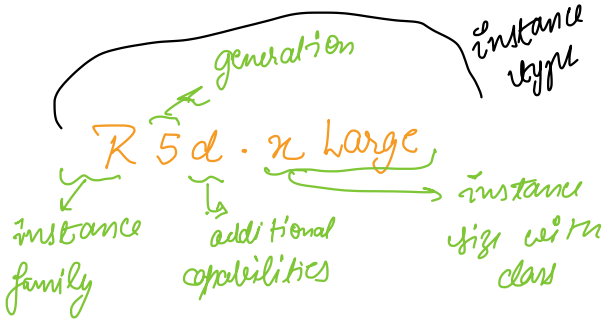
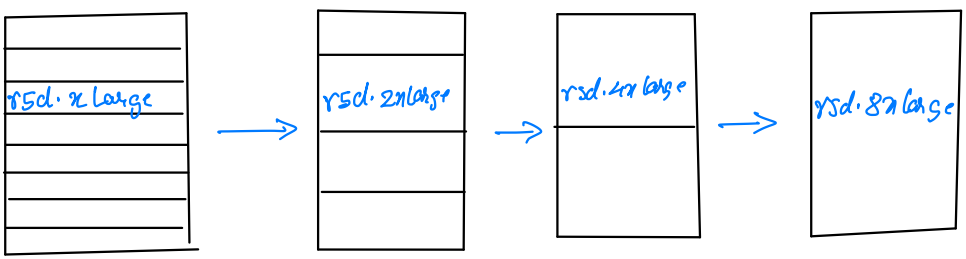
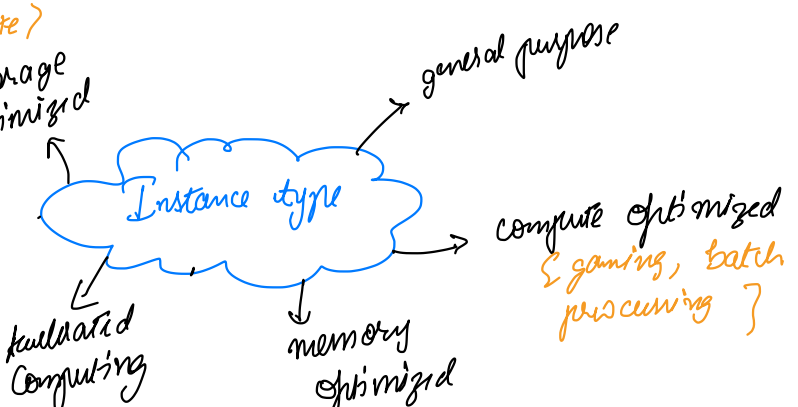
EC2 { Elastic Cloud Compute }

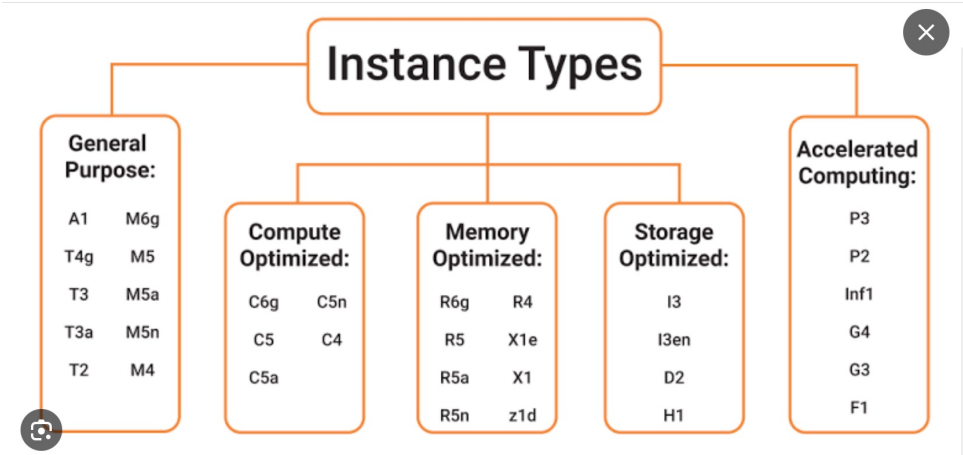
" The servers provisioned in cloud "





{ high speed sequential read & write }
{ distributed file system }
{ data warehousing }
{ OLTP }





	Type	Description	Mnemonic
General Purpose	a1	Good for scale-out workloads, supported by Arm	**a** is for Arm processor – or as light as **A1** steak sauce
	t-family: t3, t3a, t2	Burstable, good for changing workloads	**t** is for **tiny** or **turbo**
	m-family: m6g, m5, m5a, m5n, m4	Balanced, good for consistent workloads	**m** is for **main** or happy **medium**
Compute Optimized	c-family: c5, c5n, c4	High ratio of compute to memory	**c** is for **compute**
Memory Optimized	r-family: r5, r5a, r5n, r4	Good for in-memory databases	**r** is for **RAM**
	x1-family: x1e, x1	Good for full in-memory applications	**x** is for **xtreme**
	High memory	Good for large in-memory databases	High memory is for... high memory.
	z1d	Both high compute and high memory	**z** is for **zippy**
Accelerated Computing	p-family: p3, p2	Good for graphics processing and other GPU uses	**p** is for **pictures**
	Inf1	Support machine learning inference applications	**Inf** is for **inference**
	g-family: g4, g3	Accelerate machine learning inference and graphics-intensive workloads	**g** is for **graphics**
	f1	Customizable hardware acceleration with field programmable gate arrays (FPGAs)	**f** is for **FPGA** or **feel** as in hardware
Storage Optimized	i-family: i3, i3en	SDD-backed, balance of compute and memory	**i** is for **IOPS**
	d2	Highest disk ratio	**d** is for **dense**
	h1	HDD-backed, balance of compute and memory	**H** is for **HDD**

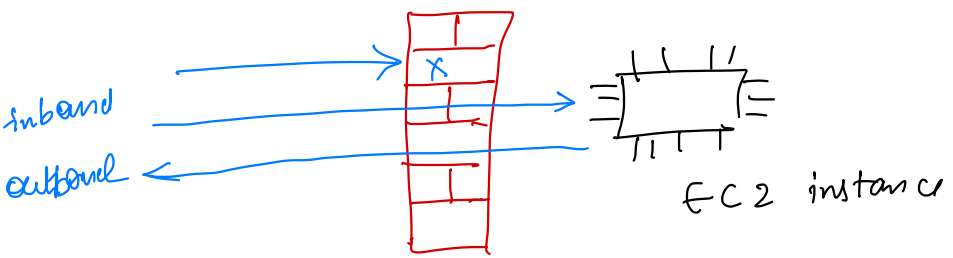
Security groups

↳ fundamental to network security for an EC2 instance

* scoped by region!

* like a firewall

* controls inbound & outbound traffic { by default all outbound traffic is allowed }



* only contains "Allow rules"

* Lives outside EC2 instances & and are like policies can be attached to multiple instances

Security groups are stateful!

⟹ All inbound traffic is blocked
All outbound traffic is allowed

- why "connection timeout" is because of SG
- "connection refused" is an application error

* SSH into a EC2 server requires public IP address of EC2 & not private

* Amazon Linux OS is a Cent-OS based & Red Hat based operating system, so to install packages we need to do "yum install" & not "apt-get" or "apt"

→ when trying to connect an EC2 instance via SSH we may encounter access denied due to many permission to solve it, go to properties > security & remove all users & add only yourself as the only user who is allowed to read & write the .pem file.

ssh -i .\EC2.pem ec2-user@3.110.223.51

file name | user | public ip

different ways to connect to EC2 instances

- Instance connect
  { cloud से दी instance खोल दो }
- SSH using port 22
  { requires aws keys & secret aws keys }
- SSM session manager
  { gives access to EC2 w/o SSH {port 22} or bastion host or SSH aws keys }
  ∵ port 22 is not required hence more secure
  also log the session data to S3 or cloudwatch logs.

* Older version of windows need PUTTY to SSH
but windows 10 onwards it's easy.

EC2 purchasing options

1. On-demand {pay as you go}
2. Reserved instances { 72% discount }
   1 time payment 1y / 3y with options of upfront, partial upfront, monthly { regional or zonal scope }
   { Convertible Resci. Instan. → can change {66% dis} cheapest.
   family, type, OS, scope & tenancy
3. savings plan { 72% discount }
   Commit to an amount of usage { 1y or 3y }
   AZ specific + family specific.
   { locked to a family & flexible in family. }
   If usage exceeds then considered on demand.

(4) spot instances [90 % discount]

Bid on EC2 capacity, highly volatile, availability not guaranteed,

AWS can reclaim an instance with a 2 min warning.

For jobs that can be interrupted. {Eg: Batch processing, analysis}

* highly volatile

(5) Dedicated host

{ physical servers for an AWS acct. {can launch multiple EC2 instances }

Allows to address compliance requirements & use existing server bound software licenses (per-socket, per core, per VM

useful for softwares that have BYOL { Bring your own license } licensing model.

{ visibility to bare metal hardware }

}

(7) dedicated instances

no other customer will share the hardware

may share in same acct.

(8) Capacity reservation

Reserve capacity in a specific AZ for any duration

{ unlike RI, this pay as you go. for the time we keep the reservation active

{ for a period book the instances & pay full price even if you don't use it }

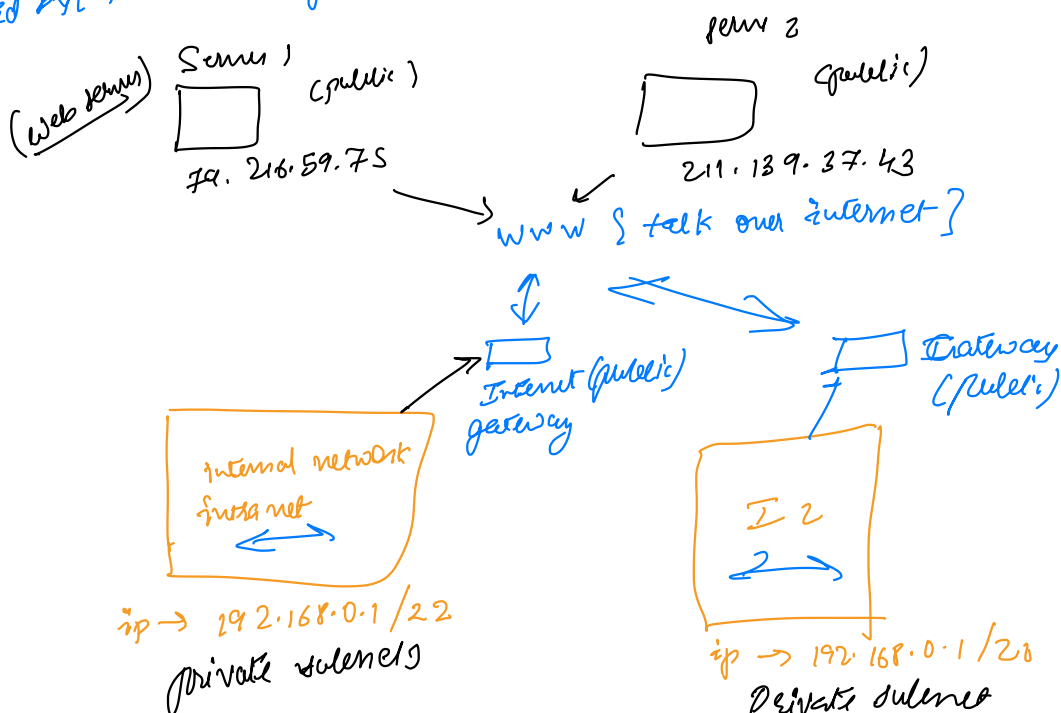
* Spot fleet is a set of spot instances & optionally on-demand instances

---

IPv4 & IPv6

IPv4 → 8 bit field x4 separated by [.]

IPv6 → 16 bit fields X 8 for IOT problem, & new (relatively)



* private network dont machines talk to internet standing behind a NAT + internet gateway which acts as a "proxy."

→ network address Translation

Elastic IPs

whenever we stop & start an instance the public IP changes, so to fix the public ip of our EC2 instance we need elastic IP. Can be assigned to instances & network interfaces.

* only have 5 Elastic IPs in our account {can be incr.}

→ if any instance goes down we can remap the traffic to another instance.

→ Considered a poor architectural design!

↳ instead use a random public IP & register a DNS name to it.

Or

Use Load balancer fully!

Q Problem with changing public IP is that it changes every time an instance is stopped & started {not reboot}.

→ Make an elastic IP & associate it with the EC2 or disassociate to detach,
But make sure to remove the ElasticIP because it costs!

Placement groups

how should our EC2 inst. be placed. in AWS

* 3 diff. strategies

① cluster → a low-latency group in a single AZ {risky} ξ

(2) spread → spread instances across different hardware { max 7 inst. per group per AZ }
{ high availability }

(3) Partition → spread instances across different logical partitions { rely on diff. set of racks }
upto 7 partitions per AZ
100's of EC2 in a grp! { for distributed workloads, Kafka, Cassandra, Hadoop }
{ connectivity }

Elastic Network interface { ENI }

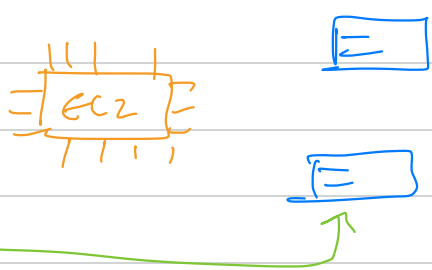
Logical component in a VPC that represents a virtual network card

instances को internet access के लिए जो network card चाहिए होता है वही ENI है।

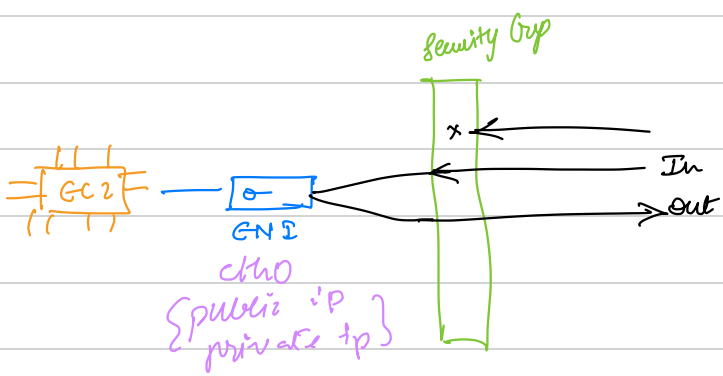
→ has 1 primary IPv4 & one or more secondary IPv4s

→ gives the private IPv4 to an EC2 instance
→ 1 Elastic IPv4 per private IPv4
→ 1 Public IPv4 auto assigned only to primary ENI (eth0)
→ 1 or more security group
→ 1 MAC address

* ENI's can be created independently & attached on fly.
* AZ specific
* eth0 will also give the public IPv4



So basically,



* ENI's give a more control over the networking

EC2 hibernate

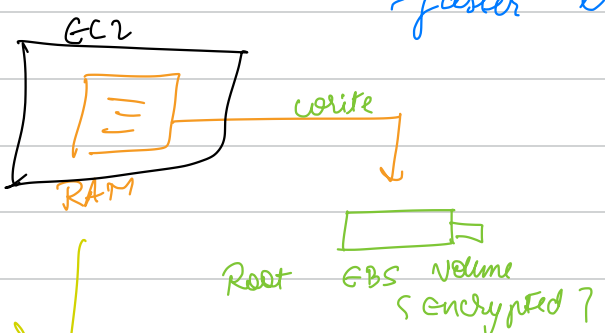
what actually happens?

- stop → The instance is just not running
  (All network volumes (EBS) are intact
- terminate → the attached network volumes are destroyed (if flag set)

starting

First start :- OS boots + EC2 user data script
Following start :- OS boots up
starting up of application, caches load, etc.

hibernate → In-memory (RAM) is preserved
⇒ faster boot



* provided root EBS volume has enough spaces to store.

* Instance RAM size < 150 GB
* Cannot hibernate for more than 60 day
* bare metal family does not support

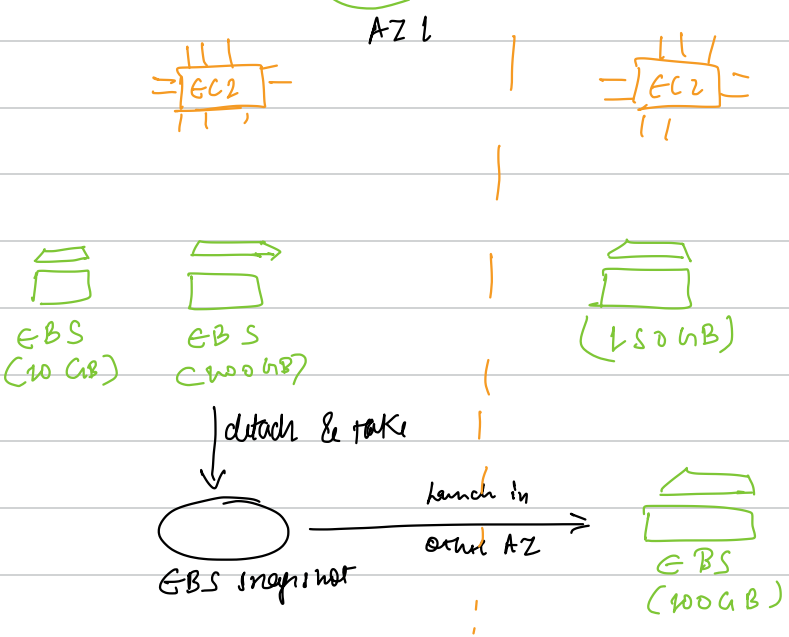
EBS volume {Elastic Block Storage}

→ a "network USB stick" {a network drive}

→ allows data to persist even after termination.

→ can be attached to multiple instances

→ AZ specific { if need in another AZ then take EBS snapshot & then make an EBS volume out of that snapshot in another AZ



→ has a "delete on termination" attribute.

default { "True" for root volume
"False" for other —11—

EBS Snapshot → A point in time snapshot of an EBS.

pricing of snapshots is based on the size of snapshot

First detach EBS & then make snapshot & reattach!

features:

1. EBS snapshot archive {1-3 days} {75% v. cheaper} {24-72 hrs to retain}
2. Recycle Bin {Specify retention period}
3. Fast Snapshot Restore (FSR) {full ini of snapshot to have no latency} {good for big snapshot}

AMI {Amazon Machine Image}

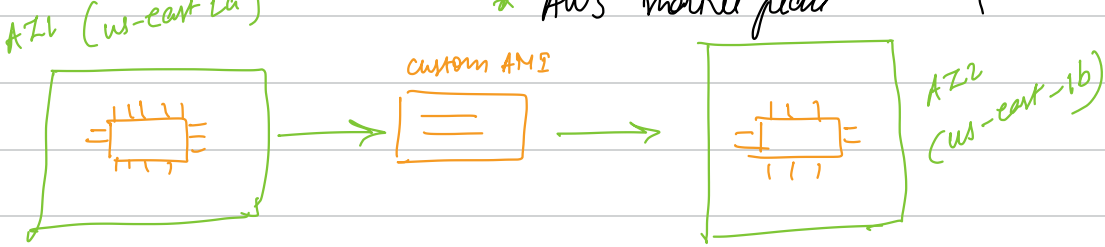
snapshot of the entire EC2 instance! with ENIs, EBS, and all the configuration

customization of EC2 instance on our use case

{Can be found @ Marketplace}

→ created for a specific AZ but can be copied across regions.

- public → AWS provided
- your own
- AWS marketplace



EC2 instance store

→ hardware disk attached to EC2 instance

→ attached to physical server

→ extremely fast I/O

→ not durable long term, will get deleted on termination, or may fail if EC2 fails.

→ Backup & replication is user responsibility.

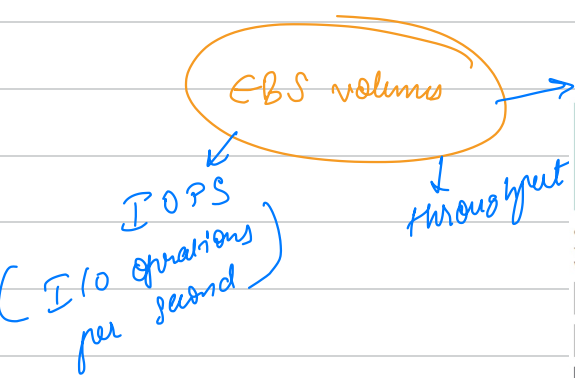
→ good for cache / buffer etc ? low time storage ?

EBS volume types

new gen

used as boot volumes

1. gp2 | gp3 (SSD) — general purpose & cost effective
2. io1 / io2 Block Express (SSD) — high perform.
3. st1 (HDD) — frequently accessed & throughput intensive
4. sc1 (HDD) — less frequent



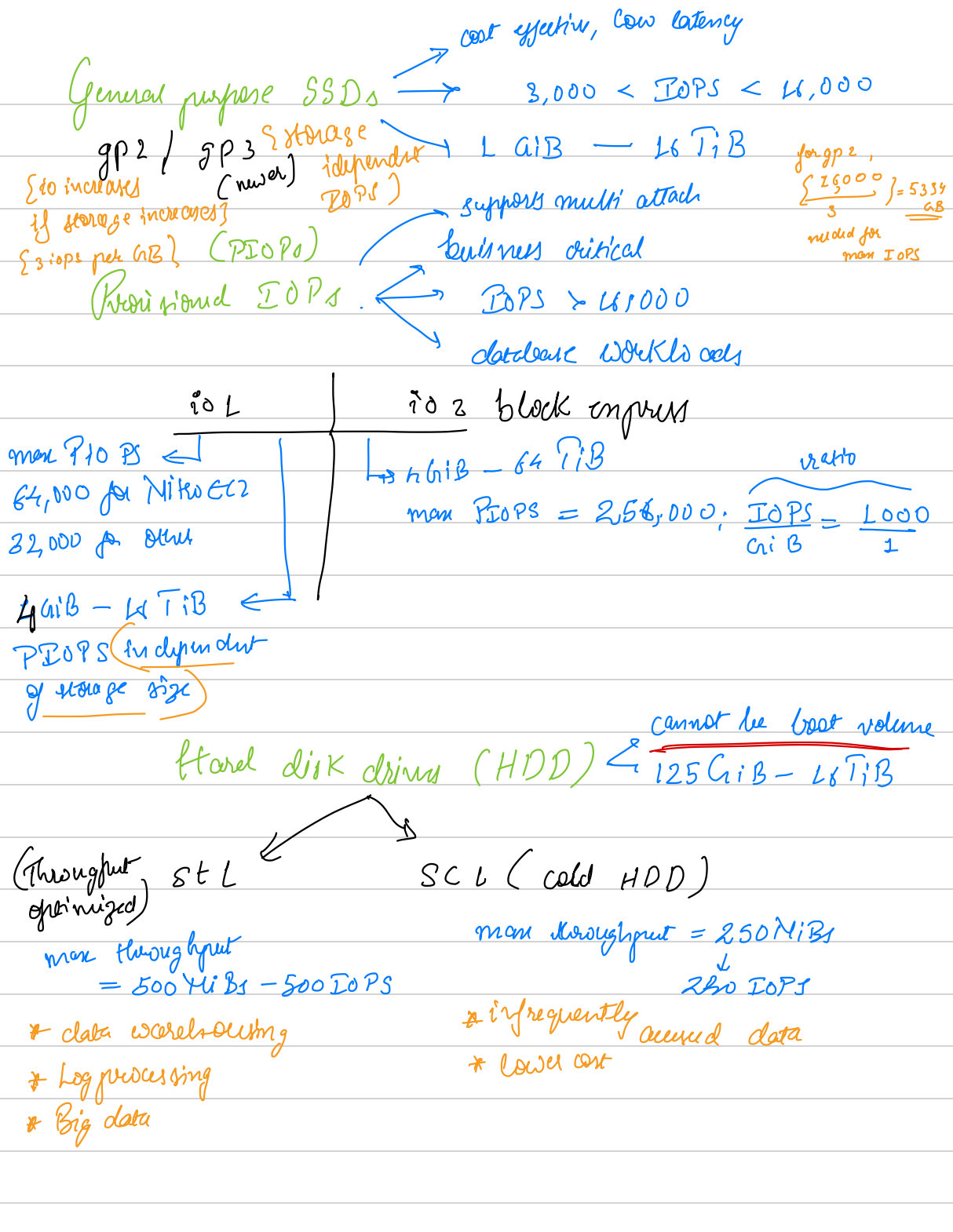
Press Esc to exit full screen

Good job!
You can run a database on an EC2 instance that uses an Instance Store, but you'll have a problem that the data will be lost if the EC2 instance is stopped (it can be restarted without problems). One solution is that you can set up a replication mechanism on another EC2 instance with an Instance Store to have a standby copy. Another solution is to set up backup mechanisms for your data. It's all up to you how you want to set up your architecture to validate your requirements. In this use case, it's around IOPS, so we have to choose an EC2 Instance Store.

Question 9:

You are running a high-performance database that requires an IOPS of 310,000 for its underlying storage. What do you recommend?

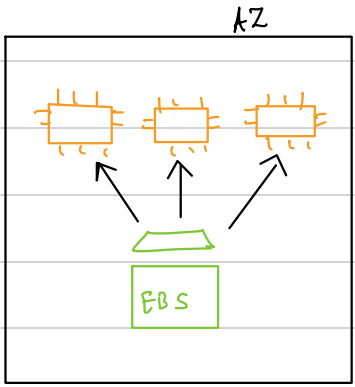
- Use an EBS gp2 drive
- Use an EBS io1 drive
- **Use an EC2 Instance Store**
- Use an EBS io2 Block Express drive


cost effective, low latency
General purpose SSDs
3,000 < IOPS < 16,000
gp2 / gp3 (newer)
storage independent IOPS
1 GiB — 16 TiB
for gp2, 16000/3 = 5334 GB needed for max IOPS
{io increases if storage increases} {3 iops per GB}
supports multi attach
(PIOPS)
Provisioned IOPS
business critical
IOPS > 16,000
database workloads
io1
io2 block express
max PIOPS
64,000 for Nitro EC2
32,000 for other
4 GiB — 64 TiB
max PIOPS = 256,000; ratio IOPS/GiB = 1000/1
4 GiB — 16 TiB
PIOPS independent of storage size
cannot be boot volume
Hard disk drives (HDD)
125 GiB — 16 TiB
(Throughput optimized) stL
SCL (cold HDD)
max throughput = 500 MiBs — 500 IOPS
max throughput = 250 MiBs
250 IOPS
* data warehousing
* Log processing
* Big data
* infrequently accessed data
* lower cost

* Multi Attach (io1/io2 family)

→ 1 EBS to multiple EC2 instances

→ Each instance has full read & write permissions. (with (T) performance)



use case:
① Clustered Linux application ⇒ high availability
② Concurrent write operations

* 16 EC2 instances @ a time
* must use file system that is cluster aware

EBS Encryption

→ when enabled is automatically handled by EC2 & EBS.

when volume is created with EBS encryption!
- data in rest encrypted
- data in flight encrypted
- snapshots encrypted
- volumes from snapshots encrypted

→ leverages encryption keys from KMS (AES-256)

→ encrypted volumes ⇒ snapshot ⇒ encrypted

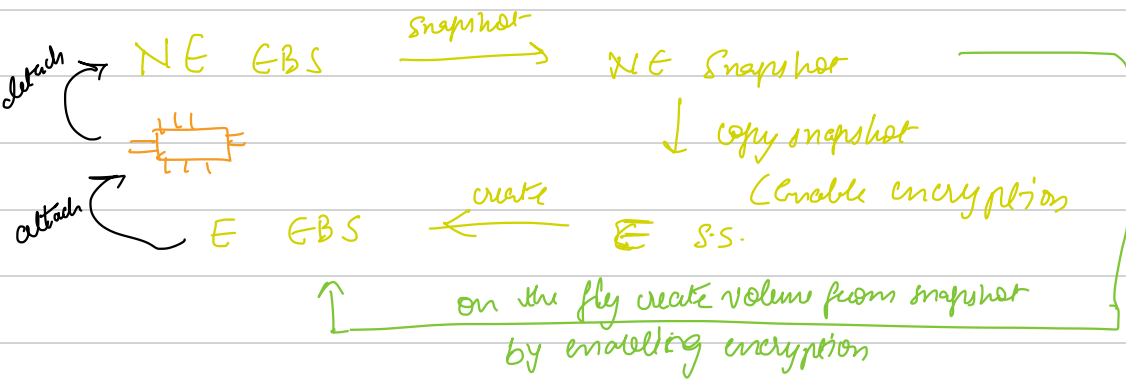
Encryption of unencrypted EBS volumes !

→ Create snapshot

→ Encrypt the EBS snapshot (using copy)

→ Create new volume from snapshot

→ attach encrypted EBS



Amazon EFS (Elastic File System)

↳ network file system (managed by AWS)

→ mount of many EC2, in multi AZ

→ high { scalable, available, cost } & pay per use

Remember

→ uses NFSv4.1 protocol

→ only compatible with Linux based AMI {Not Windows}

→ POSIX {Linux} filesystem

→ auto scaling i.e. no capacity planning.

1) Performance mode (set @ EFS creation time)

→ Max I/O {higher latency, throughput & parallel} (big data, media proc.)

* default → general purpose {latency sensitive}

throughput → measurement of bits or bytes per second by a storage device

IOPS → no. of read/write operations per second.

2) Throughput mode → Bursting {1TB = 50 MiB/s + burst up to 100 MiB/s} → throughput increases with increase in size

→ Provisioned {set throughput regardless of storage size 1GB/s for 1TB storage} {throughput independent of storage size}

→ Elastic {automatically scale throughput when unpredictable workload!}

EFS - Storage Classes

Storage tiers

1) Standard
2) Infrequent access {EFS-IA}
3) Archive {50% cheaper}

→ implement lifecycle policies to move from different tiers {to save money}

{with right option 90% of cost saving can be done.}

Availability & durability.

1) Standard {Multi AZ} * default
2) One Zone {Single AZ} {can also set up Infrequent Access one zone} → much cheaper