**Content**

[1. Questions 3](#_Toc16974733)

[1. iSCSI 3](#_Toc16974734)

[2. EMR 4](#_Toc16974735)

[2. FAQs 5](#_Toc16974736)

[3. Intro – General - IAM 6](#_Toc16974737)

[1. Well Architected Framework 6](#_Toc16974738)

[2. Edge Locations 6](#_Toc16974739)

[3. STS and Temporary Security Credentials 8](#_Toc16974740)

[4. Identity Federation 9](#_Toc16974741)

[3.4.1. Custom Identity Provider 9](#_Toc16974742)

[3.4.2. LDAB / Active Directory 9](#_Toc16974743)

[3.4.3. Web Identity 10](#_Toc16974744)

[5. Organizations 11](#_Toc16974745)

[4. EC2 12](#_Toc16974746)

[1. Tenancy 12](#_Toc16974747)

[2. Virtualization 13](#_Toc16974748)

[HVM 14](#_Toc16974749)

[Paravirtual (PV) 14](#_Toc16974750)

[4.2.1. Hypvervisors 14](#_Toc16974751)

[XEN 14](#_Toc16974752)

[KVM 15](#_Toc16974753)

[3. Placement Groups 16](#_Toc16974754)

[4. Purchasing Options 17](#_Toc16974755)

[4.4.1. On Demand 17](#_Toc16974756)

[4.4.2. Reserved 17](#_Toc16974757)

[4.4.3. Scheduled 17](#_Toc16974758)

[4.4.4. SPOT 18](#_Toc16974759)

[5. EC2 Storage Options 19](#_Toc16974760)

[4.5.1. EBS 19](#_Toc16974761)

[6. Instance store 21](#_Toc16974762)

[4.6.1. Object storage vs Block storage – general 21](#_Toc16974763)

[7. EFS 22](#_Toc16974764)

[5. Autoscaling and Elastic Load Balancing 23](#_Toc16974765)

[1. Load Balancing 23](#_Toc16974766)

[5.1.1. Classic ELB 23](#_Toc16974767)

[5.1.2. Application (Layer 7 ELB) 23](#_Toc16974768)

[5.1.3. Difference between Classic and Application ELB 24](#_Toc16974769)

[5.1.4. Network Load Balancer 24](#_Toc16974770)

[5.1.5. Certificates and protocols 25](#_Toc16974771)

[5.1.6. Round Robin and Sicky – Load Balancing methods 25](#_Toc16974772)

[5.1.7. NAT Gateway 25](#_Toc16974773)

[5.1.8. VPC endpoints 26](#_Toc16974774)

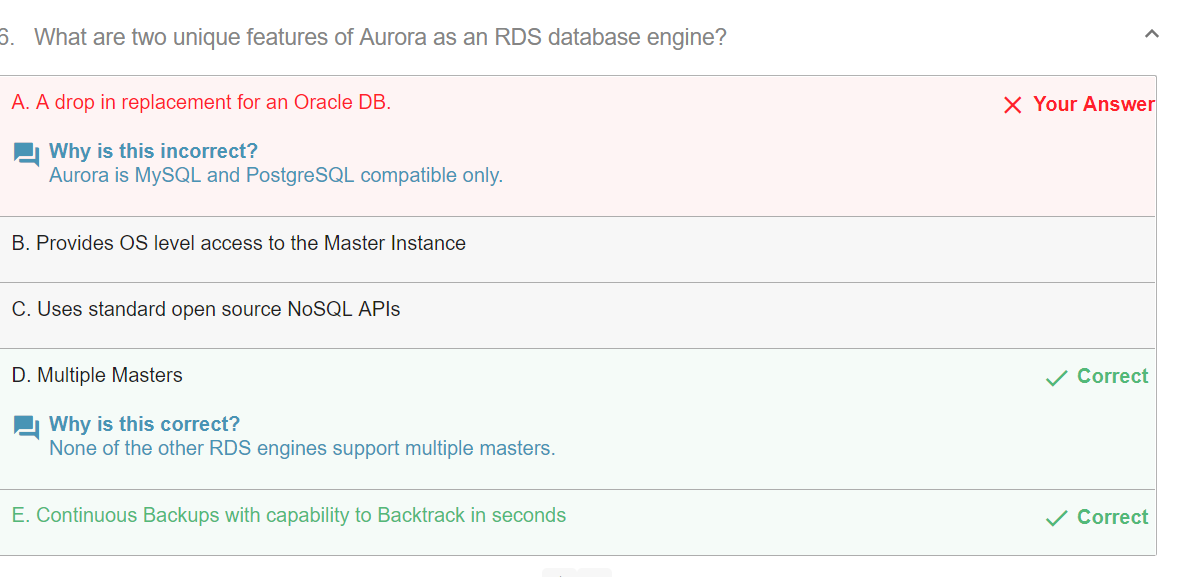
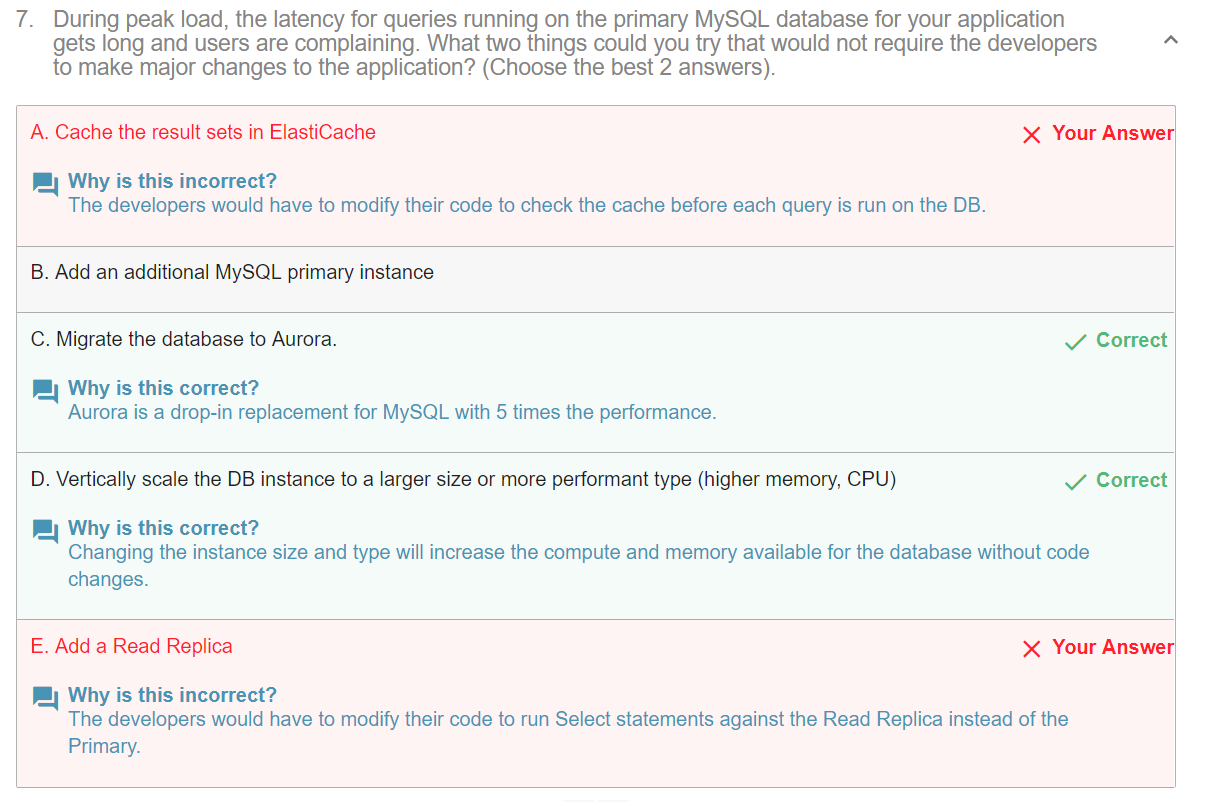
[5.1.9. Autoscaling 26](#_Toc16974775)

[5.1.10. Stateful vs Stateless applications 27](#_Toc16974776)

[6. Databases 28](#_Toc16974777)

[1. SQL Databases 28](#_Toc16974778)

[2. NOSQL 28](#_Toc16974779)

1. Questions
2. Incorrect. EBS is block storage and can only be attached to a single instance.
3. EMR is designed for processing of big data/large unstructured data sets. It's not a good fit for long term data warehousing.
4. Redshift: AWS product which will allow storage of that data to be used for long term reporting, querying, forecasting, and business intelligence.
5. Natively Highly Available services: Internet Gateway, Virtual Private Gateway, Dynamic Hardware VPC VPN, DynamoDB, S3
6. An IGW is resilient by design, and only one needs to be attached to a VPC in order to provide all subnets in all AZ's with resilient internet connectivity.
7. How to mask a failure of an instance? An Elastic IP can be remapped from a failed instance to a replacement instance. That doesn't require any application or DNS changes.
8. It is not possible to encrypt an existing EBS volume. You can take a snapshot of the unencrypted volume. Once the snapshot is taken, copy the snapshot and enable encryption on the copy so that the target snapshot is encrypted. Once the target snapshot is created, you can attach a new encrypted volume to the EC2 instance, and restore the encrypted snapshot to a new volume.
9. AWS Shield offers managed DDOS protection
10. Placement groups are a clustering of EC2 instances in one Availability Zone with fast (up to 25Gbps) connections between them. This feature is used for applications that need extremely low-latency connections between instances.
11. IOPS provisioned instances does not have an impact on the degree of latency between instances.
12. Amazon EFS now allows you to instantly provision the throughput required for your applications independent of the amount of data stored in your file system. This allows you to optimize throughput for your application’s performance needs.
13. EFS mount only for Linux instances, not for windows.
14. A read capacity unit represents one strongly consistent read per second, or two eventually consistent reads per second, for an item up to 4 KB in size.
15. 
16. 
    1. iSCSI

In computing, iSCSI is an acronym for Internet Small Computer Systems Interface, an Internet Protocol (IP)-based storage networking standard for linking data storage facilities. It provides block-level access to storage devices by carrying SCSI commands over a TCP/IP network. iSCSI is used to facilitate data transfers over intranets and to manage storage over long distances. It can be used to transmit data over local area networks (LANs), wide area networks (WANs), or the Internet and can enable location-independent data storage and retrieval.

* 1. EMR

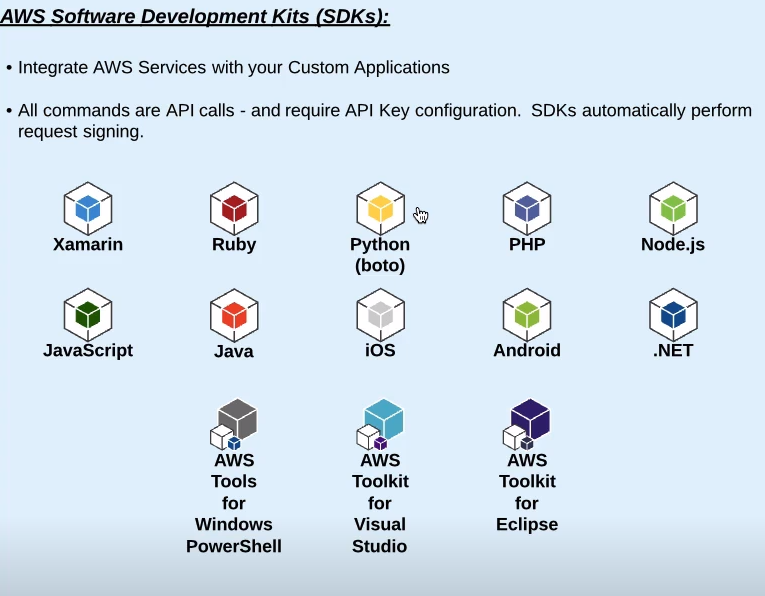
Amazon EMR is the industry leading cloud-native big data platform, allowing teams to process vast amounts of data quickly, and cost-effectively at scale. Using open source tools such as Apache Spark, Apache Hive, Apache HBase, Apache Flink, and Presto, coupled with the dynamic scalability of Amazon EC2 and scalable storage of Amazon S3, EMR gives analytical teams the engines and elasticity to run Petabyte-scale analysis for a fraction of the cost of traditional on-premise clusters. Developers and analysts can use Jupyter-based EMR Notebooks for iterative development, collaboration, and access to data stored across AWS data products such as Amazon S3, Amazon DynamoDB, and Amazon Redshift to reduce time to insight and quickly operationalize analytics.

1. FAQs

Please read below mentioned list of FAQs: From Compute: - EC2 - Auto Scaling - Lamdba - Elastic Load Balancing Storage: - S3 - EBS - EFS - Glacier - Storage gateway Database: - RDS - Aurora - DynamoDB -RedShift - ElastiCache Networking and CDN: - VPC - CloudFront - Route53 - Direct Connect. Management Tools - CloudWatch - CloudTrail Media Services and Analytics - Kinesis - EMR Security, Identity & Compliance - IAM - Key Management Service Application Integration. - MQ - SQS - SNS - SWF - Steps Link of all FAQs https://aws.amazon.com/faqs/ null

1. Intro – General - IAM

WE interact w services through service API. Service API calls are made to service API endpoints.



* 1. Well Architected Framework

Best practices and recomendations.

* Operational Excellence
* Reliability
* Security
* Performance Efficiency
* Cost Optimization
  1. Edge Locations

AWS datacenters (points of presence) located around the world, designed to give low latency access to 2 AWS services:

* Route 53 – DNS Lookups
* CloudFront – CDN, Cached content, streaming distributions, acceleration.

Services that run in CloudFront: security / accelerate other services.

1. Shield
2. WAF – web application framework
3. Lambda@Edge
4. S3 Transfer
5. API Gateway

**Point of Presence** is the point at which two or more different networks or communication devices build a connection with each other.

**Edge Locations vs Regional Edge Caches**

The nine new Regional Edge Cache locations are in Northern Virginia, Oregon, São Paulo, Frankfurt, Singapore, Seoul, Tokyo, Mumbai, and Sydney. These locations sit between your origin webserver and the 68 global edge locations that serve traffic directly to your viewers. As the popularity of your objects reduce, individual edge locations may evict those objects to make room for more popular content. Regional Edge Caches have larger cache-width than any individual edge location, so your objects remain in cache longer at these locations. This helps keep more of your content closer to your viewers, reducing the need for CloudFront to go back to your origin webserver, and improving overall performance for viewers. For instance, our edge locations in Europe now go to the regional edge cache in Frankfurt to fetch an object before going back to your origin webserver.

To deliver content to end users with lower latency, Amazon CloudFront uses a global network of 187 Points of Presence (176 Edge Locations and 11 Regional Edge Caches)

**Vertical scaling** – increase the capacity of a single instance or server

**Horizontal scaling** – add or terminate the number of instances

**Overprovisioning** – when deploying counting w peak loads + some buffer. Not cost efficient.

Root account is logging in with email address.

Resource Groups – a collection of resources that you want to identify to being part of a group. You can use it in conjunction with Tag Editor. Group together resources and manage the tags.

Power user access – Admin access except it does not allow user/group management.

Inline policy – when a user gets a special rights.

* 1. STS and Temporary Security Credentials

You can use the AWS Security Token Service (AWS STS) to create and provide trusted users with temporary security credentials that can control access to your AWS resources. Temporary security credentials work almost identically to the long-term access key credentials that your IAM users can use, with the following differences:

* Temporary security credentials are short-term, as the name implies. They can be configured to last for anywhere from a few minutes to several hours. After the credentials expire, AWS no longer recognizes them or allows any kind of access from API requests made with them.
* Temporary security credentials are not stored with the user but are generated dynamically and provided to the user when requested. When (or even before) the temporary security credentials expire, the user can request new credentials, as long as the user requesting them still has permissions to do so.

These differences lead to the following advantages for using temporary credentials:

* You do not have to distribute or embed long-term AWS security credentials with an application.
* You can provide access to your AWS resources to users without having to define an AWS identity for them. Temporary credentials are the basis for roles and identity federation.
* The temporary security credentials have a limited lifetime, so you do not have to rotate them or explicitly revoke them when they're no longer needed. After temporary security credentials expire, they cannot be reused. You can specify how long the credentials are valid, up to a maximum limit.

STS allows you to create temporary security credentials that grant users access to your AWS resources. These temp creds are for short-term use, with configurable session duration between 15 min and 12h / 36h

When requested thorugh an STS API call, a credential object is returned containing:

* Session Token
* Access Key ID
* Secret Access Key
* Expiration Timestamp

**API keys are needed when working "programmatically" through the CLI, PowerShell, Direct HTTP calls, and SDK API access.**

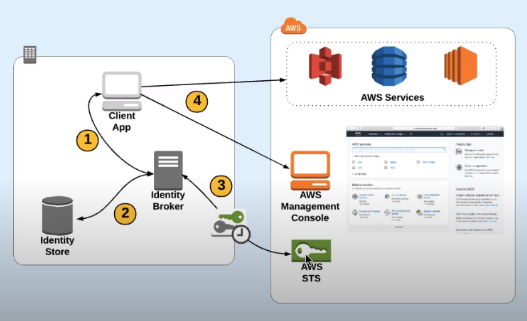
Benefit: avoid distributing or embedding long-term AWS security credentials in an application.

* 1. Identity Federation

Authenticate users using an Identity Broker Application running outside of AWS.

* + 1. Custom Identity Provider

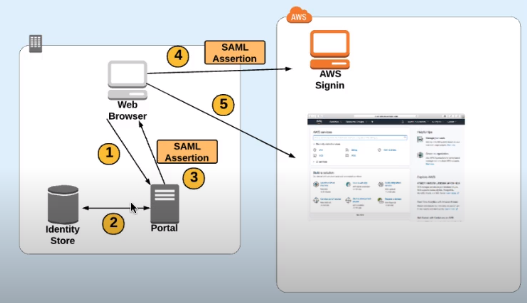
Authenticate against the identity broker app then it checks if you provided valid creds. Requests for you an STS temp cred (you are gonna be write a little bit of code at step 3 to request the credentials and forward them) and you are ready to go.



* + 1. LDAB / Active Directory

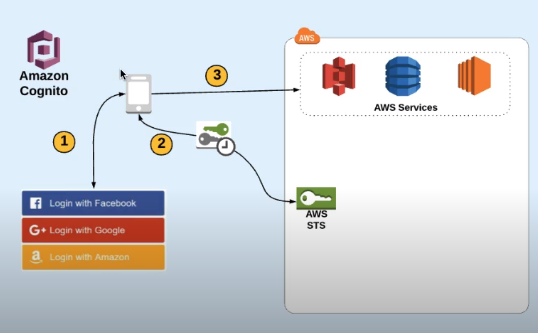
SAML – Security Assertion Markup Language – an open standard for exchanging authentication and authorization data between 2 different parties, such as your active directory and AWS.

In the console you choose identity provider. A user browses to a webpage, where he type in its active directory credentials, they got authenticated. Then the SAML compatible application, like Active Directory generates, what’s called a SAML assertion. Sends back to browser, than it post to a special sign in URL, which validates the assertion. Then automatically sends back a response which redirects the user to the console. The permission that the user will have is defined by a role, which is pre-associated with the users active directory group. That’s how to do SSO



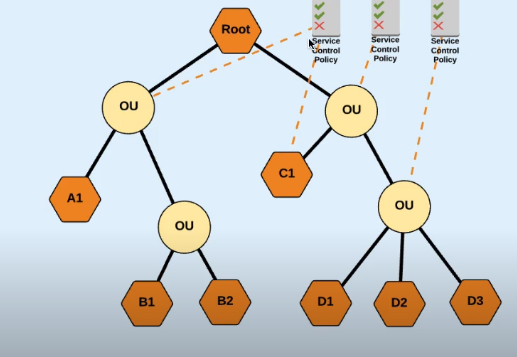
* + 1. Web Identity

You want to use them in web-based / mobile applications. When you have a mobile application, instead of having every user of you app needing a set of AWS credentials, you allow the users to log in with what’s called an **Open ID connect provider like Facebook / Google / Amazon.** You can also create your own as long as its **open id connect compatible**. Once the user is authenticated against the choosen platform the app requests credentials. Here helps Amazon Coginito – takes care of the part of exchangeing your web identity authorization token for STS credentials



* 1. Organizations

To enforce what accounts what are allowed to do. Which services? Regions? How much money they are able to spend? Allows you to create a root account then invite other accounts to join to the organization beneath that root account. You can group together accounts in OUs and then apply SCPs. You can’t create any IAM policies that are in violation with SCPs. Consolidated Billing – all of the accounts payments will roll up to the root account.



1. EC2

An AMI includes the following:

* A **template for the root volume** for the instance for e.g. an operating system, an application server, and applications
* **Launch permissions** that control which AWS accounts can use the AMI to launch instances for e.g. AWS account ids with whom the AMI is shared
* **A block device mapping** that specifies the volumes to attach to the instance when it’s launched

**AMIs are specific to a region and if needed in other region must be copied over.** (Actions, copy AMI, specify region… )

Launch permissions define who has access to the AMI

* Public – Accessible to all AWS accounts
* Explicit – Shared with specific AWS accounts
* Private – Owned and available for AMI creator AWS account only

To create an EC2 instance you are making a call to service API endpoint. So you are running a software application to launch a machine.

Automated bootstrapping launch commands can be passed to the instance via user-data scripts.

* 1. Tenancy

AWS provides a few options for tenancy including dedicated or the default type of shared. These models work in a very similar fashion to the housing example above.

**Shared tenancy** means that multiple EC2 instances from different customers may reside on the same piece of physical hardware.

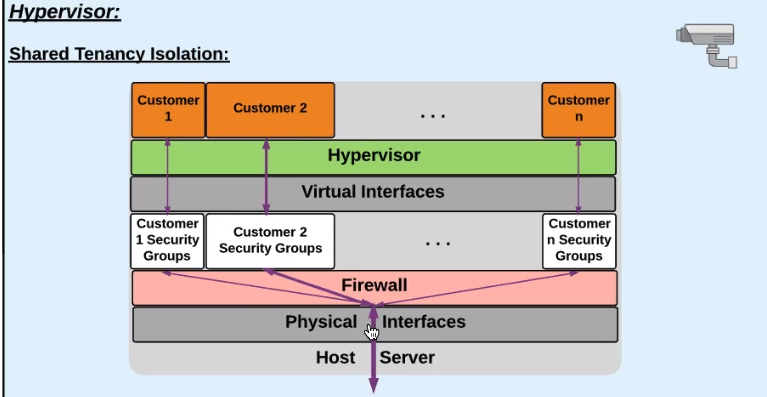
The **dedicated model** means that your EC2 instances will only run on hardware with other instances that you’ve deployed, no other customers will use the same piece of hardware as you. In some cases due to licensing restrictions some software isn’t allowed to be run on a shared tenancy model. For instance if you’re trying to use Bring Your Own License (BYOL) to AWS, some licenses are based on the Socket model where the number of hosts sockets are used for licensing. In other circumstances, regulatory compliance may dictate that you can’t use the shared model. HIPAA up until earlier this year required dedicated tenancy to ensure data confidentiality. This restriction has since been removed.

There are two different options for dedicated tenancy with AWS:

1. **Dedicated Hosts**
2. **Dedicated Instances.**

With a **dedicated host, you purchase an entire physical host from AWS** and that host is billed to you on an hourly basis just like EC2 instances are billed. Once you’ve purchased that host, you’re allowed to spin up as many EC2 instances as that host will allow for no additional charges

With a dedicated instance, you’re **still receiving the benefits of having separated hosts** from the rest of the AWS customers but you are not paying for the entire host all at once. You do not need to worry about the capacity of the hosts but **you’re being charged a higher rate for the instances.** This type of instance model is similar to the default model where you don’t worry about where the instances are, but it does ensure they’re kept separate. In addition to the higher rate that you’re charged for dedicated instances, you’re also charged a $2 per hour charge per region where dedicated instances are being used.



* 1. Virtualization

*How the instance is emulating a bare hardware machine.*

Linux Amazon Machine Images use one of two types of virtualization:

1. **paravirtual (PV)**
2. **hardware virtual machine (HVM)**

The main differences between PV and HVM AMIs are the way in which they boot and whether they can take advantage of special hardware extensions (CPU, network, and storage) for better performance. It is recommended that you use current generation instance types and HVM AMIs when you launch your instances for the best performance.

HVM

HVM AMIs are presented with a fully virtualized set of hardware and boot by executing the master boot record of the root block device of your image. This virtualization type provides the ability to run an operating system directly on top of a virtual machine without any modification, as if it were run on the bare-metal hardware. The Amazon EC2 host system emulates some or all of the underlying hardware that is presented to the guest.

Runs more like bare metal hardware, there is no modifications necessary. With it you can take advantage of hardware extensions

ENA – enhanced networking adaptor

Paravirtual (PV)

PV AMIs boot with a special boot loader called PV-GRUB, which starts the boot cycle and then chain loads the kernel specified in the menu.lst file on your image. Paravirtual guests can run on host hardware that does not have explicit support for virtualization, but they cannot take advantage of special hardware extensions such as enhanced networking or GPU processing. Historically, PV guests had better performance than HVM guests in many cases, but because of enhancements in HVM virtualization and the availability of PV drivers for HVM AMIs, this is no longer true.

Paravirtual guests traditionally performed better with storage and network operations than HVM guests because they could leverage special drivers for I/O that avoided the overhead of emulating network and disk hardware, whereas HVM guests had to translate these instructions to emulated hardware. Now PV drivers are available for HVM guests, so operating systems that cannot be ported to run in a paravirtualized environment can still see performance advantages in storage and network I/O by using them. With these PV on HVM drivers, HVM guests can get the same, or better, performance than paravirtual guests.

* + 1. Hypvervisors

XEN

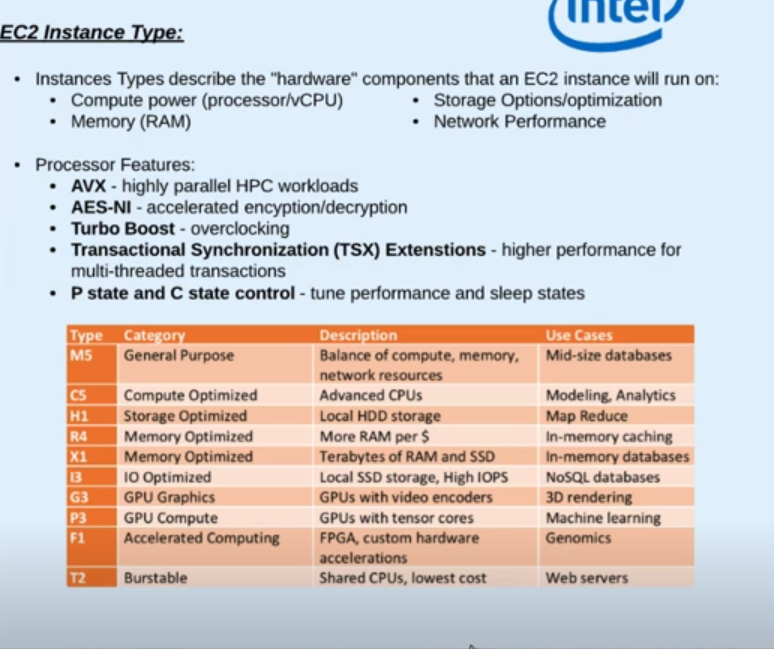
Xen Project (pronounced /ˈzɛn/) is a type-1 hypervisor, providing services that allow multiple computer operating systems to execute on the same computer hardware concurrently. It was developed by the University of Cambridge and is now being developed by the Linux Foundation with support from Intel.

KVM

**Intel XEON**

Xeon is a brand of x86 microprocessors designed, manufactured, and marketed by Intel, targeted at the non-consumer workstation, server, and embedded system markets. It was introduced in June 1998. Xeon processors are based on the same architecture as regular desktop-grade CPUs, but have some advanced features such as support for ECC memory, higher core counts, support for larger amounts of RAM, larger cache memory and extra provision for enterprise-grade reliability, availability and serviceability features responsible for handling hardware exceptions through the Machine Check Architecture.

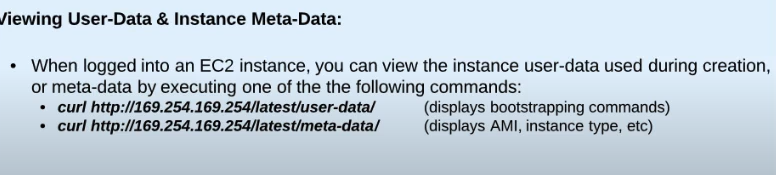
HDFS – Hadoop Distributed File System



T-s don’t give you dedicated CPU s (they are the only instance types)

Network interfaces – you can detach the secondary network interface from a failed instance and attach to a newly created one.

When you need to use variables, like IP Addresses about the instance in the bootstrapping script you can get them by querying the metadata.



On the linux servers – the public key will stay on the ec2 you will get the private. The instance sends a message encrypted w pubic key, It is decrypted by your private key and gets sent back. Usually there is a default user, ec2-user or Ubuntu for Ubuntu AMIs.

Windows – you will get an admin pw, which you need to decrypt. After you can RDP (Remote Desktop Protocol ) to the instance.

Stop writes before issuing a snapshot for data consistency: fsfreeze / stopping the instance. Why? Because if there is any data in the memory of the instance waiting to be written in the ebs will be not concluded in the snapshots. You can create AMI / Volume from the snapshot.

* 1. Placement Groups

When you launch a new EC2 instance, the EC2 service attempts to place the instance in such a way that all of your instances are spread out across underlying hardware to minimize correlated failures. You can use placement groups to influence the placement of a group of interdependent instances to meet the needs of your workload. Depending on the type of workload, you can create a placement group using one of the following placement strategies:

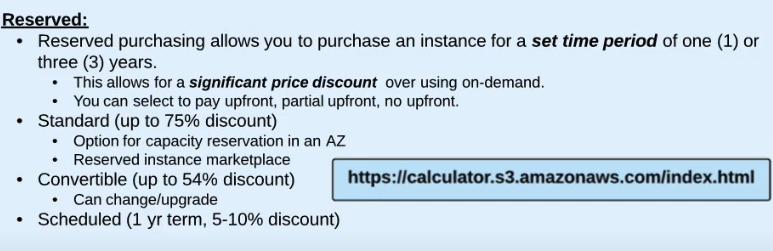
1. Cluster – packs instances close together inside an Availability Zone. This strategy enables workloads to achieve the low-latency network performance necessary for tightly-coupled node-to-node communication that is typical of HPC applications.
2. Partition – spreads your instances across logical partitions such that groups of instances in one partition do not share the underlying hardware with groups of instances in different partitions. This strategy is typically used by large distributed and replicated workloads, such as Hadoop, Cassandra, and Kafka.
3. Spread – strictly places a small group of instances across distinct underlying hardware to reduce correlated failures.

There is no charge for creating a placement group.

* 1. Purchasing Options
     1. On Demand

Paying for seconds, minimum 60s. License charged instances are billed by hour. Expensive for long term but if you are turning on / off then it can be much better.

* + 1. Reserved



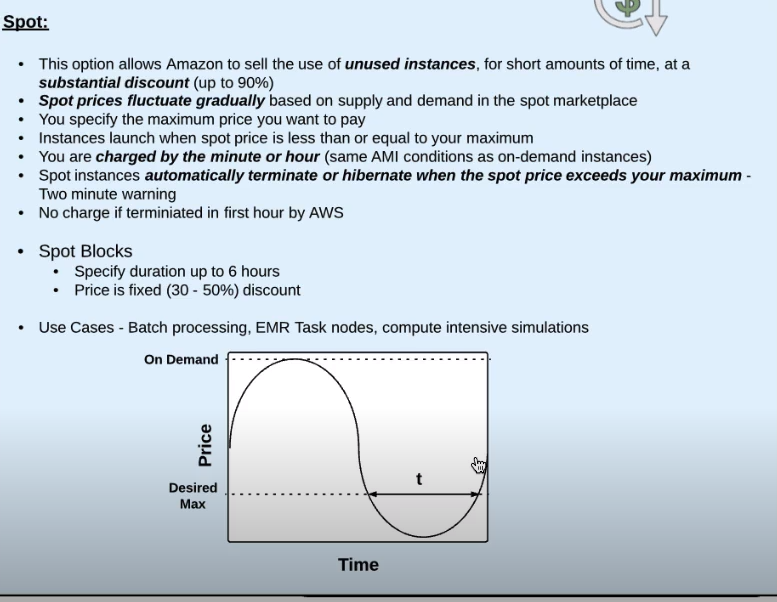
Reserve an instance in an availability zone –

Reserved instances and Organizations (Consolidated Billing) if 1 acc reserved an instance the second can get the discount by running the same instance in same region?

* + 1. Scheduled

When you don’t need non-stop for 1 or 3 years the instance, but for certain period of time. I wanna schedule for these time windows, and you are able to launch only in these windows. (5-10% discount)

* + 1. SPOT

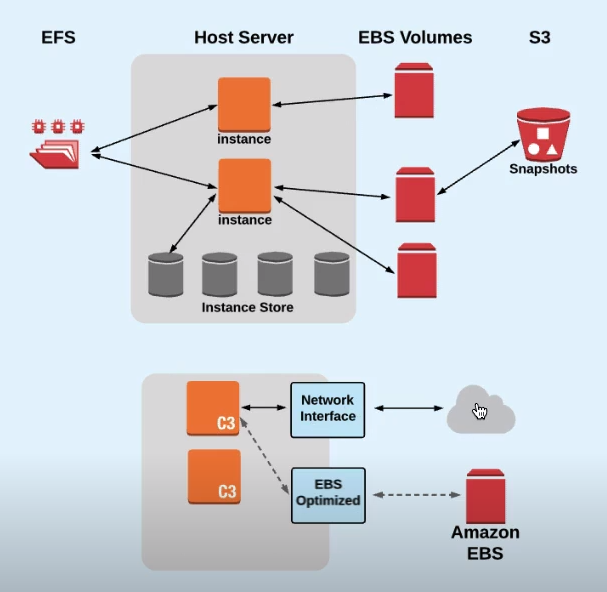


Hibernation?

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html>

<https://aws.amazon.com/blogs/compute/new-amazon-ec2-spot-pricing/>

* 1. EC2 Storage Options



* + 1. EBS

You have a network interface that is used for general network traffic. But in addition to that on all of the current generation EC2 instances there is a second network interface, that’s known as EBS Optimized. It allows the instance to communicate with the EBS volumes without having to contend the regular network traffic going over the network interface. Some of older instances was not EBS optimized by default but you could enabled this feature, now every current gen. instances have this feature.

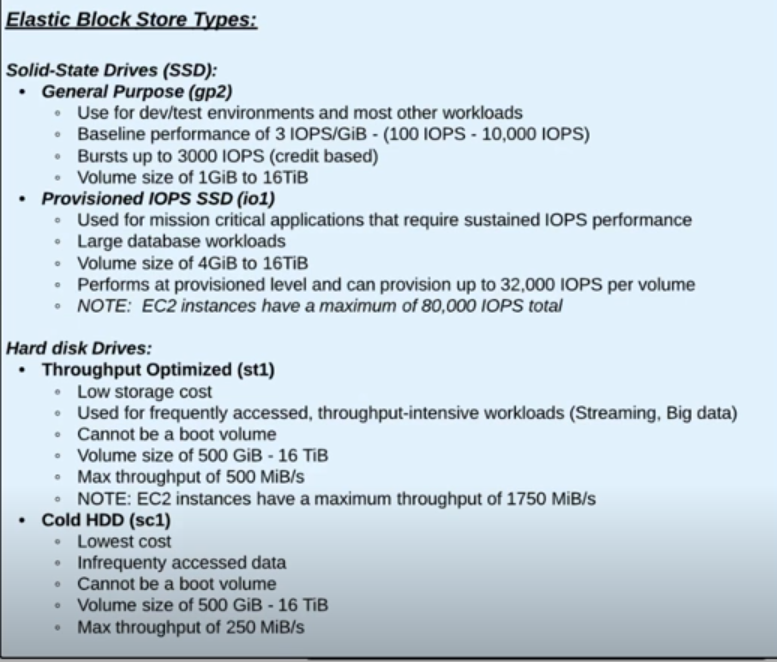
EBS performance

EBS volumes measure I/O operations in IOPS. (input output operations pre secod)

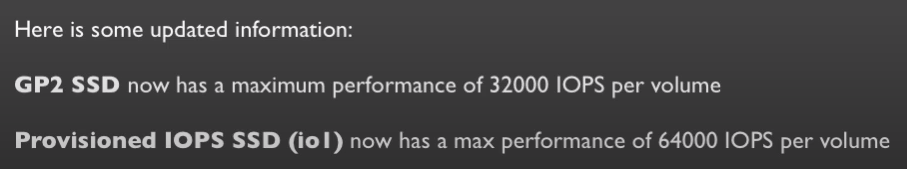
AWS measures IOPS in 256KB chunks (or smaller) Operations greater than 256KB are separated into individual chunks. The type of EBS volume you specify greatly influences the I/O performance your device will receive. Even volumes with ‘provisioned IOPS’ may not produce the performance you expect. If this is the case, an EBS optimized instance type is required, which prioritizes EBS traffic.

100/3000 IOPS – 100 is the baseline performance, and 3000 is where it can burst up to.

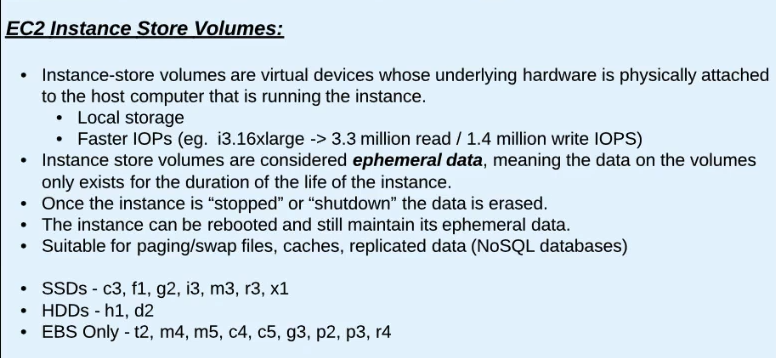
When you restore your data into a new EBS volume the volume will be created immediately, however when you are using an EBS volume which has been restored from a snapshot, the files from the snapshot are not copied into the blocks of the volume until you try to read that block – so it’s lazy loaded. You may see some performance degradation and it starts to perform better and better. – Script for reading every volume of the volume (initialization).



Credit system – when you are working with the EBS in the baseline interval you are accumulating credits. Then you can burst, achieve higher IOPS but you will use these credits and when they are all used up you will be forced back to the baseline interval. The baseline and the credits goes up by provisioning higher EBS volume.



* 1. Instance store



Fixed size and numbers / images, high IOPS, but ephemeral.

Why ephemeral? With stopping the instance you remove the instance from that hypervisor, so there is no more direct connection. But an another instance can take that slot, so before they have to wipe the storage so that nobody can see the data. You cannot count on ec2 instances running years without an issue, so you have to count w the fact that you can loose these data. BUT YOU CAN REBOOT. – paging, cache, nosql db when clustering, swap files? Not available on all instnaces. IF you want to use instance storage you are limited by the instance types.

To get the root value encrypted you need a custom encrypted AMI.

* + 1. Object storage vs Block storage – general

Object storage (also referred to as object-based storage) is a general term that refers to the way in which we organize and work with units of storage, called objects. Every object contains three things:

1. The data itself. The data can be anything you want to store, from a family photo to a 400,000-page manual for assembling an aircraft.
2. An expandable amount of metadata. The metadata is defined by whoever creates the object storage; it contains contextual information about what the data is, what it should be used for, its confidentiality, or anything else that is relevant to the way in which the data is used.
3. A globally unique identifier. The identifier is an address given to the object in order for the object to be found over a distributed system. This way, it’s possible to find the data without having to know the physical location of the data (which could exist within different parts of a data center or different parts of the world).
4. <https://www.druva.com/blog/object-storage-versus-block-storage-understanding-technology-differences/>
   1. EFS

1994 – Sun Microsystem invented a concept of network file system and the NFS protocol. You have a server with file system and that fs. is shared to NFS clients. You could configure an EC2 with EBS volumes to act as an NFA server, but AWS has a service for it. Only Linux compatible.

EFS file systems can be mounted to on-prem. servers, when connected to your VPC via AWS Direct Connect. This allows you to migrate data from on-prem. servers to EFS (EFS file sync) and use it as a backup.

You can provision multiple mount points in multiple AZs – if you have your web server distributed into multiple AZs behind a Load Balancer each instance can be attached to a mount point closest to them and that could reduce latency.

Big Data / Analytics, Content Mangement, Web Serving, Media processing workflows

1. Autoscaling and Elastic Load Balancing

Because everything is an API call it’s very easy to automate the process of adding more capacity proportional to how many requests are coming to the application. (matching the capacity to the load). Reliability and cost efficient.

* 1. Load Balancing

ELB can be paired w Auto Scaling to enhance high availability and fault tolerance, and allow for automated scalability and elasticity. An ELB has its own DNS record set, that allows for direct access from the open internet access.

ELB can be launched in the public subnet (public – facing) or used as an internal in the private subnet. ELB will automatically stop serving traffic to an instance that becomes unhealthy (via health checks).

An ELB or ALB can help reduce compute power on an EC2 instance by allowing for an SSL certificate to be applied directly to the ELB. You can upload your SSL certificate to en ELB or ALB and have your HTTPS request terminated at the LB, and then the LB can forward the request to the instances via regular HTTP. Encrypting and decrypting a stream of requests can be CPU intensive. (SSL, TLS).

* + 1. Classic ELB

Designed for simple balancing of traffic to multiple EC2 instances. There are no granular routing “rules” – all instances get routed evenly and no special routing request can be made based in specific content request from the user.

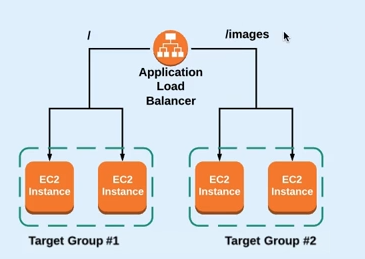
TCP, SSL, HTTP, HTTPS

Round Robin (latency – as an additional metric to CPU utilization)

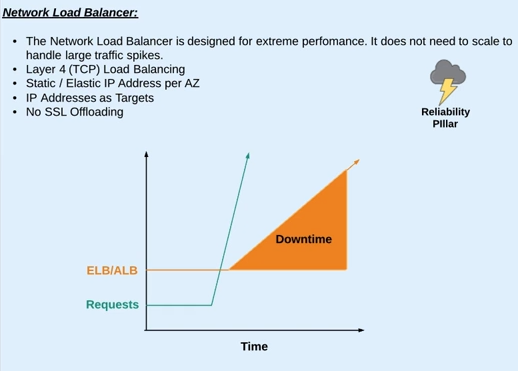
* + 1. Application (Layer 7 ELB)

More advanced LB. Can do everything that a classic ELB + something more. If you have a LB which operates on lvl 7 (application) and not on lvl 4 (transport, TCP) it can inspect the HTTP request, and then based upon the nature of the request will forward the request further.

* An Application Load Balancer is designed for balancing traffic to one or more instance target groups using **Content-based rules.**
* Content-based rules (setup on the listener) can be conjured using:
  1. **Host-based rules:** Route traffic based on the host field of the HTTP header
  2. **Path-based rules**: route traffic based on the UTL path of the HTTP header
  3. This allows you to structure your application as smaller services, and even monitor/auto-scale based on traffic to specific “**target groups**”
  4. Can balance traffic to multiple ports
* An application LB also supports **ECS and EKS, HTTPS, HTPP/2, WebSockets, AccessLogs, Sicky Sessions and AWS WAF.**



* + 1. Difference between Classic and Application ELB
    2. Network Load Balancer



**Suddenly increased load. Does not launch instances, just coded. Handles the spikes.** Supports static and elastic IP.

* + 1. Certificates and protocols

HTTP – one of the most widely used protocol, used for viewing web pages on the Internet. In standard http all the information is sent in clear text.

HTTPS – secure HTTP, encrypts data that is being retrieved by HTTP

HTTPS protect the data with SSL (Secure Sockets Layer), it uses public key encryption to secure data.

TLS (Transport Layer Security) can secure HTTP too, it’s the latest industry standard cryptographic protocol. It is the successor to SSL, and it’s based on the same specifications. A lot of websites are now using HTTPS by default, regardless if sensitive data is going to be exchanged or not. Google will penalize websites without SSL in it’s search ranking.

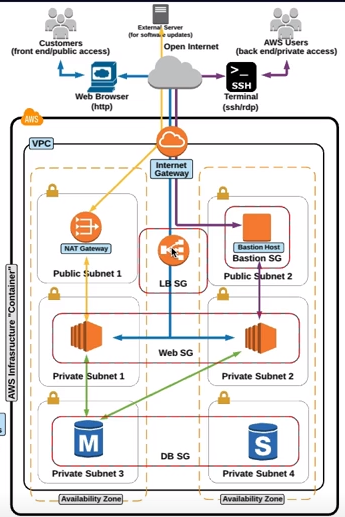
<https://aws.amazon.com/blogs/aws/elastic-load-balancer-support-for-ssl-termination/>

SSL termination?

* + 1. Round Robin and Sicky – Load Balancing methods

<https://avinetworks.com/glossary/round-robin-load-balancing/>

* + 1. NAT Gateway

 NAT Gateway for one-way internet access – so private servers will be able to install packages.

Bastion host so you can ssh into the private instances.

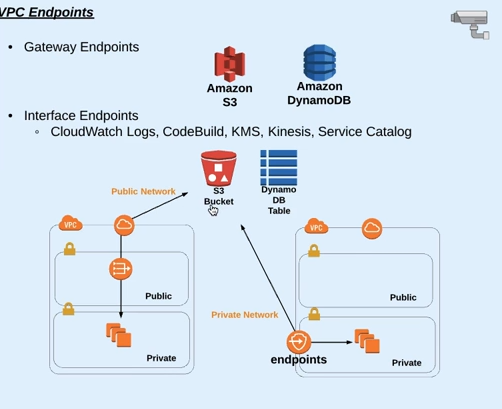
Public facing LB so you can balance the traffic between the private servers.

Nat Gateway MUST be created in a public subnet, MUST be part of the private subnets route table.

NAT Instance is identical to NAT gateway in its purpose, however, it is executed differently by configuring an actual EC2 instance to do the same job. Legacy. IP maskarading.

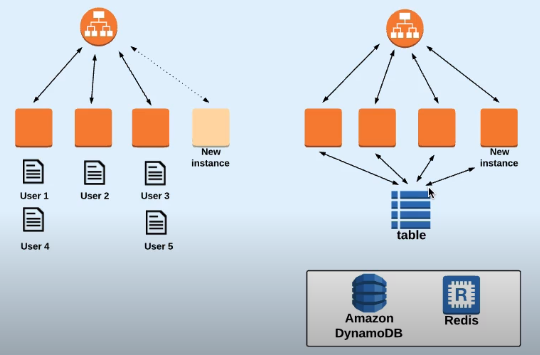
* + 1. VPC endpoints

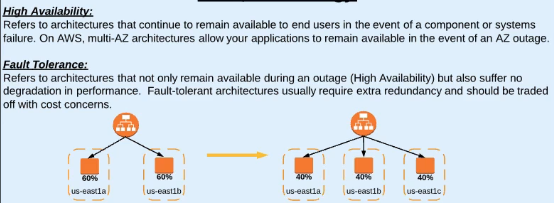
*When private instances need to reach AWS’s public services.*

Without endpoint traffic will go through the internet. After you add the endpoint you need to update the route table. (it will be done automatically during the creation.)

* + 1. Stateful vs Stateless applications

Stateful applications do not combo well with the LB – apps that maintain session. Maintain that state object – saves it locally (s3, db)

Left Stateful (session send there and back w cookies) Right Stateless much better, RDS is not good to store session.



* 1. Autoscaling

Increasing availability – horizontal scaling – adding more instances, more capacity when needed. LB will start use these.

Cost – terminate the unneeded instances.

* **Launch Configuration**

EC2 template used when the auto scaling group needs to provision an additional instance (i.e. AMI, instance type, user-data, storage, security groups)

**Launch Configuration vs Launch Template (new)** Launch templates give you the option of launching one type of instance, or a combination of instance types and purchase options. Launch templates include the latest Amazon EC2 features and can be updated and versioned.

* **Auto Scaling Group**

All the rules and settings that govern if/when an EC2 instance is automatically provisioned or terminated.

* + Number of MIN & MAX allows instances
  + VPC & AZs to launch instanes into
  + If provisioned instanes should receive traffic from an ELB
  + Scaling policies (cw metrics thresholds that trigger scaling)
  + SNS notifications
* **CloudWatch Alarms**
  + Metrics are selected that indicate load on instances(CPU Utilization, Latency)
  + Alarms are triggered when metrics exceeds thresholds
  + Alarms trigger Autoscaling policies
  1. CloudFront

CF – accelerate delivery by caching content on locations closer to the enduser. (Edge locations) Caches static content for website, so origins are offloaded from extra traffic. Data transfer from s3 to cloudfront is free. Live video streaming and VOD or web distribution. Cloudfront.net domain name by default, you can use HTTPS by default. GET ofc, but you can use POST/PUT too – long connections may not be that stable. TTL – how long are objects cached for. If you need to change the cache – rename the logo for logo2 or invalidate – costly.

RTMP distribution – you must store the media files in an S3 bucket, use CF live streaming, create web distribution.

• AWS Shield

• AWS WAF

Signed URL or Signed Cookie (expiration, restrict by IP ..)

Dynamic objects are still cached, connection optimization.

1. Databases and S3
   1. SQL Databases

ACIT – Atomicity, Consistency, Isolation, Durability

OLAP - Online Analytical Processing, The primary objective is data analysis and not data processing. – optimized for queries. Data Warehouse: large datasets, running queries aginst the data.

OLTP - Online transaction processing, the primary objective is data processing and not data analysis – optimized for inserts/updates. Typycal databases, like oracle, ms sql server, mysql, postgre

* 1. NOSQL
* Column – fast in terms of querying
* Key-Value
* Document – json/xml
* Graph – optimized for reltionships

Casssandra, MongoDB, HBase

ON AWS: managed service (may not support your engine) / deploy db on an instance

* 1. DynamoDB

API – no servers, serverless architecture. Build as a fault tolerant highly available service. On the back end, it fully synchroniuzes the data across all of the availability zones within the region you created the DynamoDB tables in.

* + - ITO (storing metadata)
    - Gaming (storing session info, leaderborads)
    - Mobile (storing user profiles, personalization)

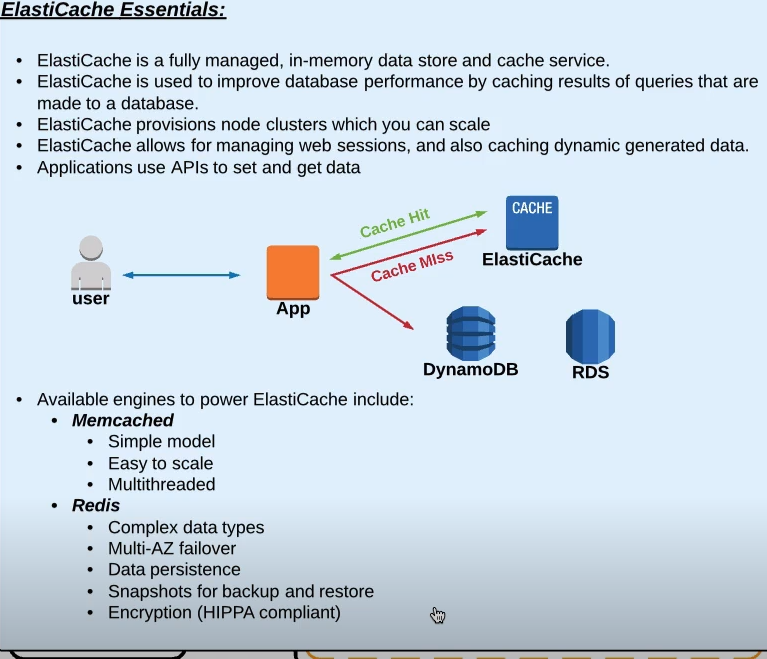
Built on top of SSD + parallelism.

* + - Partition key (if it was no unique i need to add sort key.
    - Sort key
    - Composite key

Default settings – 5 reads/s and 5 writes/s. You can set higher, or leave 5-5 but allow autoscaling (needs role). Items dont have to hafve the same attributes. You can add book and bicycles to the product table too.

* + - Scan
    - Query
  1. ElastiCache

Used to offload other databases, so you are not loading them w the same query statements again and again. Good for sessions.

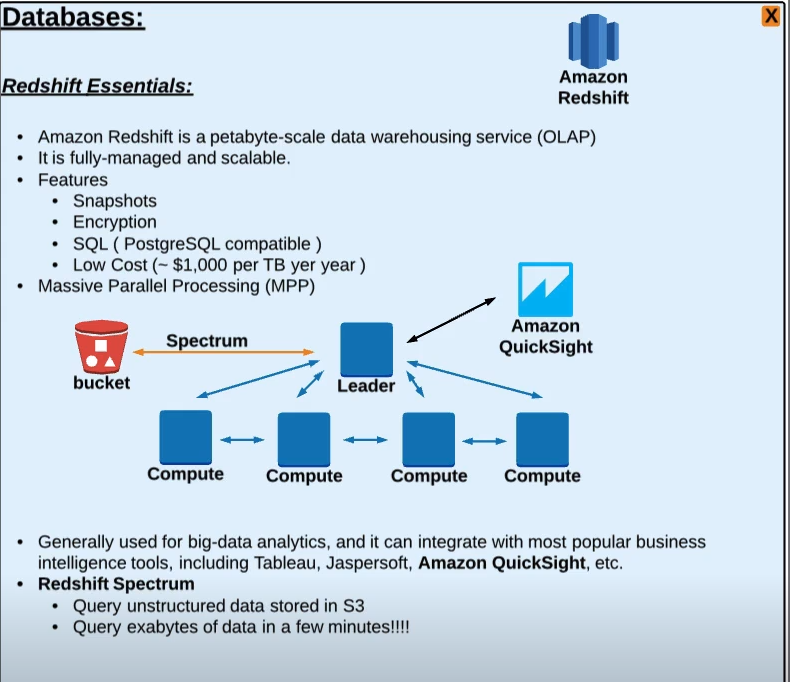


* 1. Redshift

Optimized for analytics. Large, systems feed them. Validation, cleaning up the data. Reports.

You can connect to the leader w sql client. It passes the work to compute nodes. You can use any postgres compatible tool. QuickSight – visual reports of the queries.

Spectrum – extend quetries , add unstructured data stored on s3.



* 1. S3

Backups, replicating objects

File shared and backups/archiving for hybrid networks (via AWS Storage Gateway)

You should always create an S# bucket in a region that makes sense to it’s purpose, serving content to customers, sharing data with ec2.

All regions support:

* Red-after-write consistency for PUTS of new objects
* Eventual consistency for PUTS overwriting existing objects and DELETES of objects

In theoretical computer science, the CAP theorem, also named Brewer's theorem after computer scientist Eric Brewer, states that it is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees:

* Consistency: Every read receives the most recent write or an error
* Availability: Every request receives a (non-error) response – without the guarantee that it contains the most recent write
* Partition tolerance: The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes