IAM:

* STS API Calls: credentials are returned with the following components
  + Session Token
  + Access Key ID
  + Secret Access Key
  + Expiration

Web Identity Federation: Authenticate to AWS from third-party providers such as facebook or google.

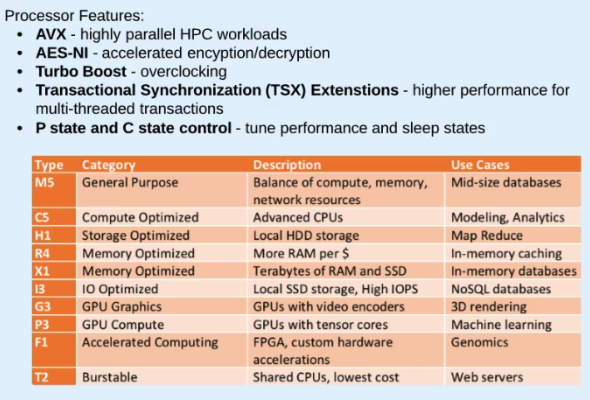
Server-Based Compute Services

# EC2:

Types of Virtualization for Linux instances

* HVM AMIs (Hardware Virtual Machines (recommended standard)
  + Runs like bare metal hardware
  + Get hardware extensions (works like normally do in datacenter)
* PV AMIs (Paravirtual)
  + Host hardware that does not have explicit support for virtualization
  + No special hardware extensions (enhanced n etworking or GPU processing)

# Instance Types:



User data command on instance:

* Curl http<nolink>://169.254.169.254/latest/user-data

Public IP

* Static public IP = Elastic IP Address
  + When you stop and start instance: keep same public IP
  + When you terminate instance: only then will the public IP get put back into the pool
  + Have to attach elastic IP address and it will replace the default public IP assigned
* Public IP Address: when you stop the instance AWS takes back public address

EBS Snapshots: stored in S3

* Can create AMI (image) for an instance from a snapshot
* Create a volume from a snapshot
* Network attached volumes (persistent volumes, they can live past the life of an EC2 instance)

EFS (Elastic File System)

* Shared file system for EC2
* Elastic (thin provisioned)
* Only compatible with Linux Servers \*\*\*\*\*\*\*\*
* Can mount to on prem servers

Placement Groups: (NO EXTRA CHARGE FOR THIS)

* Cluster Placement Group
  + Cluster of instances (AWS DRS)
  + Example: Can put all instances on one host in datacenter or physically close as possible to reduce latency and maximize network throughput
  + Use Enhanced networking to maximize benefit
  + Not supported for T2 Instances
  + Best to launce all instances needed at ONE TIME
* Spread Placement Group
  + Place instances on distinct host hardware (spread it out amongst hardware)

Billing:

* On demand –
  + most expensive for long running instances
  + Most flexible
  + Charged per second or per hour
* Reserved: set time period of 1 – 3 years
  + Pay upfront, partial upfront, no upfront
    - Standard (up to 75% discount)
      * Capacity reservation in an AZ
    - Convertible (up to 54% discount)
      * Can change/upgrade policy
    - Scheduled (1yr term, 5-10% discount)
      * Schedule exact times when the instance can run/launch

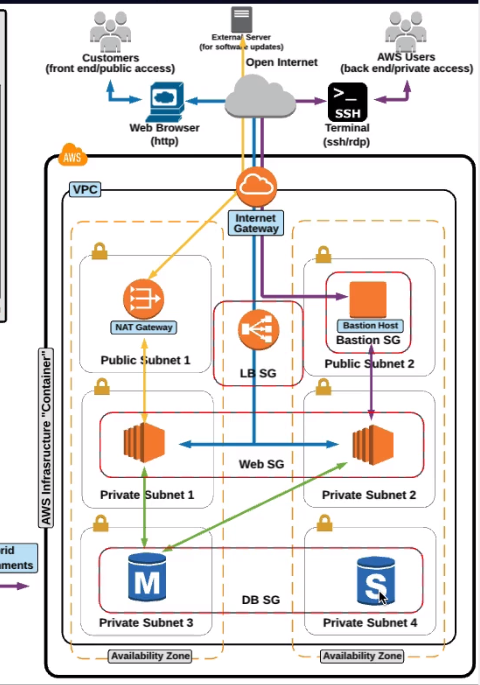
VPC (Virtual Private Cloud)

* If you do not specify a subnet when creating an instance it will be created in the default VPC and subnet

High Availability

Elastic Load Balancing

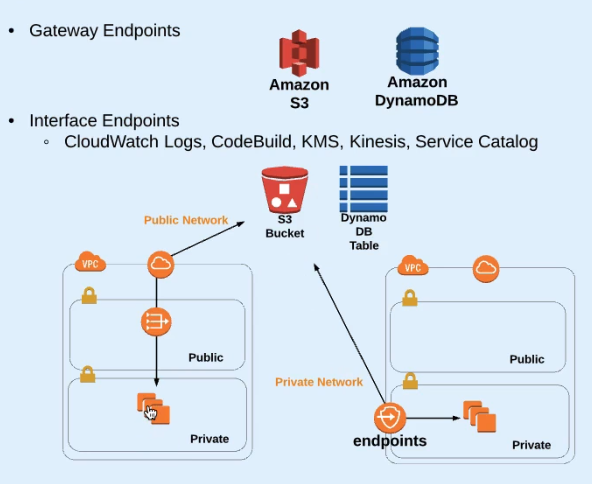
* EC2 instances being launched behind the scenes to load balance traffic coming in from the interweb
* Three types of ELB
  + Classic load balancer
    - Simple balancing of traffic
    - Support TCP, SSL, HTTP, HTTPS
    - No special routing requests
  + Application ELB
    - Content-based “rules”
    - Route based on HTTP headers
    - Target groups (groups of EC2 instances)
    - Supports ECS (Elastic container service), EKS (kubernetes), HTTPS, HTTP/2, WebSockets, Sticky Sessions, WAF
  + Network LB
    - Layer 4 TCP/UDP load balancing
    - Designed for extreme performance
      * Does not need to scale, already designed to handle huge traffic spikes
    - No SSL offloading
    - No instances just a software designed load balancer
      * Just passes traffic
    - Multiple IP addresses, must point to DNS that has all IP addresses associated with DNS name
* Bastion Host: EC2 instance that lives in a public subnet and is used as a gateway for traffic that is destined for instances in private subnets. Admins use for SSH/RDP access from web
  + All traffic must pass through it first from the web
  + Used for Administrators/users to access the servers when a Load balancer is present (all traffic is traveling through this load balancer and offloading ssl so admins can’t get to the instances behind the LB



* NAT Gateway
  + For traffic to get a route from private subnet to the internet
  + NAT Instance
    - Identical to NAT Gateway except it is executed differently by configuring an actual EC2 instance
    - More of a legacy feature in AWS

VPC Endpoints:

* Used when you don’t want certain data to travel from private subnets over the internet (needs to stay off the web)
  + Essentially adds a route for the vpc to the service you want so that it bypasses the internet gateway and you use internal IP addresses
* Gateway Endpoints
  + S3
  + DynamoDB
* Interface Endpoints
  + CloudWatch Logs
  + CodeBuild
  + KMS
  + Kinesis
  + Service Catalog



Stateless Applications: (Store state information off-instance)

* Stores data from a session to a shared common database table (no SQL database)
  + Instead of saving data locally to the instances that the load balancer sends the traffic too
* Shared Filesystem
* All instances point to a common database so no matter what instance it hits, it can see all the data

HA vs Fault Tolerance

* HA – architectures that continue to remain available to end users in the event of a systems or component failure
* Fault Tolerance – architectures that not only remain available during an outage but also suffer no degradation in performance during the outage
* Scalability – ability to easily increase or decrease in size and capacity
  + Vertical scaling – increase capacity of single instance
  + Horizontal scaling – add or remove number of instances
* Elasticity – ease of a systems ability to change or adapt
  + Scaling in or out
  + Updating firewall rules
  + Remapping IP addresses

AutoScaling:

* Automatically increase or decrease instances available for your application
* Based on metrics from Cloudwatch

DNS – Route 53

Hosted Zones – records for your domain

Failover routing policy –

* Using S3 for DNS failover
  + Used as a primary endpoint as a static webpage
* Setup routing Policy to “Failover” and set Primary as Load balancer and secondary as S3 bucket containing website

Cloud Front: Content Delivery Network

* Web Distribution:
  + Used for distribution of static and dynamic content
    - .html
    - .css
    - .php and graphic files
    - Media files HTTP or HTTPS
    - Web forms
    - Live streaming
* RTMP: streaming media files using Adobe flash media servers
  + Must be stored in an S3 bucket
* Accelerate delivery of content to users by caching content at edge locations and will pull the content based off of geo-location from the closest cached edge location
* Origin: the source of content
  + S3 Bucket
  + ELB
  + Public HTTP endpoint
  + Video streaming
* Large objects at the origin, you want a longer TTL so it isn’t always trying to move the large files as much
* Query strings and TTL need to be set based off of usage, size, and variables changing

Databases

* Relation (SQL) databases (Amazon RDS and Datawarehouse (RedShift)
  + Data is normalized and spread across multiple tables
  + SQL Query Language
  + Vertical scaling and read replicas
  + Examples: Oracle, SQL, MySQL, PostgrSQL
* NoSQL Databases
  + Semi-structured data
  + Flat table structure
  + API queries
  + Horizontal Scaling
  + Examples: MongoDB, CouchDB, HBase, Cassandra
  + Four Types:
    - Column: instead of storing data in rows it stores them by column
      * Fast querying
      * HBase or EMR
    - Key-Value: simple.. store data as Key=value
      * Amazon ElastiCache
    - Document: storing json or xml documents
      * Amazon DynamoDB
    - Graph: social media applications use (optimized for relationships)
      * Amazon Neptune
* Managed Database services
  + Pros: no overhead or maintenance, highly available, automatic failover
* Cons: Don’t get all tuning features that a DBA might require

Amazon RDS:

* Amazon’s Relational Database Service
* Automatic Snapshots can only be retained for up to 35 days
* License is already included in the hourly fee
* Provision/resize hardware on demand for scaling
* Multi-AZ deployment for backup and HA
  + Two identical copies (primary, standby)
  + Automatic Failover
* Supported DB’s
  + MySQL
  + MariaDB
  + PostgreSQL
  + Oracle
  + MS SQL Server
  + Aurora (Amazon’s home grown DB)
    - Cheaper
    - Continuous backups
    - Backtrack (can roll back database in seconds to point in time)
  + Aurora Serverless
    - Autoscaling
    - No management what so ever (scales in the background)
    - Pay as you go (Storage, I/O)

DynamoDB:

* Stateless (noSQL)
  + Runs on a cluster of nodes and can scale in and out
  + Schema-less
    - Every item in a table can have different attributes
* Easier to scale behind a load balancer
* Good for a flat table structure (json documents or key values)
  + Storing meta data
  + Gaming (storing session information, leaderboards)
  + Mobile (user profiles, personalization)
* Not a good solution for multiple tables joining together (relational database)

Neptune (noSQL)

* Not Server-less
* Graph Database (relationships type database)
  + Used for social networking platforms
  + Knowledge graphs
  + Fraud detection

ElastiCache:

* Improves database performance by caching results of queries that made to a database
* Provisions node clusters which you can scale
* Offloads databases
  + When query is run, it checks the cache before it checks the database
  + If it is not in cache and query hits database, it will cache that query in cache for the next time
* Possible Elasticache engines
  + Memcached
    - Simple
    - Easy to scale
    - multithreaded
  + Redis
    - More complex data types
    - Multi-AZ failover
    - Saved on disk and memory

Redshift

* Data Warehouse Database
* Built from PostgreSQL
  + PostgreSQL compatible
* Spectrum: query against an S3 bucket

S3/Glacier

S3:

* Distributed storage system (in multiple AZ)
  + Data is replicated out once added to bucket
* 11 9’s availability uptime
* S3 Objects
  + Encryption
    - SSE (Server Side Encryption)
      * Encrypted in S3
    - Client Side Encryption
      * Application encrypt the data before sending it into S3
    - SSL terminated endpoints
      * Encrypted in transit as it goes over the network
* S3 Folders
  + S3 is not a file system so these are more like prefixes than folders
* Versioning
  + Applied at the bucket level
  + Puts a delete marker on the object and you can un delete it
  + Cross-region replication
    - Creates another bucket in another region that is replicated over too
* Storage Classes
  + Standard
    - Best availability, general all-purpose storage
    - Eleven nines uptime
    - Most expensive
  + Infrequent Access (S3-IA)
    - Used for objects that you don’t frequently access
    - Three nines uptime
    - Storage costs drop
    - Access fees are charged more
  + One Zone Infrequent access
    - 99.5% uptime
    - Much lower pricing
    - 30 day minimum
    - Keeps all data in a single AZ
  + Glacier
    - Archiving
    - Cheapest pricing
    - 90 day minimum
    - Offline Cold storage (data retrieval times
      * Expedited: 1-5 minutes (most expensive)
      * Standard: 3-5 hours
      * Bulk: 5-12 hours (least expensive)
* Lifecycle Polices = retention policy
  + Can move data to different policies based on days old
* Hosting Websites with S3
  + Static Web Hosting
  + Make public bucket and add HTML, CSS, javaScript files
  + Point Route 53 to the bucket
  + Cross-Origin Resource Sharing
    - Allows website to be accessed from any browser

Adding Large files into S3

* Multipart upload
  + Uploads a single object as a set of parts
  + Uploads objects up to 5TB
* S3 Transfer Acceleration
  + Uploads data through cloud Front (must faster)
* Snowball
  + Petabyte-scale data transport solution
    - Up to 80TB per device
  + They send you a device to take to your datacenter and plug directly into your environment and move data directly (bypassing network transfer)
* Snowmobile
  + Can hold up to 100PTB of data
* Storage Gateway
  + Connects local datacenter software appliances to cloud based storage
  + VMware or Hyper-V
* Volume gateway
  + EBS snapshots for DR (holds snapshots in AWS so if your datacenter goes down you can set up EBS snapshots as ec2 instances while datacenter is down
  + Gateway-Cached Volumes
    - Storage volumes mount as iSCSI
    - Gateway stores data written in S3 and caches frequently access data
  + Gateway-stored Volumes
    - Stores all data locally in storage volumes
    - Periodically take snapshots of the data and store in S3
* File Gateway
  + Local NFS
  + Objects stored and retrievable in S3
* Tape Gateway
  + Tape library solution in S3

Hybrid Environments (on prem and cloud environment)

Direct Connect: (dedicated private connection) (mpls fiber cable) (direct connection to AWS network)

* Faster and more secure than VPN
* 1gbps up to 10gbps connection
* Cheaper costs to transfer data over line
* Private Virtual interface
  + Allows you to interface with AWS (VPC)
  + Uses BGP for route discovery
* Public Virtual Interface
  + Let you connect to public aws endpoints using private line

Connect datacenter via IPSec VPN

VPC Peering

* Two VPC’s to communicate
* Can’t have overlapping CIDR ranges

Serverless

         Microservices

o   Communicate via API’s

Lambda

         Scales automatically

         Upload code to service

         Only pay for compute time when function/app runs

         When to use over EC2

o   Run code that is in response to events such as:

  Changes to S3 Buckets

  Updates to Dynamo DB and Arurora DB tables

  Custom events generated by your applications or devices

  Kinesis Streams

         To increase the CPU in Lambda, you must increase the RAM and it will increase the CPU along with it

VPC Flow Logs

* Located in CloudWatch (VPC Flow Logs)
* Collects information about IP traffic going to and from network interfaces in your VPC
  + Source : Destination : Source Port : Dest Port : Action
* To Enable
  + Go to VPC – Actions – Create Flow Log (create role to have access to create flow logs)

Acces Logs (Delivered to S3 bucket)

* ELB Logs (http requests/loadbalancer traffic)
* CloudFront Logs (Edge location and client IP traffic hitting edge services in Cloud Front)
* S3 Access Logs (access to a bucket)

Kinesis Essentials

        Real time processing of big data

        Data from multiple sources can be processed and streamed out to many different AWS service

        A kinesis stream can support multiple instances/databases at one time

        Data Streams Components

o   Contains one or more shards

  Shars (processing power) 1MB/sec data input and 2MB/sec data output

o   Producers (data creators)

  Anything that sends data to Kinesis

o   Consumers (data consumers)

  Applications that consumes the streams data (EC2 instance, database, Lambda, etc..)

        Benefits

o   Real-time processing

o   Parallel Processing

o   Durable

  Synchronously replicates data across three datacenters within AWS region for up to 7 days retention

o   Scales automatically

        When to use

o   Gaming (collecting player actions and real time actions of players)

o   Real-time analytics

  IOT (Sensors) high amounts of frequency

o   Application alerts

o   Log/Event data collection

o   Mobile data capture (mobile applications for real time dashboards)

Kinesis Video Streams

        Stream video to AWS

        Real-time or batch video processing and analytics

Kinesis Data Streams

        Ingest data from many sources

        Real time data processing applications

        Server side encryption

Kinesis Firehouse

        Load streaming data to S3, Redshift, Splunk, Elasticsearch

Kinesis Data Analytics

        Run SQL queries against data streams or firehouse

        Send results to output Data Stream or Firehouse

Elastic MapReduce:

        Simplifies running big data frameworks

        Processing of data in large batches

        Advantages and Features

o   S3 for storage

o   Transient clusters

  (provide set of steps and task you want cluster to do then it will terminate itself)

  Spins up to do its job then terminates

o   Spot instances

  Saves money

o   Bootstrapping

  Customized processing during spin up

o   Scalability

o   Preconfigured application frameworks

Pillars of Excellence (best ways to architect the environment for best results)

* AWS Best practices

AWS Config:

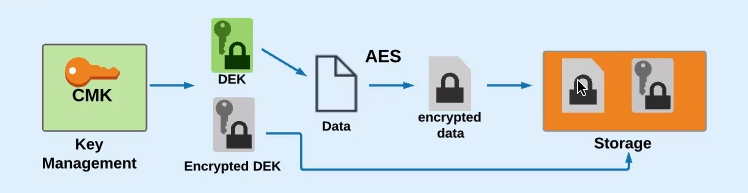
* Gives you exactly what is running in your AWS environments
* Visibility into resource configuration
  + Snapshots of resource inventory
  + Delivers reports to S3
  + Historical Timelines
  + Relationships between resources
* Sends out notifications whenever a configuration change is made
* Evaluates compliance/security

**Reliability**:

* Ability to recover from failure and mitigate disruptions
* Chaos engineering
* Trusted Advisor (best to run at first in a new AWS environment)
  + Generates recommendations for:
    - Cost Optimization
    - Performance
    - Security
    - Fault Tolerance
    - Service Limits

Encryption on AWS

* Amazon KMS
  + Master keys never leave KMS
  + Integrated with database services, S3, Kinesis, EBS, etc..
  + Server-side encryption



* Cloud HSM
  + Single tenant hardware (dedicated key management hardware)
  + Needed when compliance requires it
    - Federal Information Processing Standard (FIPS) 140-2
    - Common Criteria EAL4+ standards
  + AWS manages firmware, updates, automatic backups

RTO (Recovery Time Objective) – How long it takes to recover

RPO (Recovery Time Objective) – How much data is lost during down time

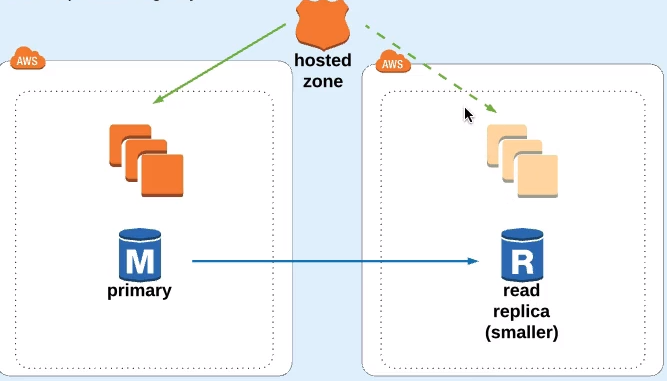
Backup and Restore:

* Backup data to AWS or second region
* Have AMIs in recovery region
* Keep CloudFormation templates standing by to spin up back up environment in case of disaster
* Script everything
* Have Route 53 (DNS) ready to flip over

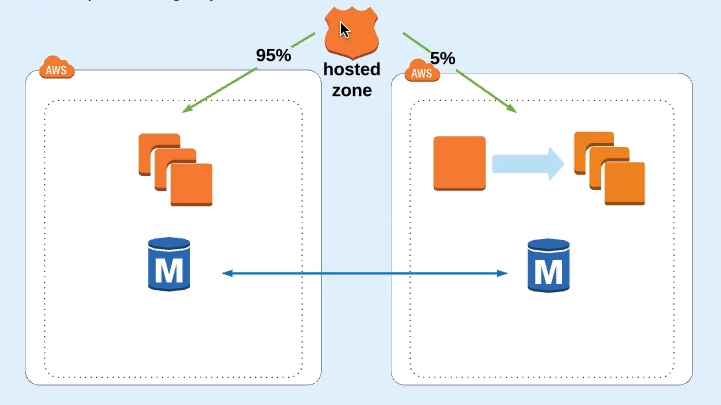
Cross Region Replication

RDS, DynamoDB, S3

Pilot Light Standby



Low Capacity Standby



Multisite Active:Active

