**(important components)**

EC2-elastic compute cloud (VM)

ECS- ec2 container service

Lambda- Serverless

Provisioning: over API (automation) or web console

VPC: virtual private cloud

ACL: access control

IAM: identity management

S3: storage

EBS: elastic block storage

Cloudwatch: Monitoring AWS workload

Cloud Formation (with CLI): automate

**Support Plans**

<https://aws.amazon.com/tr/premiumsupport/plans/>

Developer:

**AWS\_POWERSHELL CONNECTOR**

<https://sdk-for-net.amazonwebservices.com/latest/AWSToolsAndSDKForNet.msi>

Module name: AWSPowerShell

**CONFIGURING AWS CLI (VERSİON2)**

[**https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-windows.html**](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-windows.html)

[**https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html**](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html)

aws configure (uses access keys of iam user)

MFA App ex. Google Authenticator

!! Set Password Policy

**IAM POLICY** DENY overrides ALLOW

5000 IAM account limit

IAM POLICY JSON SYNTAX

{

"Version": "xx",

"Statement": [

{

"Action": [

"xx:\*"

],

"Effect": "Allow",

"Resource": "xx"

}

]

}

Easier management: Prefer not to use inline iam policy for one user but create policy for all user

AWS Security Configuration Scanner: CloudSploit

ROOT USER:

* Billing
* Authorize (create IAM entities, permissions)
* Authenticate to portal

Eu-central1: region

Eu-central1a: availability zone

Eu-central1b: availability zone

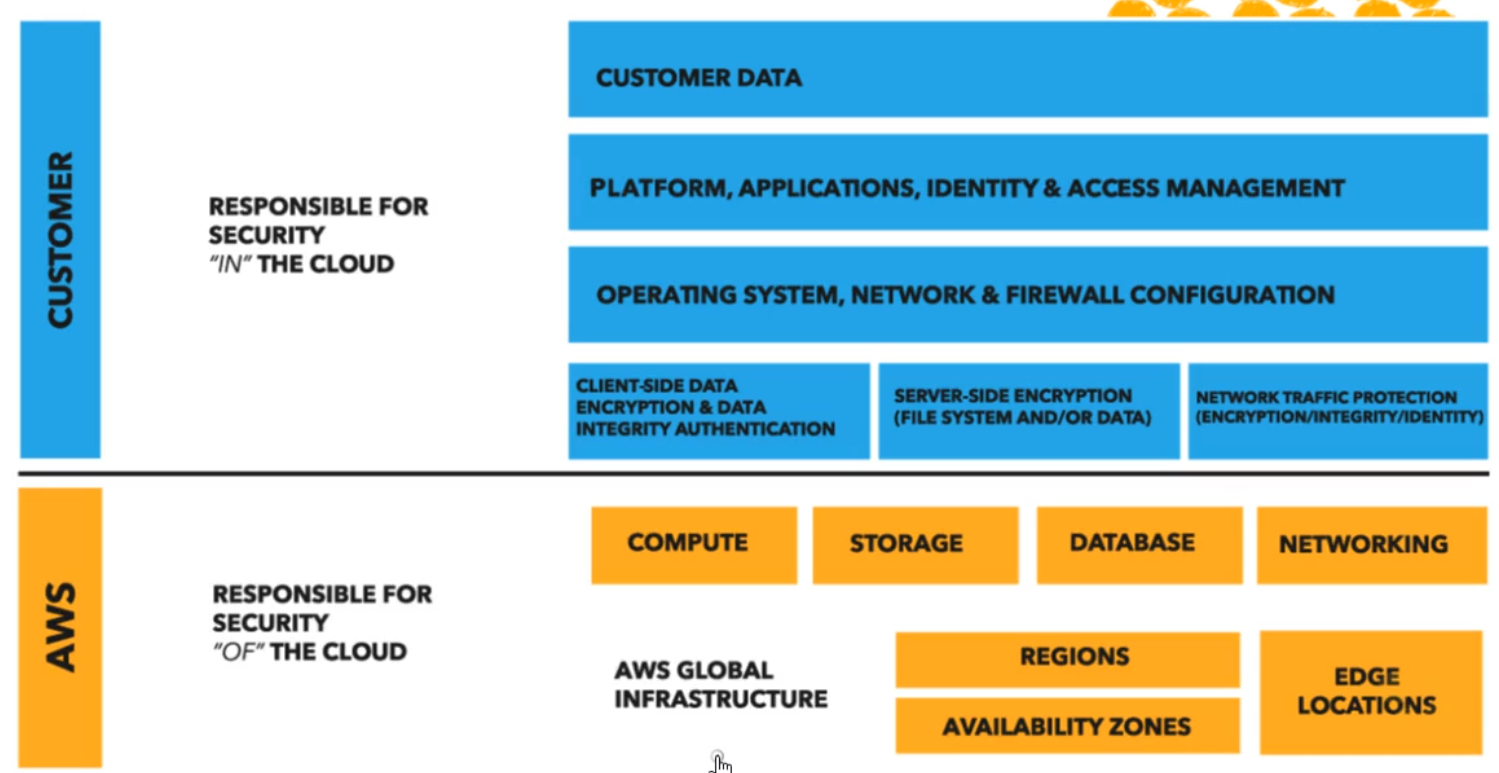
Each availability zone has own compute resources

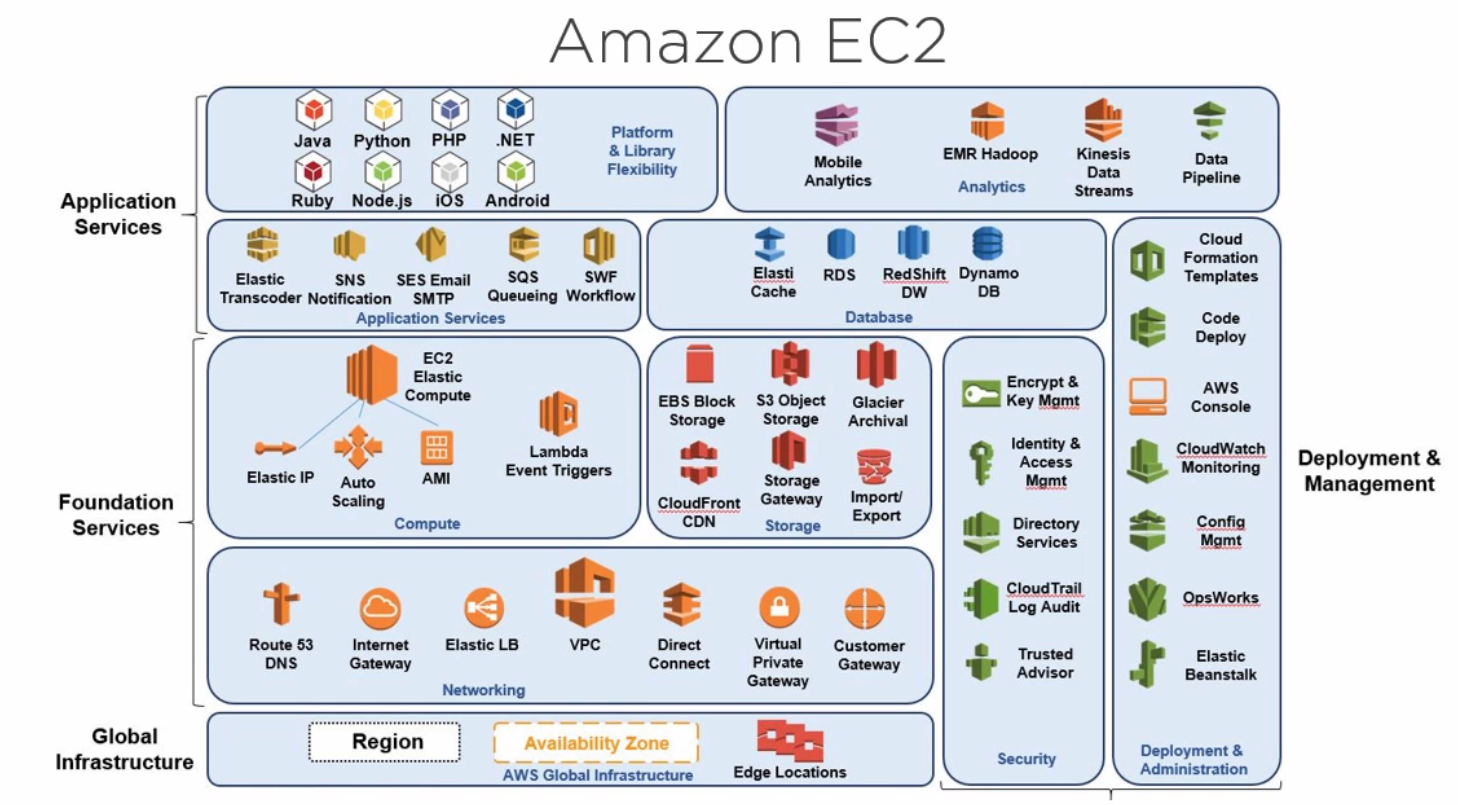
AWS CONFIG: AWS resources for which you want AWS Config to record configuration changes

It records status changes , e.x ec2 instance shutdown

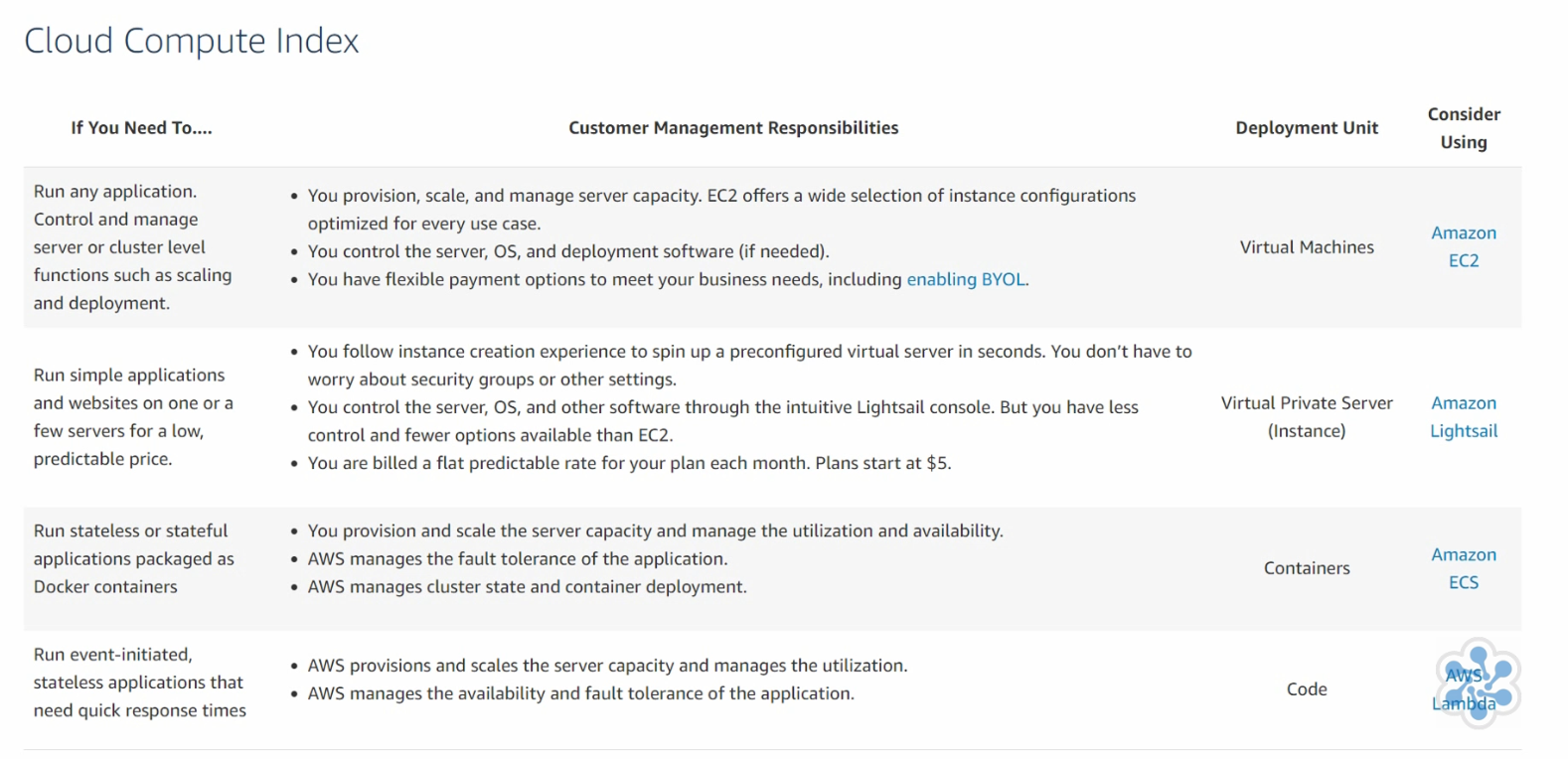
Usage: For compliance checks

**(SHARED RESPONSIBILITY MODEL)**





(COMPUTE PART)



EC2

* AMI (amazon machine image OS+ applications) : you can create own image

From vendors (paid), from aws, or open source images

* Instance types

Type: memory optimized, storage optimized…

https://www.amazonaws.cn/en/ec2/instance-types/

ECU: number of ec2 compute units

VCPU: virtual cpu

Processor type

Clock speed

Memory

Network performance: (according to data transfer rates)

IPV6 support

Processor architecture

* Instance purchasing options

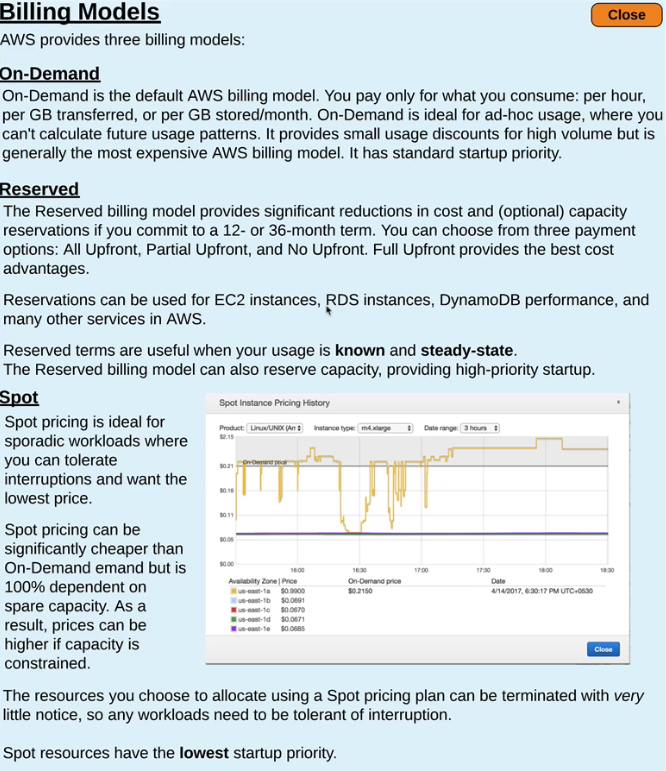
**On-demand** (can be launched at any time, use as long as needed, pay according to uptime, short term uses, use case: workload interruption possible, testing and development, if stop instance you will not pay for it)

**Reserved** (buy for a set of period of time, usage case: for long term, predictable workloads)

**Scheduled instances** (for recurring workloads)

**Spot instances** (usage case: background processing of data)

**On-demand capacity reservations**



* Tenancy

**Shared tenancy**: ec2 can be on any host, this host can be used by other customers

**Dedicated instances**: hosted on hardware that no other customer can access

**Dedicated hosts:** allows to use the same host for number of instances

* User data

Enter commands that will run during the first boot cycle of that instance

* Storage options

**Persistent storage** (by attaching EBS volumes (network attached storage)-isolated from instance)

Possible to disconnect

Possible to connect to other instance (but can’t be shared)

Backup snapshots, and encryption

Durable for power events (start, stop, restart)

Example: EBS, EFS

EBS types can be ssd, hdd and IOPS criteria

EBS snapshots are incremental, stores changes since the most recent snapshot

EBS snapshots are in S3 and region resilient

EFS provides shared storage

**Ephemeral storage** (temporary, attached to instance, provide high I/O than EBS persistent storage)

If instance is stopped, powered off or terminated, data will be lost

If instance is rebooted, data will be kept

Example: instance store volume, elastic cache

* Security

Security Group: instance level firewall (multiple security group can be linked to ec2 instance interface or single security group can be linked to ec2 instance interfaces)

Key pair: (download and secure .pem file)

One created private key can be used for other instances as well

Patching is your responsibility

* Status checks

System Status Check (AWS responsible: stop the instance and restart)

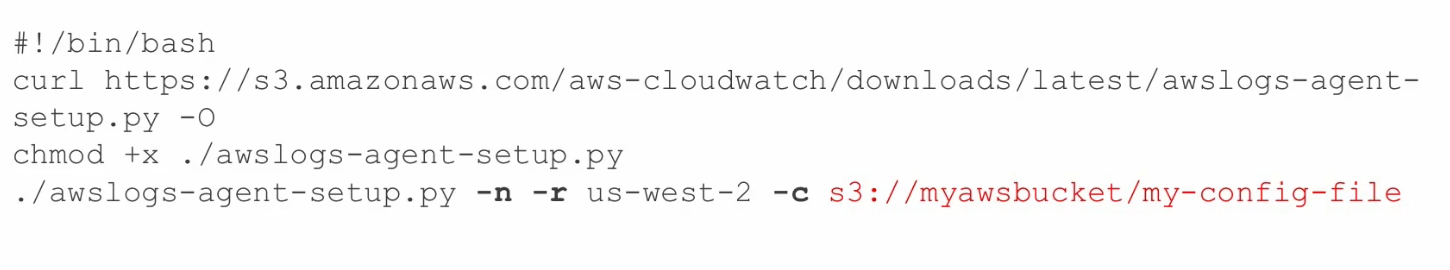
Instance Status Check (network failures)

* Installing cloudwatch agent.rpm on ec2 instance OR Aws System Manager

Configure agent config: /etc/aws/amazon-cloudwatch-agent/etc/

It includes which logs to collect and where

OR



* Create cloudwatch alarm according to your need
* Create backup strategy for ec2 instance
* TO EBS volume snapshots

CRON: aws ec2 create-snapshot –volume-id <value>

* For i/o intensive instances during operation = hot backups(data stored in memory are taken into account)

(OPTİON1) Fsfreeze for ext4 volumes (disable access)

(OPTION2) unmount volume, create snapshot, then remount volume

* HOW TO RESTORE FROM EBS SNAPSHOT

Launch new volume from snapshot and attach to ec2 instance and copy lost data

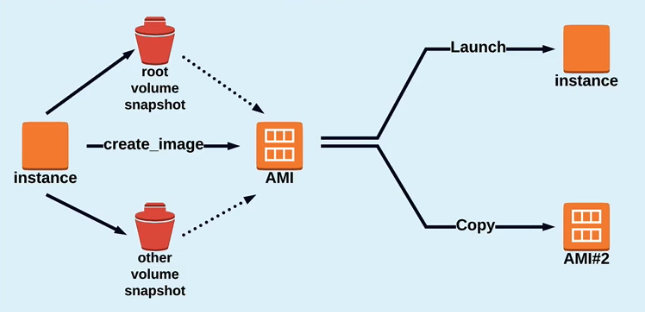
Running Amazon Linux on premise

<https://medium.com/shehuawwal/download-and-run-amazon-linux-2-ami-locally-on-your-virtualbox-or-vmware-b554a98dcb1c>

<https://cdn.amazonlinux.com/os-images/2.0.20200304.0/vmware/> (.ova file)

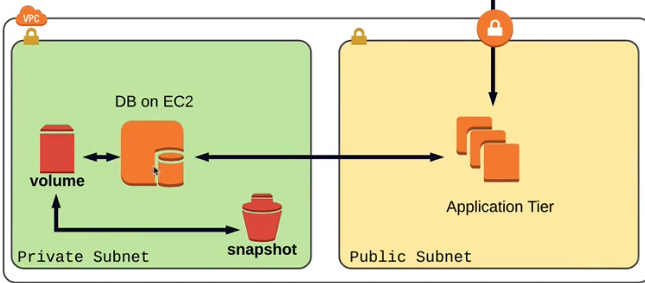
<https://cdn.amazonlinux.com/os-images/2.0.20190612/virtualbox/> (.vdi file)

**AMI usage**



**EC2-SELFMANAGED DATABASE**

* If you don’t find specific database service, create ec2 instance and install db manually but you should administer it!
* Backups can be done with ebs snapshots



**EC2 METRICS**

* **CPUUtilization** (percentage of allocated compute units that are currently in use on instance)
* **DiskReadOps**
* **DiskWriteOps**
* **DiskReadBytes**
* **DiskWriteBytes**
* **NetworkIN :** number of bytes received on all network instances of a single instance
* **NetworkOut:** number of bytes sent on all network instances of a single instance
* **NetworkPacketsIN, NetworkPacketsOUT**
* **StatusCheckFailed\_instance, StatusCheckFailed\_System**

**ECS dimensions**

* İmageid
* Instanceid
* İnstancetype

**VIRTUAL PRIVAE CLOUD**

!! before deleting VPC and subnets, terminate running instances and NAT device

**ROUTE53**

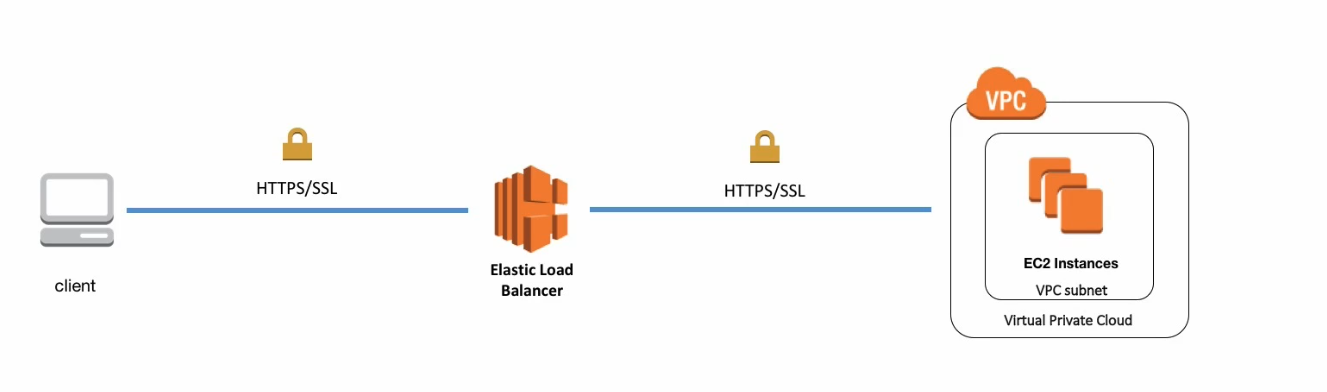
Public& Private hosted zone

NS,SOA pre-populated with AWS

Suggested: Create HealthCheck on route53 for ElasticLoadBalancers

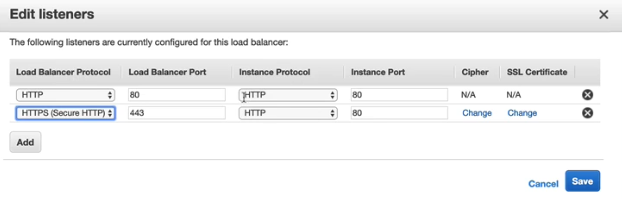
Set Routing type according to need: Simple,Weighted

**ELASTIC LOAD BALANCER**



* HARDWARE (standard, depreciated, round robin fashion forwarding, possible to set session stickiness, during user session connected server is remembered)

Route53 create friendly record for load balancer external IP



By this way, you do not need to deal with backend ssl sertificate work

Deploy SSL certificate to load balancer with ACM manager

* APPLICATION –layer7(rule based forwarding)
* NETWORK – layer4 (designed for performance )

Provides more resilience with auto scaling

Can have public or private interface

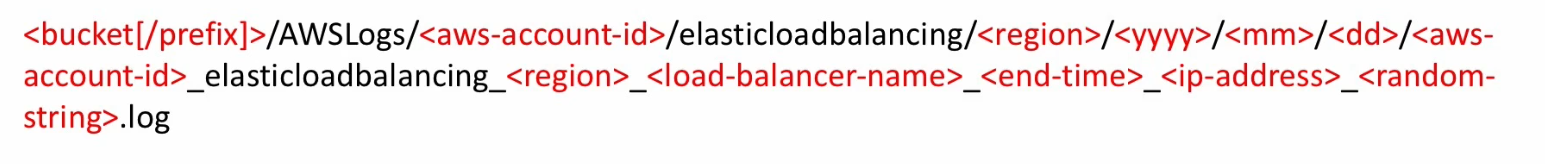
Load balancer metrics

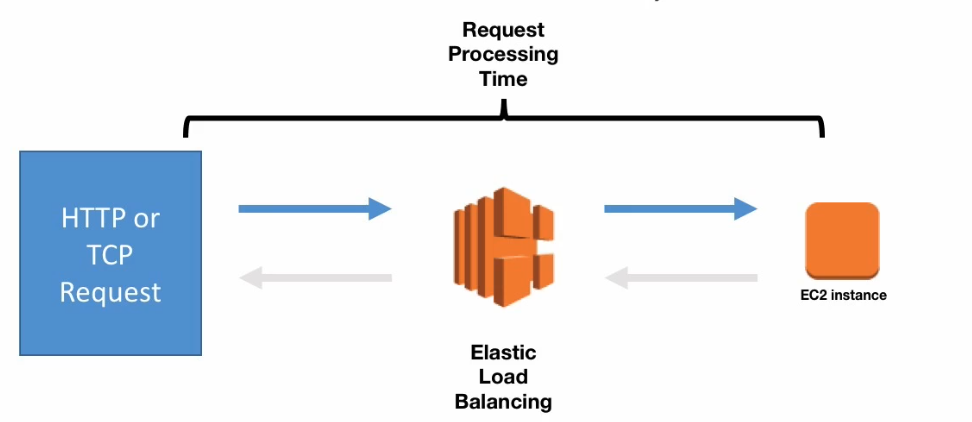
* **Backend connection errors** ( the number of connections that were not successful between load balancer and its instances
* **Healthy-unhealthy Host Count** (used to determine how many healthy and unhealthy instances are registered with the load balancer)
* **HTTP code backend**
* **HTTP code ELB\_4XX** (response code from load balancer)
* **HTTP code ELB\_5XX** (no healthy instances or if the request rate is more than instance)
* **Latency** (time elapsed after request leaves the load balancer until the headers of the response are received)
* **Request Count** (number of request completed or connections made during specified interval)
* **SurgeQueueLength** (number of requests that are pending routing)

Load balancer logs

Logs are turned off by default, enable it and log to S3

* Time of request
* The client’s IP address
* Request paths
* Server responses
* Latencies





**AWS SNAPSHOT**

Go to instance> find volume ID in elastic storage volume> Chose volume > Action: create snapshot

(possible to copy snapshot to other location)

(recover)

* Ec2 create volume
* Unmount
* Detach volume
* Attach volume
* Remount

! RDS (aws database system) has already automated backups

! Backup AMI files

! Possible to create AMI file from volume snapshot

! On Premise to AWS : s3 and glacier

**AUDITING: CLOUD TRAIL**

**TRUSTED ADVISOR**: SECURİTY, PERFORMANCE RECOMMENDATIONS

**IAM POLICY SIMULATOR**: check applicability of assigned user role

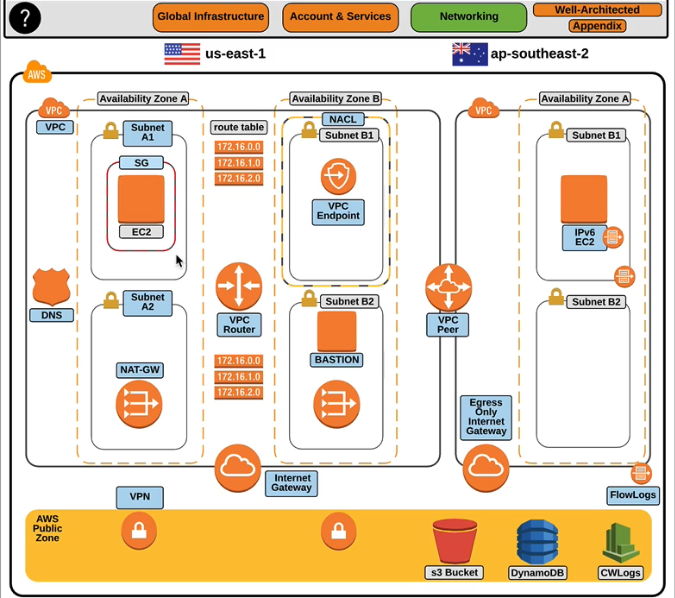
**CLOUDWATCH>Billing> Alerts**

**Free cloudwatch:** 5min interval cpu, network, disk, instance and system status

**Advanced cloudwatch: baseline-metrics**

**(SAS) AWS Directory Service(simple ad),AWS Microsoft AD**

**NETWORKING**



* Private: vpc
* AWS public zone(example:vpn,s3,cloudwatch logs)
* Internet

Largest possible IPv4 range in aws is when subnet mask is 16

Tenancy modes in VPC

Default: shared resources

Dedicated: dedicated host resources

**Networking configuration steps:**

1. Create vpc (name:vpc1) **CIDR:10.10.0.0/16 10.10.0.1 – 10.10.255.254**
2. Create subnet and connect to vpc1

**vpc1-sn-a1 CIDR:10.10.0.0/20 10.10.0.1 – 10.10.15.254 dns(R53 resolver)=10.10.0.2**

**vpc1-sn-b1 CIDR:10.10.16.0/20 10.10.16.1-10.1.0.31.254 dns(R53 resolver) =10.10.16.2**

**vpc1-sn-c1 CIDR:10.10.32.0/20 10.10.32.1-10.1.0.47.254 dns(R53 resolver) = 10.10.32.2**

(yellow ones are vpc router=default gw)

Create vpc (name:vpc2) **CIDR:10.11.0.0/16 10.11.0.1 – 10.11.255.254**

Create subnet and connect to vpc1

**Vpc2-sn-a1 CIDR:10.11.0.0/20 10.11.0.1 – 10.11.15.254**

1. One Network acls are assigned to every subnet on vpc (filtering traffic, subnet firewall)

Subnets can communicate thru assigned allow network acl (it should be two way!)

1. create internet gateway and attach it to vpc (internet gateway has dynamic address)
2. Define private and public subnets by creating new route tables

Create routing table name:

Public Subnet 0.0.0.0/0 >> internet gw

10.10.0.0/16>> local

Private Subnet 10.10.0.0/16>> local

and edit subnet associations (assign subnets to route tables)

for public subnets > auto assign IP address

1. Create Elastic IP(static), create NAT gateway using elastic IP and subnet IP

Assign nat gateway to route table of each subnet 0.0.0.0/0 >> natgatewayID

1. Create Private Hosted Zone, create A records of ec2 instances

VPC resolver inbound endpoints (direct dns queries coming from on premise to route53 resolver)

VPC resolver outbound endpoints (direct dns queries coming from vpc to onpremise dns)

1. VPC logs >> cloudwatch or s3 bucket

VPC settings >> create flow log

(subnets automatically send logs because of VPC inheritance)

Then it possible to track network logs from cloudwatch)

1. VPC gateway endpoint, aws services to connect to ec2 that doesn’t have internet access

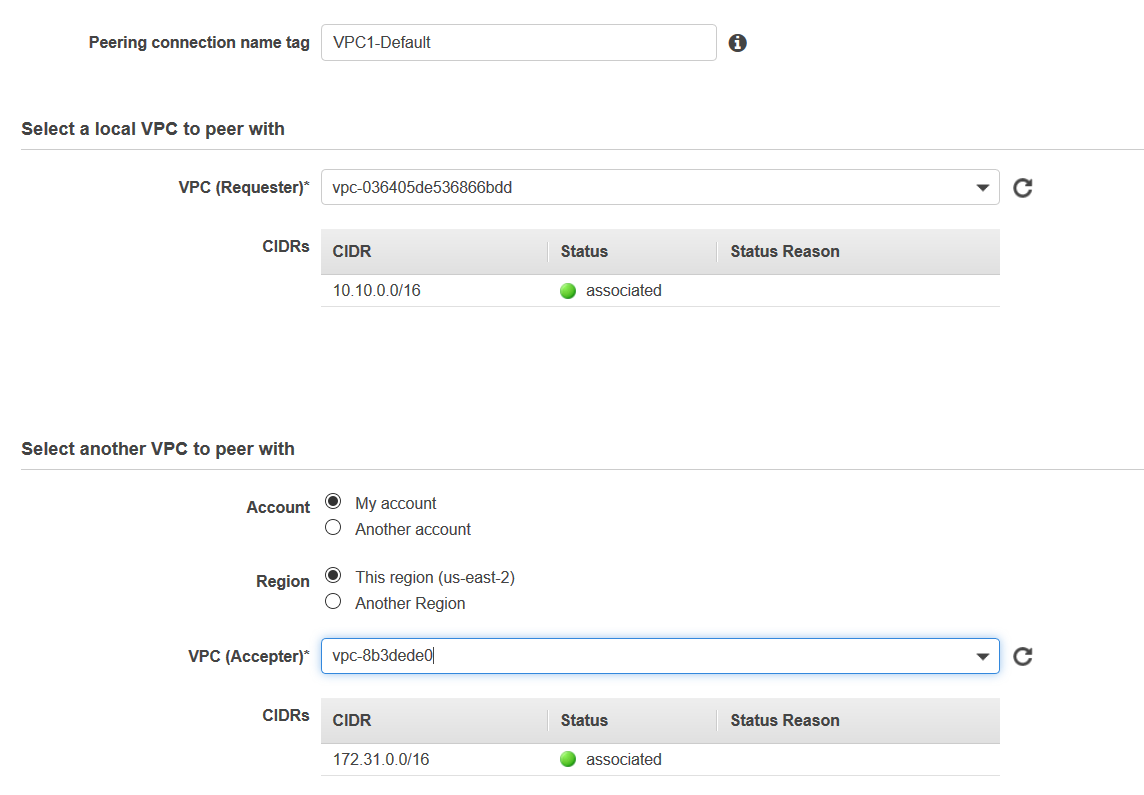
(for example s3 gateway)

It adds extra rule to routing tables designed for subnets

Means if there is traffic to s3 service from ec2 instance, use gateway endpoint

1. VPC Peering (should be done for both way)

VPC Dashboard: Peering connection



Add accept peering request

Enable dns resolution (to resolve public dns names between different VPCs)

*Add peering to Route table*

Destination (the vpc2 that needs to be accessed)

10.11.0.0/16

Target

(VPC peer id)

*Add vpc CIDRs in security group for inbound traffic*

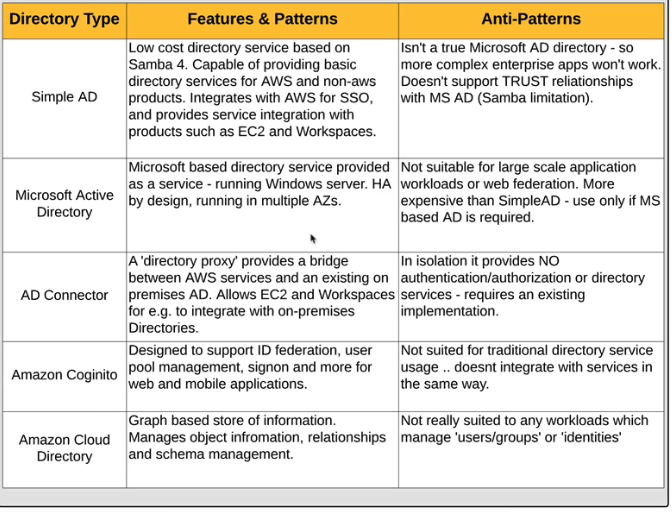
**SITE TO SITE VPN FOR ON-PREMISE TO CLOUD**

Cheap, performance depends on on premise firewall-cpu, charged according to data

Hardware based

* Add customer gateway (on premise Firewall-static external ip)
* Create Virtual Private Gateway and attach it to VPC
* Create Site to site VPN (aws HA multiple tunnels on aws side)
* Download configuration settings and use it on client side
* Add route table Destination network(on premise) Target(Virtual private gateway)

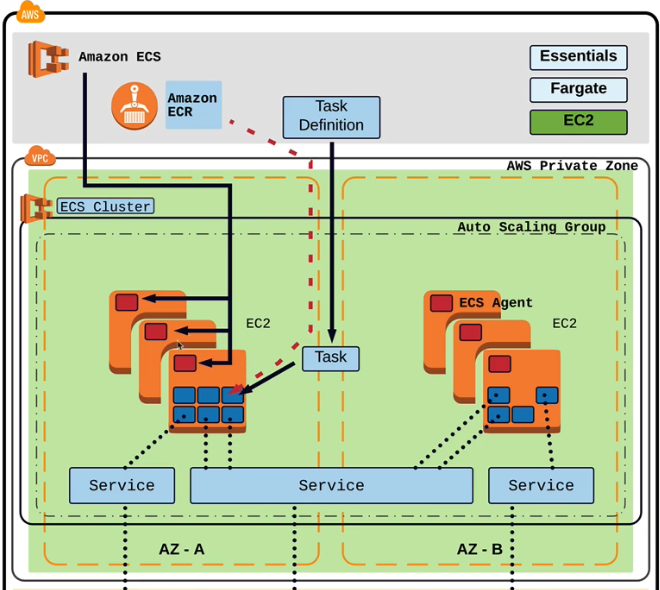
**DIRECTORY SERVICES**

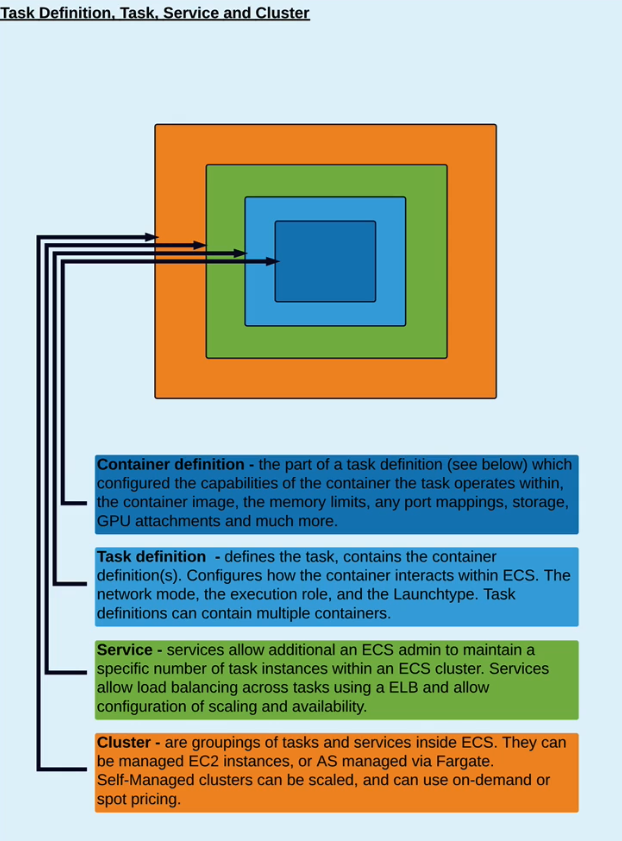


Ec2 instance creation: domain join directory

**ECS CLUSTER (Container)**

(standard)





Docker enable instances

Collection of ec2 instances

Dynamically scalable

Region specific

İnstances within cluster has ecs agent and docker daemon

AWS fargate: (you don’t need to do cluster management)

**Fargate Launch**: app packaging, cpu, ram, policies are your responsibility

**EC2 Launch**: patching, scaling, defining how many containers in cluster is your responsibility

**ELASTIC CONTAINER REGISTRY (ecr)**

Provides secure location to store and manage docker images

To push, pull and manage library of docker images in a central and secure location

**ELASTIC CONTAINER SERVICE FOR KUBERNETES (EKS)**

Provision and maintaining worker node is your responsibility

Kubernetes control plane(on master node) is aws responsibility

Worker node is on demand ec2 instance

**LAMBDA**

You give responsibility of server administration to AWS

Run application code without having to manage ec2 instances

Upload code to lambda

Configure lambda functions to execute upon specific triggers

Use compute power as only required

**AWS STORAGE**

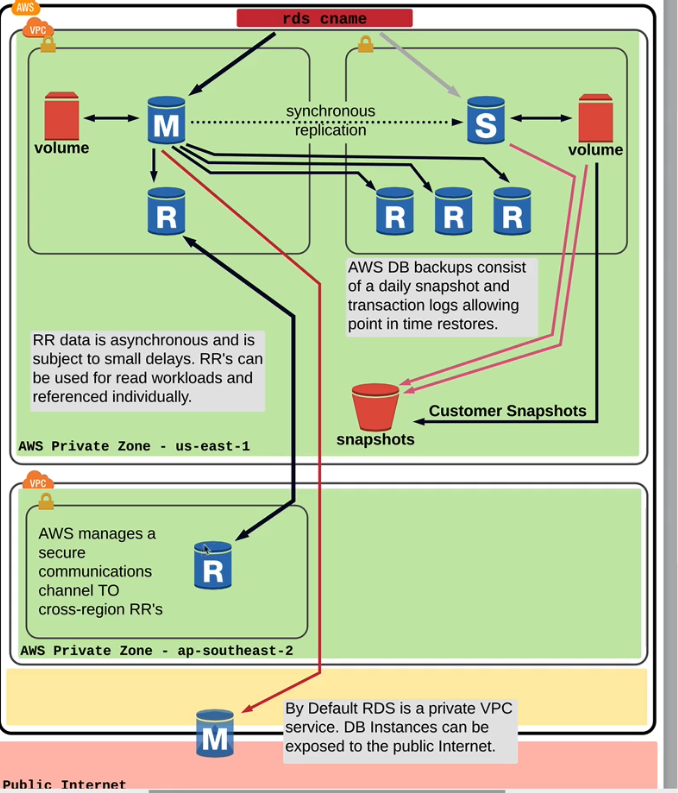
Per GB cost, upload, transfer, restore requests

**RDS**

M: master

R: read replica

S: slave (not accessible)

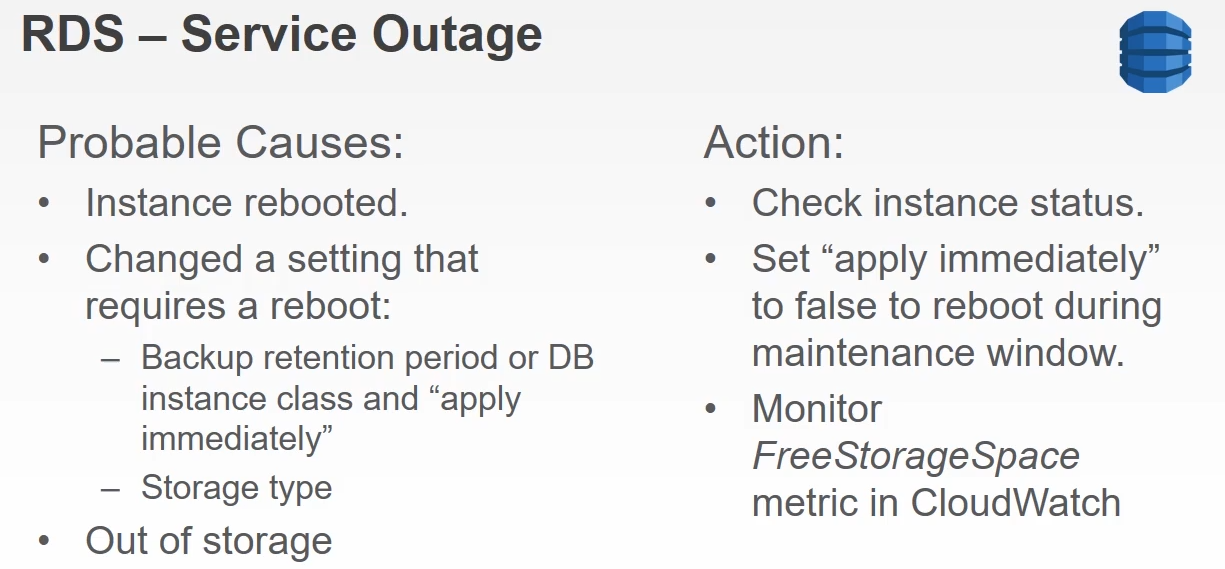


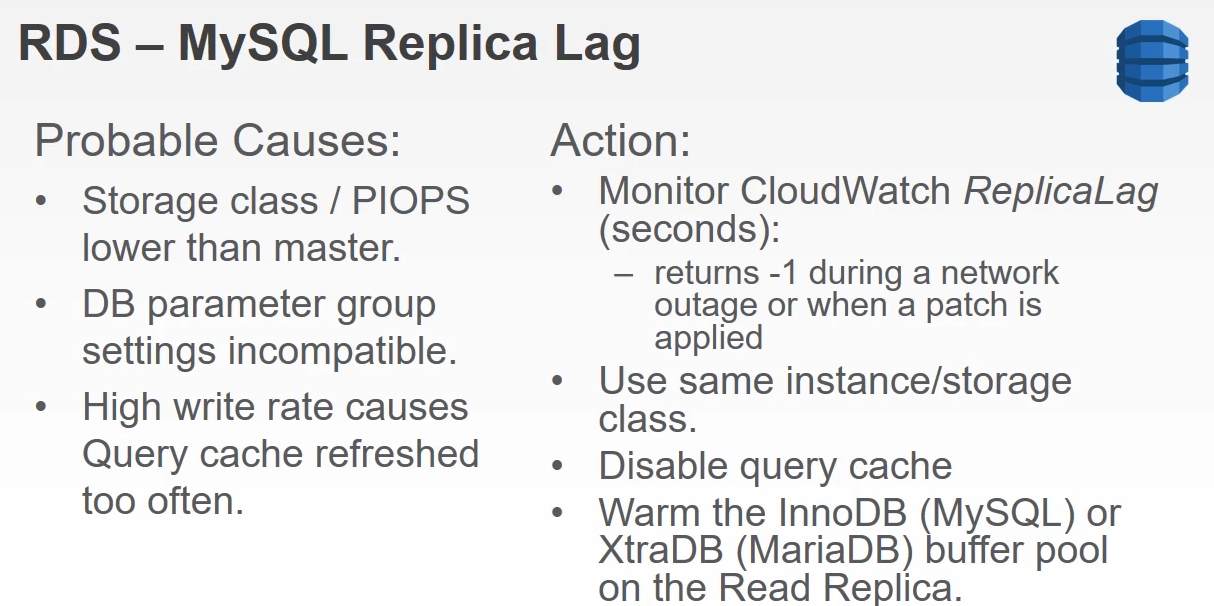
Create read replica after db creation (check replica leg seconds!)

Query cache size >> replica leg

Maintanence window: patches, changing db instance times

Modify parameter group according to your needs





Monitoring

RDS enhanced monitoring: real time os metrics (not available for t1.micro m1.small)

++ performance insight (for aurora aws)

**S3-OBJECT STORAGE** (also called bucket)

NAS, CIFS, NFS – unstructured data

Interface app: cloudberry explorer

Automatically backed up, no need extra DR solution (replication to other region)

Static object storage (file shares, backup/archive, host files, content, static web hosting)

S3 is region less, global product

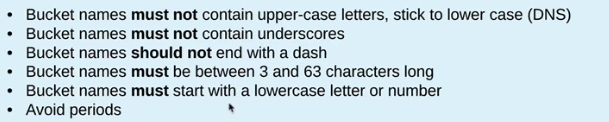
Encryption of content(object) is possible

Uploading (single path-upto5GB or multipath-parallel-upto5TB) enabled per bucket

Transfer acceleration:

Cross site replication is possible (after enabling this and set region, uploaded files automatically transferred) It is one way

Bucket naming requirements



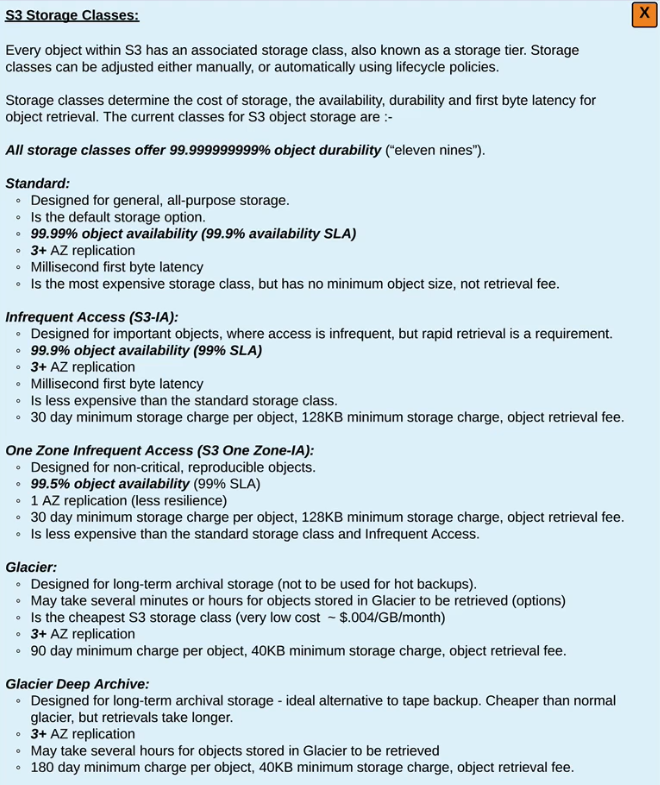
Security: Permissions Types

* Bucket policy

One policy per bucket possible

* ACL for files

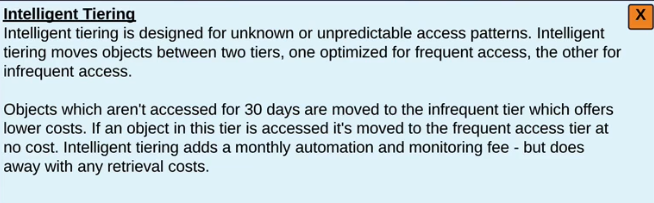
Events: Receive notifications when specific events occur in your bucket



Storage class can be changed per file basis

**Billing improvements**

1. **Intelligent Tiering**



1. **S3 Object Lifecycle Management Policies**

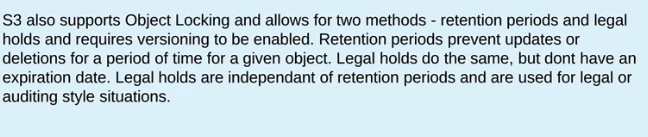
! DESIGNED FOR BUCKET LEVEL POLICIES

**Versioning**

It is defined on bucket level.

Files will have identifiers (delete markers to delete file)

Object Locking



**EBS-BLOCK STORAGE**

SAN, ISCSI – EC2 instance persistent storage

(can be used when to create new filesystem on linux)

**EFS**- file system mount for linux ec2 instances (shared file system so that other instances can connect)

Possible to attach it to instances in same availability zone

Mount targets created in aws File System ID

Protocol: NFS

Client side:amazon-efs-utils

Mount –t efs File System ID:/ /mountdir

(possible to use on premise data on cloud)

**G**- Long term cold storage, for tape archieve

**DOCKER DEPLOYMENT WITH ELASTIC BEANSTALK**

* Prepare docker file
* Build docker file and check image is built(docker images)
* Install elastic beanstalk

<https://github.com/aws/aws-elastic-beanstalk-cli-setup>

* eb init (continue with selection steps)
* eb create “environment name”
* Go to Elastic Beanstalk dashboard in aws console and check

OPSWORK- instaces,layers..