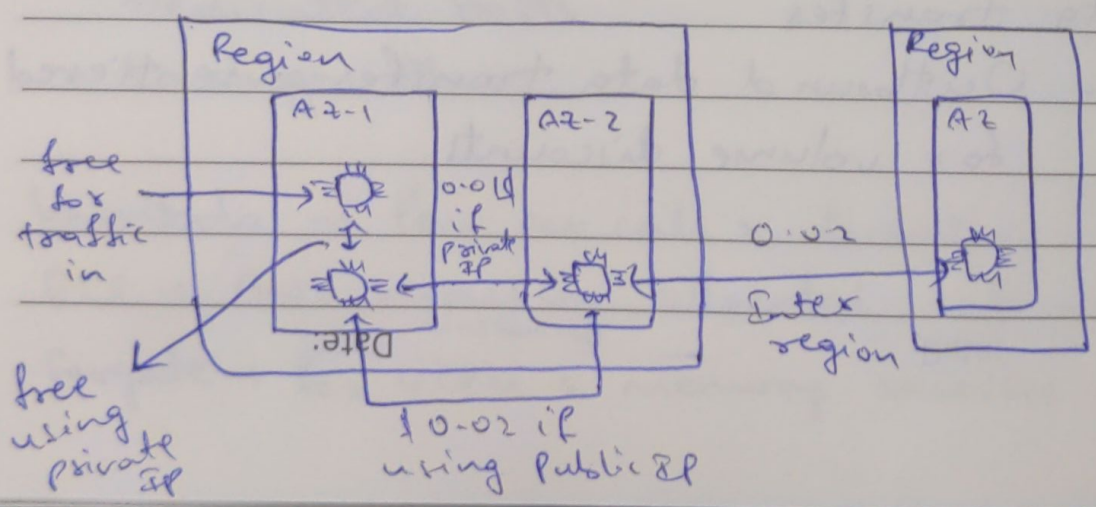
- Per hour billing
- Characteristics of DB
- Purchase type (on-demand/ Reserved)
- Backup storage is free.

- Additional storage (per GB per month)
- Number of IOPS.
- Deployment type (Single AZ, Multi AZ)
- Data transfer (OUT is cost, IN is free)

## Content Delivery - Cloudfront

- different regions, diff costs.
- Aggregated for each edge location, then applied to our bill.
- No. of HTTP/HTTPS requests.
- Data Transfer OUT.

## Network Costs per GB



# Savings Plan

* Commit a certain $ amount per hour for 1/3 yrs
* Easiest way to setup long-term commitment in AWS

* EC2 Savings Plan
  - 72% discount
  - Commit to usage of instance families (C5, M5)
  - Regardless of AZ, size, OS, tenancy
  - Upfront, partial & no upfront

* Compute Savings Plan
  - 66% discount
  - Regardless of family, region, size, OS, tenancy, compute options

    ↓

    EC2, fargate, lambda.

* Machine learning Saving Plan: SageMaker
* Setup from AWS Cost Explorer Console.

Date:

# AWS Compute Optimizer

* Reduce cost & improve perf by recommending optional AWS resources for our workload.
* Helps us choose optimal configs (over/under provisioned)
* Uses ML to analyze resource config and their utilization CloudWatch metrics.
* Supported Resources
  - EC2, ASG, EBS, Lambda
* Lower costs upto 25%.
* Recommendations can be exported to S3.

# Billing & Costing Tools

## AWS Pricing Calculator

* Estimate the cost for our architecture per year

## AWS Billing Dashboard

* Overview of cost for month

## AWS free Tier Dashboard

Date:

* Shows free tier usage & forecast

## Cost Allocation Tags

* Use to track our AWS costs on detailed lev
* AWS generated tags
  * Automatically applied to the resource we create
  - → Starts with prefix aws (aws: CreatedBy)
* User Defined tags
  - → Defined by User
  - → Starts with prefix user: (owner, cost center, application)

## Tagging & Resource Groups

* Tags can be used for organizing resources
  - → EC2: instance, image, load Balancer, sec gp.
  - → RDS, VPC Res, Route 53, IAM users, etc.
  - → Resources created by Cloud formation are all tagged the same way

* It's free naming, common tags are Name, Env, Team.

* Resource gps can be used to
  * create, maintain & view a collection of Resources that share common tags.
  - → Tag Editor

Date:

## Cost & Usage Reports

* Dive Deeper into AWS costs & Usage.
* Contains the most comprehensive set of AWS cost & usage data available.
* Lists usage of each service category used by an acent & it's IAM users in hourly or daily line items, as well as tags.

* Can be integrated with Athena, Redshift & Quicksight.

## Cost Explorer

* Visualize, understand & manage your AWS costs and usagl
* Custom reports.
* Analyze data at high level
* Monthly, hourly, resource level, granularity
* Will help us choose an optimal Saving & Plan

* forecast usagl upto 12 months based on prev usage

Date:

## Billing Alarms in CloudWatch

* Only us-east-1
* Billing data are for overall costs
* It's for actual cost, not projected
* Intended a simple alarm.

## AWS Budgets

* Create Budget and send alarm when costs exceed budget
* 3 types → Usage, Cost, Reservation.
* for Reserved Instances
  - → Track utilization
  - → Supports EC2, Elasticache, RDS, Redshift
* Upto 5 SNS notifications per budget
* can filter by :- Service, Tag, Purchase Type, Region, AZ, etc.
* Same options as AWS cost explorer.
* 2 budgets are free, then $0.02/day/budget

## Trusted advisor

* No installation, high level AWS acount assessment

* Analyzes & provides recommendation on
  - → Cost Optimization
  - → Performance
  - → Security
  - → fault tolerance
  - → Service limits

(Cost Per Refase)

## Support Plans

| 7 Core checks | 4 Full checks |
|---|---|
| Basic & Dev Support plan | Business & enterprise Plan |
| S3 Bucket Permission | * Full checks on all 5 categories |
| Security Grps | * Cloudwatch alarms on reaching limits |
| IAM use | * Programmatic Access Using AWS Support API |
| MFA on root account | |
| EBS Public snapshot | |
| RDS Public snapshot | |
| Service limits | |

# Support Plans for AWS

## (i) Basic Support Plan

* free.
* Customer Service & Communities
* 7 checks on Trusted Advisor
* AWS Personal Health Dashboard

## (ii) Development Support Plan

(i) +

* Business hours Email Access
* Unlimited cases / 1 Primary contact
* Case response times
  - General < 24 business hrs
  - System impaired < 12 " "

## (iii) Business Support Plan

* for Production Workloads
* full checks + API → Trusted Advisor
* 24*7 phone, email & chat access
* Unlimited cases / Unlimited contacts
* Access to Infra Event Mgmt at additional cost
* Response times → (ii) +
  - Prod. system impaired < 4 hrs
  - Prod system down < 1 hr.

## (iv) Enterprise On-Ramp Support Plan

* Prod or Business Critical Appln's
* (iii) +
* Pool of Technical Account Managers (TAM)
* Concierge Support Team (billing & acunt best practices)
* Infra Event Mgmt, Well Architectured & Ops Reviews.
* Response times

  (iv) +

  Business critical system down 30 min

## (v) Enterprise Support Plan (24/7)

* Mission Critical workloads
* (iv) +
* designated TAM
* Concierge Support Tem
* Infra Mgmt, Reviews.
* Response time.

  (v)

  + Business-Critical system down 15 min

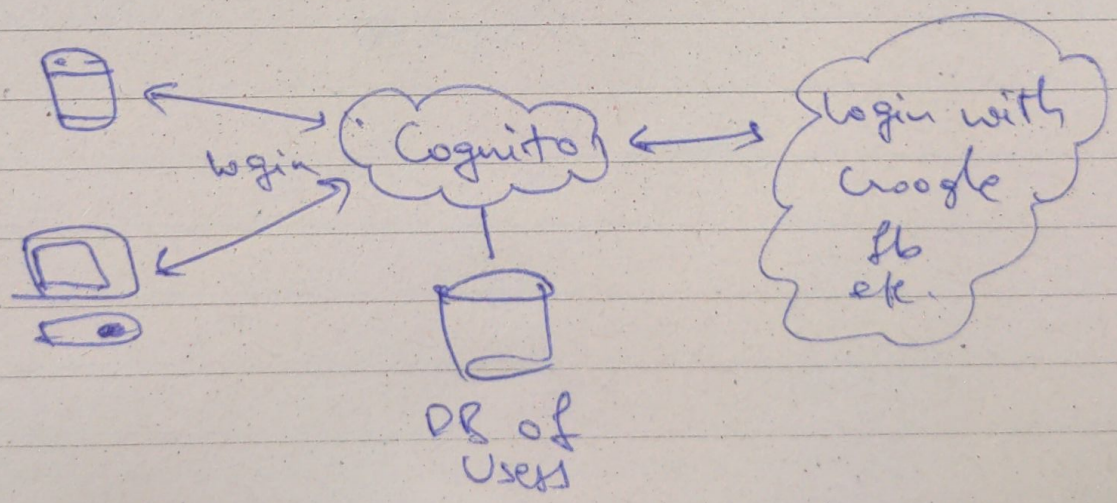
# Advanced Identity

## AWS STS

* Security Token Service
* Enables us to create temp, limited privileges credentials to access our AWS Resources.
* Short term creds: expiry date.
* Use Cases
  - → Identity federation
  - → IAM Roles for cross/same account access
  - → IAM Roles for EC2

## Cognito

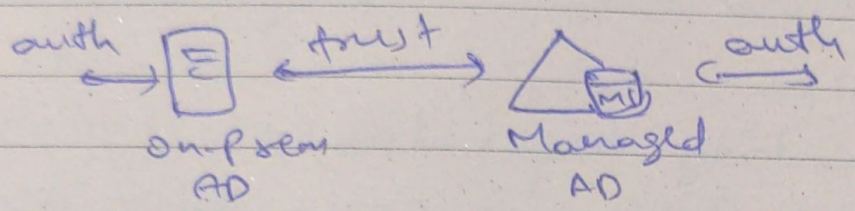
* Identity for our web/Mobile apph users
* Instead of creating them IAM user we create cognito user.



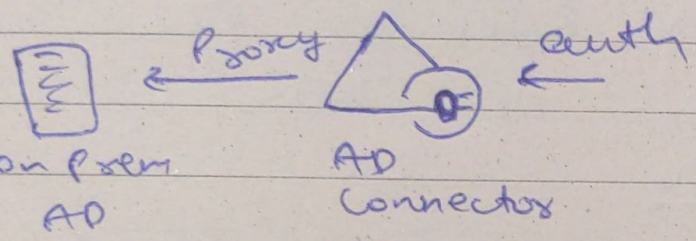
# AWS Directory Services

* Extends Microsoft Active Directories
  (logging in anywhere in our company)

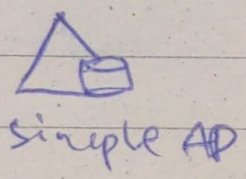
## AWS Managed Microsoft AD




## AD Connector




## Simple AD



# AWS IAM Identity Center

* Successor to AWS Single Sign On.

* One login for all
  - → AWS accounts in org
  - → Business cloud apps
  - → SAML 2.0
  - → EC2 Windows Instances

* Identity Providers
  - → Built in Identity Store in IAM Identity center
  - → 3rd party: Active Directory, One login, Okta.

# Other Services

## Amazon Workspaces

- Managed DaaS
- virtual Desktop provision.
- fast & scalable, secure (KMS), Pay as you go
- Can make multi-Region to min. latency

## AppStream 2.0

- Desktop Application Streaming Service
- Appln is delivered from within a web browser.
  Ex:- Blender on chrome/firefox.

## Sumerian

* Create and run VR, AR & 3D apps
* Ready to use templates.
* Accessible via Browser.

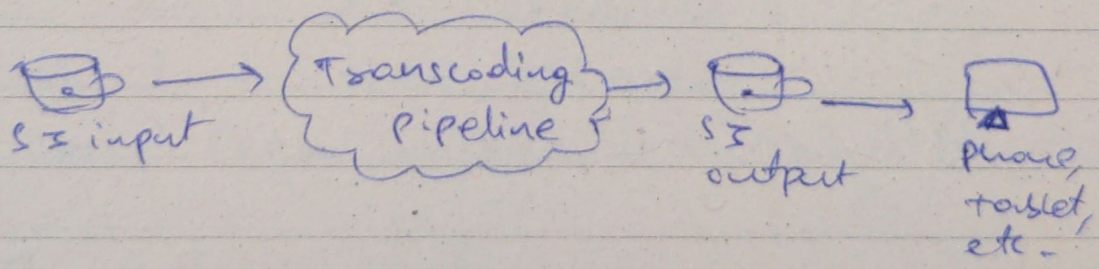
## IoT Core

* easily connect IoT Devices with cloud
* Serverless, secure & scalable.

## Elastic Transcoder

* Used to convert media files storage in S3 to media files in formats required by consumer playback services (phones, etc..)
* Benefits.

  easy to use, scalable, cost effective, manage & secure, pay for what you use.



## AppSync

* Backend for our app web in real time
* Uses GraphQL
* Automatic client code generation
* Integrates with DynamoDB, Lambda
* Real time subscriptions.
* Offline data synchronizations
* Fine grain security.

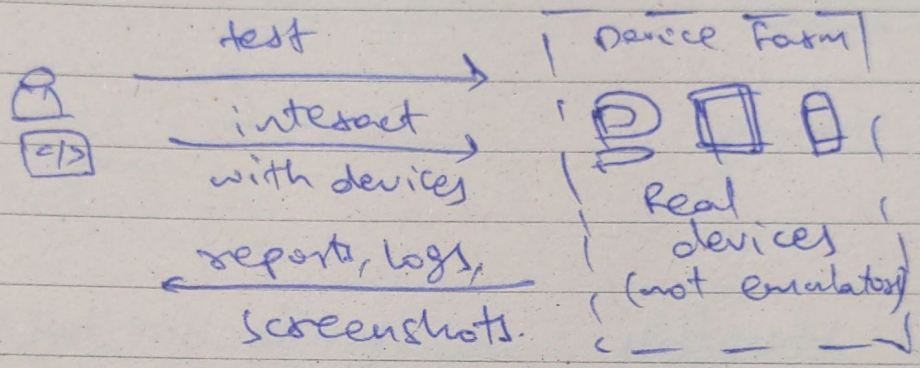
## Amplify

* Set of tools & services that helps us develop and deploy scalable full stack web & mobile appns.

* Auth, Storage, API, CI/CD, PubSub, Analytics, AI/ML, Monitoring, Source Code, etc.

## Device farm

* fully managed service that tests our web & mobile apps against multiple browsers, real mobile devices & tablets.
* Run tests concurrently on all devices
* Ability to configure device settings (Bluetooth, wifi, etc..)



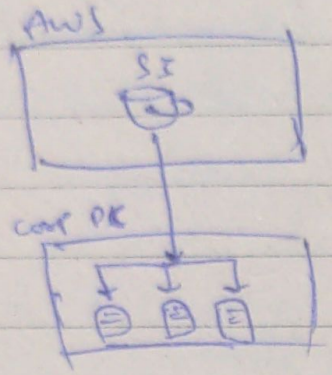

## Backup

* fully managed service to centrally manage and automate backups across AWS services.
* On demand & scheduled backups
* Point-in-time recovery.
* Cross Region backup, retention, lifecycle

Backed up into S3.

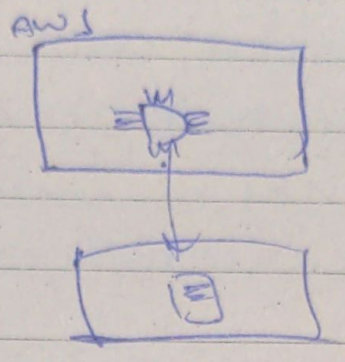
# Disaster Recovery Strategies

## Backup & Restore
cheapest



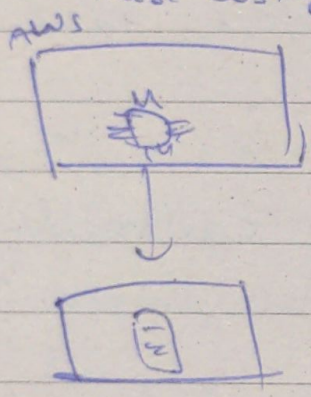

Just storing

## Pilot Light
bit costly



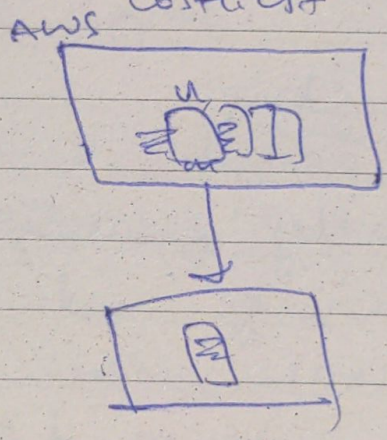

core fn's run in cloud.

## Warm Standby
more costly




full version of app but at small size.

## Multi-site/hot-site
costliest

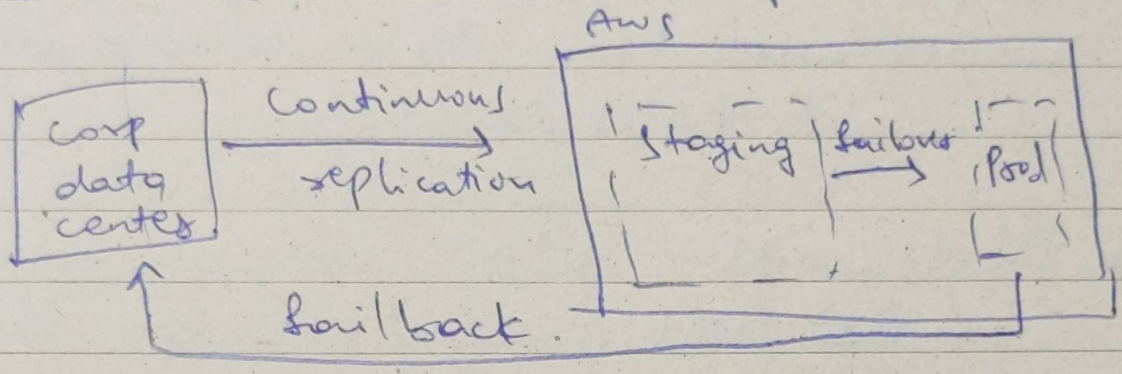



full version at full scale.

Also Failover.

# Elastic Disaster Recovery

* Quick recovery of data center.
* Continuous block level replication




# Datasync

* Move large amount of data from on-Prem to AWS.
* Can Synchronize to S3, EFS, FSx.

* Scheduled hourly, daily, weekly
* The replication tasks are

incremental after first load

## Application Discovery Service

* When we are moving totally from on-prem to AWS, this will help us understand how to move our data, in what order, etc..

* Two ways
  - → Agentless
  - → Agent-based

* Result can be viewed in Migration Hub.

## Application Migration Service (MGN)

* To actually move from on-prem to AWS
* Lift & shift.
* Replication agent installed in our on-prem.
* At last we cutover.

## Fault Injection Simulator (FIS)

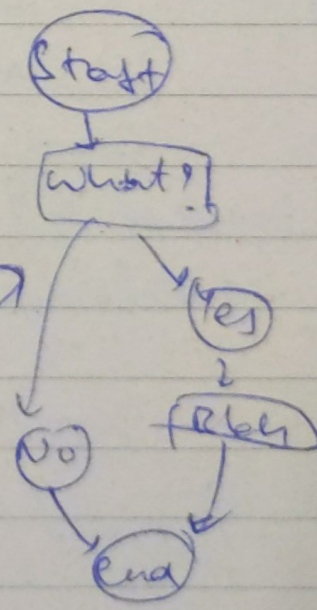
* Fully managed service for running fault injection on AWS workloads.
* Based on

Chaos Engineering

* Helps uncover hidden bugs & stuff.
* Use pre-built templates that generate desired disruptions.

## Step functions

* Build serverless visual workflow to orchestrate lambda fn's
* Features: sequence, parallel conditions, error handling.
* Integrate with ECS, EC2, on-prem, gateway.
* Possible to human approval




* Use Cases:-
  - Order fulfillment
  - Data Processing
  - web applns
  - Any workflow

# Ground Station

* fully managed service that lets us control satellite comms, process data, & scale satellite ops.
* Provides a global network of satellite ground stations near AWS Regions.
* Allows us to download satellite data within seconds into VPC
* Send data to S3 or EC2

* Use cases:
  - → Weather forecasting
  - → surface Imaging
  - → Communications
  - → Video broadcasts.

# Pinpoint

* Scalable 2-way (inbound/outbound) marketing comm service.
* Supports E-mail, SMS, push notifications, in-app messaging.

* Ability to personalize & segment customers.
* Possible to receive replies.
* Scales to Billions of msgs per day.

* Use cases:
  - → Run campaigns
  - → Marketing
  - → Bulk transactional SMSs.

An evolution of SNS or SES (email) where we need not manage the audience too.

# AWS Architecting & Ecosystem

## General guiding principles

* Stop guessing capacity needs
* Test systems at production scale
* Automate to make architectural experimentation easier
* Allow for evolutionary architectures
  - Design based on changing reqs
* Drive architectures using data
* Improve through game days
  - Simulate apps for flash sale day

## Design Principles

* Scalability
* Disposable Resources
* Automation
* Loose Coupling
* Think not Services, not Servers!

# 6 Pillars

## (i) Operational Excellence

* Includes the ability to run and monitor systems to delivers business value and to continually improve supporting processes & procedures.

* Design Principles
  - → Performs operations as code.
  - → Annotate documentation
  - → Make frequent small, reversible changes
  - → Refine procedures, frequently
  - → Anticipat failure
  - → Learn from failures.

Prepare → CloudFormation, Config

Operate → CloudFormation, Config, CloudTrail, CloudWatch, X-Ray

Evolve → CloudFormation, CodeBuild, CodeCommit, CodeDeploy, CodePipeline.

## (ii) Security

* we know what it is

* Design principles
  - → Implement a strong identity foundation
  - → Enable traceability
  - → Apply security at all layers
  - → Automate security best practices
  - → Protect data in transit & at rest
  - → Keep ppl away from data
  - → Prepare for security events.

IAM → IAM, STS, MFA, Org

Detective Controls → Config, CloudTrail, Cloudwatch

Infra Protection → Cloudfront, VPC, Shield, WAF, Inspector

Data Protection → KMS, S3, ELB, EBS, RDS

Incident Response → IAM, Cloud Formation, Cloudwatch Events

## (iii) Reliability

* Basically appln runs no matter what

* Design Principles
  - → Test recovery procedures
  - → Automatically recover from failure
  - → Scale Horizontally
  - → Stop guessing capacity
  - → Manage changes in automation

Foundations → IAM, VPC, Quotas, Trusted Advisor

Change Mgmt → ASG, CloudWatch, CloudTrail, Config

Failure Mgmt → Backups, Cloud formation, S3, S3 glacier, Route 53

## (iv) Performance Efficiency

* Adapting & Providing best perf.

* Design Principles
  - → Democratize advanced tech
  - → Go global in minutes
  - → Use serverless architectures
  - → Experiment more often
  - → Mechanical Sympathy

Selection → ASG, lambda, EBS, S3, RDS

Review → Cloudformation, AWS News Blog

Monitoring → Cloudwatch, lambda

Trade offs → RDS, Elasticache, Snowball, Cloudfront

## (v) Cost Optimization

- lowest price point

- Design principles
  - → Adopt a consumption mode
  - → Measure Overall efficiency
  - → Stop spending money on data center opns.
  - → Analyze & attribute expenditure
  - → Use Managed Services.

Expenditure Awareness → Budgets, Cost & Usage Report, Cost Explorer, Reserved Instance Reporting

Cost Effective Resources - Spot Instance, Reserved Instance, S3 Glacier

Matching Supply & Demand → Auto Scaling, Lambda

Optimizing over time → Trusted Advisor, cost & usage report, News Blog

## (vi) Sustainability

* min. env. impacts.

* Design Principles
  - → Understand impact
  - → Establish sustainability goals
  - → Maximize utilization
  - → Adopt to new, efficient h/w or s/w
  - → Use Managed Services
  - → Reduce downstream impact of cloud workloads.

→ EC2 Auto Scaling, Serverless (lambda, fargate)
→ Cost Explorer, Graviton 2, T instances, Spot instances.
→ EFS-IA, S3 Glacier, Cold HDD vols.
→ S3 lifecycle configs, S3 Intelligent tiering
→ Amazon data lifecycle Manager

→ Read local, Write Global : RDS Read Replicas, Aurora Global DB, DynamoDB Global Table, Cloudfront.

# AWS Well-Architected Tool

* free tool to review our architecture against 6 pillars & adopt best practice

* How?
  - → Select workload & answer questions
  - → Review
  - → Obtain advice: videos, documentations,

Hand on → lens,

P.T.O

# Right Sizing

* Choosing best instance type is not best choice, as cloud is elastic.
* Choose at lowest cost possible.
* Scaling up is easy so start small
* It's imp to right size
  - before cloud migration
  - continuously after cloud onboarding process.
* Cloudwatch, Cost Explorer, Trusted Advisor, 3rd party tools can help.

# AWS Ecosystems

## Free Resources

* AWS Blogs
* AWS Forums (Community)
* AWS Whitepapers & Guides
* AWS Quick Starts
* AWS Solutions.

## AWS Support

Support based on our bought level
Developer, Business & Enterprise

## AWS Marketplace

* Custom AMI
* Cloudformation Templates
* SaaS
* Containers

* If we buy through marketplace, it goes into our AWS Bill.
* We can also sell our own SOL's

## AWS Training

* AWS Digital & Classroom Training
* AWS Private Training (for orgs)
* Training & Certification for US Gov.
* Training & " for Enterprice.

* AWS Academy: for universities

* AWS certifications.

## AWS Professional Services & Partner Network

* Global team of experts
* work alongside our team and a chosen member of APN
  AWS Partner Network

* APN Tech Partners
* APN Consultation "
* APN Training "
* AWS Competency Programs
* AWS Navigate Program

# AWS Knowledge Center

common questions & stuff

## AWS IQ

* Quickly find professional help in aws projects
* freelancing
* Video-conferencing, contact mgmt, secure collaboration, integrated billing

### Customers:

Request → Review responses → Select Expert → Work securely → Pay per milestone

direct AWS Bill

### Expert:

Create Profile → Connect with customers → Start a proposal → Work securely → Get Paid

## AWS re:Post

* AWS Managed Q&A service
* like stack overflow.
* Part of free tier

* Can build expertise & reputation.

* If we are premium customers and don't get an answer, it is passed to AWS support Engineers.

* Not intended to use for time-sensitive or proprietory info.