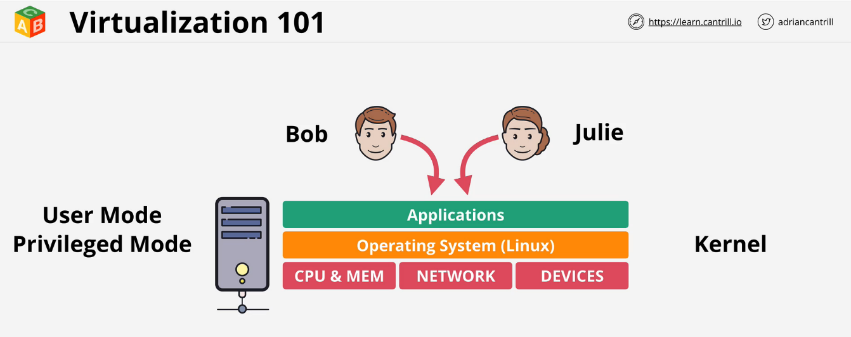
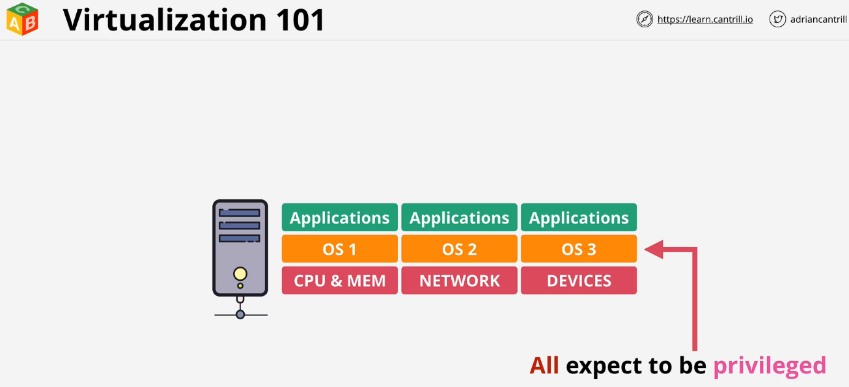
**Elastic-Cloud-Compute-EC2**

EC2 provides Infrastructure as a Service (IaaS Product)

**Virtualization 101**

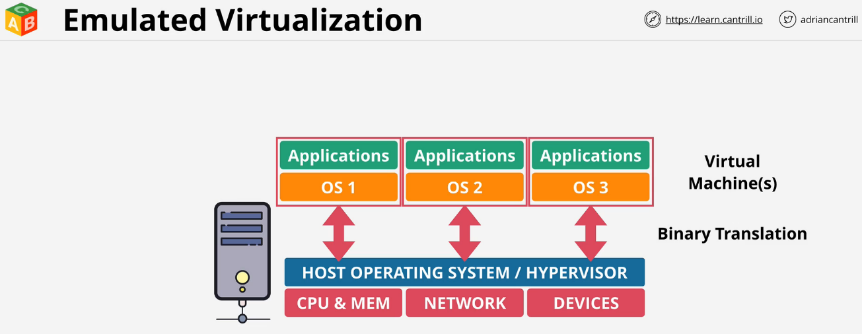
Servers are configured in three sections without virtualization.

* CPU hardware
* Kernel
  + Operating system
  + Runs in **privileged mode** and can interact with the hardware directly.
* User Mode
  + Runs applications.
  + Can make a **system call** to the Kernel to interact with the hardware.
  + If an app tries to interact with the hardware without a system call, it will cause a system error and can crash the server or at minimum the app.

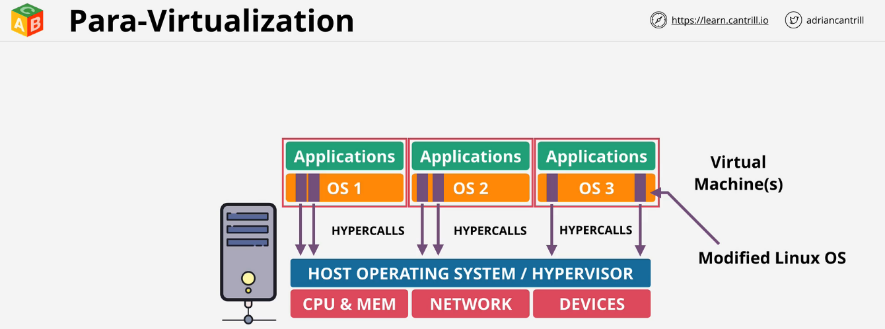
**Emulated Virtualization - Software Virtualization**

Host OS operated on the HW and included a hypervisor (HV). SW ran in privileged mode and had full access to the HW. Guest OS wrapped in a VM and had devices mapped into their OS to emulate real HW. Drivers such as graphics cards were all SW emulated to allow the process to run properly.

The guest OS still believed they were running on real HW and tried to take control of the HW. The areas were not real and only allocated space to them for the moment.

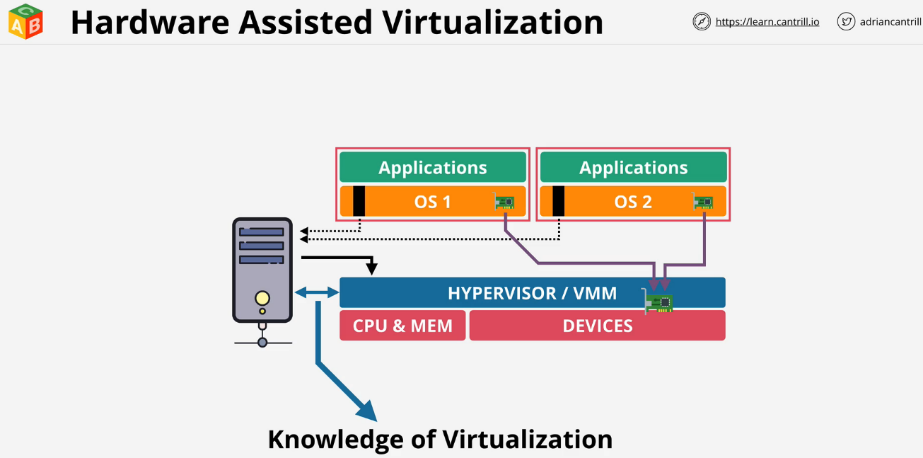
The HV performs **binary translation**. System calls are intercepted and translated in SW on the way. The guest OS needs no modification, but slows down a lot.

**Para-Virtualization**

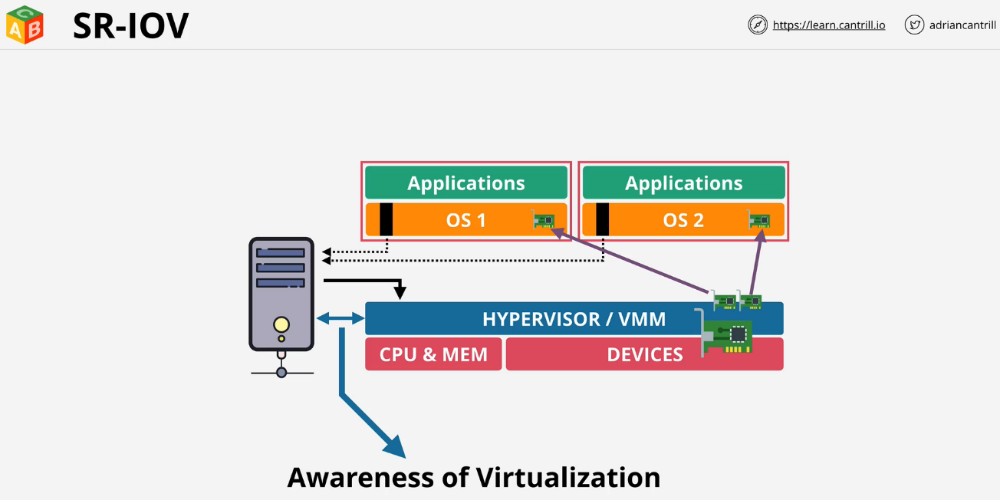
Guest OS are modified and run in HV containers, except they do not use slow binary translation. The OS is modified to change the **system calls** to **user calls**. Instead of calling on the HW, they call on the HV using **hypercalls**. Areas of the OS call the HV instead of the HW.

**Hardware Assisted Virtualization**

The physical HW itself is virtualization aware. The CPU has specific functions so the HV can come in and support. When guest OS tries to run privileged instructions, they are trapped by the CPU and do not halt the process. They are redirected to the HV from the HW.

What matters for a VM is the input and output operations such as network transfer and disk IO. The problem is multiple OS try to access the same piece of hardware but they get caught up on sharing.

**SR-IOV (Singe Route IO virtualization)**

Allows a network or any card to present itself as many mini cards. As far as the HW is concerned, they are real dedicated cards for their use. No translation needs to be done by the HV. The physical card handles it all. In EC2 this feature is called **enhanced networking**.

**EC2 Architecture and Resilience**

EC2 instances are virtual machines run on EC2 hosts.

Tenancy:

* **Shared** - Instances are run on shared hardware, but isolated from other customers.
* **Dedicated** - Instances are run on hardware that's dedicates to a single customer. Dedicated instances may share hardware with other instances from the same AWS account that are not Dedicated instances.
* **Dedicated host** - Instances are run on a physical server fully dedicated for your use. Pay for entire host, don't pay for instances.
* AZ resilient service. They run within only one AZ system.
  + You can't access them cross zone.

EC2 host contains

* Local hardware such as CPU and memory
* Also have temporary instance store
  + If instance moves hosts, the storage is lost.
* Can use remote storage, Elastic Block Store (EBS).
  + EBS allows you to allocate volumes of persistent storage to instances within the same AZ.
* 2 types of networking
  + Storage networking
  + Data networking

**EC2 Networking (ENI) [Elastic Network Interface]**

When instances are provisioned within a specific subnet within a VPC. A **primary elastic network interface** is provisioned in a subnet which **maps to the physical hardware on the EC2 host**.

Subnets are also within one specific AZ. Instances can have **multiple network interfaces**, even **within different subnets so long as they're within the same AZ**.

An instance runs on a specific host. If you restart the instance it will stay on that host until either:

* The host fails or is taken down by AWS
* The instance is stopped and then started, different than restarted.

The instance will be **relocated to another host in the same AZ**. Instances **cannot move to different AZs**. Everything about their hardware is locked within one specific AZ.

A migration is taking a **copy** of an instance and moving it to a different AZ.

In general **instances of the same type and generation will occupy the same host**. The only difference will generally be their size.

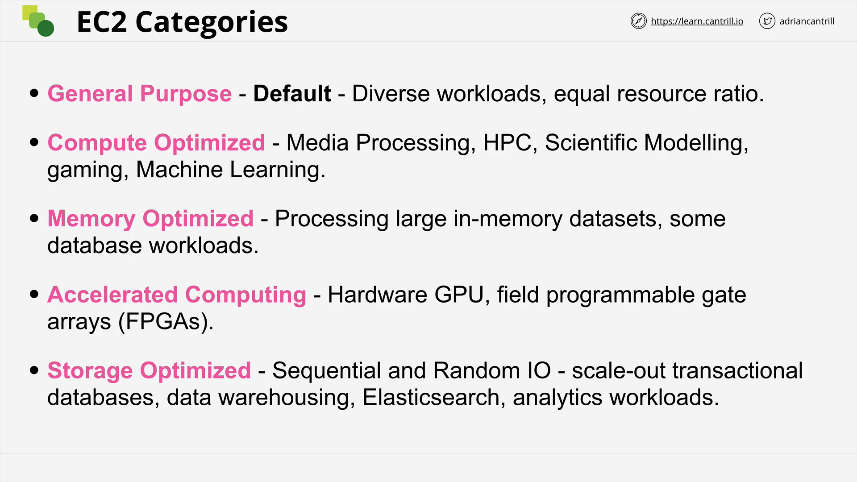
**EC2 Strengths**

Long running compute needs. Many other AWS services have run time limits.

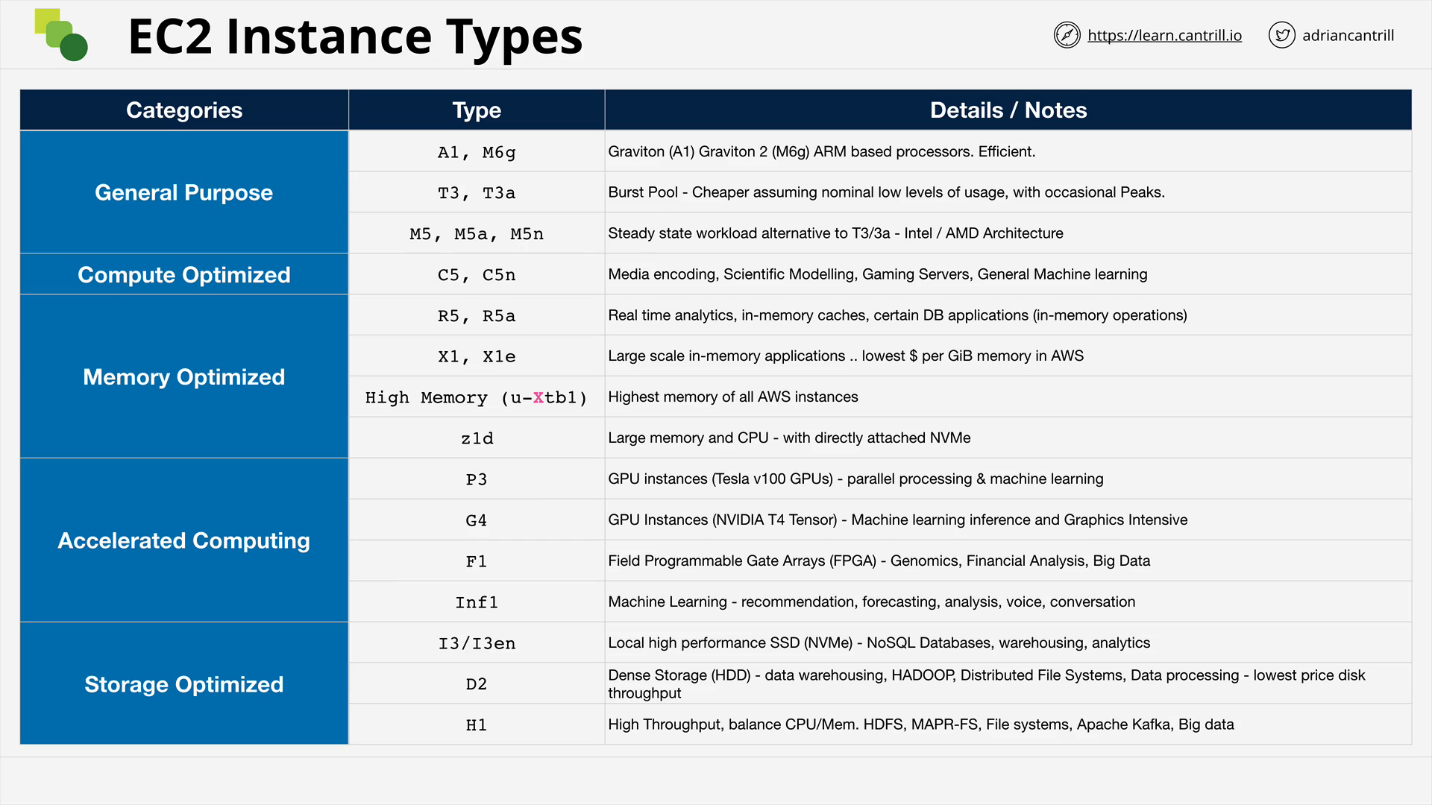
Server style applications

* things waiting for network response
* burst or stead-load
* monolithic application stack
  + middle ware or specific run time components
* migrating application workloads or disaster recovery
  + existing applications running on a server and a backup system to intervene

**EC2 Instance Types**

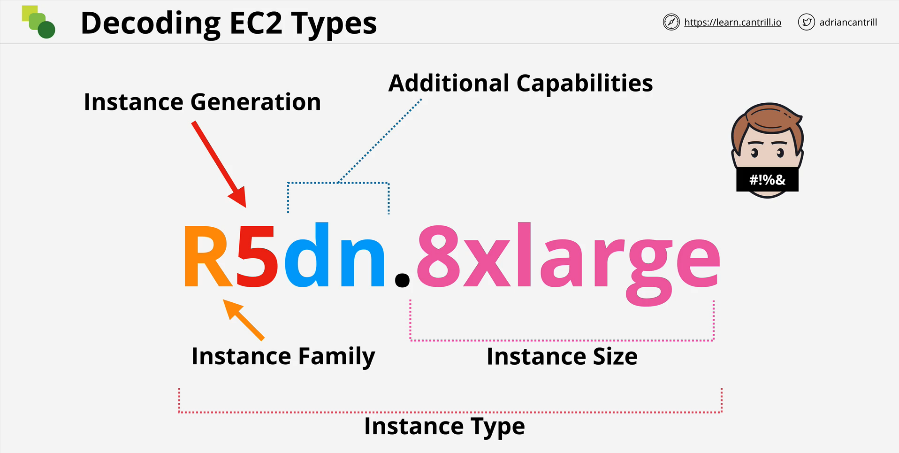
* **General Purpose** (T, M) - default steady state workloads with even resources
* **Compute Optimized** (C) - Media processing, scientific modeling and gaming
* **Memory Optimized** (R, X) - Processing large in-memory data sets
* **Accelerated Computing** (P, G, F) - Hardware GPU, FPGAs
* **Storage Optimized** (H, I, D) - Large amounts of super fast local storage. Massive amounts of IO per second. Elastic search and analytic workloads.

<https://ec2instances.info/>



**Naming Scheme**

R5dn.8xlarge - whole thing is the instance type. When in doubt give the full instance type

* 1st char: Instance family.
* 2nd char: Instance generation. Generally always select the newest generation.
* char after period: Instance size. Memory and CPU considerations.
  + Often easier to scale system with a larger number of smaller instance sizes.
* 3rd char - before period: additional capabilities
  + a: amd cpu
  + d: NVMe storage
  + n: network optimized
  + e: extra capacity for ram or storage

**Storage Refresher**

* **Instance Store**
  + Direct (local) attached storage
  + Super-fast
  + Ephemeral storage or temporary storage
* **Elastic Block Store (EBS)**
  + Network attached storage
  + Volumes delivered over the network
  + Persistent storage lives on past the lifetime of the instance

**Three types of storage**

* **Block Storage:** Volume presented to the OS as a collection of blocks. No structure beyond that. These are mountable and bootable. The OS will create a file system on top of this, NTFS or EXT3 and then it mounts it as a drive or a root volume on Linux. Spinning hard disks or SSD. This could also be delivered by a physical volume. Has no built in structure. You can mount an EBS volume or boot off an EBS volume.
* **File Storage:** Presented as a file share with a structure. You access the files by traversing the storage. You cannot boot from storage, but you can mount it.
* **Object Storage:** It is a flat collection of objects. An object can be anything with or without attached metadata. To retrieve the object, you need to provide the key and then the value will be returned. This is not mountable or bootable. It scales very well and can have simultaneous access.

**Storage Performance**

* **IO Block Size:** Determines how to split up the data in the disk.
* **IOPS:** How many reads or writes a system can accommodate per second.
* **Throughput:** End rate achieved, expressed in MB/s (megabyte per second).

Block Size \* IOPS = Throughput

This isn't the only part of the chain, but it is a simplification. A system might have a throughput cap. The IOPS might decrease as the block size increases.

**Elastic Block Store (EBS)**

* Allocate block storage **volumes** to instances.
* Volumes are isolated to one AZ.
  + The data is highly available and resilient for that AZ.
  + All of the data is replicated within that AZ. The entire AZ must have a major fault to go down.
* Two physical storage types available (SSD/HDD)
* Varying level of performance (IOPS, T-put)
* Billed as GB/month.
  + If you provision a 1TB for an entire month, you're billed as such.
  + If you have half of the data, you are billed for half of the month.
* Four types of volumes, each with a dominant performance attribute.
  + General purpose SSD (gp2)
  + Provisioned IOPS SSD (io1)
    - maximum IOPS such as databases
  + T-put optimized HDD (st1)
    - maximum t-put for logs or media storage
  + Cold HDD (sc1)

**General Purpose SSD (gp2)**

Uses a performance bucket architecture based on the IOPS it can deliver. The GP2 starts with 5,400,000 IOPS allocated. It is all available instantly.

You can consume the capacity quickly or slowly over the life of the volume. The capacity is filled back based upon the volume size. Min of 100 IOPS added back to the bucket per second.

Above that, there are 3 IOPS/GiB of volume size. The max is 16,000 IOPS. This is the **baseline performance**

Default for boot volumes and should be the default for data volumes. Can only be attached to one EC2 instance at a time.

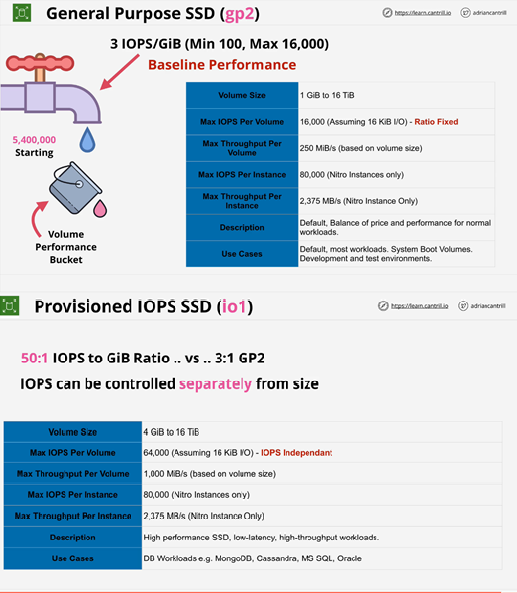
**Provisioned IOPS SSD (io1)**

You pay for capacity and the IOPs set on the volume. This is good if your volume size is small but need a lot of IOPS.

50:1 IOPS to GiB Ratio 64,000 is the max IOPS per volume assuming 16 KiB I/O.

To be used in situations where volume is small but IO is high

Good for latency sensitive workloads such as mongoDB.

**Multi-attach** allows them to attach to multiple EC2 instances at once.

**HDD Volume Types**

* great value
* great for high throughput vs IOPs
* 500 GiB - 16 TiB
* Neither can be used for EC2 boot volumes.
* Good for streaming data on a hard disk.
  + Media conversion with large amounts of storage.
* Frequently accessed high throughput intensive workload
  + log processing
  + data warehouses
* Bad for random access.
* The access patterns should be sequential
  + Massive inefficiency for small reads and writes

Two types

* st1
  + Starts at 1 TiB of credit per TiB of volume size.
  + 40 MB/s baseline per TiB
  + Burst of 250 MB/s per TiB
  + Max t-put of 500 MB/s
* sc1
  + Starts at 1 TiB of credit per TiB of volume size.
  + Designed for less frequently accessed data, it fills slower.
  + 12 MB/s baseline per TiB
  + Burst of 80 MB/s per TiB
  + Max t-put of 250 MB/s

**EBS Exam Power Up**

* Volumes are created in an AZ, isolated in that AZ.
* If an AZ fails, the volume is impacted.
* Highly available and resilient in that AZ. The only reason for failure is if the whole AZ fails.
* Generally one volume to one instance, except **io1** with multi-attach
* Has a GB/m fee regardless of instance state.
* EBS maxes at 80000 IOPS per instance and 64000 vol (io1)
* Max 2375 MB/s per instance, 1000 MiB/s (vol) (io1)

**EC2 Instance Store**

* Local **block storage** attached to an instance.
* Physically connected to one EC2 host.
  + They are isolated to that one specific host.
  + Instances on that host can access them.
* Highest storage performance in AWS.
* Included in instance price, use it or lose it.
* Can be attached ONLY at launch. Cannot be attached later.

Each instance has a collection of volumes that are locked to that specific host. If the instance moves, the data doesn't.

Instances can move between hosts for many reasons:

* If an instance is stopped and started, that migrates hosts.
* If a host undergoes AWS maintenance, it will be wiped.
* If you change the type of an instance, these will be lost.
* If a physical hardware fails, then the data is gone.

The number, size, and performance of instance store volumes vary based on the type of instance used. Some instances do not have any instance store volumes at all.

**Instance Store Exam PowerUp**

* Instance store volumes are local to EC2 host.
* Can only be added at launch time. Cannot be added later.
* Any data on instance store data is lost if it gets moved, or resized.
* Highest data performance in all of AWS.
* You pay for it anyway, it's included in the price.
* TEMPORARY

**EBS vs Instance Store**

If the read/write can be handled by EBS, that should be default.

When to use EBS

* Highly available and reliable in an AZ. Can self-correct against HW issues.
* Persist independently from EC2 instances.
  + Can be removed or reattached.
  + You can terminate an instance and keep the data.
* **Multi-attach feature of io1**
  + Can create a multi shared volume.
* Region resilient backups.
* Require up to 64,000 IOPS and 1,000 MiB/s per volume
* Require up to 80,000 IOPS and 2,375 MB/s per instance

When to use Instance Store

* Great value, they're included in the cost of an instance.
* More than 80,000 IOPS and 2,375 MB/s
* If you need temporary storage, or can handle volatility. (For DB Servers we can use gp2 as boot volumes, io1 for the persistent datastore & instance store volumes for the caches)
* Stateless services, where the server holds nothing of value.
* Rigid lifecycle link between storage and the instance.
  + This ensures the data is erased when the instance goes down.

**EBS Snapshots, restore, and fast snapshot restore**

* Efficient way to backup EBS volumes to S3.
  + The data becomes region resilient.
* Can be used to migrate data between hosts.

Snapshots are incremental volume copies to S3. The first is a **full copy** of data on the volume. This can take some time. EBS won't be impacted, but will take time in the background. Future snaps are incremental, consume less space and are quicker to perform.

If you delete an incremental snapshot, it moves data to ensure subsequent snapshots will work properly.

Volumes can be created (restored) from snapshots. Snapshots can be used to move EBS volumes between AZs. Snapshots can be used to migrate data between volumes.

**Snapshot and volume performance**

* When creating a new EBS volume without a snapshot, the performance is available immediately.
* When restoring from S3, performs **Lazy Restore**
  + If you restore a volume, it will transfer it slowly in the background.
  + If you attempt to read data that hasn't been restored yet, it will immediately pull it from S3, but this will achieve lower levels of performance than reading from EBS directly.
  + You can force a read of every block all data immediately using DD.

**Fast Snapshot Restore (FSR)** allows for immediate restoration. You can create 50 of these FSRs per region. When you enable it on a snapshot, you pick the snapshot specifically and the AZ that you want to be able to do instant restores to. Each combination of Snapshot and AZ counts as one FSR set. You can have 50 FSR sets per region. FSR is not free and can get expensive with lots of different snapshots.

**Snapshot Consumption and Billing**

Billed using a GB/month metric. 20 GB stored for half a month, represents 10 GB-month.

This is used data, not allocated data. If you have a 40 GB volume but only use 10 GB, you will only be charged for the allocated data. This is not how EBS itself works.

The data is incrementally stored which means doing a snapshot every 5 minutes will not necessarily increase the charge as opposed to doing one every hour.

**EBS Encryption**

Provides at rest encryption for block volumes and snapshots.

When you don't have EBS encryption, the volume is not encrypted. The physical hardware itself may be performing at rest encryption, but that is a separate thing.

When you set up an EBS volume initially, EBS uses KMS and a customer master key. This can be the EBS default (CMK) which is referred to as aws/ebs or it could be a customer managed CMK which you manage yourself.

That key is used by EBS when an encrypted volume is created. The CMK generates an encrypted data encryption key which is stored on the volume with the physical disk. This key can only be encrypted by KMS when a role with the proper permissions makes the request.

When the volume is first used, EBS asks CMS to decrypt the key and stores the decrypted key in memory on the EC2 host while it's being used. At all other times it's stored on the volume in encrypted form.

When the EC2 instance is using the encrypted volume, it can use the decrypted data encryption key to move data on and off the volume. It is used for all cryptographic operations when data is being used to and from the volume.

When data is stored at rest, it is stored as ciphertext.

If the EBS volume is ever moved, the key is discarded.

If a snapshot is made of an encrypted EBS volume, the same data encryption key is used for that snapshot. Anything made from this snapshot is also encrypted in the same way.

Every time you create a new EBS volume from scratch, it creates a new data encryption key.

**EBS Encryption Exam Power Up**

* AWS accounts can be set to encrypt EBS volumes by default.
  + It will use the default CMK unless a different one is chosen.
  + Each volume uses 1 unique DEK (data encryption key)
  + Snapshots and future volume use the same DEK
* Can't change a volume to NOT be encrypted.
  + You could mount an unencrypted volume and copy things over but you can't change the original volume.
* The OS itself isn't aware of the encryption, there is no performance loss.
  + The volume itself is encrypted using AES256
  + This occurs between the EC2 host and the EBS system itself.
  + The OS does not see any encryption. It simply writes data out and reads data in from a disk.
  + If an exam question does not use AES256, or it suggests you need an OS to encrypt or hold the keys, then you need to perform full disk encryption at the operating system level.

**EC2 Network Interfaces, Instance IPs and DNS**

An EC2 instance starts with at least one ENI - elastic network interface. An instance may have ENIs in separate subnets, but everything must be within one AZ.

When you launch an instance with Security Groups, they are on the network interface and not the instance.

**Elastic Network Interface (ENI)**

Has these properties

* MAC address
* **Primary IPv4 private address**
  + From the range of the subnet the ENI is within.
  + Will be static and not change for the lifetime of the instance
    - 10.16.0.10
  + Given a DNS name that is associated with the address.
    - ip-10-16-0-10.ec2.internal
    - Only resolvable inside the VPC and always points at private IP address
* **0 or more secondary private IP addresses**
* **0 or 1 public IPv4 address**
  + The instance must manually be set to receive an IPv4 address or spun into a subnet which automatically allocates an IPv4. This is a dynamic IP that is not fixed. If you stop an instance the address is removed. When you start up again, it is given a brand new IPv4 address. Restarting the instance will not change the IP address. Changing between EC2 hosts will change the address. This will be allocated a public DNS name. The Public DNS name will resolve to the primary private IPv4 address of the instance. Outside of the VPC, the DNS will resolve to the public IP address. This allows one single DNS name for an instance, and allows traffic to resolve to an internal address inside the VPC and the public will resolve to a public IP address.
* **1 elastic IP per private IPv4 address**
  + Can have 1 public elastic interface per private IP address on this interface. This is allocated to your AWS account. Can associate with a private IP on the primary interface or secondary interface. If you are using a public IPv4 and assign an elastic IP, the original IPv4 address will be lost. There is no way to recover the original address.
* **0 or more IPv6 address on the interface**
  + These are by default public addresses.
* **Security groups**
  + Applied to network interfaces.
  + Will impact all IP addresses on that interface.
  + If you need different IP addresses impacted by different security groups, then you need to make multiple interfaces and apply different security groups to those interfaces.
* **Source / destination checks:** 
  + Each EC2 instance performs source/destination checks by default. This means that the instance must be the source or destination of any traffic it sends or receives. However, a NAT instance must be able to send and receive traffic when the source or destination is not itself. Therefore, you must disable source/destination checks on the NAT instance.

**Secondary interfaces** function in all the same ways as primary interfaces except you can detach interfaces and move them to other EC2 instances.

**ENI Exam PowerUp**

* Legacy software is licensed using a mac address. Since the ENI has mac address.
  + If you provision a secondary ENI to a specific license, you can move around the license to different EC2 instances.
* Multi homed (subnets) management and data.
* Different security groups are attached to different interfaces.
* The OS doesn't see the IPv4 public address.
* You always configure the private IPv4 private address on the interface.
* Never configure an OS with a public IPv4 address.
* IPv4 Public IPs are Dynamic, starting and stopping will kill it.

Public DNS for a given instance will resolve to the primary private IP address in a VPC. If you have instance to instance communication within the VPC, it will never leave the VPC. It does not need to touch the internet gateway.

**Amazon Machine Image (AMI)**

Images of EC2 instances that can launch more EC2 instance.

* When you launch an EC2 instance, you are using an Amazon provided AMI.
* Can be Amazon or community provided
* Marketplace (can include commercial software)
  + Will charge you for the instance cost and an extra cost for the AMI
* AMIs are regional with a unique ID.
* Controls permissions
  + Default only your account can use it.
  + Can be set to be public.
  + Can have specific AWS accounts on the AMI.
* Can create an AMI from an existing EC2 instance to capture the current config.

**AMI Lifecycle**

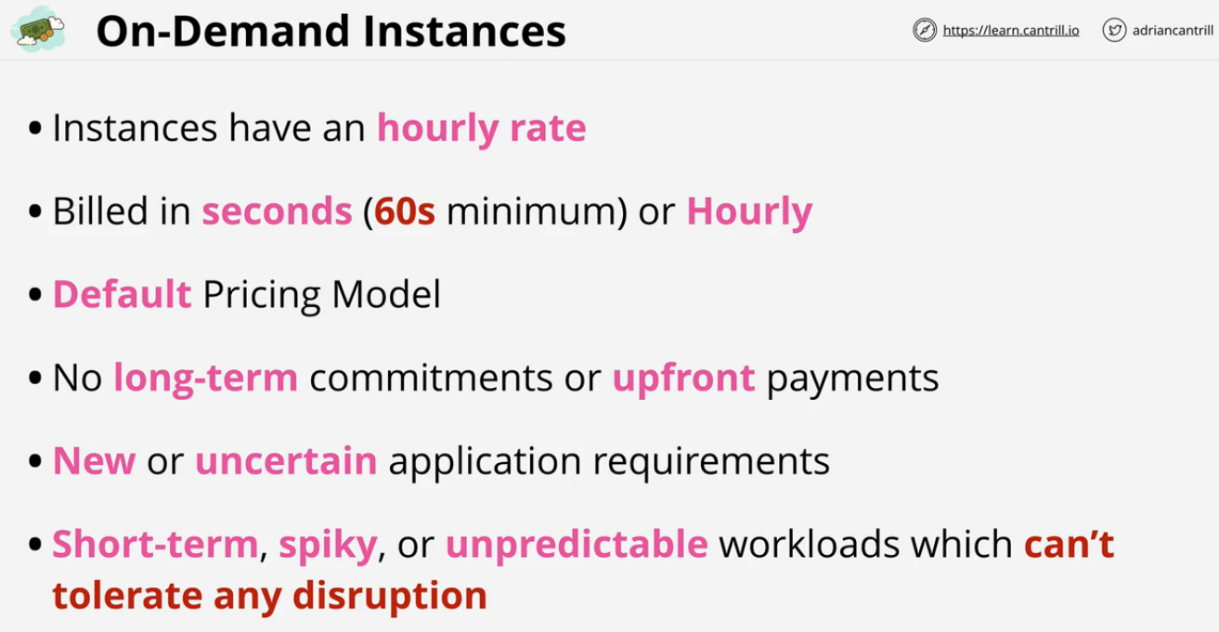
1. **Launch the Instance:** EBS volumes are attached to EC2 devices using block IDs.
   * BOOT /dev/xvda
   * DATA /dev/xvdf
2. **Configure:** customize the instance from applications or volume sizes.
3. **Create Image or AMI**
   * **AMI contains:**
     + **Permissions:** who can use it, is it public or private
     + **EBS snapshots are created from attached EBS volumes**
       - Snapshots are referenced inside the AMI using **block device mapping**.
       - Table of data that links the snapshot IDs that you've just created when making that AMI and it has for each one of those snapshots, a device ID that the original volumes had on the EC2 instance.
4. **Launch:** When launching an instance, the snapshots are used to create new EBS volumes in the AZ of the EC2 instance and contain the same block device mapping.

**AMI Exam PowerUps**

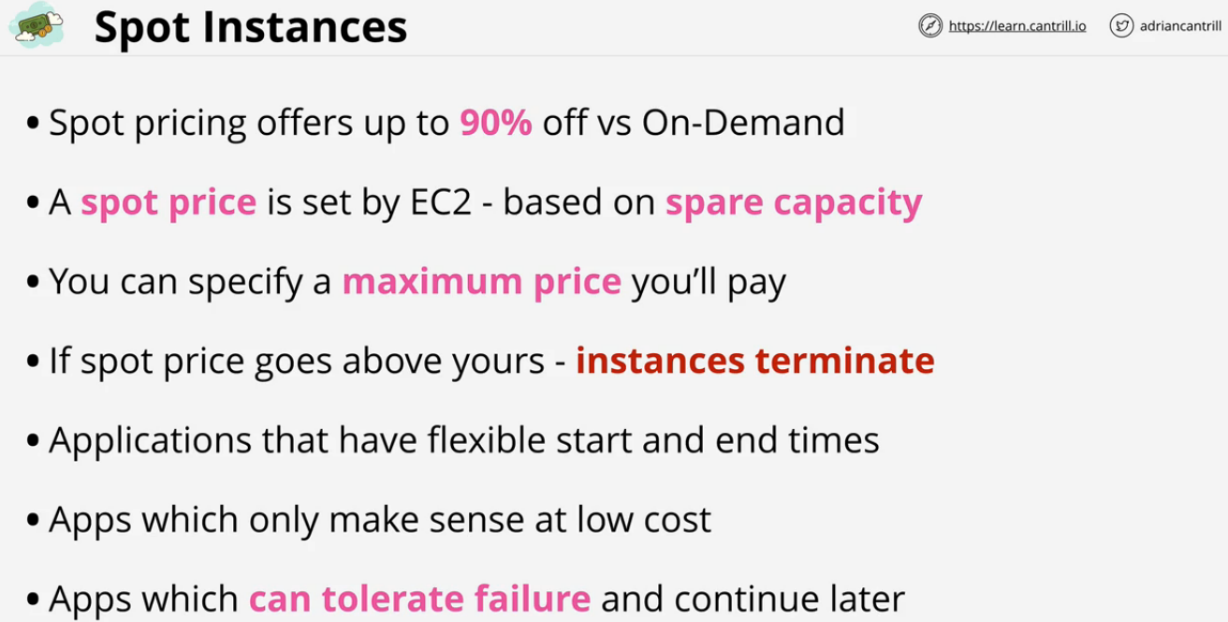
* AMI can only be used in one region
* AMI Baking: creating an AMI from a configuration instance.
* An AMI cannot be edited. If you need to update an AMI, launch an instance, make changes, then make new AMI
* Can be copied between regions
* Remember permissions by default are your account only
* Billing is for the storage capacity for the EBS snapshots the AMI references.

**EC2 Pricing Models**

**On-Demand Instances**

* Hourly rate based on OS, size, options, etc
* Billed in seconds (60s min) or hourly
  + Depends on the OS
* Default pricing model
* No long-term commitments or upfront payments
* New or uncertain application requirements
* Short-term, spiky, or unpredictable workloads which can't tolerate disruption.

**Spot Instances**

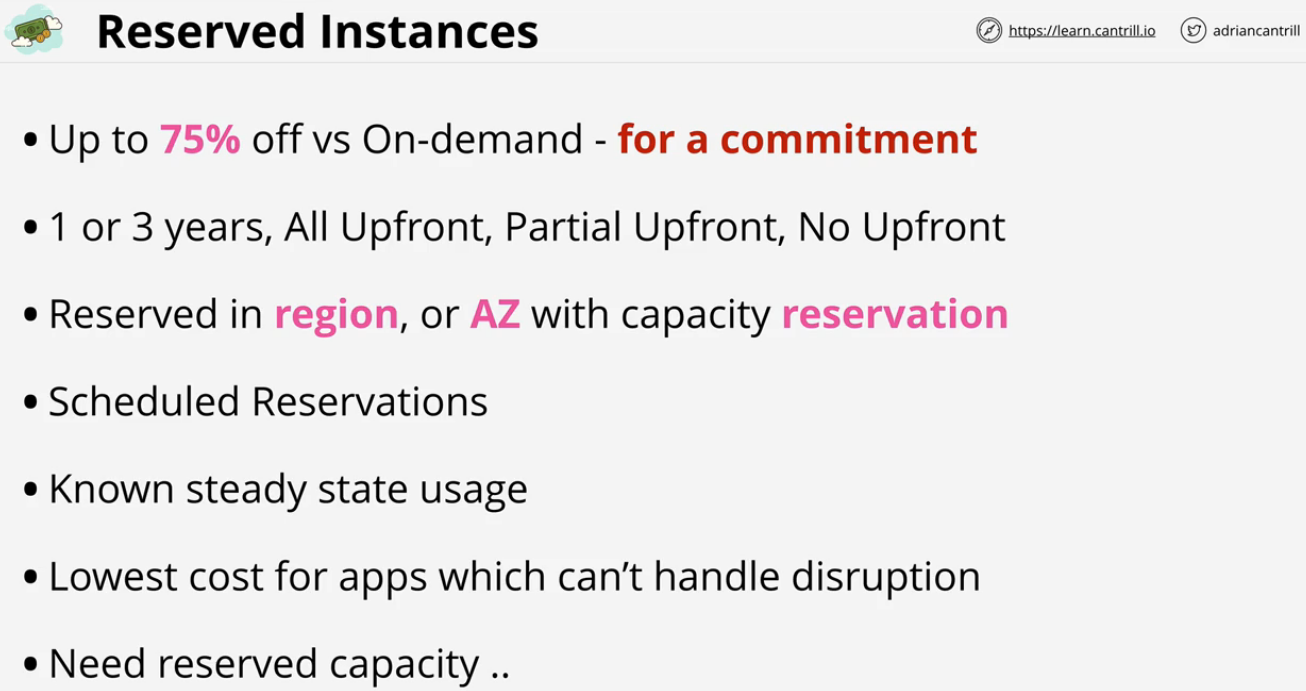
Up to 90% off on-demand, but depends on the spare capacity. You can set a maximum hourly rate in a certain AZ in a certain region. If the max price you set is above the spot price, you pay only that spot price for the duration that you consume that instance. As the spot price increases, you pay more. Once this price increases past your maximum, it will terminate the instance. Great for data analytics when the process can occur later at a lower use time.

**Reserved Instance**

Up to 75% off on-demand. The trade off is commitment. You're buying capacity in advance for 1 or 3 years. Flexibility on how to pay

* All up front
* Partial upfront
* No upfront

Best discounts are for 3 years all up front. Reserved in region, or AZ with capacity reservation. Reserved instances takes priority for AZ capacity. Can perform scheduled reservation when you can commit to specific time windows.

Great if you have a known stead state usage, email usage, domain server. Cheapest option with no tolerance for disruption.

**Instance Status Checks and Autorecovery**

Every instance has two high level status checks

* **System Status Checks**
  + Failure of this check could indicate SW or HW problems of the EC2 service or the host.
* **Instance Status Checks**
  + Specific to the file system or has a corrupted Kernel.

**Auto recovery** can kick in and help:

* Recover this instance
  + can be a number of steps depending on the failure
* Stop this instance
* Terminate this instance
  + useful in a cluster
* Reboot this instance

**Horizontal and Vertical Scaling**

**Vertical Scaling**

As customer load increases, the server may need to grow to handle more data. The server can increase in capacity, but this will require a reboot.

* Often times vertical scaling can only occur during planned outages.
* Larger instances also carry a $ premium compared to smaller instances.
* Instance size is an upper cap on performance.
* No application modification is needed.
  + Works for all applications, even monoliths (all code in one app)

**Horizontal Scaling**

As the customer load increases, this adds additional capacity. Instead of one running copy of an application, you can have multiple versions running on each server. This requires a load balancer. When customers try to access an application, the load balancer ensures the servers get equal parts of the load.

* Sessions are everything.
* With horizontal scaling you can shift between instances equally.
* This requires either application support or off-host sessions.
* Servers are **stateless**, the app stores session data elsewhere.
* No disruption while scaling up or down.
* No real limits to scaling.
* Uses smaller instances so you pay less, allows for better granularity.

**Instance Metadata**

EC2 service provides data to instances Accessible inside all instances

Memorize http://169.254.169.254/latest/meta-data/

Meta-data contains information on the environment the instance is in. You can find out about the networking or user-data among other things. This is not authenticated or encrypted. Anyone who can gain access to the instance can see the meta-data. This can be restricted by local firewall