docker version

client
server (daemon) → pulls image from docker hub.

docker info

docker run hello-world

run → run container
hello-world → hello-world image.

docker ps → lists running containers

docker ps -a → images that have ran but exited

docker images

images stored locally

Docker engine = Client + Daemon

Instally docker gives you client + daemon on the same host

**Docker run hello-world**

① Client calls daemon
   └ implements docker remote API

② Daemon checks if there is hello-world image locally (checks local image store)

③ No image, fetches it from docker hub (docker image registry)

④ Found the image pulls to create a local copy.

Images => Stopped containers.
Containers => running image.

docker pull alpine

↑ copies image to docker host

↑ alpine image

ubuntu

↑ ubuntu image (latest)

ubuntu:14.04

" (14.04)

* pulled from hub.docker.com

docker rmi ubuntu:14.04

↑ remove images.

docker start <container>

stop "

rm "

data is persisted until you run rm command (if persistance is local)

docker run -d --name web -p 80:8080 nigelp/phoralsight-ci

- run in the background detached mode
- name of container
- Map 8080 inside of container to host 80
- which image to use. nigelp 2nd level repo.

docker stop $(docker ps -aq)

stop all containers

docker rm ''

remove all containers.

docker rmi $(docker images -q)

remove all images.

# SWARMING

```
docker swarm init

--advertise-addr 172.31.12.161:
                 2377

--listen-addr 172.31.12.161:
              2377
```

nodes must be able to
reach 172.31.12.161
engine port: 2375
secure engine port: 2376
swarm port: 2377

docker swarm join \

--token xxxx

172.31.12.161.2377

--advertise-addr

172.31.12.164:2377

--listen-addr

172.31.12.164:2377

own machine's IP address that is used for the swarm

docker node ls

can only execute on manage machine.

docker service create --name psight1

-p ~~8080:8080~~ 8080:8080

--replicas 5

nijelp/pluralsight

replicas/tasks to create

(5) 5 ~~tasks~~

(psight1) - service name.

(nijelp/pluralsight) - image name.

(8080:8080) map port 8080 on entire swarm to 8080

---

~~do~~ service

docker ~~service~~ ls

inside of each container that's part of the service.

docker service ps

docker service inspect psight1

You can hit any node in the swarm and you'll always get to the service. ("routing mesh")

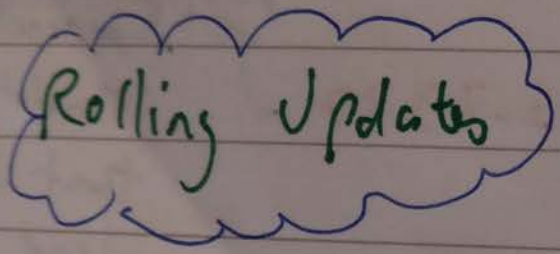
load balance

=> docker service scale psight1 = 7

// set psight1 to 7 replicas

same as

=> docker service update --replicas 7 psight1




=> docker service update --image nijel/tv-demo:v2
--update-parallelism 2
--update-delay 10s
psight2

Update 2 tasks at a time, wait 10s, then update 2 tasks, etc.
Use image nijel/tv-demo v2.
psight2 - name of service

service name.

⇒ docker service ps psight2 | grep :v2

label & image.

easier to read!

⇒ docker service inspect --pretty psight2

# Stacks & Dist. Application Bundles

1.12 experimental

Stack defines service in app.
Deploy whole app from file.

Stack is application comprising of multiple services.
Deploy stacks from DAB file.

⇒ docker stack

file must be in DAB format

⇒ docker stack deploy voteapp

voteapp ⇐ dab file

## Virtual Machine.


O/S only exists to facilitate the application.

Each VM needs an O/S.

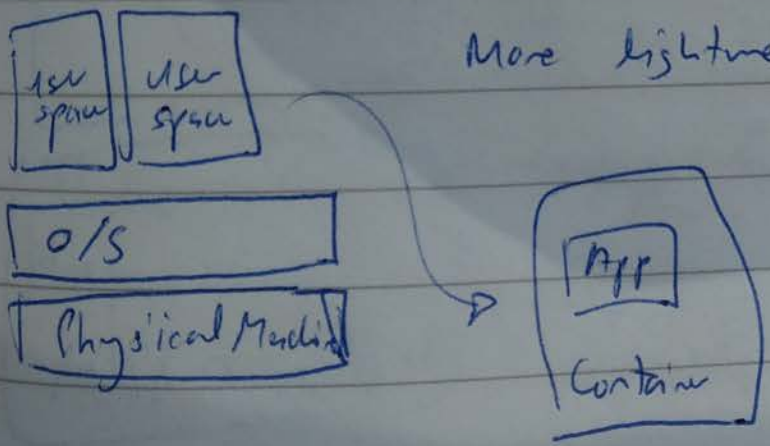
More O/S ≠ Business Value

More O/S = More overhead

## Containers

Container is lightweight!

More lightweight than VM.




(OS level virtualisation)

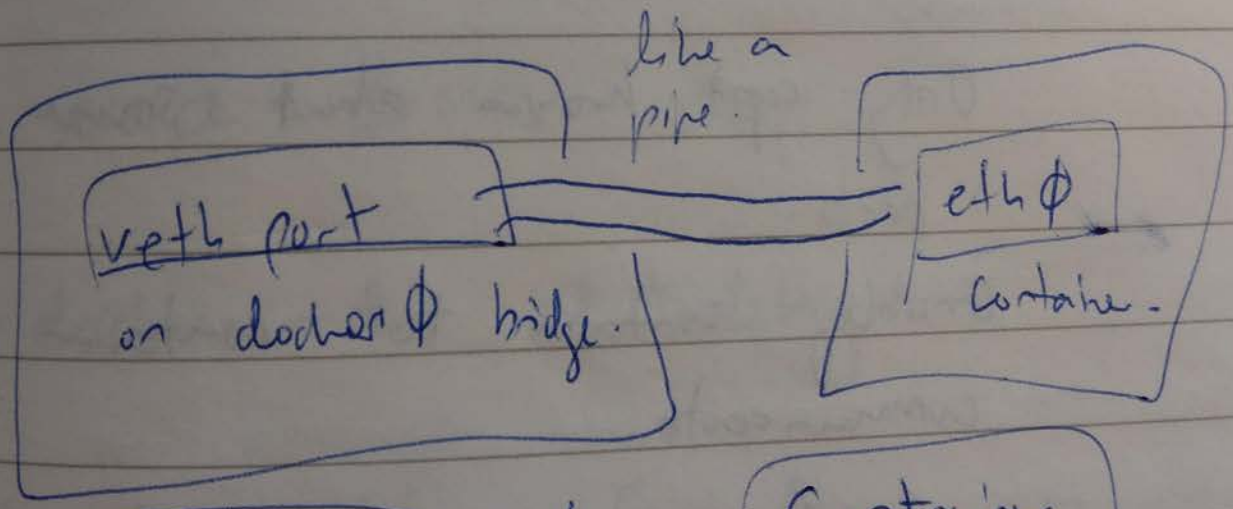
Container in a isolated area of user space.

Containe shares a common kernel.
The kernel & the operating system
& the host.

# Networking

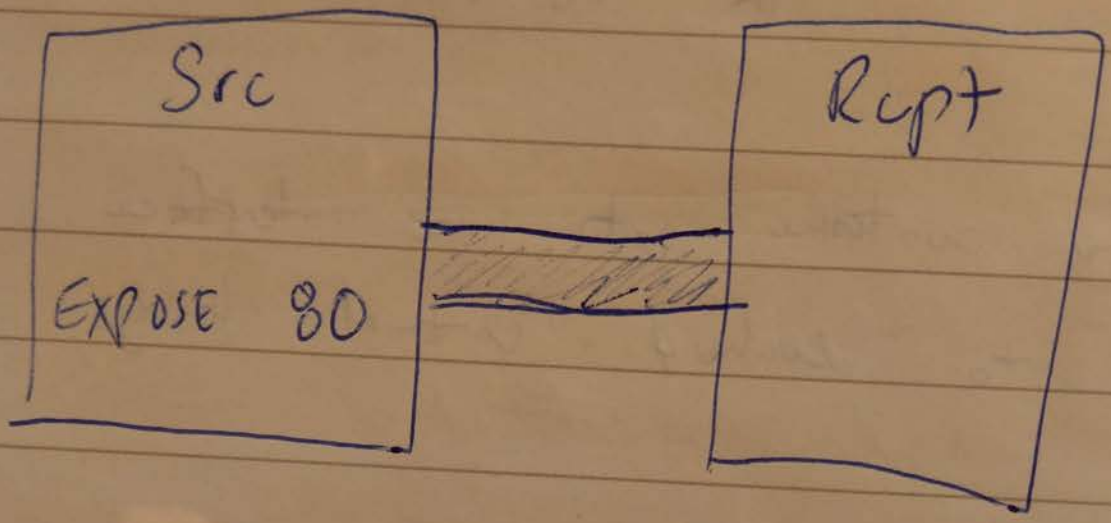
docker0 — switch implemented in software

Each new container gets one interface assigned to docker0 virtual bridge




Docker Host /etc/resolv.conf ~~Target~~ → ~~copy~~ copied to Container

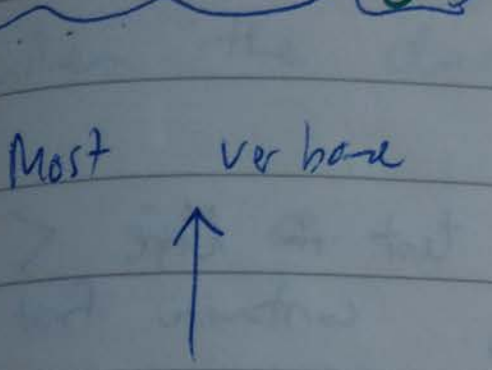
# Linking Containers




Only rcpt knows about source ports

Enabling container to container communication.

# Troubleshooting



Most verbose

Least verbose

Debug

Info

Error

Fatal

> service docker stop

> docker -d -l debug

Specify options for logging in

/etc/[redacted] default/docker

logs to syslog on Docker host

## Testing Dockerfile.

1. Spin up container; test in live
2. ~~Create Dockerfile~~. container first.
   Run commands first
3. Create Dockerfile; after all commands are working.

Use intermediate image for troubleshooting if build fails by spinning up the image that didn't dispose.

## docker0 bridge config.

possible that addresses are already in use somewhere else on the network. So you may need to tell docker which network address range to assign

to the bridge. This gets done when the daemon starts.

> ip a

inet 172.17.42.1/16

(same as 172.17.42.1 with subnet mask 255.255.0.0)

16 bit subnet mask.

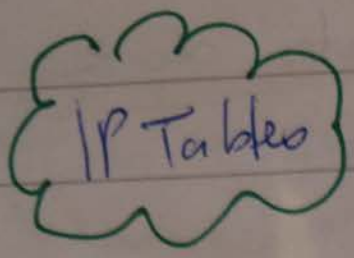
bip option (bridge IP option)

So any address from

host address 172.17.0.0 to 172.17.255.255

address space

is part of the network.




true or false

--icc (inter container communication)

--iptables. (false means Docker should not mess w/ iptables rules)

* both default to true

>iptables -L -v

Inspect containers by doing

>docker inspect containername.

Show version of Linux kernel

uname -r

Verify NAT functionality & Docker after install

iptable -t nat -L

CMD instructions get interpreted as agument to ENTRYPOINT

# docker swarm

Native to Docker.

Swarm is a clustering tool.
→ Cluster docker hosts into a single pool.
→ managing at scale easier.

## Docker architecture

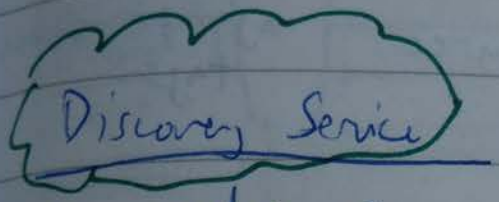
client - server.

client sends command to daemon over port 2375/tcp

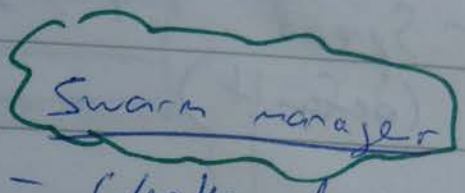
native client or 3rd party product.

native or 3rd party that implements docker remote API

Swarm api, mostly compatible with Docke remote API.




⇒ Key-value store
⇒ Stores cluste state and config
⇒ Needed for cluste to operate properly.
⇒ Supports High Availability eted or Zookeepe.
as discovery servic.




- Cluste admin
- Accepts Docke commands.
- Executes command on the cluster.
- Only 1 ever primary manage, N secondaries.
- Primary manage sends all commands to cluste, secondaris proxy commands to primary.
- Implements Swarm API
- Support High Availability

(cluster wide)

Scheduling
- Random
- Binpack (opposite of spread)
- Spread (default)

Filtering
- Affinity
- Constraint
- Resource!

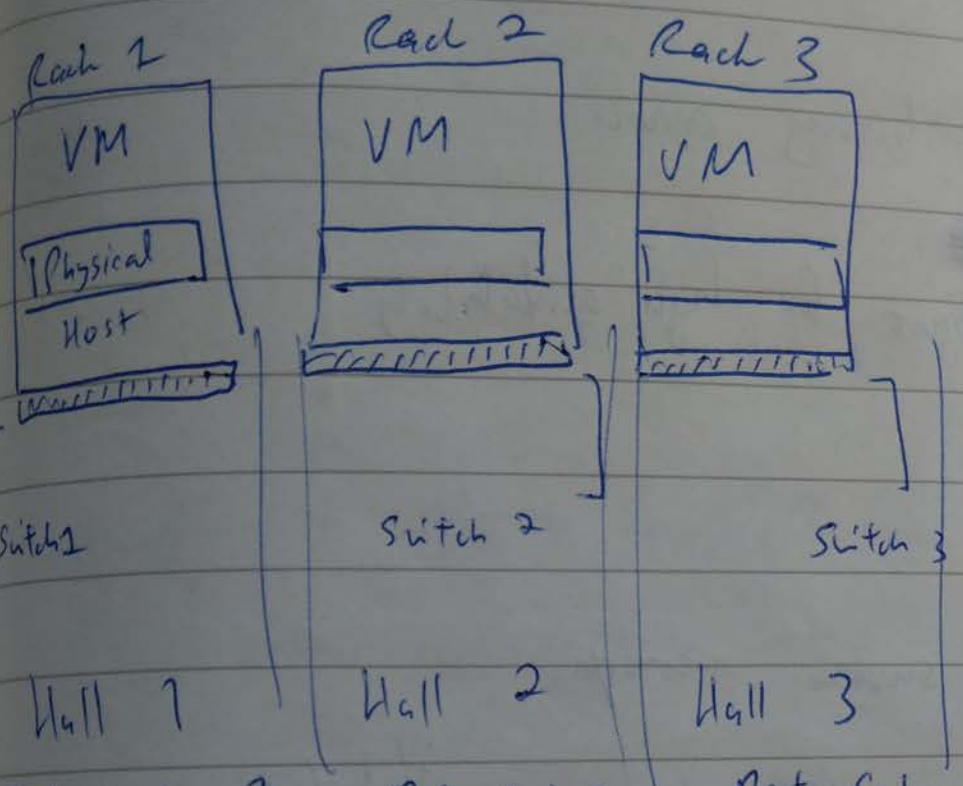
→ Lets you tag nodes like location, environment, zone, etc. then filter against those tags.

Binpack
- Starts on the smallest node. (Max out nodes before moving to next node)

Binpack
- Even stopped containers are considered

Spread

Only 1 scheduling strategy applies to the cluster.




* Avoid Single Point of failure.

Principles are universal.

☐ Step 1

- Build discovery service

Configure for high availability

☐ Step 2

- Build swarm managers

Configure for high availability.

Install on same nodes as discovery service.

☐ Step 3

- Add nodes to cluster

Securing Swarm clusters using TLS

Public Key Infrastructure.

docker

Configure daemons to run TLS

Configure clients to issue signed commands.

// set DOCKER_HOST to swarm manager

// export DOCKER_TLS_VERIFY = 1

// tells TLS to use for verification

// or authentication

// export DOCKER_CERT_PATH =

/home/ubuntu/.docker

// where to find keys.

(ca.pem, key.pem, cert.pem)

e.g export DOCKER_HOST = manager1:3376

# Lesson 1 8/10/18 AWS

NIST definition
- On-demand.
- Broad network access
- Resource pooling
- Rapid elasticity or expansion.
- Measured service

Service model - how you are consuming cloud

Deployment model - how you are building the cloud.

Private cloud for private uses. Have to build everything yourself.

Public cloud for public, everything is already set up for you. Spend what you use.

Hybrid models — adv

also allows for higher availability. since you also have a private cloud.

- so you don't have vendor lock-in.
- Increasing on a daily basis.

## leading cloud providers

| AWS | Azure | Google |
|---|---|---|
| ① Iaas | ① Paas | ① Saas |
| Saas | Iaas | Iaas |
| Paas | | Paas |

## AWS

- Lauced in 2006.
- Leading in Iaas space. (2017)
  80% market share.
  MSFT - 20%
  everything else - 10%

## AWS Regions

where the data centers are.

## Edge locations

not a data center
for caching ~~data~~ only.
- used by cloudfront, ~~transfer~~ acceleration, etc. (and other services which aren't transparent)

AZ - collection of data centers seperated physically between ~~the~~ each other.

China region is ~~done~~ controlled and managed by 3rd party provider.

## Lesson 2 9/10/18

### EC2

↳ just a virtual server / machine.

EC2 uses kvm. for virtualization and nitro.

70+ types of EC2 instances

Pay for what you use.

Diff families for diff workloads.

Some families come with Disk.
Most families are CPU + memory.
Some come with disk and memory. graphics.
(light sail)
LS has bandwidth. (to be used for web applications)

F1 → FPGA - building logic boards. (programmable CPUs)

T2 → CPU + memory. (burstable, cannot always use 100% of CPU)

R4 - 32 gig, 64 GB, etc.
X1e - 2 tb of memory, etc.

Bare metal

-> dedicated server.

-> currently beta.

Elastic GPU

-> Use any of instance types and attach an additional GPU. 1 gig, 2 gig 4 gig etc.

generation

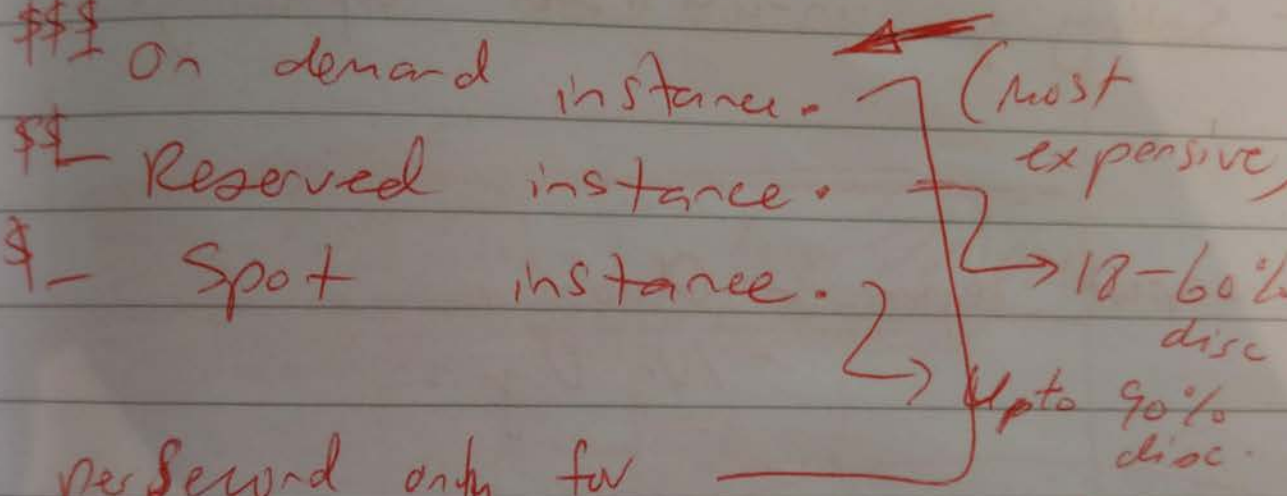
m5.large

"clothing size."

instance family

Compute opt → more CPU
Mem opt → more RAM
Gen purpose → about the same.

## Instance Type (billing models)



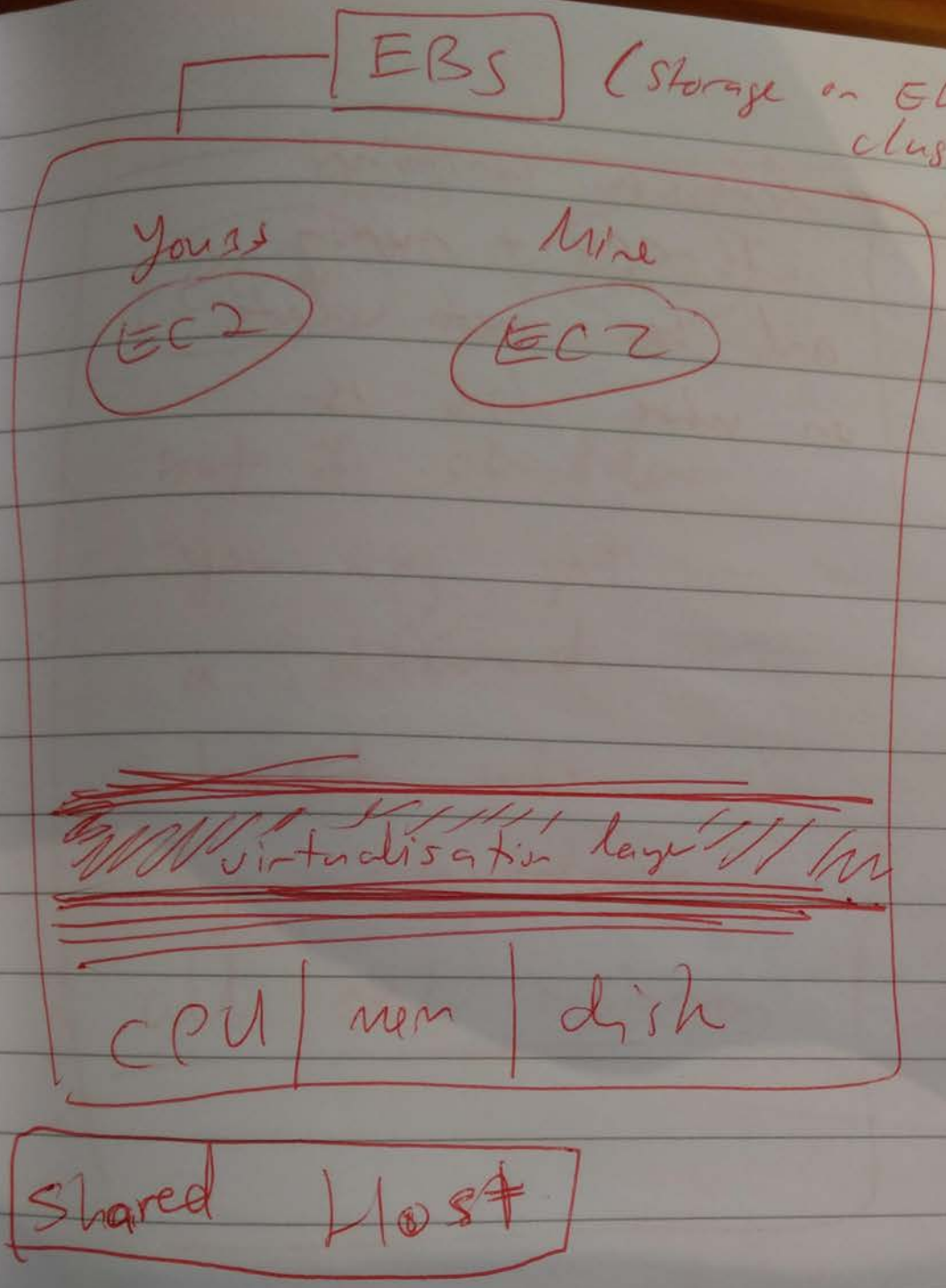
Reserved instance
- 1yr or 3yr term. contract. Upfront or scheduled monthly cost.

Spot instance $
- Selling of un-utilized capacity.

Cheapest region - Ohio
- N. Virginia.

## Tenancy Models

Where instance is, and how it behave.
- Shared Host
- Dedicated instance
" host



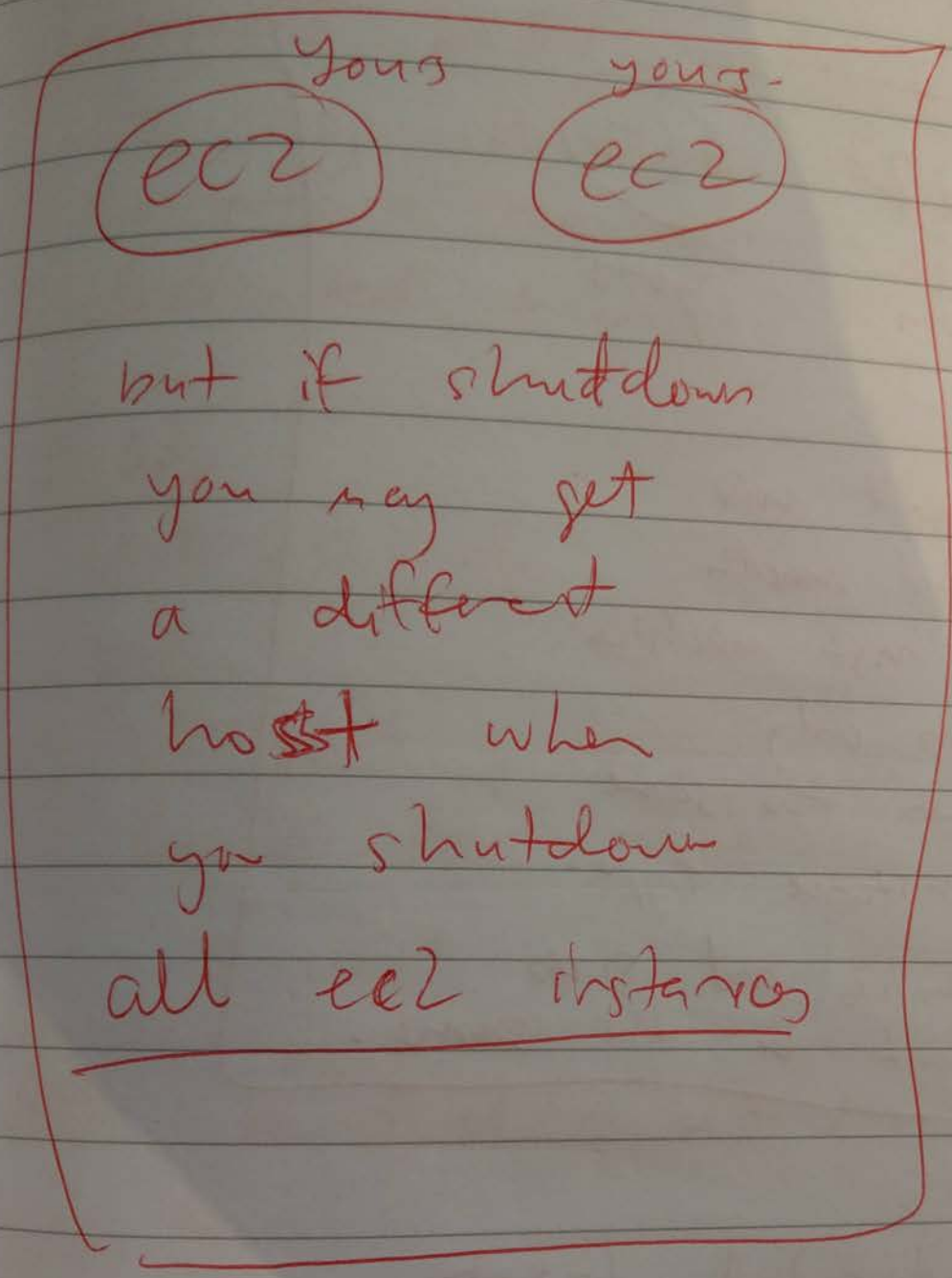

**EC2**

Stop — on virtualisation layer.

Restart — on o/s. layer only ⇒ may start on another physical host.

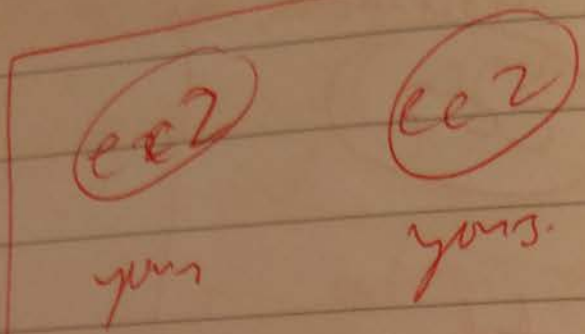
Shutdown — may be on diff. physical host machine.

EC2 — like a container
with CPU + memory
and has meta data
on where disk is.




Dedicated Instance

⇒ no one else can start their ec2 on this machine




yours yours.

Can't mix and match host machines.

Can only run the same instance type.

families and sizes must be the same.

dedicated host

(~~your~~ always yours)

even if you have no ec2 instances.

## Dedicated host usage

- Licenses based on CPU core eg Windows licenses.
- Compliance requirement eg dedicated machine

## Dis/Adv

Under utilisation

cost

Single point of failure. (so at least have 2)

Amazon

AMIs are stored in S3 system.

Tagging ec2 instances is important to specify meta data.

/etc/ssh/
sshd_config

↓

if you want to
chage ec2-user
or use root

Keep
~~Use~~ bootstrap down to
5 minutes, otherwise use
AMI instead.

# Lesson 3 EBS

EBS volumes.
(Elastic block store)

EBS is a service independent of EC2

| block | object | file |
|---|---|---|
| ebs | S3 | efs |

like a hard disk for EC2 instance.

Snapshots to private S3

Pros

(network ~~dat~~)

T2 has no EBS data pipeline, shared between data and network.

Instance store No foote
but only temporary.
Can use database but
need a plan to sync
the data. like Goldengate.

---

Writing /
Reading large chunk &
data better to use HDD
since data is sequential.
and HDD has high throughput.

Getting to the file → SSD
is better. like no many
perf.

| Volume type | provisioned IOPs ↑ | (General purpose SSD) |
|---|---|---|
| SSD - | io1 | / gp2 |
| HDD - | st1 | / sc1 |

if you don't know, gp2 almost works for everything

(IOPS)

(Throughput) w/ large sequential I/O

large block, high throughput (good for logs)

cold HDD

Provisioned IOPS

|

input output operations per second.

gp2

→ 1 gb = 3 IOPS

hard disk gives you ↑

→ Minimum IOPS is 100.

so if you have 1 tb hd you get 3000 iops.

but you can provision IOPS using provisioned IOPS; always guaranteed.

Speed & read and write is determined by IOPS.

Non-provisioned IOPS does not give you a guarantee, but you can burst occasionally.

EBS snapshots are stored in S3.

Snapshots can be spin up a volume, or an AMI

↓

but cannot be used as root volume for AMI

By default use AMI if you need a root volume. (like an iso)

AMI is an image of a volume.

Snapshoting is always incremental.

Root volume is always EBS

# Lesson 4

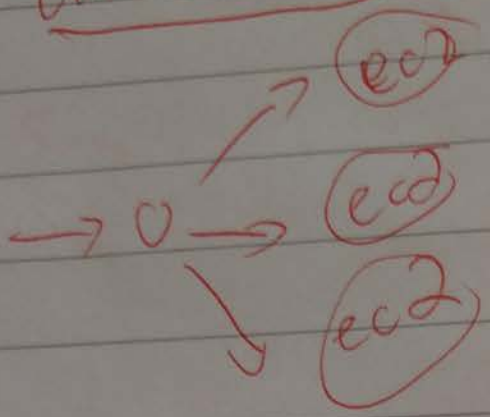
## Load balancers.

### ELB

Application load balancing
→ does health check of EC2 instances
→ ELB offered as service
→ automatically scaled
→ Integrate with auto scaling.

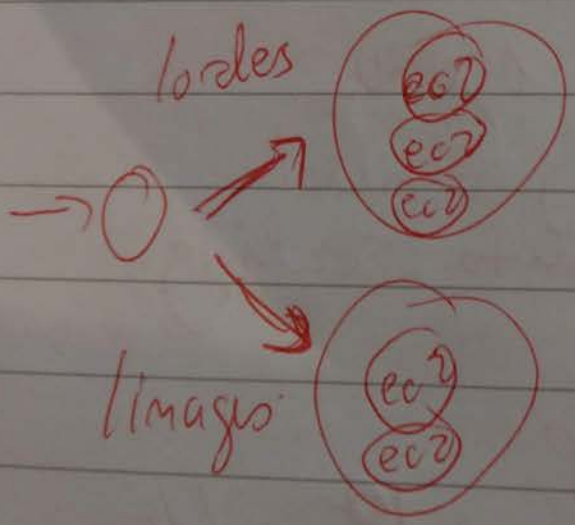
→ http https tcp traffic to any IP address or ECS endpoints
→ Can also do SSL offloading so ec2 instances do not need to spend time on SSL encryption.

# Classic load balancer (ELB)




can't use target group.

# Application load balancer




target group for orders

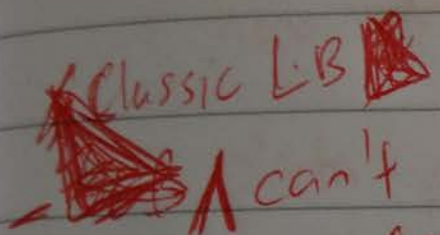
images for [redacted]

## Dis Adv

Single point of failure for load balancer

# Network load balancer

- no such thing as rules.
- only route traffic based on port numbers.




- fast
- can't scale when traffic spikes. (Solution is to "pre-warm")

- network load balancer doesn't have this problem.
- Application load balancer also is prone to spikey problems.

Classic charged per gigabyte.

# Lesson 5

## S3 (object storage)

An object is basically a file.

11 9s of reliability (but not a confirmed number)

Stored across against 1 & 6 devices at least on multiple AZs.

S3 has a public interface as well.

Standard
Inf. Access
Reduced Redundancy
Single Zone IA (infrequent access)

} 4 storage classes.

## Buckets

Are like root directory

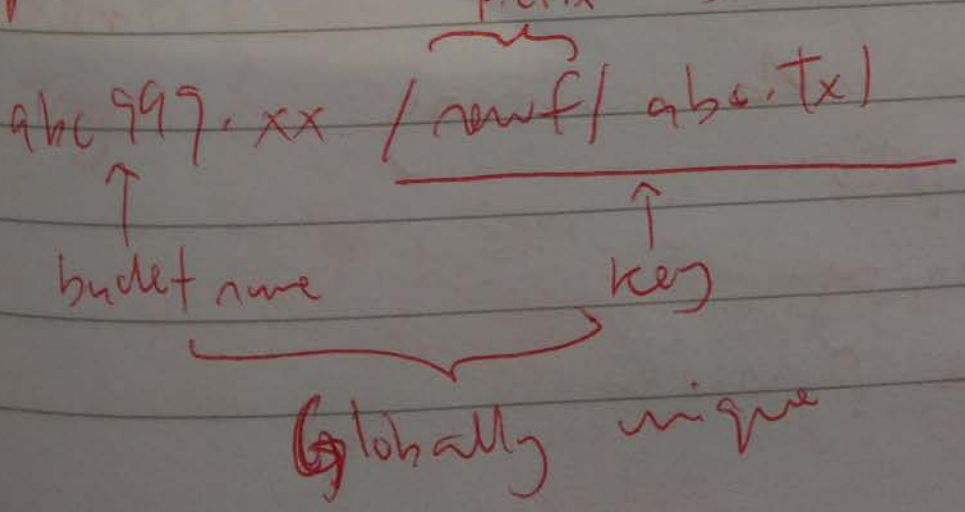
## Objects

Like files.

- has data & meta data.
- data portion is opaque to S3.
- meta data set of name-value pairs.
- every object on every bucket is unique.

**S3 structure**

S3 has no hierarchy.



## Structure of S3

Web storage NOT file system

- Write once, read many
- eventually consistent

store at least across 3 AZs.

meta data in indexing
data in storage.

## Features of S3

- Static web hosting.
- Encryption
- Versioning.
- Bucket policies
- ...

- Transfer acceleration.
  (upload to closest edge location so it might be faster)

- Signed url/cookie (using Cloud front)

- Logging. (who is accessing and operations performed)

- Requestor pays (only work with other AWS accounts)

- Storage management.
- Events (used by developers)
- Tags (e.g. any file with tag & log can be deleted after 60 days with a lifecycle policy)

- API / CLI

Charge basis
- size
- access freq.
- redundancy

Glacier
Archiving only
for long term storage.

Vault lock, stored data cannot be deleted for X amount of time.

No UI for glacier to upload

Everything you upload by default is private.

Bucket name must match domain name

e.g. abc.com

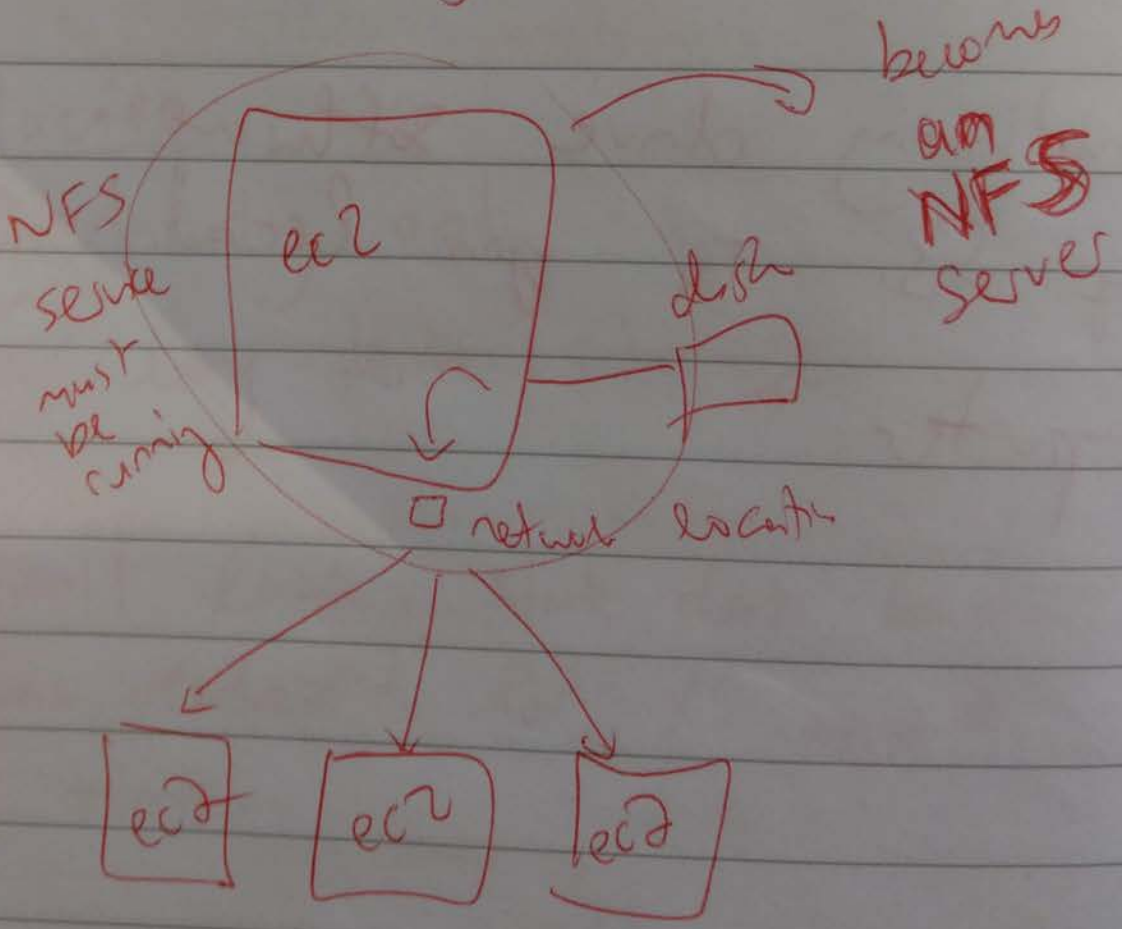
Cloudberry drive software map S3 to your local computer.

# Lesson 6

## EFS & other storage

## NAS in the cloud

EBS
Think of ebs like a hard drive.
Can't attach to multiple EC2s directly.



This server is now called EFS, so you don't need to do it manually

NFS v 4.1 compatible.

EFS has direct mappable path.

Pay as you go service.

Windows still does not support v4. So you can only do this with Linux.

## Other storage

1. Storage gateway
2. Virtual tape library
3. AWS import/export

20 TB to get to AWS?
- Use import/export service (3)
- Safe and easier.
- Physical drive transfer

250 TB to get to AWS?
- Use AWS snowball.
  - 50 TB
  - 80 TB
  - 100 TB
  } options
- Physically shipped to your location.
- Military grade device

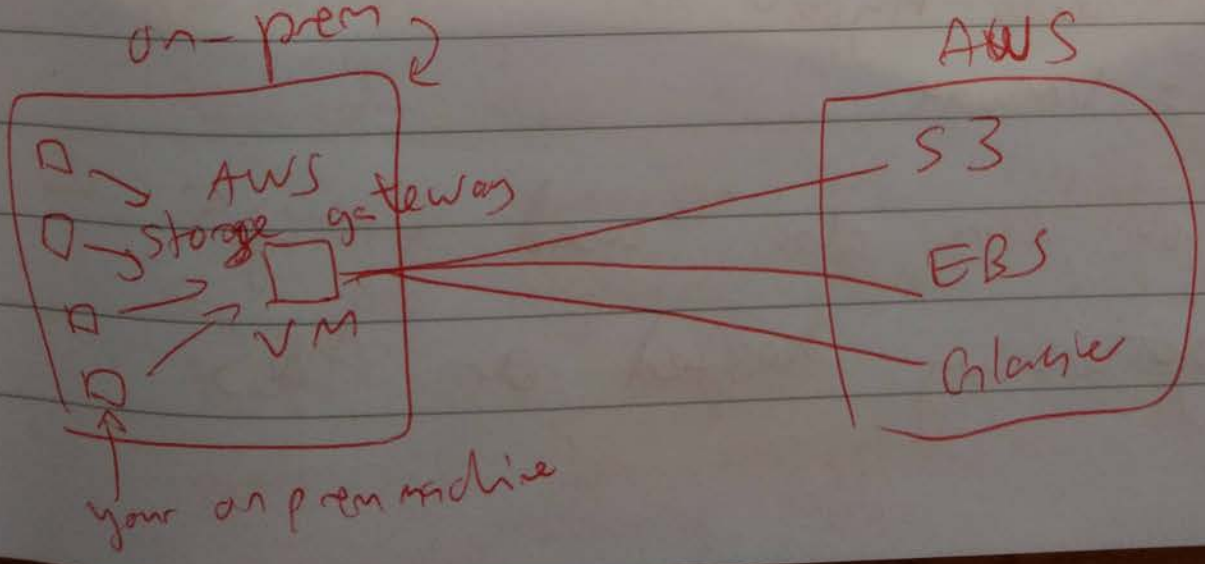
100 TB option ~~add~~ also comes with a CPU in the box.
EC2 + Snowball option.

- So you can work on data while it is processing.

50 Petabyte of data?
- AWS snowmobile
  100 PB ~~transfer~~ of data
- It's a truck!

takes ~6 months to transfer.

① Storage gateway (run as virtual machine)
- can bring data
- hybrid storage connectivity



## Adv of Storage gateway.

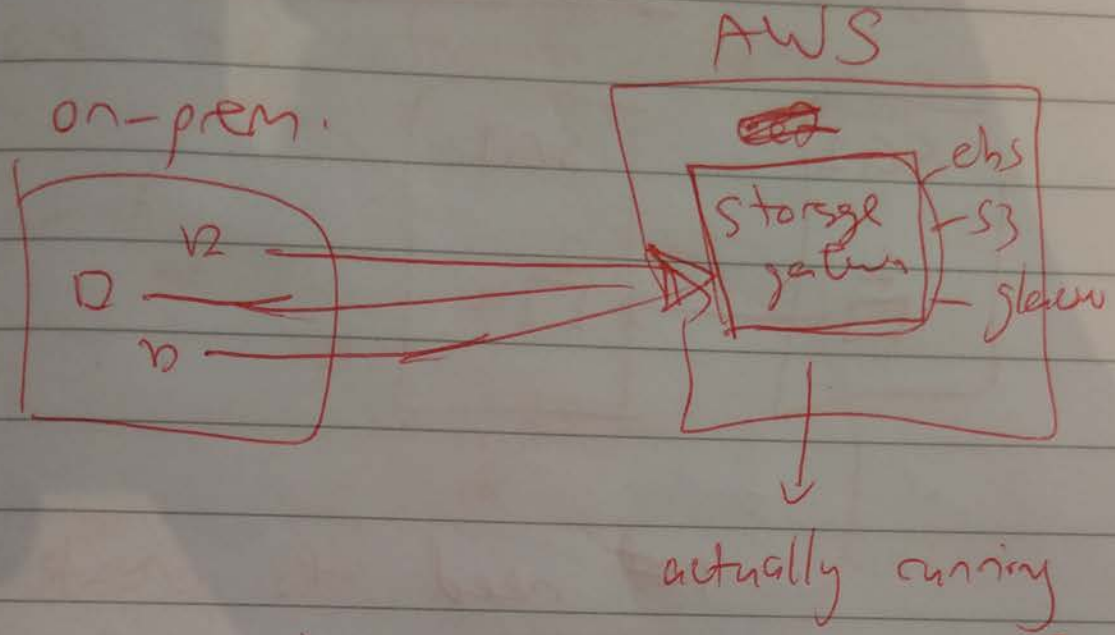
- has ability to do caching.
- very fast access of frequently used files.
- configure size of cache, cache is on virtual machine.

Virtual tape library (VTL) also provided by Storage gateway. Appears as virtual tape which gets backedup to glacier.

Storage gateway
- needs hyperV
- Vmware.

You can also used Storage gateway hosted on AWS.

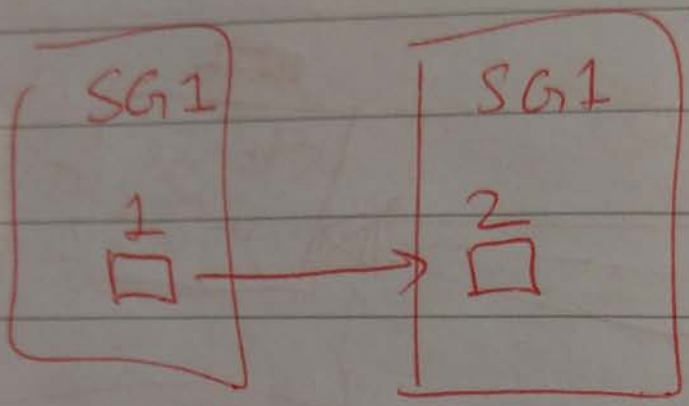
No real point to use caching on AWS.




actually running Storage gateway on an ec2. software

Security groups
- by default, all outbound allowed, all inbound is blocked.
- security groups are stateful. So anything sent out that is accepted on the same port will be accepted on the incoming side.

- security group covers systems individually.




~~not~~ need to create a rule that says SG1 can connect to SG1.

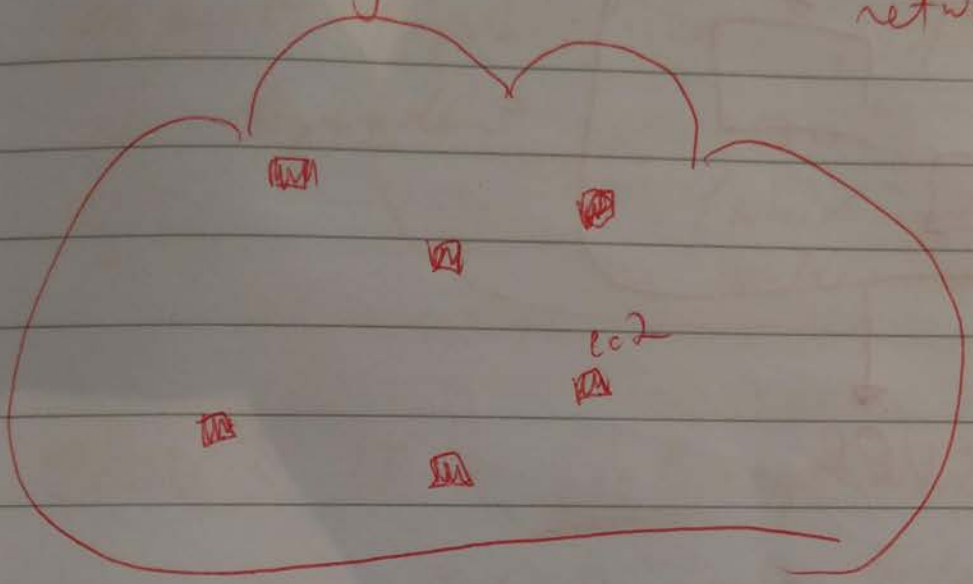
SG1 → SG1 for port 22 for example

~~EFS~~ uses regular network traffic

# Lesson 7 VPC

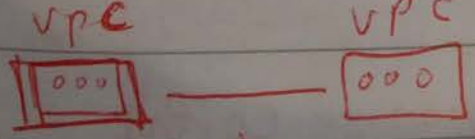
Before VPC ...

one big network. Same network.



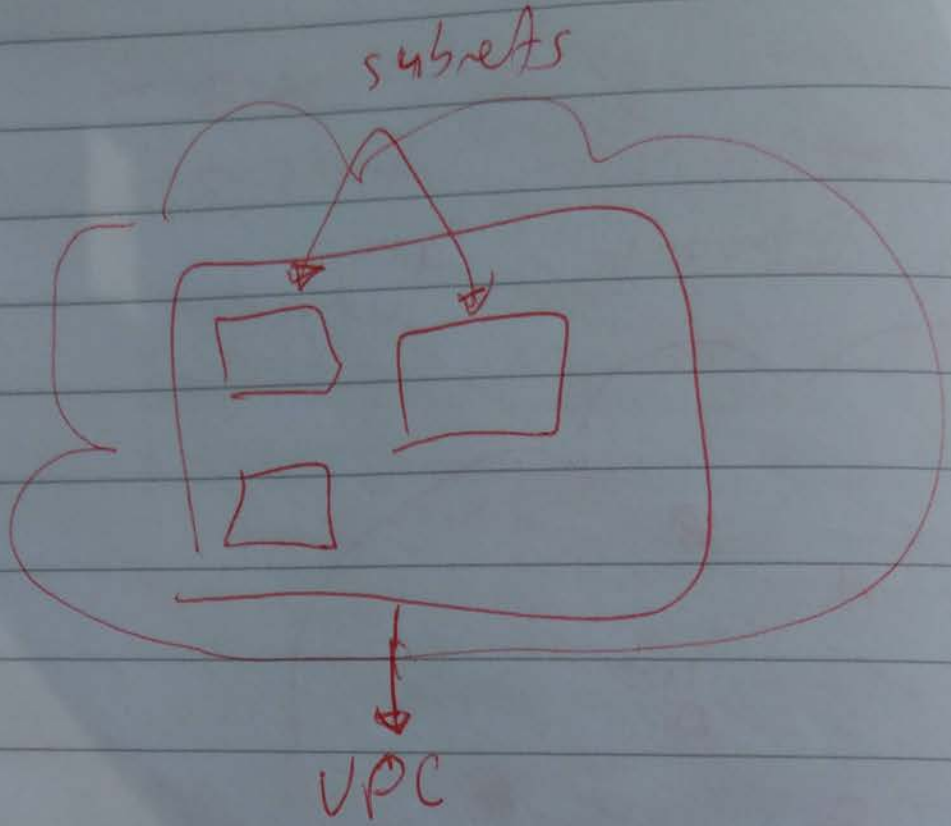

Had to configure each instance individually.

Virtually seperating them. using VPC




↓

VPC peering. a connection between VPC to VPC.




Subnets can always talk to each other since they are on the same network.

private

VPC created across region, covers all AZs.

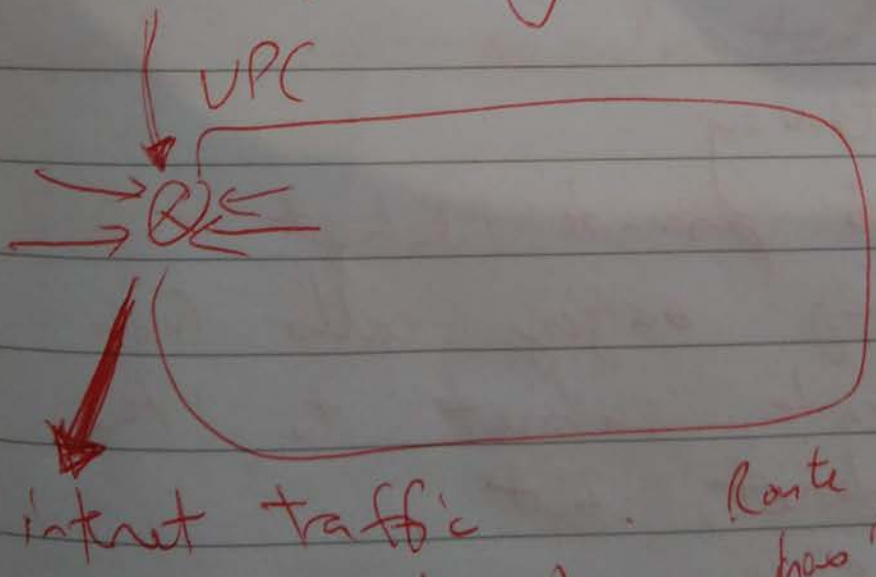
An AZ will have at least 1 subnet.

Subnets "public" or "private" is dependent on route table. (which route uses)

Elastic IP – Static public IP

ENI needs assigned IP address.

Internet gateway



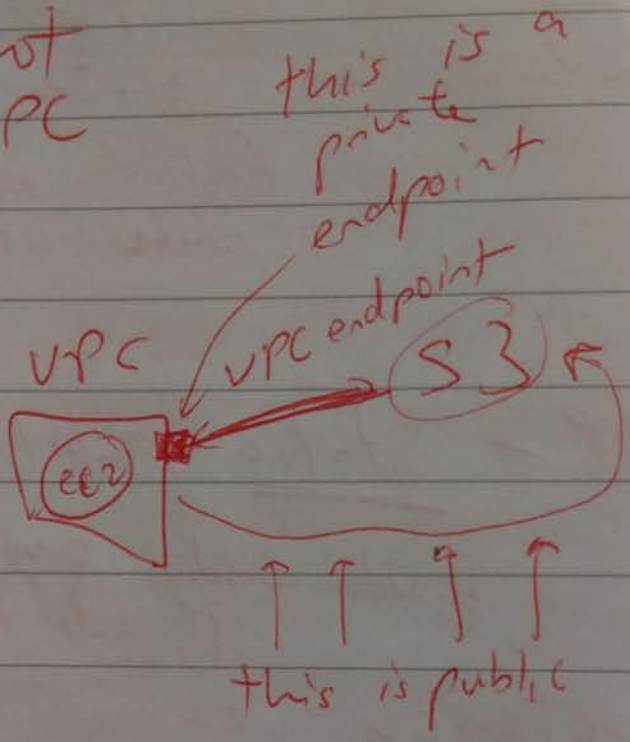

internet traffic in and out.

Route table has internet gateway? Then subnet can talk to the internet.

S3 is not in VPC

API calls to EC2
ELB

are not in VPC




NAT gateway
- protect private subnet allowing outgoing calls from private subnet to the internet, but
- only IPv4

IPv6 use Egress only NAT gateway.

## IP addresses and subnets

subnet calculator

:odies.de/ipcalc

In VPC
Max is /16
and smallest /28
for CIDR ranges.

## Route tables

Most specific (closest) match rule is evaluated.

Default routetable is private

# Lesson 8

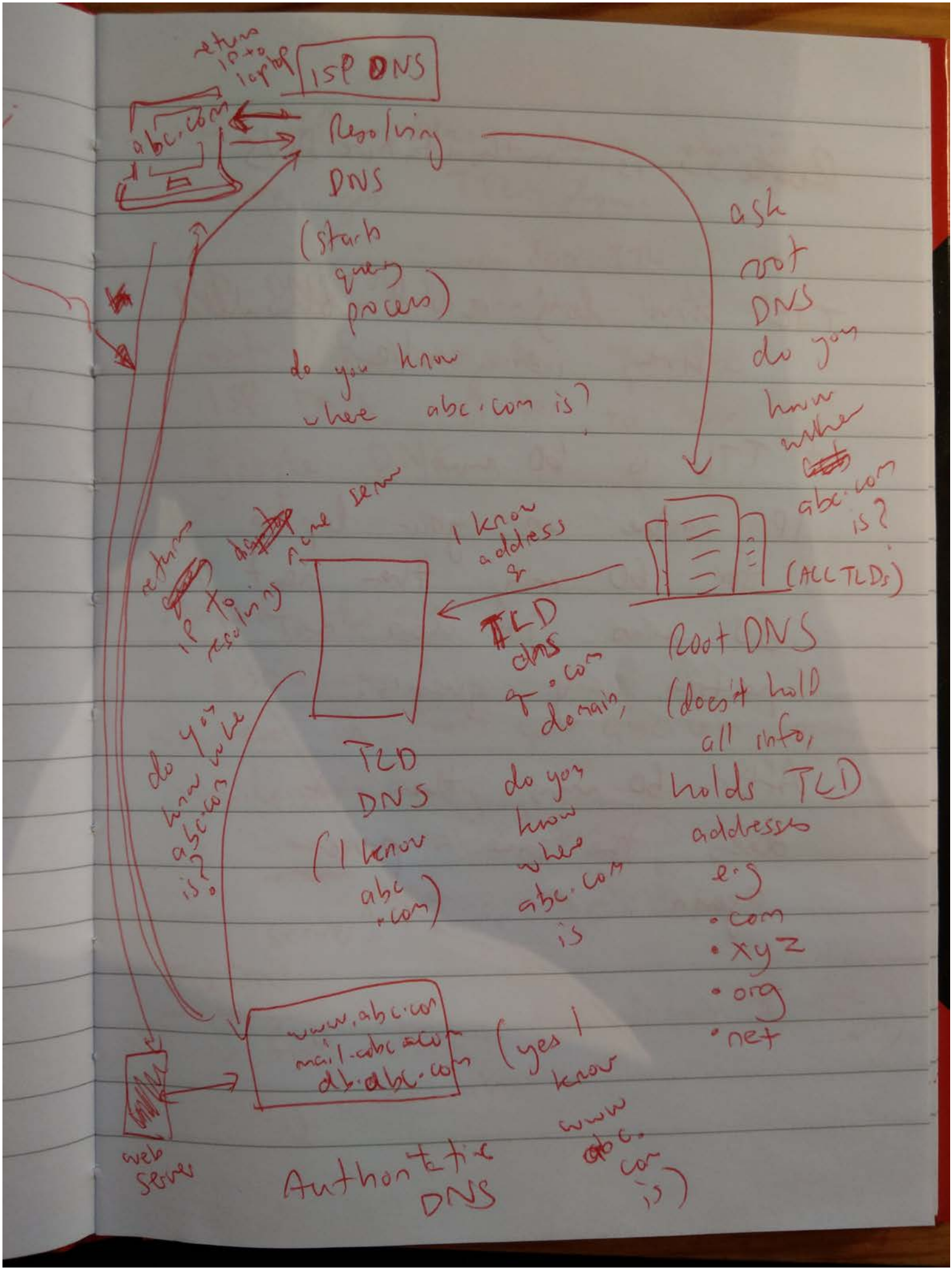
## Route 53

## DNS routing

connect to website

Convert names to IPs
and IPs to names

Name lookup system


return IP to laptop
ISP DNS
abc.com
Resolving DNS
(starts query process)
do you know where abc.com is?
ask root DNS do you know either abc.com is?
(ALL TLDs)
Root DNS
(doesn't hold all info, holds TLD addresses
e.g.
.com
.xyz
.org
.net
I know address & TLD dns for .com domain
TLD DNS
(I know abc.com)
do you know where abc.com is
do you know where abc.com is?
www.abc.com
mail.abc.com
db.abc.com
(yes I know www abc.com is)
web server
Authoritative DNS

Route 53 is authoritative DNS

TTL how long record should live, i.e. cache

If TTL is 60 mins,
IP cached on your laptop for 60 mins, then next 60 mins it does not perform any queries.

After 60 mins, then it does the same process again

If IP address keeps changing, keep your TTL low.

ISP DNS may hold very long TTL caches, that's why not that good to use ISP DNS. Better to use google DNS.

## Naked domain

abc.com

whatever in front of abc.com is subdomain eg

www.abc.com

server1.nv.abc.com

## FQDN

is full address (subdomain + abc.com)

eg www.abc.com

# Route53 - 100% uptime

ns1...com 4 diff clusters
ns2...co.uk 4 diff TLDs
ns3...net
ns4...org

abc.com → 123.123.123.123

* A record - convert name to IP address

* Cname - name pointing to another name

www.abc.com → abc.com

* NS - lists nameservers.
* MX - mail exchange records (IP address of email server)

* Every domain has SOA record

start of authority
first nameserve, who has authority over the domain

123 * TXT record = holds additional info about domain (how an spf1 record) validating a certain request.

* A (point to IPv4)
* AAAA — point to IPv6 address

## Routing policies

- Simple Based Routing
- Latency Based Routing (LBR)
  (check fastest route for user)
- Weighted routing policy
  - Key TTL to ~~set~~ size was low

- Geo location routing policy
  determines which user
  from where goes
  to which region.

- Failover routing policy
  if primary region fails
  return secondary region.
  (for example)

## Lesson 9 IAM

AWS — Hardware

Customer — Guest O/S

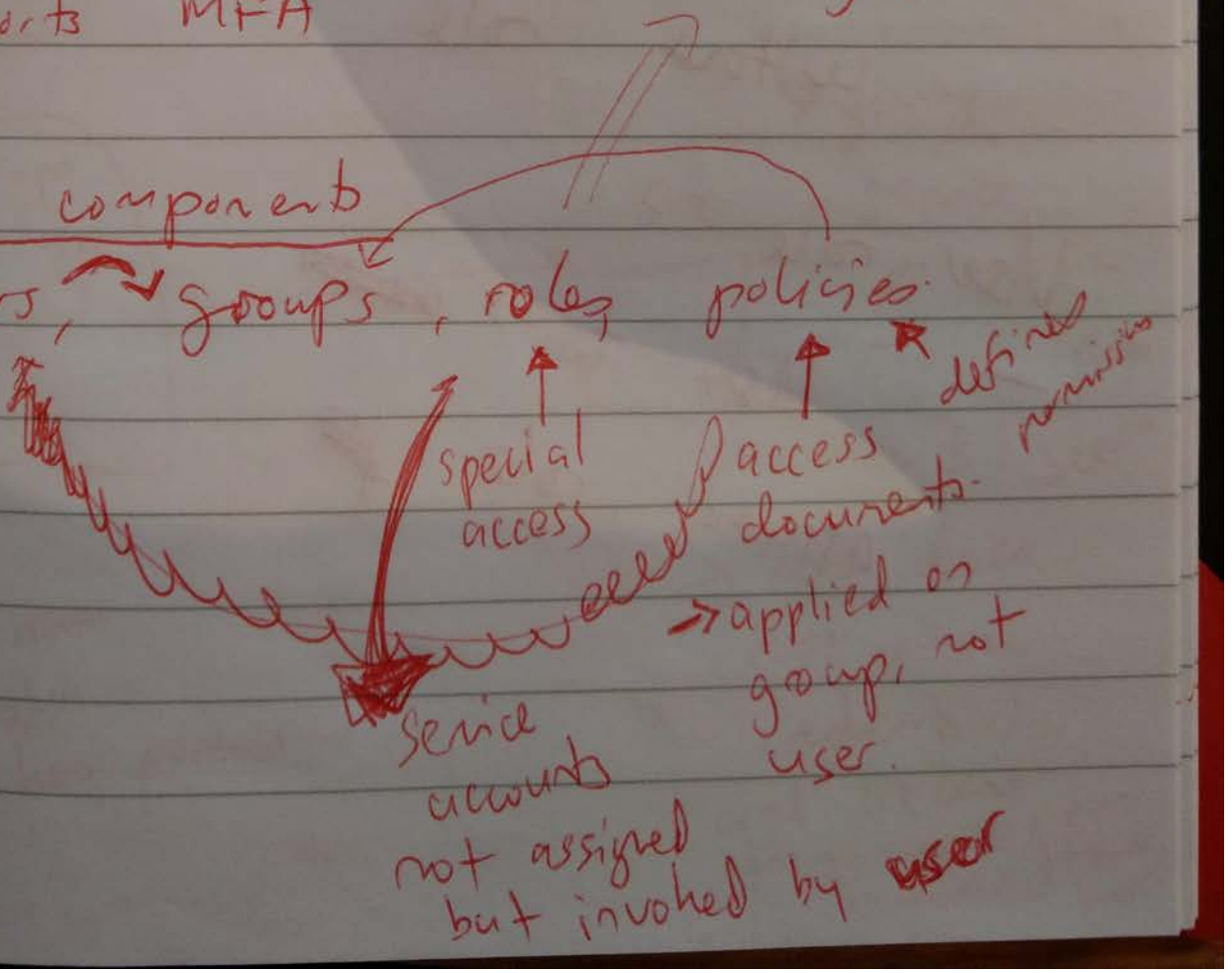
...

Security level are different
depending on service.

# IAM (identity & access management)

- controlling services to AWS.
- users, groups and roles
- ~~Cost~~ can federate with other systems
  eg Active Directory.
- cross account access
- granular control of permissions
  - what
  - when
  - where
- Supports MFA

## IAM components



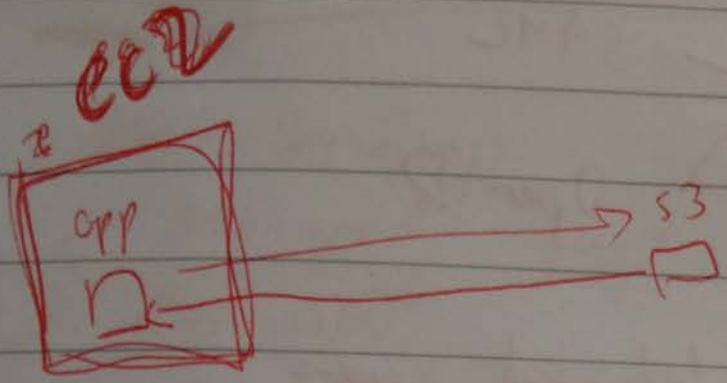
All policies are combined and evaluated.

S3 Bucket policy for public not limited to just users of AWS.

## Roles

Service role
- 1 system to access certain role

User role
- User to assume role




User role

programmatic access

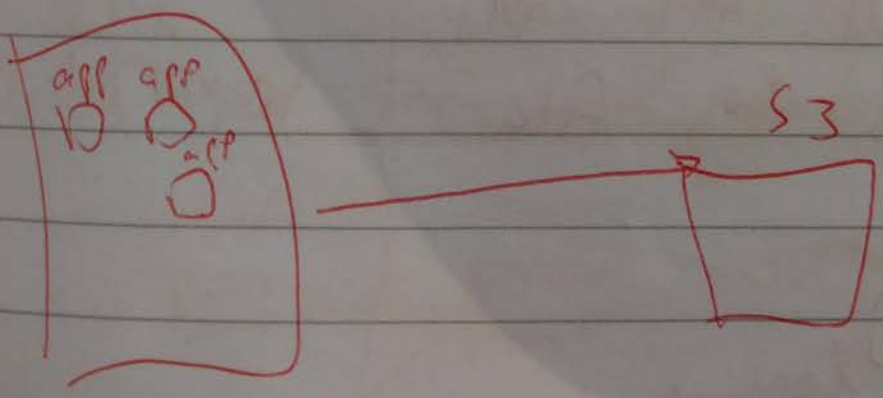
A/C access key
access id

lose account
have to
delete key id
and id

Service role

ec2

↳ giving service permission to access S3 [another service]




app knows also has permissions

→ role does not store credentials, and access keys are rotated every few hours

Identity providers
- Federation { SAML, OpenID

- configure a federated access with SAML or OpenID authentication

* Cloudtrail
  - logs all API calls.
  - creates log files
* VPC
  Flow log
  - VPC logging

} Security

Encryption keys - AWS key management service for storing keys for encryption

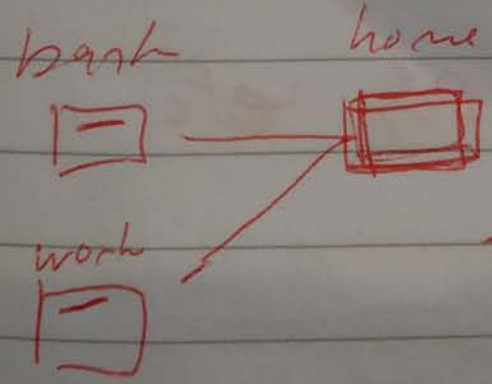
# Lesson 90
## AWS Database Service

DB Services
- Managed, just use it databases
- no full control

① RDS - MySQL, Aurora (built for cloud) 64 tb limit
NoSQL - DynamoDB
Elasticache -
Redshift -
Aurora -

---

RDS
① Managed relational databases




- Adv
- Master and standby db.
- DisAdv
- Pull on multiple tables
- Could be slow

- Service layer only
- no root permission

② No (not only) SQL
- key, value pairs
- can be text, documents, text, etc.
no fixed structure.
- faster, but do not have ACID compliance.
- Eventually consistent

→ Absolute accuracy

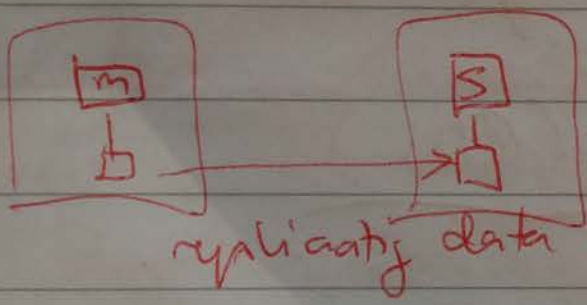
SQL → used for banking compliance for example.

NoSQL → Mongo DB, etc.

RDS availability

- Single AZ - not good if you want availability.

- Multi AZ — failover is automatic, endpoint is the same.




replicating data across

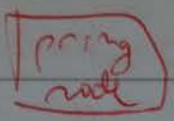
(synchronous replication using MySQL) only

## Aurora

relational database

64 TB, 15 read replicas

AZ1 AZ2 AZ3




Continuous backups to S3

## Read replicas (RDS)

within a region
- MySQL
- MariaDB
- PostgreSQL
- Aurora.

Cross region
- MySQL
- MariaDB

## Read Replicas (Oracle and MS SQL)

### Oracle

Golden gate, etc.

### MS SQL

Snapshots, third party apps

Relational databases
scale vertically better.
But single point of failure.

Today more (more volume) reads than writes, hence concept of read replicas.

## AWS Database Migration Service

- Convert one database type to another database type.
- AWS Schema Conversion tool.

DynamoDB (will be in exam)
NoSQL data store.

- no concept of slave, master !
- read units, write units (like IOPs)
- no concept of server.
- ~~Just a table~~
- All SSD storage.
- Partitions, have to think about how you partition data.
- No relational tables, but key value tables.

Elasticache
In-memory cache.

Redshift
for data warehousing & dumping data for ~~queries~~ which you might later query.

- Redshift stores data in column format.
- can do massive parallel queries across columns.
- petabyte storage
- uses columnar storage
- used by BI tools.

## Front ends for connecting to DB

MySQL workbench or heidiSQL

## Dynamo DB

- NoSQL, managed
- Document or key-value
- Scales to any workload
- Fast and consistent
- Acces control
- Event-driven programming

# Day 11

~~Beanstalk~~

(EB)

Elastic beanstalk and cloudwatch.

## Elastic beanstalk

Allows fast deployment
Automated deployment tool
Give it your app and
it will build environment
for you.
Use cases : for web ~~app~~

~~Still~~ allows full control.
Preconfigured infra. model.

Elastic beanstalk came out same time appx at cloudformation.

Elastic beanstalk has a dashboard.

## Cloudwatch

Monitors things on a VM level.

Resource and application monitoring

Also triggers autoscaling.

log aggregation, monitoring and troubleshooting with cloudwatch logo.

By default 5 min monitoring but can go down to 1 min.

Custom metrics must be configured in EC2.

Cloudwatch agent must be installed for cloudwatch logs.

SSM is an interface to run commands. Instance also needs SSM agent running.

~~SSM agent must be running installed~~

To install Cloudwatch agent.

/var/log/httpd/error_log

you can define the path + name to log

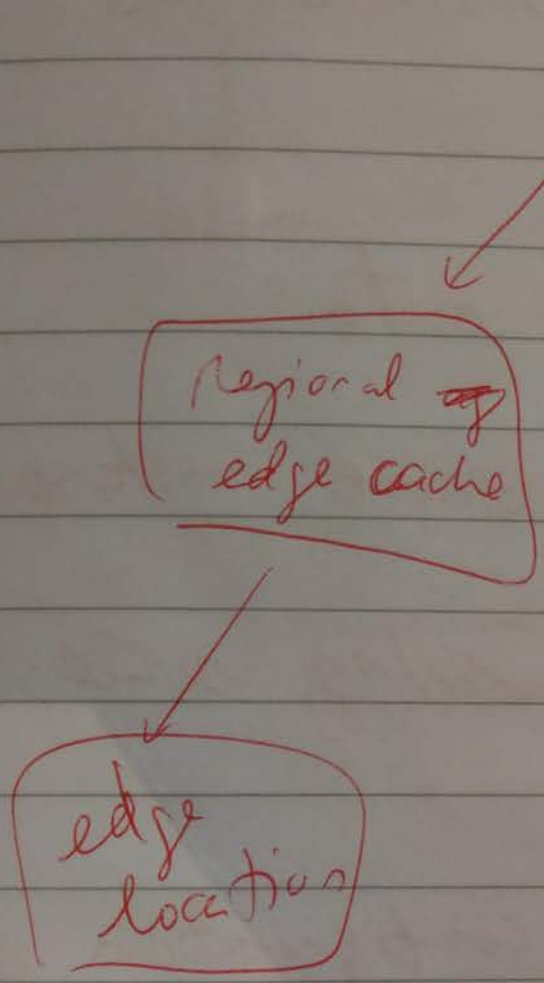
Elastic beanstalk
- easy way to do blue-green deployment

SSM makes it easy to ~~deploy~~ run one thing to many systems. (e.g a shell script)
~~eg~~ run shell script ~~the command~~ (select the right ~~the~~ a action)
No charge ~~for~~ SSM.

# Day 12

## Cloudfront

- 100+ edge locations
- for caching
  - static and dynamic content
  - SSL, geographic restriction, private content




Web -

RTMP - distribute videos based on RTMP protocol.

# Auto scaling

Only for EC2 instances
for now.

Horizontally scale out.

Add or ~~delete~~ remove instances.

No charge for auto scaling service.

Create
1. Autoscaling launch configuration
   - launches requested AMI into group.
   - what EC2 instance, security group, etc.

Autoscaling group can also have scheduled actions.
Lifecycle hook - useful to add software when AMI coming up.

cloud watch ~~watches~~ ec2 metrics

↓

cloudwatch metric

↓

alarm → Autoscaling policy, takes autoscaling launch configuration. (or creates) and launches ec2 into autoscaling group.

# Day 13 Cloudformation

1. VPC
2. DB
3. Main instance for creating AMI
4. Install WP applications
5. Create AMI
6. Create load balancer
7. Auto ~~Scat~~ scaling group

Use S3 for media storage

# Day 14

Cloudformation → Security & Admin

Automate infrastructure deployment

Template → Stack → Change Stacks (dry run)

(review change before updating existing stack)

no interruption

some "

replacement

# Security & Admin

AWS management tools.
- resource provisioning.
- config management.
...

Trusted adviser
- Check AWS account for cost, performance, fault tolerance, security.

ACM
→ SSL cert auto-renews.
→ soft limit of 50 certs.
all free.

KMS
→ integrate w. EBS, S3, RDS, Redshift
Elastic Transcoder,
Workmail, EMR

AWS config
- Inventories AWS resources.
change log &
all config changes.
(resource changes)

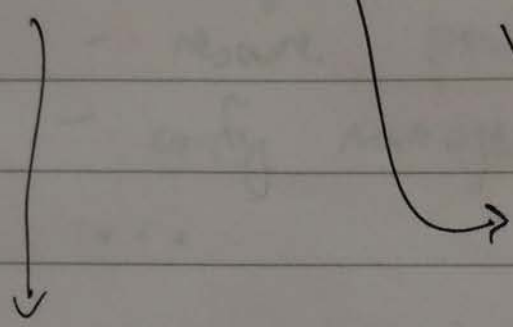
AWS service catalog.
- IT service portal.

OpsWorks
- Provide you with Chef or Puppet server.
Managed service

EC2 Systems Manager
- Can also manage on-prem, EC2 instances,
To manage instances.
- need SSM agent and IAM permissions. Default ~~avaitable~~ is active on Amazon AMI.

## Enterprise Apps

WorkSpaces — desktops in the cloud (windows & linux)
WorkDocs — login from any device.
WorkMail — VDI environment



AWS version of sharepoint.

AWS outlook
for use $5 per month.

Consolidated billing
- include charges from other AWS accounts.
- 3yA will lose free tier of other accounts.
- ok for companies ☺

# Day 15 Lambda, SQS, SNS

Lambda.

- function as a service
- Using compute cycles
- Still a growing concept.
- Delay (needs some warm up) starting from cold start.
- Integrates w/ alot of services. S3 upload, DynamoDB, Kinesis, API gateway requests.
- Works system that does one action, best for that kind of work. Use in places to replace a system, or small batch jobs.

Application Services.

- SES, SNS, SQS, API gateway.

SES

SMTP gateway,  uses common pool of IP addresses. Sometimes email doesn't get delivered (!?) due to IP reputation issues.

- Use static IP on SES gateway. Maintain this static IP. for better reputation. About $30 a month.
- You can also recive email. on the inbound gateway.

SNS - notifies subscribers.
- push message to mobile devices.
- also supports SMS.
- pub and sub model.
* Check for SLA.

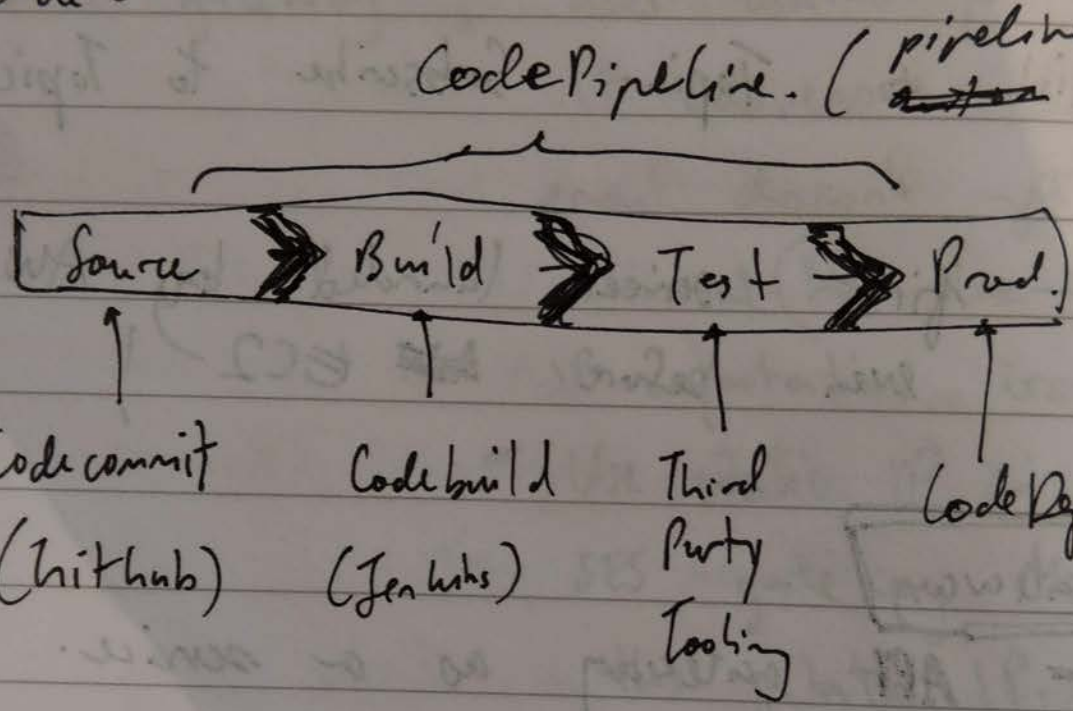
Publish to Topic. Subscribe to Topic.

SQS — first device launched by AWS.
even before ~~AWS~~ EC2 !

API gateway
- API gateway as a service.

# Developer Tools

CodeDeploy
CodePipeline
CodeCommit.




CodeDeploy
- Can use on AWS or on-prem.

CodeCommit
- Source Control.

# EXAM info

- No longer need pre-requisites.
- No one can teach you at professional level.

① Storing access keys is a bad idea.

White papers

* Well architected framework
* Best practises.
* Best for DDos Resiliency practises

cli, cloudformation, ....
discussaws.com

coding, on-site.

## Operational excellence pillar (operations as code)

Prepare: AWS config
Operate: Amazon cloudwatch
Evolve: Amazon elasticsearch

* AWS cloudformation is key service

## Security pillar

IAM: IAM, MFA Token
Detective: Cloudtrail, AWS Config, Cloudwatch
Inf. Protection: VPC, WAF
Data Protection: Macie, KMS, S3,
Incident response: IAM, Cloudformation

* AWS IAM is key service.

## Reliability pillar

* Cloudwatch is key service.

Foundations: IAM, VPC, Trusted Adv, Shield
Change management: CloudTrail, Config, Auto Scaling, Cloudwatch.
Failure management: C. Formation, S3, Glacier, KMS

## Reliability pillar

- Recover from failures, dynamically acquire computing resources, mitigate disruptions.
- Plan network topology.
- Manage service limits.
- Monitor behaviour of system
- Automate responses to demand
- Backup data regularly.

## Performance Efficiency

Use IT resources efficiently.

* Cloudwatch is key service.

Selection: EBS
Autoscaling
S3
RDS
DynamoDB
VPC
Route 53
Direct Connect

Review: Blog

Monitoring: Cloudwatch
Lambda

Tradeoffs: Cloudfront
Elasticache
Snowball
RDS

Selecting resources - Reference architecture
Quick start reference deployments
Benchmarking
Load testing
Cost / Budget
Monitoring & notification

Review: Load Testing.

Monitoring: Use cloudwatch to monitor and send notification alarms.
Use automation to work around performance issues by triggering actions through:
- cloudwatch, Kinesis, SQS, Lambda

Performance efficiency Trade off: (memory or storage) to reduce compute.

Proximity & Caching Trade off: Cloudfront, Elasticache RDS read replicas.

## Performance pillar

Use compute resources to meet system requirements
Select appropriate resource type
Benchmark.
Monitor performance
Optimise location of resources, data, processing.

## Cost optimisation

*Key service is Cost Allocation Tags.

Cost Eff Resources: Reserved EC2
Spot EC2
Cost Explorer

Matched Supply & Demand: Auto Scaling.
Expenditure: SNS, Cloudwatch
Optimising over time: Trusted Advisor, Blog.

Cost-effective resources: Cloudformation, beanstalk, ElasticSearch
Managed Services: RDS, EMR, DynamoDB, Elastic Cache

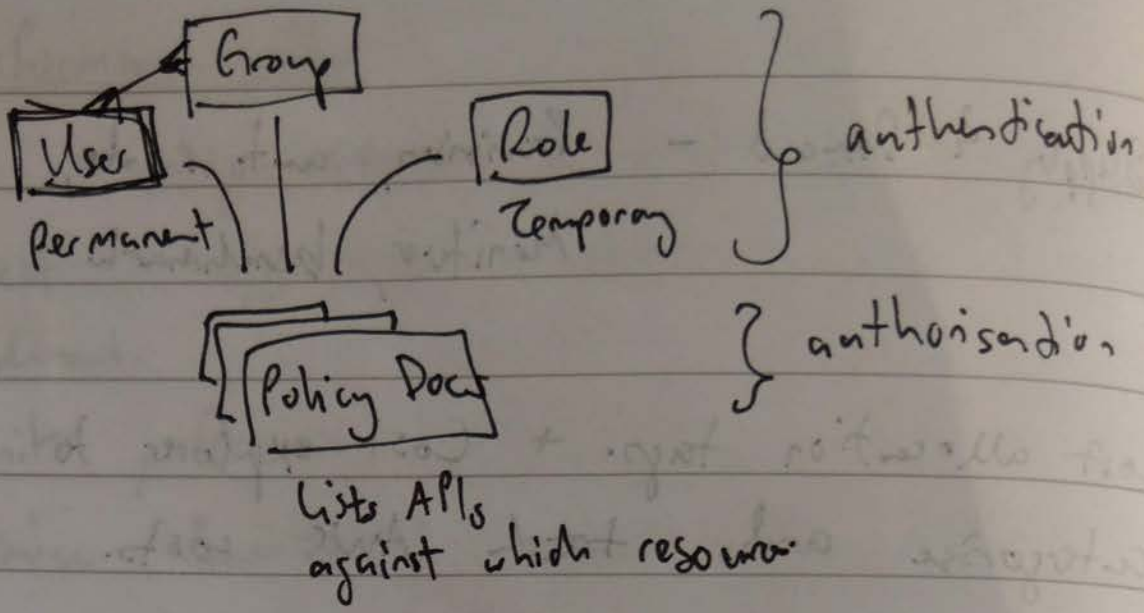
Supply & Demand - Provision, auto scale, Monitor, benchmark

Cost allocation tags. + Cost explorer to categorise and track AWS costs.

## Cost optimisation

Tag resources.
Add/Remove resources to match demand
RIs and reserved capacity to reduce cost
AWS trusted adviser.
Monitoring service.
Be aware of new services and feature.

## IAM




## DynamoDB

2 operations. Query and Scan.

Query - Find items in table using pri key attribute values.

Scan - Returns all data attributes for every item in the table or index.