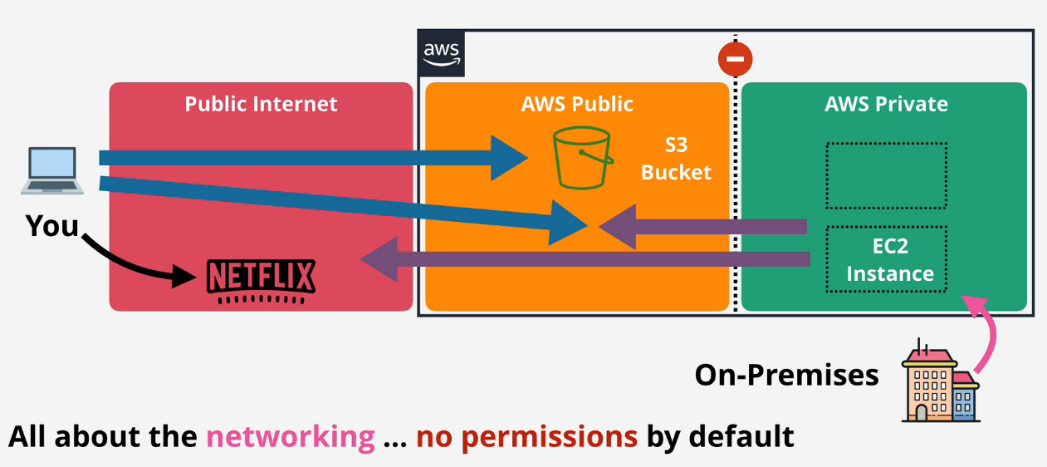
**AWS-Fundamentals**

**AWS Support Plans**

* Basic (free)
* Developer (one user, general guidance)
* Business (multiple users, personal guidance)
* Enterprise (Technical account manager)

**Public vs Private Services**

Refers to the networking only, not permissions.

* **Public Internet:** AWS is a public cloud platform and connected to the public internet. It is not on the public internet, but is next to it.
* **AWS Public Zone: Attached to the Public Internet.** For example-**S3 Bucket is hosted in the Public Zone**, not all services are in the AWS Public Zone. Also Just because you connect to a public service, that does not mean you have permissions to access it.
* **AWS Private Zone: No direct connectivity** is allowed between the **AWS Private Zone and the public cloud** unless this is configured for that service. **Connectivity between AWS Private Zone & Public Cloud** is done by taking a part of the private service and projecting it into the AWS public zone which allows public internet to make inbound or outbound connections.

[**AWS Global Infrastructure**](https://www.infrastructure.aws/)

<https://aws.amazon.com/about-aws/global-infrastructure/>

**Regions**

AWS Region is an area of the world they have selected for a full deployment of AWS infrastructure.

Areas such as countries or states

* Ohio
* California
* Singapore
* Beijing
* London
* Paris

AWS can only deploy regions as fast as their planning allows. Regions are often not near their customers.

**AWS Edge Locations**

Local distribution points. Useful for services such as Netflix so they can store data closer to customers for low latency high speed transfers.

If a customer wants to access data stored in Brisbane, they will stream data from the Sydney Region through an Edge Location hosted in Brisbane.

**AWS Management**

Regions are connected together with high speed networking. Some services such as EC2 need to be selected in a region. Some services are global such as IAM

**Region's 3 Benefits**

* **Geographical Separation**
  + Useful for natural disasters
  + Provide isolated fault domain
  + Regions are 100% isolated
* **Geopolitical Separation**
  + Different laws change how things are accessed
  + Stability from political events
* **Location Control**
  + Tune architecture for performance
  + Duplicate infrastructure at closer points to customers

**Regions and AZs**

Region Name: Asia Pacific (Sydney) Region Code: ap-southeast-2

AWS will provide between 2 and 6 AZs per region. AZs are isolated compute, storage, networking, power, and facilities. Components are allowed to distribute load and resilience by using multiple zones.

AZs are connected to each other with high speed redundant networks.

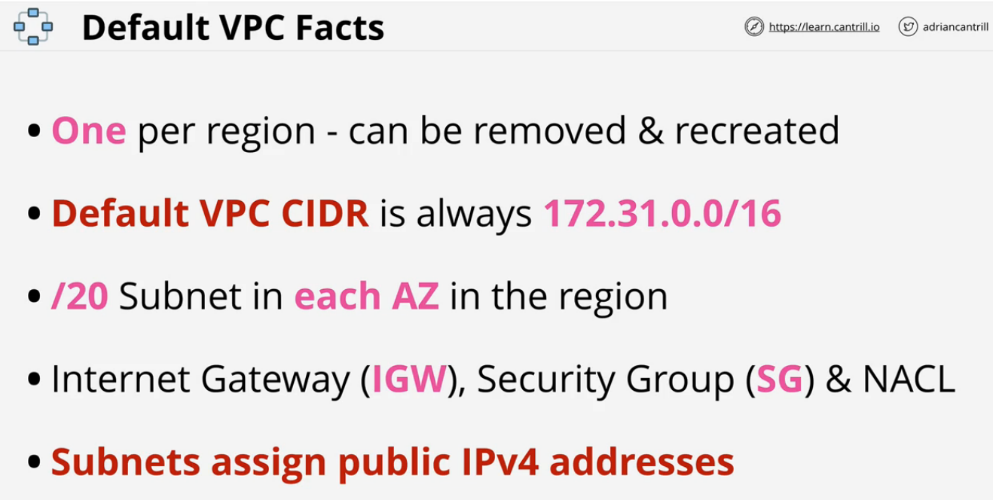
**Service Resilience**

1. **Globally Resilient:** Data is replicated throughout multiple regions. Example: IAM or Route 53. No way for them to go down.
2. **Region Resilient:** Operate as separate services in each region. Generally replicate data to multiple AZs in that region. If an AZ fails the service still runs but if the regions fails it goes down.
3. **AZ Resilient:** Run from a single AZ. If an AZ fails the service fails. It is possible for hardware to fail in an AZ and the service to keep running because of redundant equipment, but should not be relied on.

**AWS Default VPC**

* VPC is a virtual network inside of AWS.
* A VPC is within 1 account and 1 region which makes it **Regionally Resilient**.
* A VPC is **private and isolated** until decided otherwise.
* **One “Default VPC” per region**. Can have many “Custom VPC”(s) which are all private by default.

**Default VPC Facts**

* **Default VPC** can be **one per region**.
* **VPC CIDR** - defines **start and end ranges of the VPC**. IP CIDR of a default VPC is always: **172.31.0.0/16**
* Configured to have **one subnet in each AZ in the region by default**.
* Subnets are given **one section of the IP ranges for the default service**.
* Default VPC is large because it uses the **/16 range.** A subnet is smaller such as **/20.**
* **The higher the / number is, the smaller the grouping.** Two /17's will fit into a /16, sixteen /20 subnets can fit into one /16.
* In general **do not use the Default VPC** in a region **because it is not flexible**.

**Elastic Compute Cloud (EC2)**

* Default compute service.
* Provides access to virtual machines called instances.
* **IaaS** - Infrastructure as a Service
* The unit of consumption is an **instance.**
* EC2 instance is configured to launch into a single VPC subnet. **Private service by default**, public access must be configured. The VPC needs to support public access. If you use a custom VPC then you must handle the networking on your own.
* EC2 deploys into one AZ. If it fails, the instance fails.
* Different sizes and capabilities all use On-Demand Billing - Per second. Only pay for what you consume.
* Charge for running the instance, CPU, memory and storage.
* Extra cost for any commercial software(in form of AMI) the instance deploys with.
* Storage: **Local on-host storage(Instance Store Volumes)** or **Elastic Block Storage(EBS)**

Pricing based on:

* CPU
* Memory
* Storage
* Networking

**Running State**

Charged for all four categories.

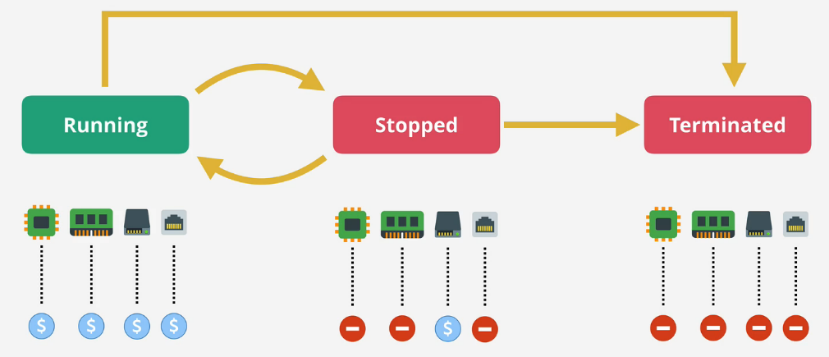
* Running on a physical host using CPU.
* Using memory even with no processing.
* OS is stored on disk allocated
* Networking is always ready to transfer information.

**Stopped State**

Charged for EBS storage only.

* No CPU resources are being consumed
* No memory is being used
* Networking is not running
* Storage is allocated to the instance for the OS.

**Terminated State**

No charges, deletes the disk and prevents all future charges.

**AMI (Server Image)**

AMI can be used to create an instance or created from an instance.

**AMIs in one region are not available from other regions.**

**Contains:**

* **Launch Permissions:** control which accounts can and can't use the AMI.
  + **Public**: Anyone can launch it.
  + **Owner** - Implicit allow, only the owner can use it spin up new instances
  + **Explicit** - owner grants access to AMI for specific AWS accounts
* **Template for the Root Volume:** contain the **Boot Volume**
* **Block Device Mapping:** links the volumes that the AMI has and how they're presented to the operating system. Determines which volume is a boot volume and which volumes is a data volume.

**AMI Types:**

* Amazon Quick Start AMIs
* AWS Marketplace AMIs
* Community AMIs
* Private AMIs

**Connecting to EC2**

* **Windows using RDP** (Remote Desktop Protocol), Port **3389**
* **Linux SSH protocol**, Port **22**

**Login to the instance using an SSH key pair.**

* **Private Key** - Stored on your local machine to initiate connection.
* **Public Key** - AWS places this key on the instance.

**Status Checks while Launching Instances:**

* **System Status Checks:** These checks monitor the AWS systems(EC2 Hosts) required to use this instance and ensure they are functioning properly.
* **Instance Status Checks:** These checks monitor your software and network configuration for this EC2 instance.

**S3 (Default Storage Service)**

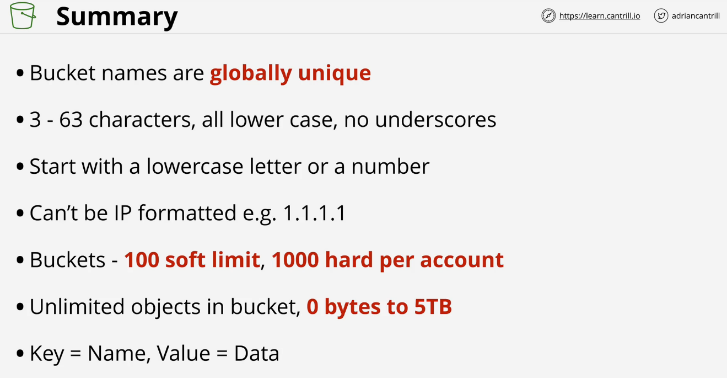
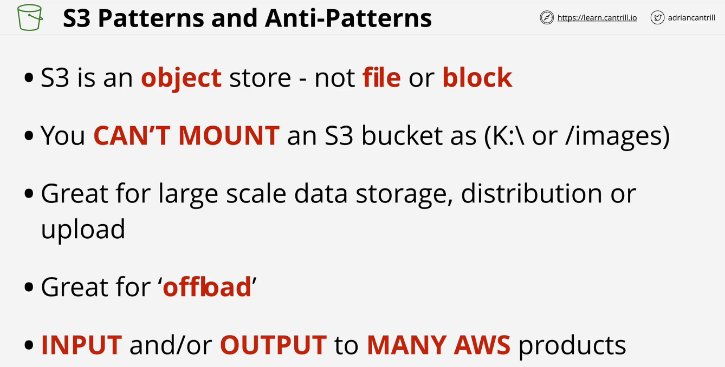
* Global Storage platform. Runs from all regions and is a public service. Can be accessed anywhere from the internet with an unlimited amount of users.
* Regional based/resilient(So data is replicated across AZ within a region)
* This should be the default storage platform
* S3 is an object storage, not file, or block storage. You can't mount an S3 Bucket.

**Objects**

Can be thought of a file. Two main components:

* **Object Key:** File name in a bucket
* **Value:** Data or contents of the object
  + 0 bytes to 5 TB
* **Other components:**
  + Version ID
  + Metadata
  + Access Control
  + Sub resources(ACL)

**Buckets**

* Containers for Objects
* Created in a specific AWS Region.
* Data has a primary home region. Will not leave this region unless told.
* Blast Radius = Region
* Unlimited number of Objects
* Bucket Name is globally unique
* All objects are stored within the bucket at the same level.
* We can specify permissions at the Bucket Level.
* Flat Structure. No concept of folders. If the objects name starts with a slash such as /old/Koala1.jpg the UI will present this as a folder. In actuality this is not true, there are no folders.

**CloudFormation Basics**

Templates can modify infrastructure to, create, update and delete.

Written in YAML or JSON

## This is not mandatory unless a description is added

**AWSTemplateFormatVersion:** "version date"

**## Give details as to what this template does.**

**## If you use this section, it MUST immediately follow the AWSTemplateFormatVersion.**

**Description:**

A sample template

## Can control the command line UI. The bigger your template, the more likely

## this section is needed

**Metadata:**

template metadata

## Prompt the user for more data. Name of something, size of instance,

## data validation

**Parameters:**

set of parameters

## Another optional section. Allows lookup tables, not used often

**Mappings:**

set of mappings

## Decision making in the template. Things will only occur if a condition is met.

## Step 1: create condition

## Step 2: use the condition to do something else in the template

**Conditions:**

set of conditions

**Transform:**

set of transforms

## The only mandatory field of this section

**Resources:**

set of resources

## Once the template is finished it can return data or information.

## Could return the admin or setup address of a word press blog.

**Outputs:**

set of outputs

**Resources**

An example which creates an EC2 instance

Resources:

Instance: ## Logical Resource

Type: 'AWS::EC2::Instance' ## This is what will be created

Properties: ## Configure the resources in a particular way

ImageId: !Ref LatestAmiId

Instance Type: !Ref Instance Type

KeyName: !Ref Keyname

* Once a template is created, AWS will make a stack. This is a living and active representation of a template. One template can create infinite amount of stacks.
* For any **Logical Resources** in the stack, CF will make a corresponding **Physical Resources** in your AWS account.
* It is cloud formations job to keep the logical and physical resources in sync.
* A template can be updated and then used to update the same stack.

**CloudWatch Basics**

Collects and manages operational data on your behalf.

Three products in one

* **CloudWatch Metrics:** Data relating to AWS products, apps, on-premises solutions. Collection of metrics, monitoring of metrics and actions based on metrics. Data gathering inside or outside AWS. Example- CPU Utilization, Number of visitors to your website, **Memory Utilization(Cloudwatch Agent is required)**
* **CloudWatch Logs:** collection, monitoring & actions based on logs. Example -Windows Server logs, Linux Logs, Firewall logs etc. Cloudwatch Agent is required for anything which is not visible within cloud watch.
* **CloudWatch Events:** event hub
  + **AWS Services:** If an AWS service does something for example EC2 instance is terminated , CW events can generate an event perform another action
  + **Schedules:** Generate an event to do something at a certain time of day or time of week.

**Namespace**

**Container for monitoring data.** Naming can be anything so long as it's not “AWS/service” such as AWS/EC2. This is used for all metric data of that service. Name spaces contain related metrics.

**Metric**

Time ordered set of data points such as:

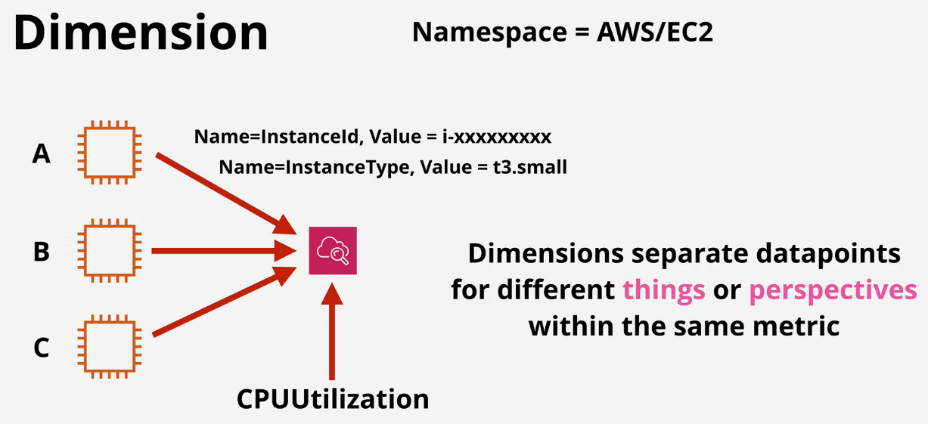
* CPU Usage
* Network IN/OUT
* Disk IO

This is not for a specific server. This could get things from different servers.

Anytime CPU Utilization is reported, the **datapoint** will have 2 parts, for example it reports

* **Timestamp** = 2019-12-03
* **Value** = 98.3

**Dimensions** separate data points for different **things** or **perspectives** within the same metric

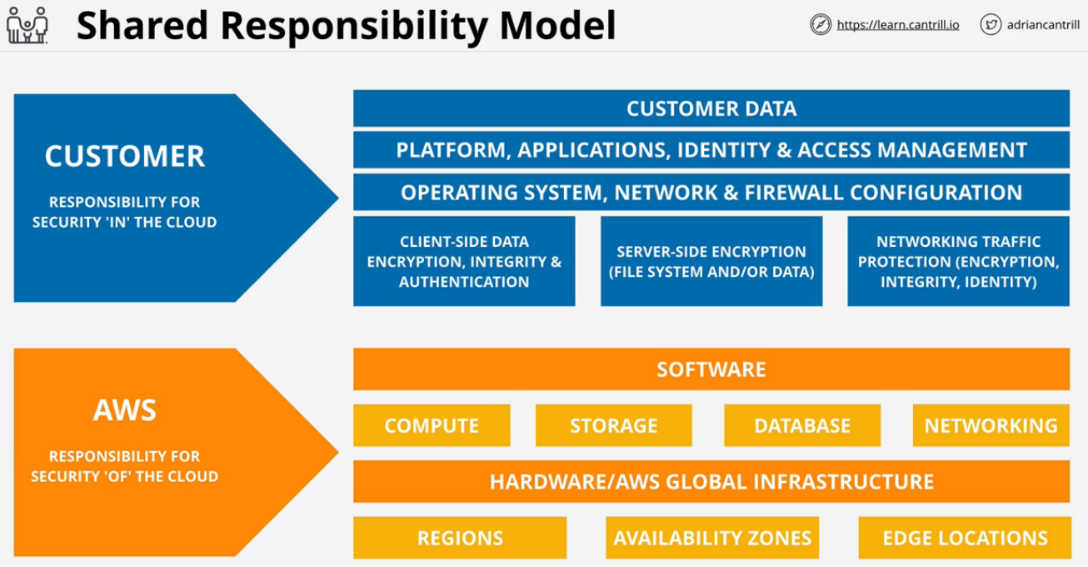


**Alarms**

Has two states **“ok”** or **“alarm”**. **State** can send an **SNS** or **action**. Third state can be **“insufficient data”** state. Not a problem, just wait.

**Shared Responsibility Model:**

* AWS: Responsible for security **OF** the cloud
* Customer: Responsible for security **IN** the cloud

**High Availability (HA), Fault-Tolerance (FT), and Disaster Recover (DR)**

**High Availability (HA)**

* Aims to **ensure** an agreed level of operational **performance**, usually **uptime**, for a **higher than normal period**
* Instead of diagnosing the issue, swap it out.
* Redundant hardware to minimize downtime
* User disruption is not ideal, but is allowed
  + The user might need to log back in or lose some data on their screen.
* Maximizing a system's uptime
  + 99.9% (Three 9's) = 8.7 hours downtime per year.
  + 99.999 (Five 9's) = 5.26 minutes downtime per year.

**Fault-Tolerance (FT)**

* System can **continue operating properly** in the event of the **failure of some** (one or more faults within) of its **components**
* Fault tolerance is much more complicated than high availability and more expensive. Outages must be minimized and the system needs levels of redundancy.
* An airplane is an example of system that needs Fault Tolerance. It has more engines than it needs for redundancy.

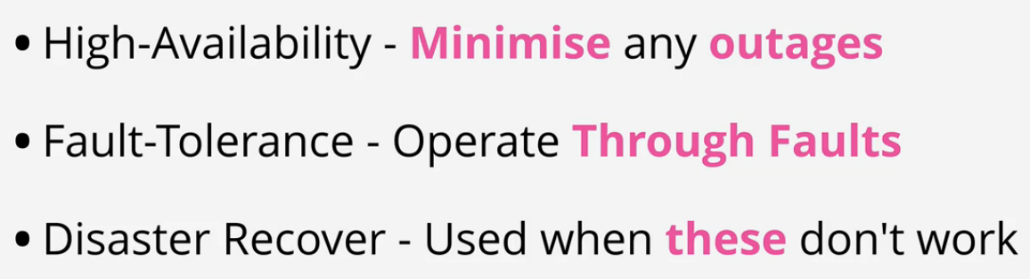
Example: A patient is waiting for a lifesaving surgery and is under anesthetic. While being monitored, the life support system is dosing medicine. This type of system cannot only be highly available, even a movement of interruption is deadly.

**Disaster Recover (DR)**

* Set of policies, tools and procedures to **enable the recovery** or **continuation** of **vital** technology infrastructure and systems **following a natural or human-induced disaster**.
* DR can largely be automated to eliminate the time for recovery and errors.

This involves:

* Pre-planning
  + Ensure plans are in place for extra hardware
  + Do not store backups at the same site as the system
* DR Processes
  + Cloud infrastructure is ready when needed

This is designed to keep the crucial and non-replaceable parts of the system in place.

**Domain Name System (DNS)**

DNS is a discovery service. Translates domain names into IP Address and visa versa. It is a huge database and has to be distributed.

**Parts of the DNS system**

* **DNS Client:** Piece of software running on the OS for a device you're using.
* **DNS Resolver:** Software on your device or server which queries DNS on your behalf.
* **DNS Zone:** A part of the DNS database.
  + This would be [www.amazon.com](http://www.amazon.com/)
  + What the data is, the substance
* **Zonefile:** physical database for a zone
  + How physically that data is stored
* **Nameserver:** where zonefiles are hosted

**Steps:**

Find the Nameserver which hosts a particular Zonefile. Query that Nameserver for a record with that Zone. It then passes the information back to the client.

**DNS Root**

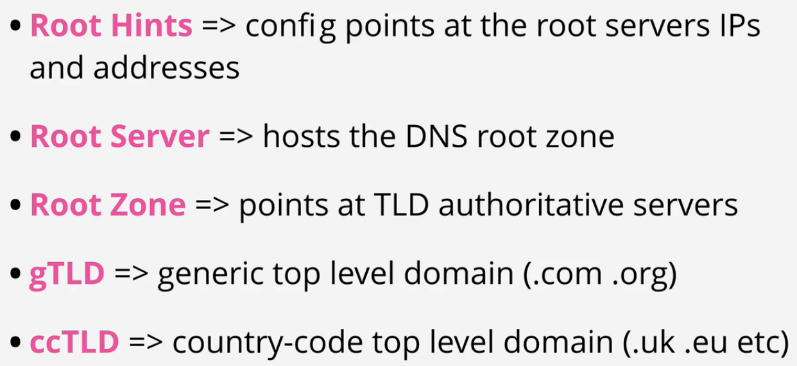
The starting point of DNS. DNS names are read right to left with multiple parts separated by periods.

www.netflix.com.

The (.) is assumed to be there in a browser when it's not present. The DNS Root is hosted on DNS Root Servers (13). These are hosted by 12 major companies.

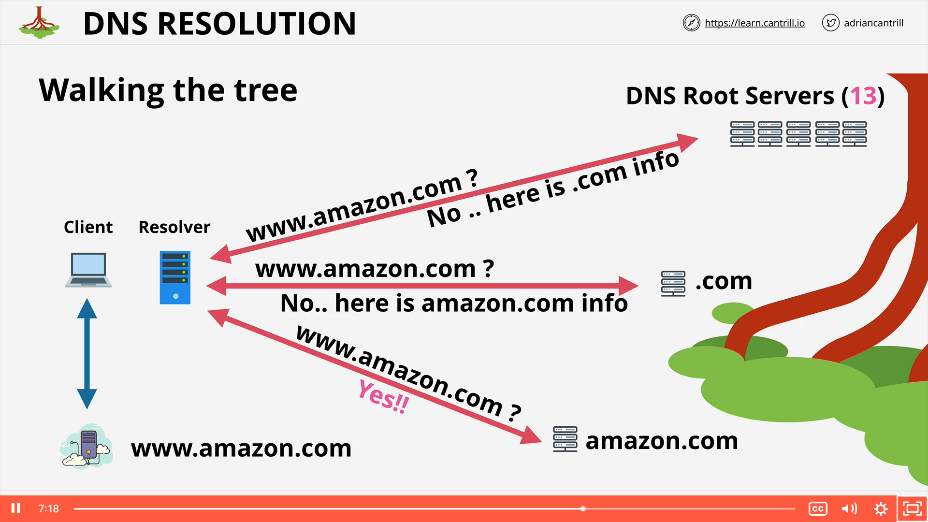
**Root Hints** is a pointer to the DNS Root server. Generally Provided by the OS Vender.

**Process**

1. DNS client asks DNS Resolver for IP address of a given DNS name.
2. Using the **Root Hints** file, the DNS Resolver communicates with one or more of the root servers to access the **Root Zone** and begin the process of finding the IP address.
3. The Root Zone is organized by IANA (Internet Assigned Numbers Authority). Their job is to manage the contents of the root zone. IANA is in charge of the DNS system because they control the root zone.

**DNS Hierarchy**

Assuming a laptop is querying DNS directly for [www.amazon.com](http://www.amazon.com/) and using a root hints file to know how to access a root server and query the root zone.

* When something is trusted in DNS, it is an **authority**.
* One piece can be authoritative for root.
* One piece can be authoritative for amazon.com
* The root zone is the start and the only thing trusted in DNS.
* The root zone can delegate a part of itself to another zone or entity.
* That someone else then becomes authoritative for that piece of itself only.
* The root zone is just a database of the top level domains.

The top level domain(TLD)s are the only things to the left of the DNS name.

* .com or .org are generic top level domains (GTLD)
* .uk is a country code top level domains (CCTLD)

**Registry** maintains the zones for a TLD (e.g .ORG)

**Registrar** has relationships with the .org TLD zone manager allowing domain registration

**Route53 Fundamentals**

* Registers domains
* Can Host Zone Files on managed nameservers
* This is a global service, no need to pick a region
* Globally Resilience
  + Can operate with failure in one or more regions

**Register Domains**

Has relationships with all major registries

* Route 53 will check with the top level domain to see if the name is available
* Route 53 creates a zonefile for the domain to be registered
* Allocates name severs for that zone
  + Generally four of these for one individual zone
  + This is a hosted zone
  + The zone file will be put on these four managed nameservers
* Router 53 will communicate with the .org registry and add the nameserver records into the zone file for the top level domain.
  + This is done with a nameserver record.

**Route53 Details**

**Zonefiles** in AWS Hosted on four managed name servers

* Can be **public** or **private**

**DNS Record**

* **Nameserver (NS):** Allows delegation to occur in the DNS.
* **A and AAAA Records:** Maps the host to a v4 or v6 host type. Most of the time you will make both types of record, A and AAAA.
* **CNAME Record Type:** Allows DNS shortcuts to reduce admin overhead. CNAMES cannot point directly at an IP address and only another name.
* **MX records:** How emails are sent. They have two main parts:
  + **Priority:** Lower values for the priority field are higher priority.
  + **Value**
    - If it is just a host, it will not have a dot on the right. It is assumed to be part of the same zone as the host.
    - If you include a dot on the right, it is a ***fully qualified domain name***
* **TXT Record:** Allows you to add arbitrary text to a domain. One common usage is to prove **domain ownership.**

**TTL - Time To Live**

This is a numeric setting on DNS records in seconds. Allows the admin to specify how long the query can be stored at the resolver server. If you need to upgrade the records, it is smart to lower the TTL value first.

Getting the answer from an Authoritative Source is known as an **Authoritative Answer**.

If another client queries the same thing, they will get back a **Non-Authoritative** response.