AWS

AWS Solns. Architect · SAA-co2: (udemy).

Amazon Web Services → Cloud Provider.

> AWS Account created ✓

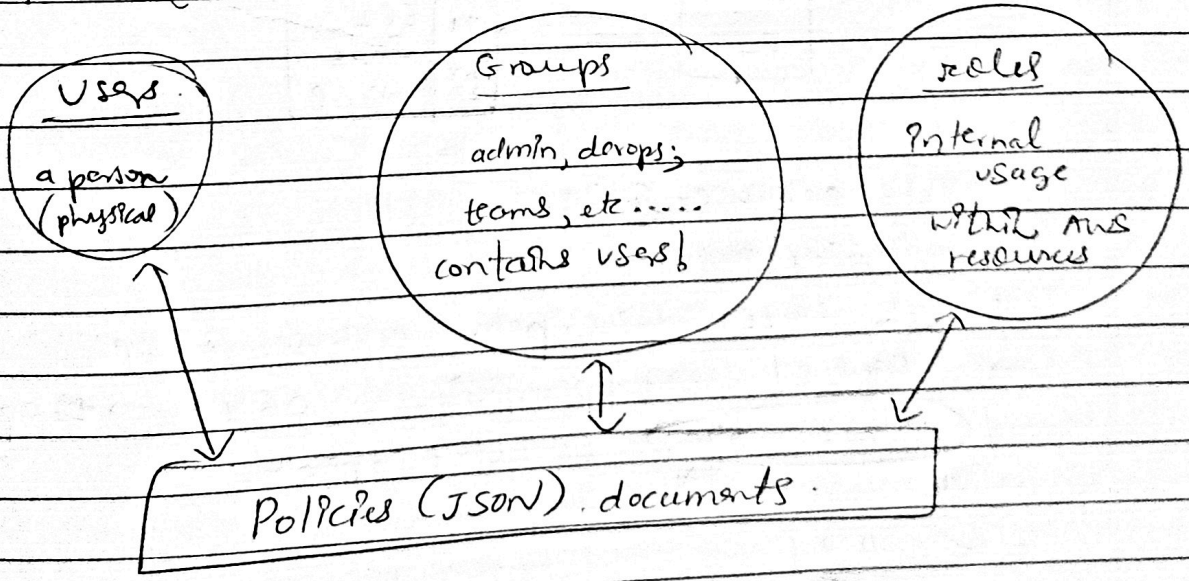
Module 3:

AWS Regions.
↳ AZ (availability zones)

> physical data center.
> all consoles. except: IAM & S3.

IAM → identity & access Management.
(heart)
users.
groups.
roles.

Root account must never be used (& shared).
> Policies are written in. JSON.



> IAM has a global view.
> MFA → multi factor authentication.
> least privilege principle → to user.

myIP ⇒ 103.69.115.10/32

(Big) Enterprise → IAM Federation.
↳ SAML standard (active directory).

> 1 IAM User per person.
> 1 IAM Role per applⁿ.
☆ Never write IAM credentials in code. EVER!.
Never use ROOT account.
> all policies → 6 ticks → updated ✓

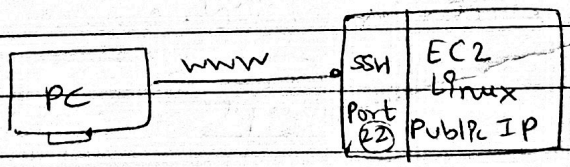
30 Aug.

EC2 → fundamental! [Elastic Compute Cloud]
→ renting VM's (EC2)
→ storing data on virtual drives (EBS)
→ distributing load across machines (ELB).
→ scaling services using auto-scaling grp (ASG)

t2.micro free

AMI ⇒ Amazon Machine Image.

> SSH into EC2 instance. timeout issue ↓ security group.
port: 22



$ ssh ec2-user@"IP"
→ Denied!.
$ ssh -i EC2Tutorial.pem ec2-user@"IP"
→ Denied! ⇒ permission ⇒ 0644 → ☆ open!
$ chmod 0400 EC Tutorial.pem ☆☆
→ " "
✓ Done!
$ ctrl+D or exit.

Date
Page

Security Groups:

↳ controls inbound / outbound traffic.
acts as "firewall" on EC2 instances.
ports, IPv4, IPv6 ✓

> can be attached to multiple instances.
> locked down to a region.
> firewall outside EC2 instance.
* maintain one separate security group for SSH access.

by default ⇒ inbound traffic → blocked.
outbound traffic → authorized.

Private vs Public IP (IPv4)

IPv4: 1.160.10.240
IPv6: 3ffe:1900:4545:3:200:f8ff:fe21:67cf
preferred ↘ for IoT

IPv4 ⇒ 3.7 billion addr. in public space.

IPV4: [0-255]. " . " . "

> Public: → unique.
→ www access ✓
→ geo located easily.

> Pvt. → unique across pvt. network.
→ identified by pvt. network.
→ 2 diff. pvt. networks (2 companies) can have same IP's. (no problems ✓).
→ www access thru Internet gateway (proxy)
→ specified range of IP's (pvt)

>> Elastic IPs: → to not change IP when on/off EC2.
↳ you own it (until you delete it)
↳ 1 instance at a time.
max (5) IP's
• avoid using them.

> Instead; use random public IP & register a DNS name to it.

or:

> use a load balancer & don't use public IP.

Install Apache on EC2:

→ Login to EC2. ✓

→ su ✓ (sudo su)

→ yum update -y ✓

yum install -y httpd.x86_64

systemctl start httpd.service

systemctl enable httpd.service

curl localhost:80

echo "Hello" > /var/www/html/index.html

echo "Hello world from $(hostname -f)" > index.html

ip-172-31-43-112.region....

EC2 User Data:

bootstrapping EC2 user data.

↳ runs only once at first start.

→ runs with root user.

launch new instance:

User data as text:

#!/bin/bash

||

||

||

✓

EC2 Instance Launch Types:

- On demand. ✓
- Reserved instances (>=1 year). ✓
- Convertible Reserved instances.
- Scheduled Reserved Instances.
- Spot Instances. ✓
- Dedicated instances. (no sharing).
- Dedicated Hosts. (entire physical server) ✓

☆ Spot fleet

Instance Types: (main ones):

R: RAM needs → in memory caches.
C: CPU needs. → compute/dB.
M: balanced ('medium) → general/web app.
I: I/O storage → dB.
G: GPU → video render/ML.

T2/T3: burstable instances (upto capacity) ✓ } 100% burst credit.
T2/T3: unlimited! ✓

to monitor => cloudwatch.

- AMI: machine image. ✓ → (for a specific region).
  - custom → pre-installed packages
    - → faster boot.
    - → monitor.
    - → security.
    - → maintenance.
    - → active dir. integration.
    - → using someone else's AMI.

OSI => APSTNDP
7 6 5 4 3 2 1

AMI storage → S3 → durable, cheap.
by default AMI's → pvt (can be made public).
cost → very low.

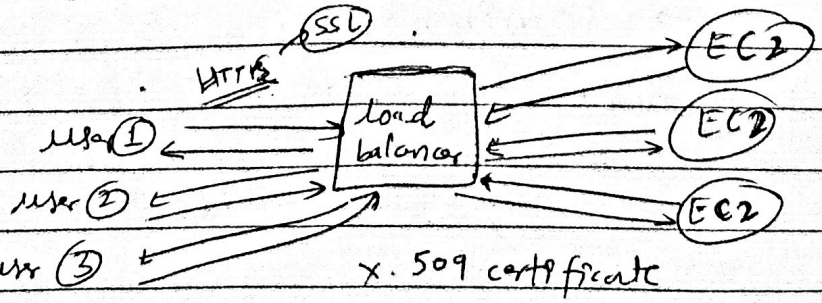
> Cross Account AMI Copy ✓
→ sharing doesn't affect ownership; but if copied onto new AMI in diff. acc; yes, ownership is affected
→ to copy => launch EC2 from your acc. using shared AMI & create AMI from instance.

> Placement Groups:
1) cluster → 1 AZ, low latency. (same hardware). → 10 Gbps on
max failure risk
2) spread → max 7 instances per AZ (hardware spread).
→ for critical applns. (high availability).
min failure risk
3) partition → instances spread on diff. partitions
→ which rely on diff. set of racks;
→ scales 100's of EC2 per grp. (Hadoop, Cassandra, Kafka).
ENI => Elastic Network Interface

> Scalability:
vertical
horizontal = elasticity → kind of → high availability.

> Load Balancing: → has static host name.

Servers that forward internet traffic to multiple servers (EC2 instances) downstream.



← has sec-grp attached

X.509 certificate

DNS => single pt of access (hostname)

ELB
CLB → connection draining
ALB/NLB → deregestration delay
~300 seconds

SSL ⇒ Secure Sockets Layer
TLS ⇒ Transport Layer Security

/health → health checks every 5 secs!

> classic LB - v1. (2009)
> Appln LB - v2 (2016)
> Network LB - v2 (2017)
} Internal (pvt) or external (public)

> Appln LB: → microservices & container based appln. (Docker, ECS (elastic container service), ...)
↳ 1 LB many applns ✓ → HTTP traffic
> supports HTTP/HTTPS & websockets protocol.
> IP of client not seen directly.
True IP of client ⇒ X-Forwarded-For ✓ - Ports, - Proto
> dynamic host port mapping with ECS.

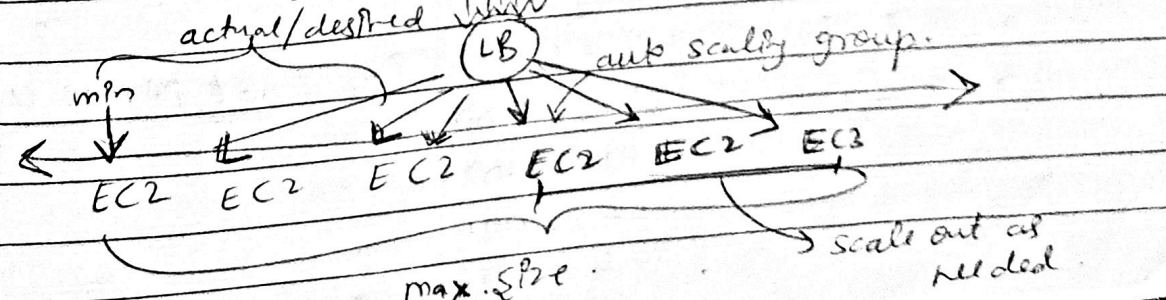
> Network LB: → forwards (TCP) traffic. → SSL termination
→ less latency ~100ms ⇒ pvt static IP ✓
sees client IP → higher performance. 1 static IP per AZ.
→ don't resolve & use underlying IP. cross-zone LB

> 4xx → client induced errors.
5xx → appln induced errors.

> LB Stickiness: → can cause imbalance (50:50 → ideal, 80:20 → may happen) CPU utilization
↳ works with Classic LB & Appln. LB
used for cookies → session data.

ACM ⇒ AWS Certificate Manager.
SNI → "server name indication" → specifies hostname.
(SSL/TLS) certificates.

> Auto Scaling Group: (ASG).
> Scale in/out EC2 & decrease/incr. load.
> auto registers new instances to LB.



Spaghetti → Amazon!

> ASG's: → smart hai!

Launch config:
- > AMI + Instance type
- > EC2 user Data.
- > EBS volumes.
- > Security groups.
- > SSH key pair.

min / max / optimal capacity
network / subnet info.
load balancer info.
scaling policies.

> auto scaling alarms → cloudwatch → tells when to scale up/down.

> auto scaling rules ✓

> ASG default termination policy. (A) + (B) ×
- → AZ → most instances ✓ (→ A).
  - → delete the one with oldest launch config ✓
    - (v1) not (v2) ×.

default cooldown → 300 secs.
- → cost effective ⇒ 180 secs ✓

Scaling Policy:
- > Target Tracking Scaling:
- > Simple / Step Scaling:
- > Scheduled Actions:

lifecycle hooks (React? ×D)
- (pending state)
- (terminating state)

> Launch template vs Launch config (better)
- → multi-use
- → re-created each-time

> EBS (Elastic Block Storage) → not a physical drive
- → Network drive you can attach to your Instances.
- > Locked to an AZ
- > billed for provisioned capacity (not PAY AS U GO!)
- > 4 types: GP2 (SSD)
  - IO1 (SSD)
  - ST1 (HDD)
  - SC1 (HDD)

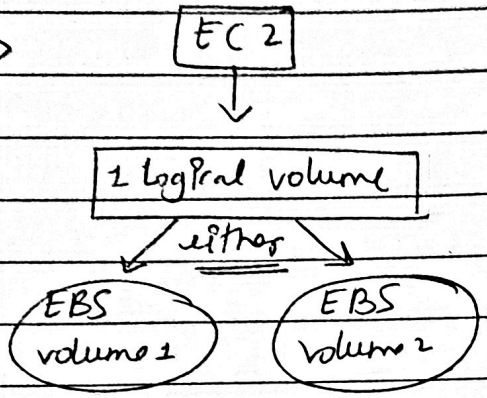
ENI → Elastic network interface.

> Only Gp2 & Io1 → used as boot volumes.

```
$ lsblk          // show all attached drives

$ sudo mkfs -t ext4 /dev/xvdb
$ sudo mkdir /data
$ sudo mount /dev/xvdb /data

$ lsblk
$ cd /data
$ sudo touch Hello.txt
$ sudo ls
$ sudo nano Hello.txt
                Hello world!
$ cat Hello.txt

$ sudo cp /etc/fstab /etc/fstab.orig
$ sudo nano /etc/fstab
    ....
    /dev/xvdb /data ext4 defaults,nofail 0 2
$ cat /etc/fstab

$ sudo file -s /dev/xvdb
        o/p ⇒ ext4 ✓
$ cd ../
$ sudo umount /data
$ lsblk          // drive will be unmounted.
$ sudo mount -a
$ lsblk
        // xvdb → data "/data
```

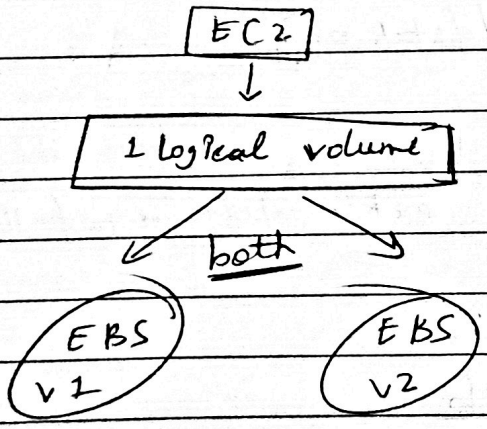
- EBS snapshots.
- EBS Volume migration.
- EBS encryption. AES-256 → KMS
- EBS volume vs Instance store [physical attached disk]
- EBS RAID options.

RAID 0 (increase performance)



> High IOPS.
> data losing ka risk!!

RAID 1 (increase fault tolerance)



> mirroring the volume.
> 2 × network.

- EFS → Elastic File System
  - > managed NFS (network file system)
  - > 3× gp2 price
  - > pay per use ✓
  - > compatible with Linux AMI (NOT WINDOWS) (∵ POSIX)

performance mode : 1) general purpose.
2) max I/O.

Storage tier : 1) standard → frequent access
2) Infrequent access (EFS IA) → cheap

Global Aurora

PostgreSQL → 5432 port

MySQL ⇒ 3306 port

↳ RTO → recovery time objective. < 1min

classmate Date Page

RDS, Aurora, ElastiCache:

Relational Database Service ⇒ SQL ✓

- ↳ PostgreSQL, MySQL, Oracle, MariaDB, Microsoft SQL Server.
- ↳ Managed service ⇒ automation ✓
- ↳ storage → EBS backed (IO1 or GP2)
- ↳ U can't (SSH) into Instances.

Read Replicas for read scalability.

- ↳ upto 5 ; async ✓
- ↳ can be promoted to own DB.
- ↳ network cost → from 1 AZ to another $$$; same AZ → free.

RDS multi AZ → disaster recovery.

↳ SYNC ; 1 DNS name ; more availability ; not used for scaling.

Read Replica → multi AZ? ⇒ YES ✓ for disaster recovery.

DB download ⇒ SQL Electron ✓

→ RDS Security → Encryption. → at rest → in-flight.

- ↳ AWS KMS - AES 256
- ↳ if master not encrypted, read replicas ~~not~~ can't be replicated.

IAM Authentication → for MySQL & PostgreSQL only.

→ Aurora: 5× SQL, 3× postgres

- ↳ 20% more cost [RDS]
- ↳ 6 copies across → 3 AZ !! self-healing & shit
- ↳ same as RDS security.
- ↳ Aurora Serverless? (pay per sec)

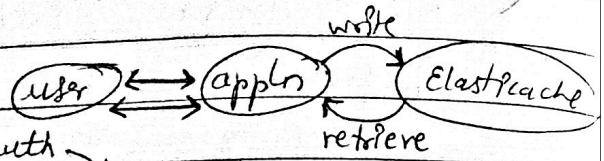
→ ElastiCache: → (RDS) for cache's.

- ↳ in-memory DB's, high performance, low latency.
- ↳ makes appln. stateless.
- ↳ sharding, read replicas ✓ SSL in-flight ✓
  - write ↑
  - read ↑
- ↳ no IAM Auth. but has Redis Auth.

> sharding → multi-node for partitioning of data.

redis ⇒ 6379 port

- DB cache
- User Session Store



→ Redis vs Memcached. → SASL auth → adv stuff

> multiAZ > non-persistent, sharding.

> Backup ✓ > Backup & restore X.

> AOF, read replicas > multi-threaded arch.

→ cache invalidation → super tough.

> Route 53 → managed DNS (domain name system)

↳ global service ✓

most common records:

> A → hostname to IPv4.

> AAAA → hostname to IPv6.

> CNAME → hostname to hostname.

> Alias → hostname to AWS resource.

> $0.50 per month per hosted zone, ∴ not free.

nslookup xyz.abc.com..

→ gives IPv4 // on mac → dig xyz.com

EC2 meta-data

EC2_AVAIL_ZONE = $(curl ..... // 169.254.169.254 ...

→ DNS records TTL (time to live)

high TTL → 24 hrs.

low TTL → 60 secs.

→ Root not allowed X

→ CNAME works for 'something.mydomain.com'

↳ 'NON-ROOT' domain

→ Alias → xyz.amazonaws.com

↳ ROOT + NON-ROOT ✓

↳ free + native health check

Routing Policy → Simple . Weighted (70 / 20 / 10) . Latency . Failover . Geolocn . Closest server
Health Checks → unhealthy if fails (3) checks . • Multivalue .

Route53 as a Registrar → 3rd party . GoDaddy, etc...
↳ !DNS ✗

Sdns. Architect Discussions:

5 pillars ⇒ Cost, Performance, Reliability, Security, Operational Excellence.

Elastic BeanStalk ⇒ developer centric view of deploying appln. on AWS
↳ 3 components:
① Appln.
② Appln version for each deployment.
③ Env. name (eg: dev, test, prodn, ...) → free naming

Amazon S3 → buckets, directories
↳ global unique name // region level defined
↳ consist of objects → have a (key) → full path.

Key = prefix + object name.
eg: s3 // my-bucket / my-folder / my-file.txt
prefix ↳ obj. name

max. obj. size → 5TB!
upload limit → 5GB (at a time) ; metadata ; tags ✓ version ID ✓
→ S3 versioning → bucket level.
→ S3 encryption → SSE-S3, SSE-KMS, SSE-C, client-side.
SSE → server side encryption.
SSEC → HTTPS is mandatory ✓
> Encryption in transit (SSL / TLS).
↳ or flight.
→ S3 Security: User Based, Resource Based.
IAM policy. ↳ ACL → access control list.

→ pre-signed URL's

ARN → amazon resource name. add (/*) at the end.

→ JSON based policies ✓
  - resources: buckets & objects
  - actions: set of API to Allow or Deny.
  - effect: Allow / Deny.
  - principal: acc/user to apply policy to...

⇒ CORS → Cross Origin Resource Sharing.
  - protocol, domain (host), port.

> AWS CLI, SDK, IAM Roles + Policies:

> Never put credentials on EC2 machine! ✗ bad practice!
  - use IAM roles instead!

> AWS policy generator, simulator.

EC2 instance metadata: → instances learn about themselves!

URL ⇒ http://169.254.169.254/latest/meta-data

AWS SDK > Exponential Backoff ⇒ wait for 2X more than last API call, if API call fails.

> Advanced S3 & Athena:

MFA-Delete → possible only w/ CLI ; not UI.
  - also only Root?

S3 Access Logs → for auditing (using Athena?)

S3 Replication (CRR, SRR) ⇒ cross region replication, same region replication.
  - async replication?

Presigned URLs: to access pvt S3 bucket object for a specific time to allowed users.

→ S3 storage classes:
  - S3 Standard - General
  - S3 Standard - IA (Infreq. access)
  - S3 Onezone - Infreq. access (IA)
  - S3 Intell. Tiering
  - Glacier & Deep Archive.

S3 object Lock | Glacier Vault Lock
→ WORM write once read many

serverless!

> S3 lifecycle rules ★ → to effectively switch S3 classes.

> Athena → S3 data analyser

> CloudFront & AWS Global Accelerator:
- ↳ CDN (content delivery network)
- distribute & cache READS w/ help of edge locations.
- → enhanced security: OAI (origin access identity)
- > signed URL / cookie
- > Global Accelerator → for low latency (global)
  - > unicast ; anycast IP's (1 IP) > nearest IP client gets

> AWS Storage Extras:

Snowball → physical transfer. (Snowball Edge → new).

Snowmobile → Truck! xD

Snowball → S3 → Glacier

> AWS Storage Gateway → S3 access on-premise!
- ↳ File, Volume, ~~Type~~ Tape

> Amazon FSx (File Server for Windows) + (for Lustre) → Linux + Cluster. (for ML, HPC)
- ∵ EFS can't run on Windows (only Linux systems)

> Decoupling Apps: SQS, SNS, Kinesis, Active MQ:

SQS queue: Simple Queue Service
- > standard queue.
- > msg visibility timeout
- > dead letter queues 💀
- > delay queues.
- > FIFO queue eg: Demo.fifo .fifo → must!

ARN → amazon resource name

SNS : Simple Notification Service :

> Protocol endpoints :→ HTTP, HTTPS, Email, Email-JSON, Amazon SQS, Lambda

> SNS + SQS : Fanout | send 1 receive all.

> SNS can't send msg to Fifo queue !

Kinesis → managed alternative to Apache Kafka.

↳ big data 'real time'

↳ Streams, Analytics, Firehose.

↳ divided into shards / partitions

> shard-Iterator, records, ...

> Amazon MQ → managed message broker service. Active MQ.

↳ endpoints : Openwire, AMQP, STOMP, MQTT, WSS

↳ might have S.G issue ✓

→ FAAS (fn as a service).

> Serverless Overview :

↳ don't need to manage servers -

↳ {Lambda, DynamoDB, Cognito, API gateway, S3, SNS & SQS, Kinesis data Firehose, Serverless Aurora, Step Fn, Fargate.}

AWS Lambda : > virtual functions + autoscaled.

> short executions + on-demand.

> pay per req & compute time.

Languages → Node, Python, Java, C#, Go, Ruby, Rust - DOCKER (or ECS)

λ Limits : memory allocn : 128 MB - 3008 MB

max exec time : 900 secs (15 mins)

env. variables : 4 KB.

/tmp size : 512 MB

concurrent exec : 1000

deploym. size : 50 MB

uncmpsd. size : 250 MB

Lambda @ Edge : deploy λ fns alongside CloudFront CDN

OLTP → online transaction processing.
OLAP → " analytical "

→ 3 AZ ✓

DynamoDB: Serverless NOSQL dB. ×Spoke: WCU → write capacity unit.
↳ provisioned throughput. RCU → read capacity unit.

DAX → DynamoDB Accelerator ⟹ seamless cache.

Streams → CRUD operations.

API Gateway: public REST API's.
↳ easy integration w/ : λ, HTTP or any AWS service.

Cognito: → v confusing?! → gives users identity to interact w/ app.
↳ User Pools
↳ Identity Pools (Federated Identity).
↳ Cognito Sync → (AppSync ☆).

SAM (Serverless Appln. Model)

> **Classic serverless REST API:** HTTPS, API Gateway, λ, DynamoDB

> **Databases in AWS:**

RDS Aurora ElastiCache DynamoDB S3
Athena Redshift Neptune ElasticSearch

5 pillars of DB: (OSCPR)

Operations Security Cost Reliability Performance

> **Monitoring & Audit:**

CloudWatch → dashboards (→ global) → includes graphs.
↳ Logs, Agent, Alarms (OK, Insuff-Data, Alarm), Events

· CloudTrail → enabled by default. (record API calls)

Config → audits & records: compliance & config. changes.

> **IAM Advanced:**

> **STS:** "Security Token Service" → backbone of services!
↳ valid for 1 hr. (grants temp, limited access).
↳ assumed Roles, cross account access!

Identity Federation → 3rd party management
↳ SAML 2.0

Cognito: provide direct access from client side. (mobile, web app).

SAML → security assertion markup lang.

> Directory services: EC2 running Windows?
Microsoft Active Directory (AD).

AWS AD:

1) AWS Managed Microsoft AD.
2) AD connector.
3) Simple AD.
opt. 4) Cognito user pools.

> Organizations: global service.
↳ to manage multiple accounts.
SCP (Service Control Policy) → whitelist or blacklist IAM act
↳ not applied to Root/Master account.

> IAM conditions → If explicit deny exists, it superimposes any underlying allow & vice versa.

> AWS RAM (resource access manager) → avoid resource duplication
↳ share pvt subnet on VPC ✓

> AWS SSO (Single sign on) → login once, access all accounts!
↳ SAML 2.0, AD integrated ✓

> Security & Encryption:

KMS (key mngmnt service)
↳ CMK (Customer Master Key) ⇒ symmetric (AES-256)
↳ Assymmetric (RSA & ECC key pair)

SSM Parameter Store: Simple ^ Systems Manager Agent.

Secrets Manager

CloudHSM > Hardware Security Model

Shield: DDOS prevention

WAF: Web Appln. Firewall (layer 7 → HTTP).
↳ deploy on ALB, API Gateway, CloudFront
Web ACL → SQL inj, xSS prevention?

AWS Shared Responsibility Model

availability is compromised by saving cost!

cost: use (pvt IP) for lower cost.

> Networking VPC:

CIDR - IPv4 : Classless Inter-Domain Routing

AWS reserves 5 IP's in each subnet. (1st 4 & last addr)

Internet Gateway: connects VPC instances w/ Internet

Route Table: ✓

better

NAT Instances: "Network Address Translation" (old way)

NAT Gateway: pvt subnet → NAT Gateway → internet gateway

pvt zone on R53 & DNS resolution is disabled by default.

Network ACL → (inbound/outbound rules → stateless).

↳ like firewall (controls traffic from & to subnet).

↳ can block IP at subnet level.

by default, everything is denied.

VPC Peering: connects 2 VPC's privately

VPC Endpoint: access services with pvt network

⇒ 2 ways: ↳ ① Interface ② Gateway.

→ One AWS CLI cmd ^ --region <your-region> ✓ ★ (coz default is ~~US-east-1~~ us-east-1).

VPC Flow Logs + Athena

↳ IP traffic info

Bastion Hosts: to SSH into pvt instances.

↳ SG must be super strict. (port 22 only).

↳ it is a public subnet (instance).

Site-to-site VPN links VPN Gateway & Customer Gateway.

Direct Connect + DC Gateway

↳ dedicated pvt connection

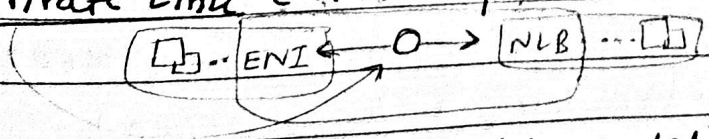
::/0 any IPv6 → all public addr.!

Egress (outgoing) Only Internet Gateway → IPv6 ✓ (only)

↳ edit Route tables too ✓

NAT → IPv4 ✓

Aws Private Link (VPC Endpoint Service)



EC2 Classic, AWS ClassicLink (deprecated) → VPC hana ✓.

VPN CloudHub: provide secure commn. bw sites on multi VPN connections

Transit Gateway: transitive peering bw 1000s of VPC & star connection.

(hub & spoke)

↳ supports IP multicast.

NLB → doesn't have SG

Disaster Recovery:

RPO → Recovery Point Objective.

RTO → Recovery Time Objective.

DMS (Database Migration Service): to migrate dBs ✓

↳ continuous data replication ⇒ CDC (change data capture)

AWS SCT (Schema Conversion Tool): convert database schema from 1 engine to another.

SMS (Server Migration Service) } on-premise

VM Import/Export; DMS ✓ }

DataSync: move large amount of data from on-premise to AWS.

↳ 1) NFS/SMB ⇒ S3, EFS, FSx 2) EFS ⇒ EFS

Other Services:

CI/CD: CodeCommit, CodeBuild, CodeDeploy

(continuous Integration ; continuous delivery.

code → build → test → deploy → provision

CodeCommit CodeBuild Elastic Beanstalk, CodeDeploy.

⇒ Orchestrate: AWS CodePipeline

CloudFormation: Infrastructure as code.

↳ StackSets: CRUD stacks across multi-acc/region with a single operation!

ECS (Elastic Container Service): run Docker on EC2.

(ECR (Elastic Container Registry)

↳ serverless? ⇒ Fargate ✓

EKS (Elastic Kubernetes Service): managed (k8s) clusters on AWS ⇒ ✓ ; ≈ ECS but diff. API

Step Functions: run serverless, λ fn (multiple) ✓ (better)

SWF (Simple Workflow service) → ext. signal, child process.

OSCPR

classmate
Date
Page

EMR (Elastic MapReduce): helps create Hadoop clusters.
→ (Big Data) analysis.

AWS Glue: fully managed ETL (extract, transform, load)
↳ serverless ✓, automated code-generation.

AWS Opsworks: managed Chef & Puppet.
↳ manage configs.

AWS Elastic Transcoder: convert media files (video + music).
↳ jobs, pipeline, presets, notifications.
↳ serverless ✓.

AWS WorkSpaces: managed secure Cloud Desktop. (VDI) ✓ (virtual desktop infrastr.)

AWS AppSync: store & sync data across mobile + web apps.
↳ real-time ✓ ⇒ uses GraphQL (mobile tech from Facebook)

AWS TA (Trusted Advisor)

★ Links: aws.amazon.com/architecture
" " " / solutions