

# Disclaimer: These slides are copyrighted and strictly for personal use only

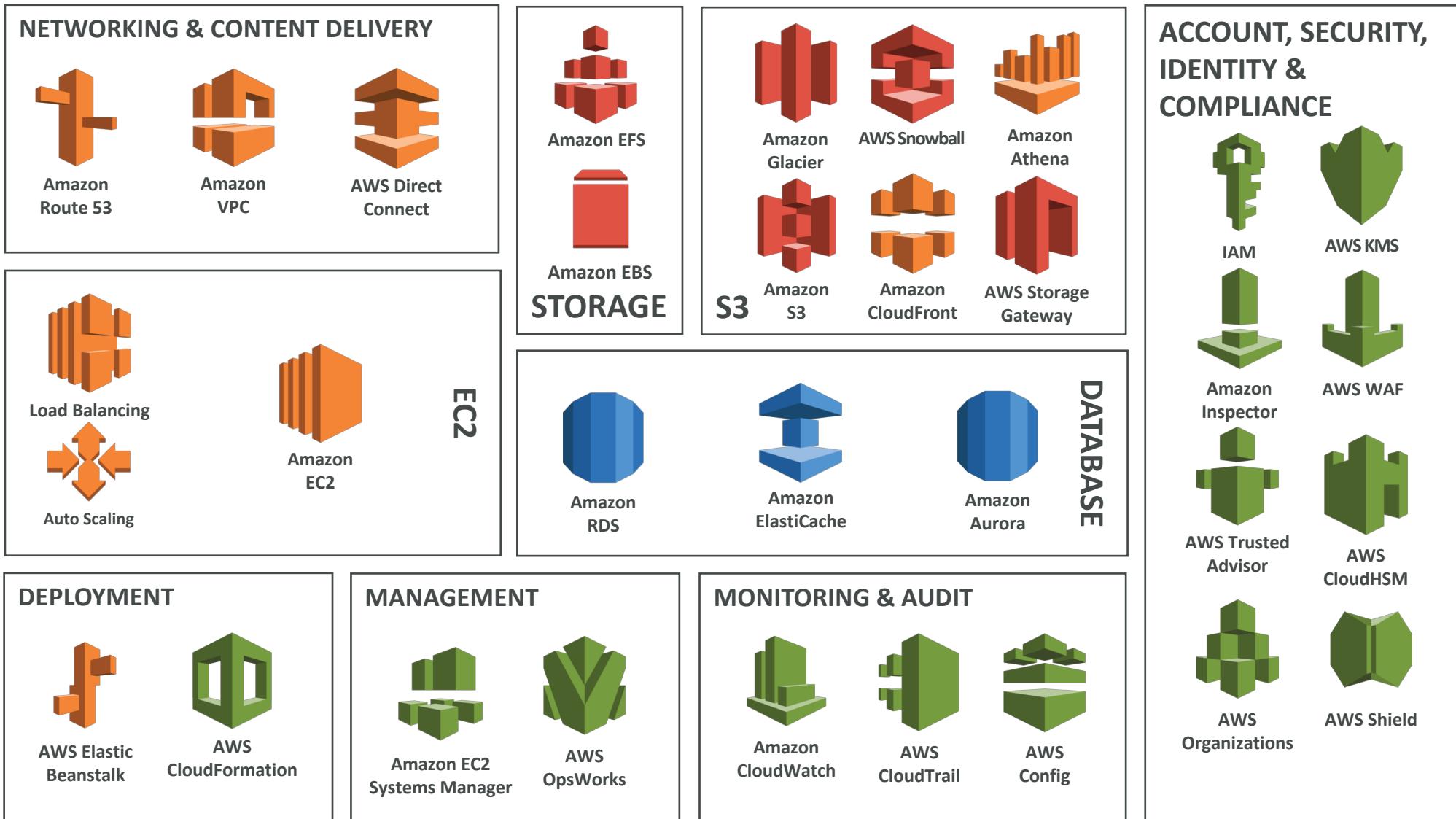
- This document is reserved for people enrolled into the [AWS Certified SysOps Administrator Associate course by Stephane Maarek.](#)
- Please do not share this document, it is intended for personal use and exam preparation only, thank you.
- If you've obtained these slides for free on a website that is not the course's website, please reach out to [piracy@datacumulus.com](mailto:piracy@datacumulus.com). Thanks!
- Best of luck for the exam and happy learning!

# AWS Certified SysOps Administrator Associate Course

## SOA-C01

# Welcome!

- We're going to prepare for the SysOps exam
- It's a challenging certification, so this course will be long and interesting 😊
- Advanced course – requirements:
  - AWS Solutions Architect Or AWS Developer certification
  - Significant Practice with AWS
  - AWS Account already setup
  - CLI knowledge + setup
  - Basics: EC2, S3, RDS, CloudWatch, Security Groups, how to SSH / Putty
- We will revisit most concepts but from a SysOps exam perspective



# My certification: 98.0%

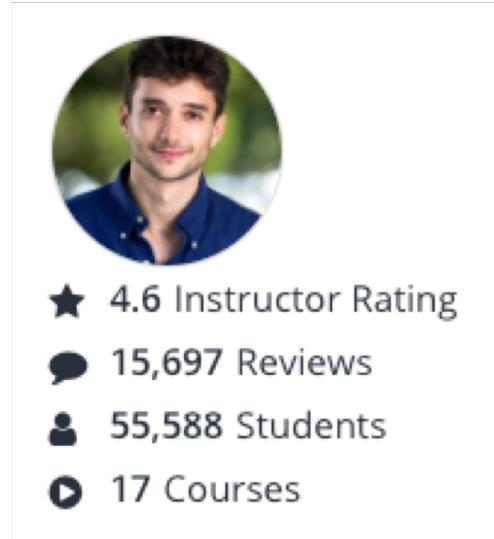
## AWS Certified SysOps Administrator - Associate

### Notice of Exam Results

Candidate: Stephane Maarek	Exam Date: December 11, 2018
Candidate ID: AWS00614912	Registration Number: 446741
Candidate Score: 980	Pass/Fail: PASS

# About me

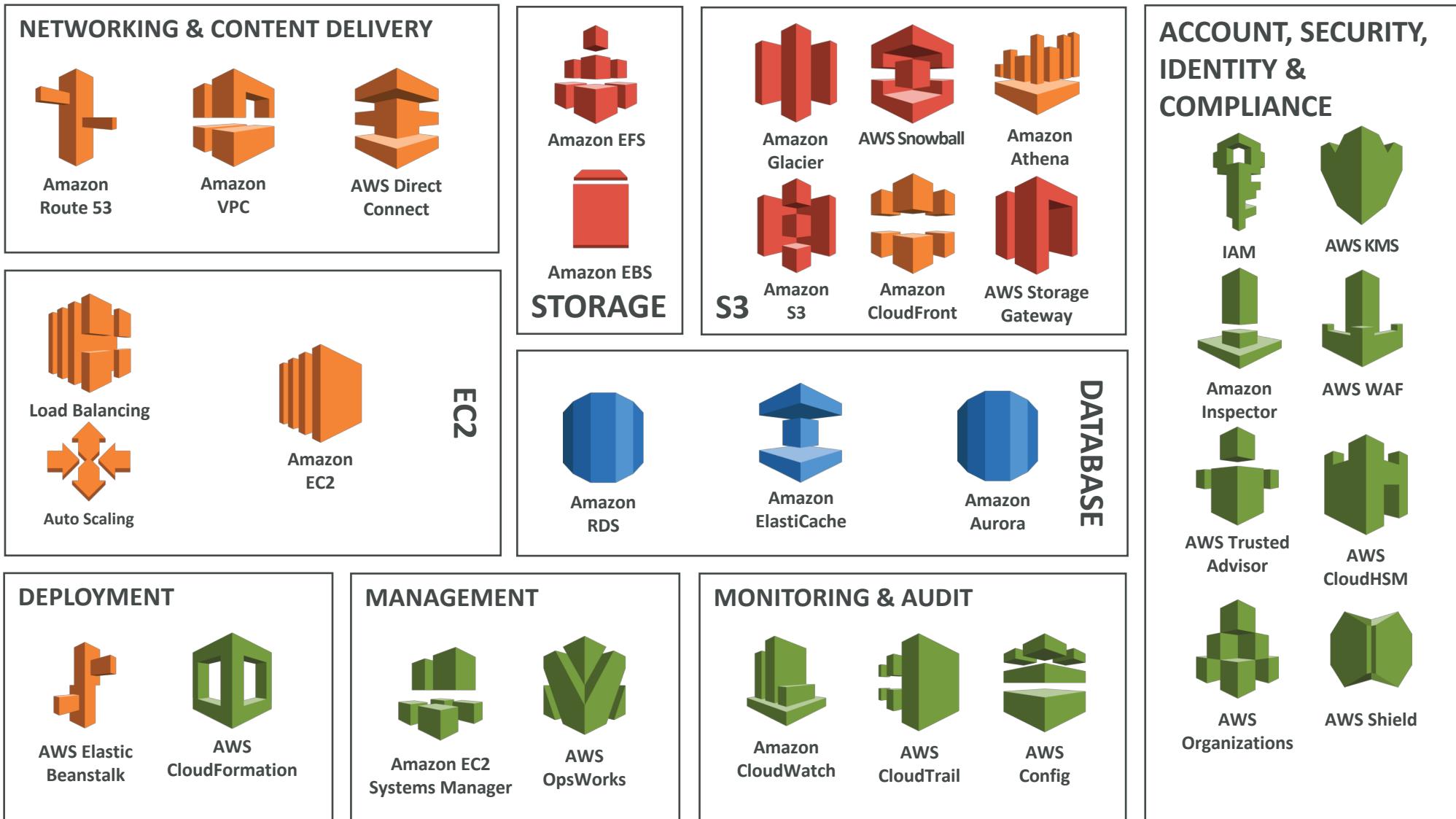
- I'm Stephane!
- Working as in IT consultant and AWS Developer, Solution's Architect & SysOps
- Worked with AWS many years: built websites, apps, streaming platforms
- Veteran Instructor on AWS (Certifications, CloudFormation, Lambda, EC2...)
- You can find me on
  - GitHub: <https://github.com/simplesteph>
  - LinkedIn: <https://www.linkedin.com/in/stephanemaarek>
  - Medium: <https://medium.com/@stephane.maarek>
  - Twitter: <https://twitter.com/stephanemaarek>



# Udemy Tips

# EC2 for SysOps

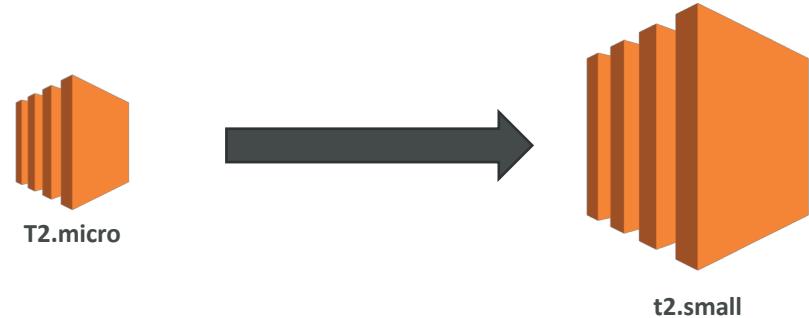
Rocking EC2 from a SysOps perspective



# EC2 Section

- Operations
- Troubleshooting
- Instance Types and Launch Modes
- AMI
- CloudWatch + EC2

# EC2 Changing Instance Type

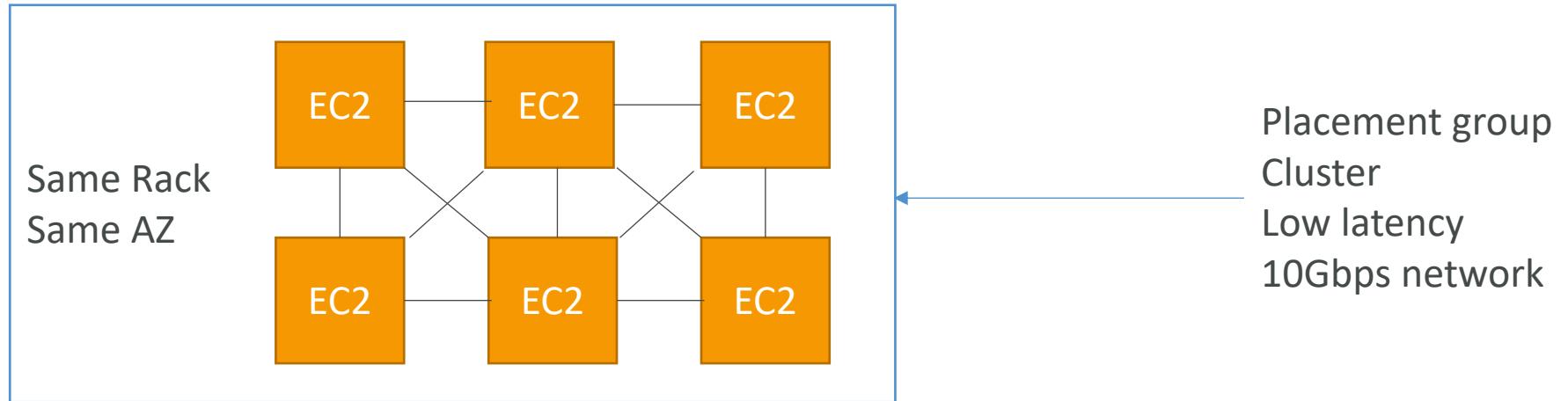


- This only works for EBS backed instances
- Stop the instance
- Instance Settings => Change Instance Type
- Start Instance

# Placement Groups

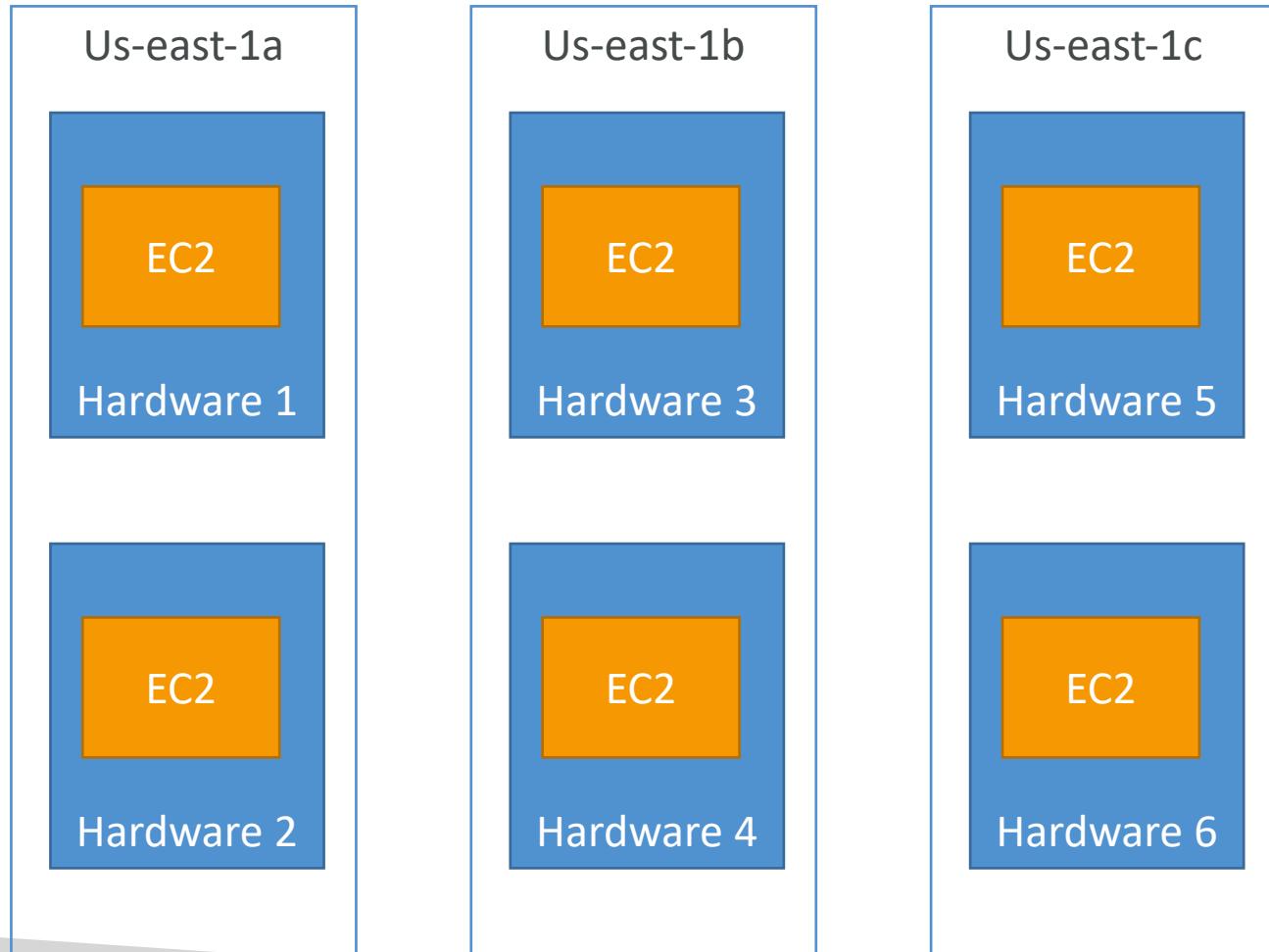
- Sometimes you want control over the EC2 Instance placement strategy
- That strategy can be defined using placement groups
- When you create a placement group, you specify one of the following strategies for the group:
  - *Cluster*—clusters instances into a low-latency group in a single Availability Zone
  - *Spread*—spreads instances across underlying hardware (max 7 instances per group per AZ)
- Not applicable to t2 instances

# Placement Groups Cluster



- Pros: Great network (10 Gbps bandwidth between instances)
- Cons: If the rack fails, all instances fail at the same time
- Use case:
  - Big Data job that needs to complete fast
  - Application that needs extremely low latency and high network throughput

# Placement Groups Spread



- Pros:
  - Can span across Availability Zones (AZ)
  - Reduced risk of simultaneous failure
  - EC2 Instances are on different physical hardware
- Cons:
  - Limited to 7 instances per AZ per placement group
- Use case:
  - Application that needs to maximize high availability
  - Cassandra Cluster, Kafka Cluster, Web application that is distributed

# Shutdown Behaviour

- **Shutdown behaviour** : How should the instance react when shutdown is done using the OS?
  - Stopped: default
  - Terminated
- This is not applicable from AWS console or AWS API.
- CLI Attribute: `InstanceInitiatedShutdownBehavior`

# Termination Protection

- Enable termination protection : To protect against accidental termination in AWS Console or CLI
- Exam Tip:
  - We have an instance where shutdown behavior = terminate and enable terminate protection is ticked
  - We shutdown the instance from the OS, what will happen ?
  - The instance will still be terminated!

# EC2 Launch Troubleshooting

- **# InstanceLimitExceeded error** : if you get this error, it means that you have reached your limit of max number of instances per region
- Resolution : Either launch the instance in a different region or request to AWS to increase your limit of the region.
- Default Limit of Instances in each region: 20

# EC2 Launch Troubleshooting

- **# InsufficientInstanceCapacity** : if you get this error, it means AWS does not have that much On-Demand capacity in the particular AZ to which the instance is launched.
- Resolution :
  - Wait for few mins before requesting again.
  - If requesting more than 1 requests, break down the requests. If you need 5 instances, rather than a single request of 5, request one by one.
  - If urgent, submit a request for a different instance type now, which can be resized later.

# EC2 Launch Troubleshooting

- # Instance Terminates Immediately (goes from pending to terminated)
  - You've reached your EBS volume limit.
  - An EBS snapshot is corrupt.
  - The root EBS volume is encrypted and you do not have permissions to access the KMS key for decryption.
  - The instance store-backed AMI that you used to launch the instance is missing a required part (an image.part.xx file).
- To find the exact reason, check out the EC2 console of AWS - instances - Description tab, note the reason next to the State transition reason label.

# EC2 SSH trouble

- Make sure the private key (pem file) on your linux machine has 400 permissions, else you will get “Unprotected Private Key File error”
- Make sure the username for the OS is given correctly when logging via SSH, else you will get “Host key not found” error.
- Possible reasons for ‘connection timeout’ to EC2 instance via SSH :
  - SG is not configured correctly.
  - CPU load of the instance is high

# EC2 Instance Launch Types

- On Demand Instances: short workload, predictable pricing
- Reserved Instances: long workloads ( $\geq 1$  year)
- Convertible Reserved Instances: long workloads with flexible instances
- Scheduled Reserved Instances: launch within time window you reserve
- Spot Instances: short workloads, for cheap, can lose instances
- Dedicated Instances: no other customers will share your hardware
- Dedicated Hosts: book an entire physical server, control instance placement

# EC2 On Demand

- Pay for what you use (billing per second, after the first minute)
  - Has the highest cost but no upfront payment
  - No long term commitment
- 
- Recommended for short-term and un-interrupted workloads, where you can't predict how the application will behave.

# EC2 Reserved Instances

- Up to 75% discount compared to On-demand
- Pay upfront for what you use with long term commitment
- Reservation period can be 1 or 3 years
- Reserve a specific instance type
- Recommended for steady state usage applications (think database)
- **Convertible Reserved Instance**
  - can change the EC2 instance type
  - Up to 54% discount
- **Scheduled Reserved Instances**
  - launch within time window you reserve
  - When you require a fraction of day / week / month

# EC2 Spot Instances

- Can get a discount of up to 90% compared to On-demand
- You bid a price and get the instance as long as its under the price
- Price varies based on offer and demand
- Spot instances are reclaimed with a 2 minute notification warning when the spot price goes above your bid
- Used for batch jobs, Big Data analysis, or workloads that are resilient to failures.
- Not great for critical jobs or databases

# EC2 Dedicated Hosts

- Physical dedicated EC2 server for your use
  - Full control of EC2 Instance placement
  - Visibility into the underlying sockets / physical cores of the hardware
  - Allocated for your account for a 3 year period reservation
  - More expensive
- 
- Useful for software that have complicated licensing model (BYOL – Bring Your Own License)
  - Or for companies that have strong regulatory or compliance needs

# EC2 Dedicated Instances

- Instances running on hardware that's dedicated to you
- May share hardware with other instances in same account
- No control over instance placement (can move hardware after Stop / Start)

Characteristic	Dedicated Instances	Dedicated Hosts
Enables the use of dedicated physical servers	x	x
Per instance billing (subject to a \$2 per region fee)	x	
Per host billing		x
Visibility of sockets, cores, host ID		x
Affinity between a host and instance		x
Targeted instance placement		x
Automatic instance placement	x	x
Add capacity using an allocation request		x

# Which host is right for me?



- **On demand:** coming and staying in resort whenever we like, we pay the full price
- **Reserved:** like planning ahead and if we plan to stay for a long time, we may get a good discount.
- **Spot instances:** the hotel allows people to bid for the empty rooms and the highest bidder keeps the rooms. You can get kicked out at any time
- **Dedicated Hosts:** We book an entire building of the resort

# EC2 Instance Types – Main ones

- R: applications that needs a lot of RAM – in-memory caches
- C: applications that needs good CPU – compute / databases
- M: applications that are balanced (think “medium”) – general / web app
- I: applications that need good local I/O (instance storage) – databases
- G: applications that need a GPU – video rendering / machine learning
- T2 / T3: burstable instances (up to a capacity)
- T2 / T3 - unlimited: unlimited burst
- Real-world tip: use <https://www.ec2instances.info>

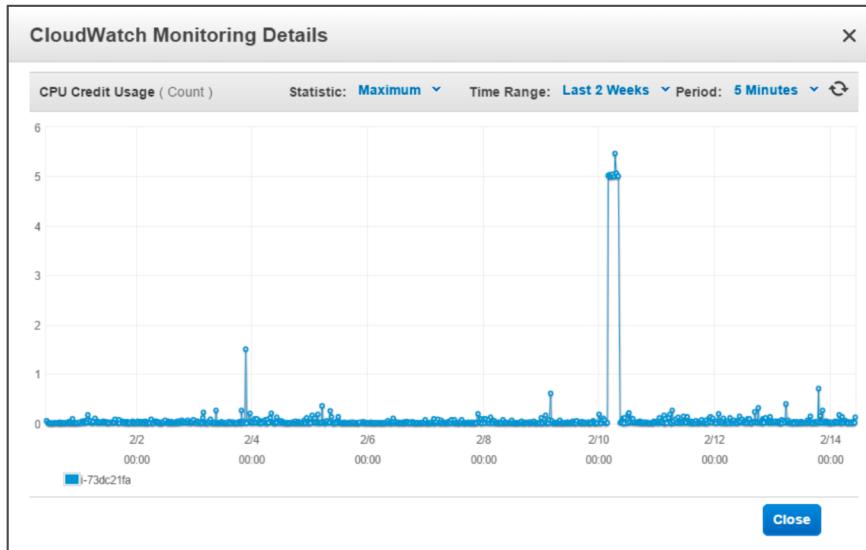
# Burstable Instances (T2/T3)

- AWS has the concept of burstable instances (T2/T3 machines)
- Burst means that overall, the instance has OK CPU performance.
- When the machine needs to process something unexpected (a spike in load for example), it can burst, and CPU can be VERY good.
- If the machine bursts, it utilizes “burst credits”
- If all the credits are gone, the CPU becomes BAD
- If the machine stops bursting, credits are accumulated over time

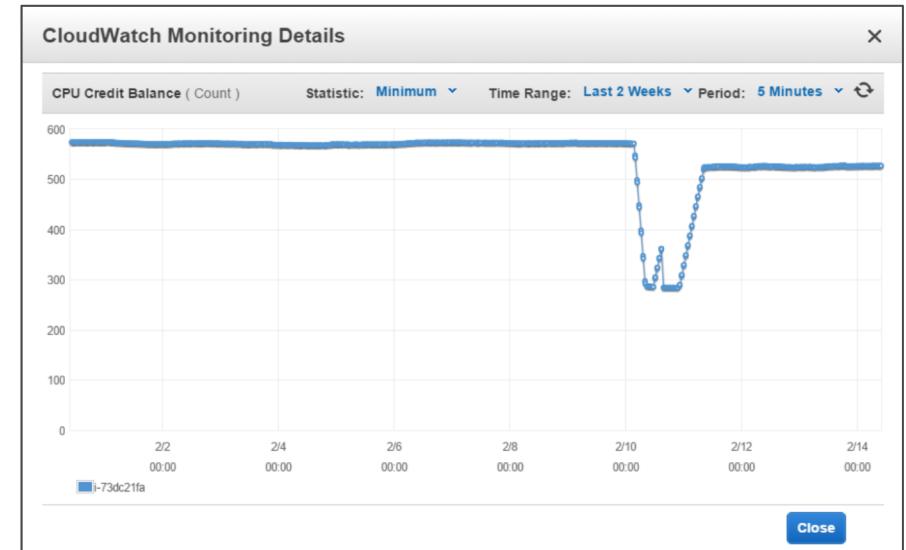
# Burstable Instances (T2/T3)

- Burstable instances can be amazing to handle unexpected traffic and getting the insurance that it will be handled correctly
- If your instance consistently runs low on credit, you need to move to a different kind of non-burstable instance

Credit usage



Credit balance



# CPU Credits

Instance type	Launch credits	vCPUs	CPU credits earned per hour	Maximum earned CPU credit balance	vCPUs	Baseline performance (% CPU utilization)
t2.nano	30	1	3	72	1	5%
t2.micro	30	1	6	144	1	10%
t2.small	30	1	12	288	1	20%
t2.medium	60	2	24	576	2	40% (of 200% max)*
t2.large	60	2	36	864	2	60% (of 200% max)*
t2.xlarge	120	4	54	1296	4	90% (of 400% max)*
t2.2xlarge	240	8	81	1944	8	135% (of 800% max)*

# T2/T3 Unlimited

- Nov 2017: It is possible to have an “unlimited burst credit balance”
- You pay extra money if you go over your credit balance, but you don’t lose in performance
- Overall, it is a new offering, so be careful, costs could go high if you’re not monitoring the health of your instances
- Read more here: <https://aws.amazon.com/blogs/aws/new-t2-unlimited-going-beyond-the-burst-with-high-performance/>

# What's an AMI?

- As we saw, AWS comes with base images such as:
  - Ubuntu
  - Fedora
  - RedHat
  - Windows
  - Etc...
- These images can be customised at runtime using EC2 User data
- But what if we could create our own image, ready to go?
- That's an AMI – an image to use to create our instances
- AMIs can be built for Linux or Windows machines

# Why would you use a custom AMI?

- Using a custom built AMI can provide the following advantages:
  - Pre-installed packages needed
  - Faster boot time (no need for ec2 user data at boot time)
  - Machine comes configured with monitoring / enterprise software
  - Security concerns – control over the machines in the network
  - Control of maintenance and updates of AMIs over time
  - Active Directory Integration out of the box
  - Installing your app ahead of time (for faster deploys when auto-scaling)
  - Using someone else's AMI that is optimised for running an app, DB, etc...
- AMI are built for a specific AWS region (!)

# Using Public AMIs

- You can leverage AMIs from other people
- You can also pay for other people's AMI by the hour
  - These people have optimised the software
  - The machine is easy to run and configure
  - You basically rent "expertise" from the AMI creator
- AMI can be found and published on the Amazon Marketplace
- **Warning:**
  - Do not use an AMI you don't trust!
  - Some AMIs might come with malware or may not be secure for your enterprise

# AMI Storage

- Your AMI take space and they live in Amazon S3
- Amazon S3 is a durable, cheap and resilient storage where most of your backups will live (but you won't see them in the S3 console)
- By default, your AMIs are private, and locked for your account / region
- You can also make your AMIs public and share them with other AWS accounts or sell them on the AMI Marketplace

# AMI Pricing

- AMIs live in Amazon S3, so you get charged for the actual space it takes in Amazon S3
- Amazon S3 pricing in US-EAST-1:
  - First 50TB / month: \$0.023 per GB
  - Next 450TB / month: \$0.022 per GB
- Overall it is quite inexpensive to store private AMIs.
- Make sure to remove the AMIs you don't use

# Cross Account AMI Copy (FAQ + Exam Tip)

- You can share an AMI with another AWS account.
- Sharing an AMI does not affect the ownership of the AMI.
- If you copy an AMI that has been shared with your account, you are the owner of the target AMI in your account.
- To copy an AMI that was shared with you from another account, the owner of the source AMI must grant you read permissions for the storage that backs the AMI, either the associated EBS snapshot (for an Amazon EBS-backed AMI) or an associated S3 bucket (for an instance store-backed AMI).
- **Limits:**
  - You can't copy an encrypted AMI that was shared with you from another account. Instead, if the underlying snapshot and encryption key were shared with you, you can copy the snapshot while re-encrypting it with a key of your own. You own the copied snapshot, and can register it as a new AMI.
  - You can't copy an AMI with an associated **billingProduct** code that was shared with you from another account. This includes Windows AMIs and AMIs from the AWS Marketplace. To copy a shared AMI with a **billingProduct** code, launch an EC2 instance in your account using the shared AMI and then create an AMI from the instance.

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/CopyingAMIs.html>

# Elastic IPs

- When you stop and then start an EC2 instance, it changes its public IP.
  - If you need to have a fixed public IP, you need an Elastic IP
  - An Elastic IP is a public IPv4 IP you own as long as you don't delete it
  - You can attach it to one instance at a time
  - You can remap it across instances
- 
- You don't pay for the Elastic IP if it's attached to a server
  - You pay for the Elastic IP if it's not attached to a server

# Elastic IP

- With an Elastic IP address, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account.
- You can only have 5 Elastic IP in your account (you can ask AWS to increase that).
- Overall, try to avoid using Elastic IP:
  - Always think if other alternatives are available to you
  - You could use a random public IP and register a DNS name to it
  - Or use a Load Balancer with a static hostname

# CloudWatch Metrics for EC2

- AWS Provided metrics (AWS pushes them):
  - Basic Monitoring (default): metrics are collected at a 5 minute internal
  - Detailed Monitoring (paid): metrics are collected at a 1 minute interval
  - Includes CPU, Network, Disk and Status Check Metrics
- Custom metric (yours to push):
  - Basic Resolution: 1 minute resolution
  - High Resolution: all the way to 1 second resolution
  - Include RAM, application level metrics
  - Make sure the IAM permissions on the EC2 instance role are correct !

# EC2 included metrics

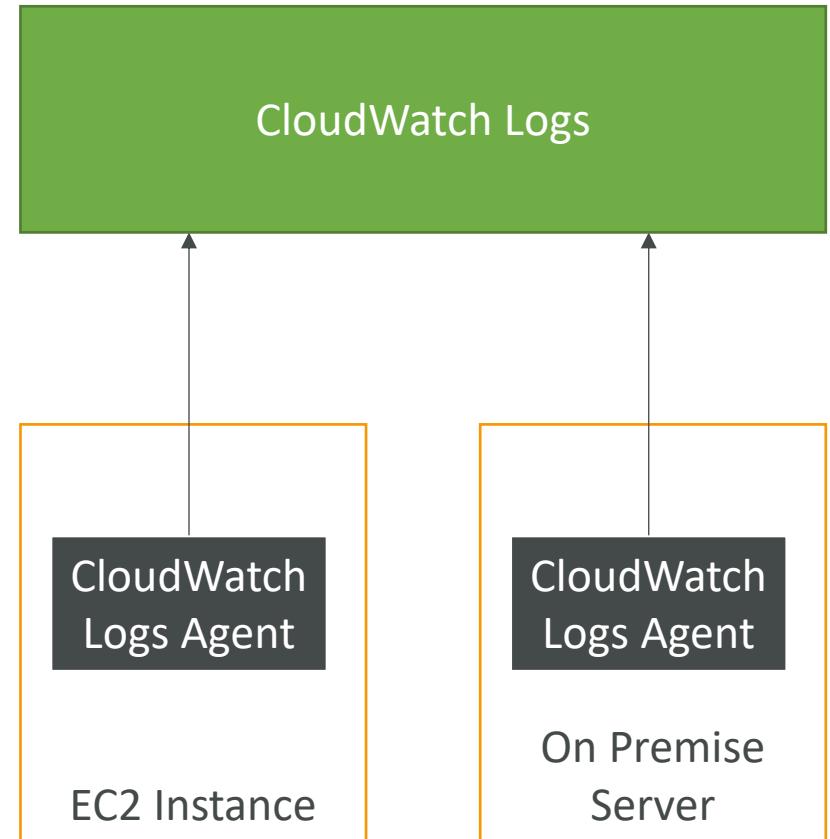
- CPU: CPU Utilization + Credit Usage / Balance
- Network: Network In / Out
- Status Check:
  - Instance status = check the EC2 VM
  - System status = check the underlying hardware
- Disk: Read / Write for Ops / Bytes (only for instance store)
- RAM is NOT included in the AWS EC2 metrics

# EC2 Custom Metrics

- Sample custom metrics for EC2:
  - RAM usage
  - Swap usage
  - Any custom metric for your application (requests per seconds, etc.. )
- Hands On - RAM as a custom metric:
- We'll download the scripts from the doc
- We'll push the RAM as a custom metric

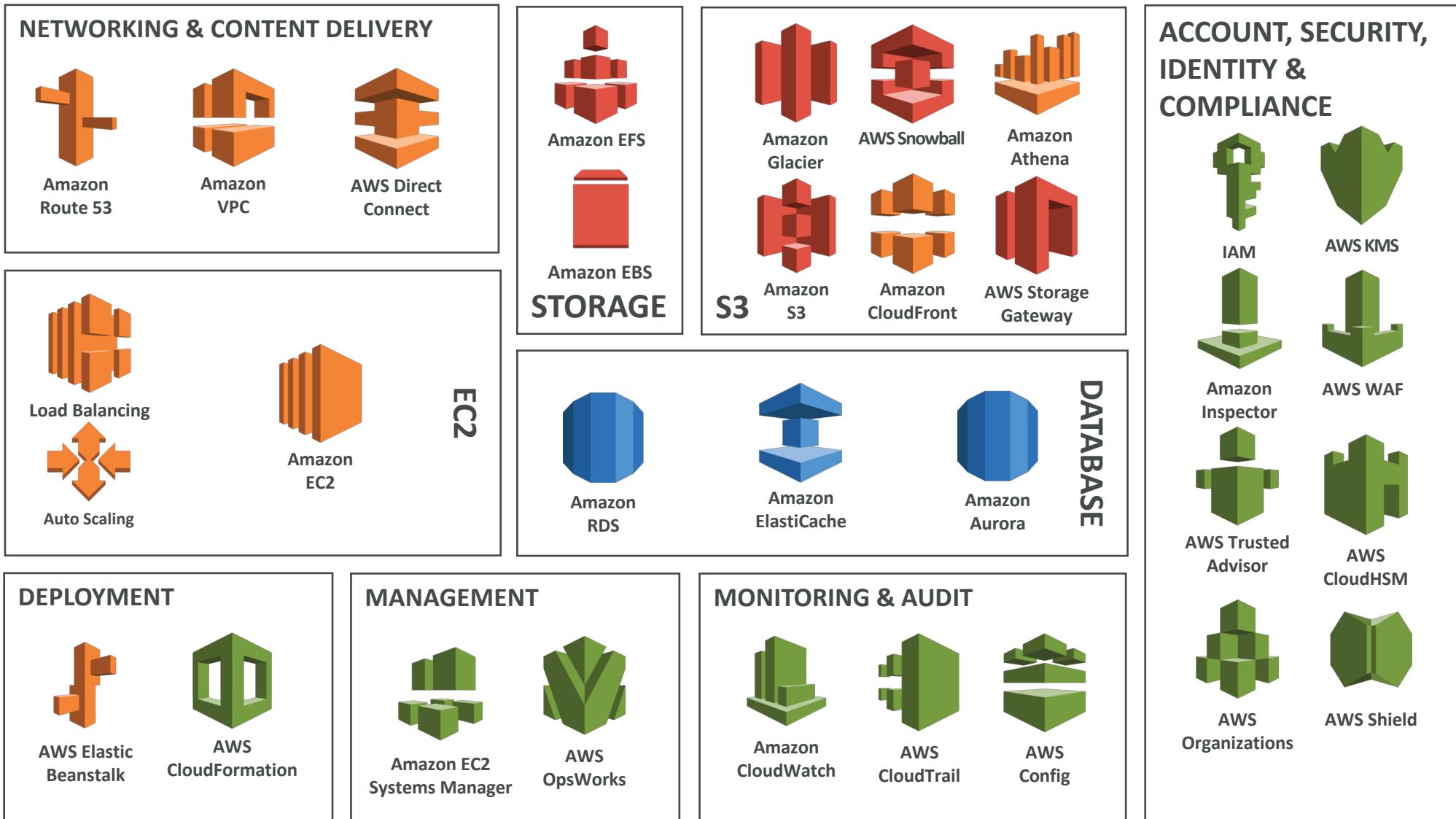
# CloudWatch Logs for EC2

- By default, no logs from your EC2 machine will go to CloudWatch
- You need to run a CloudWatch agent on EC2 to push the log files you want
- Make sure IAM permissions are correct
- The CloudWatch log agent can be setup on premise too
- Let's practice!



# Management of EC2 at scale

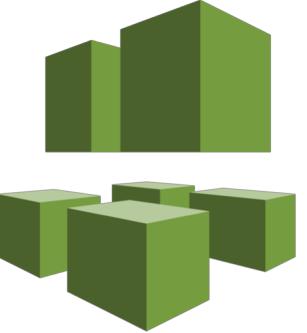
Systems Manager & OpsWorks



# SSM & OpsWorks Section

- How do we manage fleets of EC2 instances and on-premise servers?
- How to apply patches at scale?
- How to run automations?
- Store parameters?
  
- Use Chef and Puppet?

# AWS Systems Manager Overview



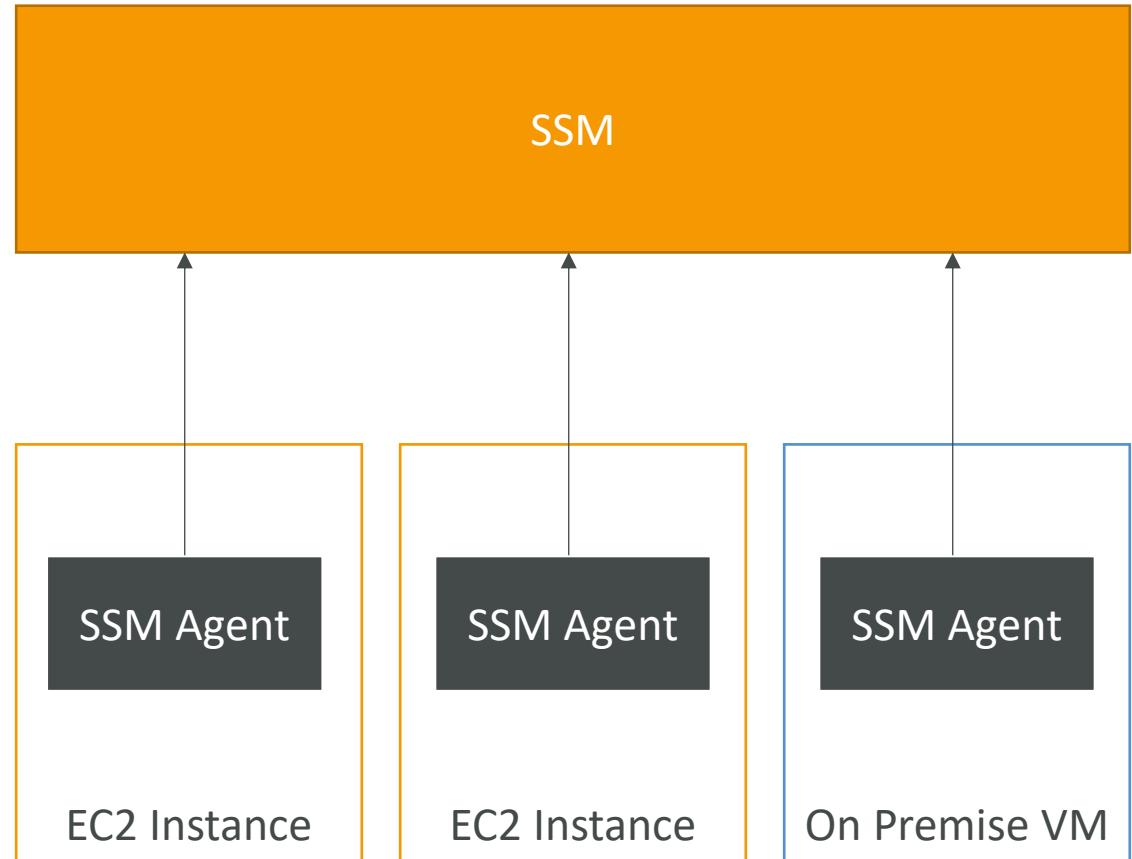
- Helps you manage your **EC2** and **On-Premise** systems at scale
- Get operational insights about the state of your infrastructure
- Easily detect problems
- **Patching automation for enhanced compliance**
- Works for both Windows and Linux OS
- Integrated with CloudWatch metrics / dashboards
- Integrated with AWS Config
- Free service

# AWS Systems Manager Features

- Resource Groups
  - Insights:
    - Insights Dashboard
    - Inventory: discover and audit the software installed
    - Compliance
  - Parameter Store
- Action:
- Automation (shut down EC2, create AMIs)
  - Run Command
  - Session Manager
  - Patch Manager
  - Maintenance Windows
  - State Manager: define and maintaining configuration of OS and applications

# How Systems Manager works

- We need to install the SSM agent onto the systems we control
- Installed by default on Amazon Linux AMI & some Ubuntu AMI
- If an instance can't be controlled with SSM, it's probably an issue with the SSM agent!
- Make sure the EC2 instances have a proper IAM role to allow SSM actions



# AWS Tags

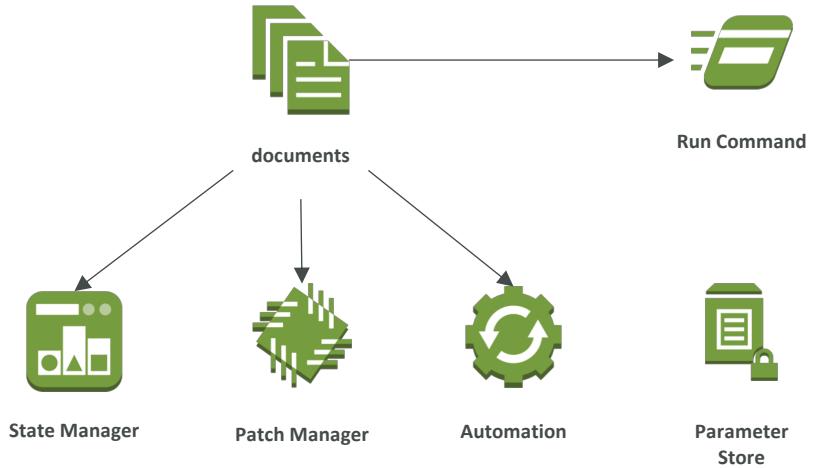
- You can add Text Key Value Pairs called Tags to many AWS resources
- Commonly used in EC2
- Free naming, common tags are: Name, Environment, Team ...
- They're used for
  - Resource grouping
  - Automation
  - Cost allocation
- Better to have too many tags than too few!

# AWS Systems Manager Resource Groups

- Create, view or manage logical group of resources thanks to tags
- Allows creation of logical groups of resources such as
  - Applications
  - Different layers of an application stack
  - Production versus development environments
- Regional service
- Works with EC2, S3, DynamoDB, Lambda, etc...

# Documents

- Documents can be in JSON or YAML
- You define parameters
- You define actions
- Many documents already exist in AWS



```
---  
schemaVersion: '2.2'  
description: State Manager Bootstrap Example  
parameters: {}  
mainSteps:  
- action: aws:runShellScript  
  name: configureServer  
  inputs:  
    runCommand:  
      - sudo yum install -y httpd24  
      - sudo yum --enablerepo=epel install -y clamav
```

# AWS Systems Manager

## Run Command

- Execute a document (= script) or just run a command
- Run command across multiple instances (using resource groups)
- Rate Control / Error Control
- Integrated with IAM & CloudTrail
- No need for SSH
- Results in the console

# Using AWS Systems Manager to PATCH

- Inventory => List software on an instance
- Inventory + Run Command => Patch Software
- Patch manager + Maintenance Window => Patch OS
- Patch manager => Gives you compliance
- State manager => Ensure instances are in a consistent state (compliance)

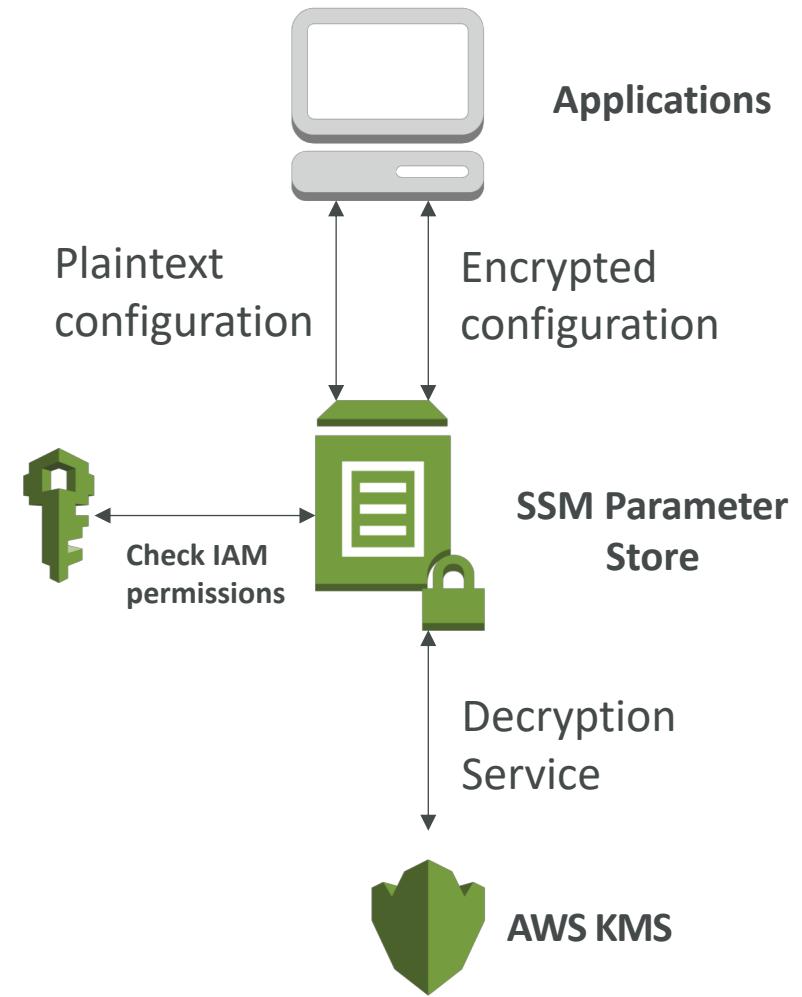
# AWS Systems Manager

## Session Manager

- Session Manager allows you to start a secure shell on your VM
- Does not use SSH access and bastion hosts
- Only EC2 for now, but On Premise soon
- Log actions done through secure shells to S3 and CloudWatch Logs
- IAM permissions: access SSM + write to S3 + write to CloudWatch
- CloudTrail can intercept StartSession events
- AWS Secure Shell versus SSH:
  - No need to open the port 22 at all anymore (security)
  - No need for bastion hosts
  - All commands are logged to S3 / CloudWatch (auditing)
  - Access to Secure Shell is done through User IAM, not SSH keys

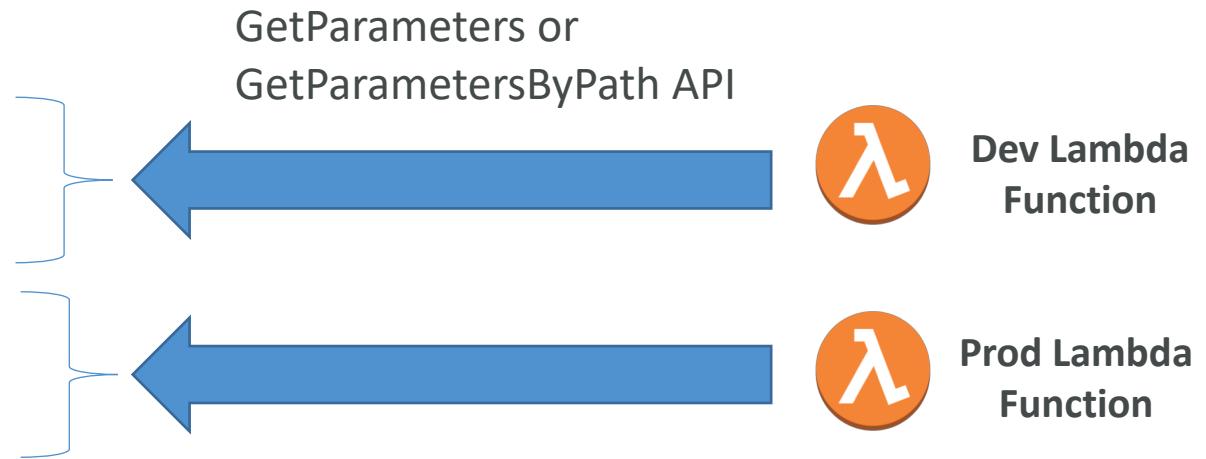
# AWS Parameter Store

- Secure storage for configuration and secrets
- Optional Seamless Encryption using KMS
- Serverless, scalable, durable, easy SDK, free
- Version tracking of configurations / secrets
- Configuration management using path & IAM
- Notifications with CloudWatch Events
- Integration with CloudFormation



# AWS Parameter Store Hierarchy

- /my-department/
  - my-app/
    - dev/
      - db-url
      - db-password
    - prod/
      - db-url
      - db-password
  - other-app/
  - /other-department/



# What if I lost my SSH key for EC2?

- (Traditional method) If the instance is EBS backed:
  - Stop the instance, detach the root volume
  - Attach the root volume to another instance as a data volume
  - Modify the `~/.ssh/authorized_keys` file with your new key
  - Move the volume back to the stopped instance
  - Start the instance and you can SSH into it again
- (New method) If the instance is EBS:
  - Run the `AWSSupport-ResetAccess` automation document in SSM
- Instance Store backed EC2:
  - You can't stop the instance (otherwise data is lost) – AWS recommends termination
  - My tip: Use `Session Manager` access and edit the `~/.ssh/authorized_keys` file directly

# AWS Opsworks



- Chef & Puppet help you perform server configuration automatically, or repetitive actions
- They work great with EC2 & On Premise VM
- AWS Opsworks = Managed Chef & Puppet
- It's an alternative to AWS SSM
- No hands on here, no knowledge of chef and puppet needed
- In the exam: Chef & Puppet needed => AWS Opsworks

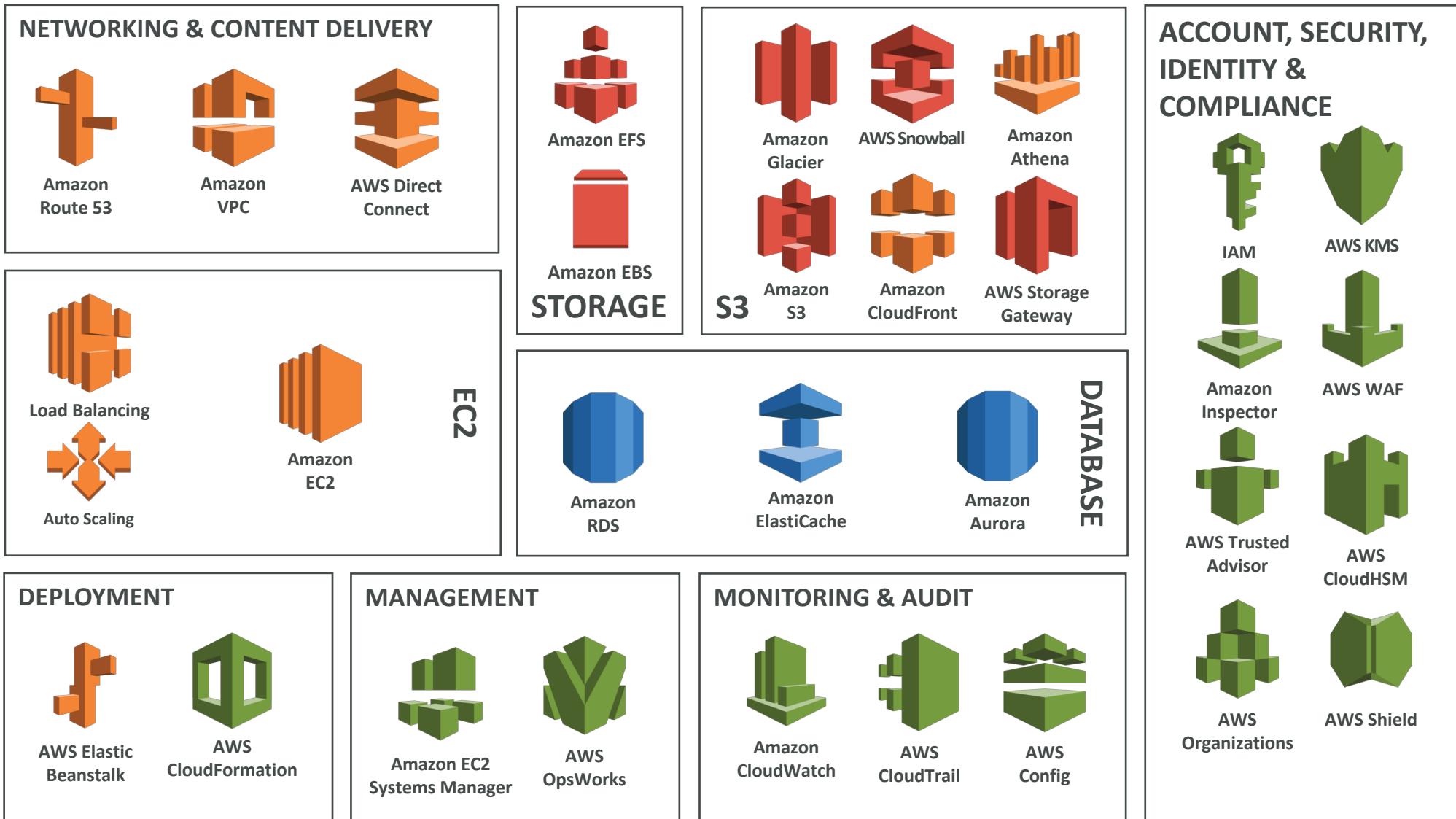


# Quick word on Chef / Puppet

- They help with managing configuration as code
- Helps in having consistent deployments
- Works with Linux / Windows
- Can automate: user accounts, cron, ntp, packages, services...
- They leverage “Recipes” or “Manifests”
- Chef / Puppet have similarities with SSM / Beanstalk / CloudFormation but they’re open-source tools that work cross-cloud

# EC2 High Availability and Scalability

Load Balancer and Auto Scaling Groups



# Scalability and High Availability Section

- Load Balancers:
  - Troubleshooting
  - Advanced options and logging
  - CloudWatch integrations
- Auto Scaling
  - Troubleshooting
  - Advanced options and logging
  - CloudWatch integrations

# Scalability & High Availability

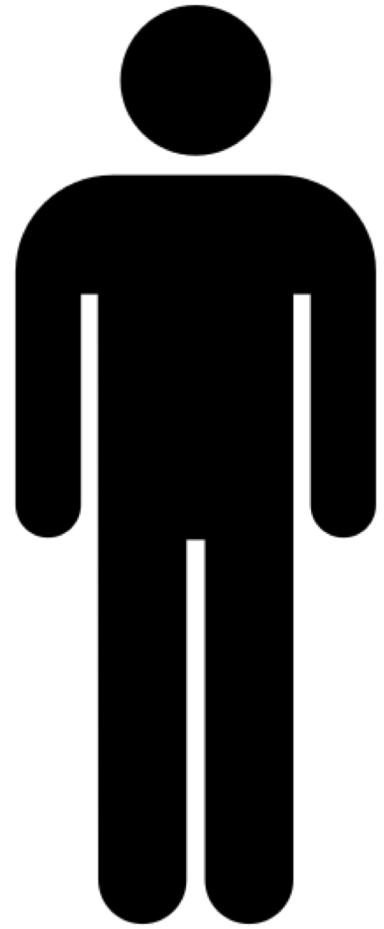
- Scalability means that an application / system can handle greater loads by adapting.
- There are two kinds of scalability:
  - Vertical Scalability
  - Horizontal Scalability (= elasticity)
- Scalability is linked but different to High Availability
- Let's deep dive into the distinction, using a call center as an example

# Vertical Scalability

- Vertically scalability means increasing the size of the instance
- For example, your application runs on a t2.micro
- Scaling that application vertically means running it on a t2.large
- Vertical scalability is very common for non distributed systems, such as a database.
- RDS, ElastiCache are services that can scale vertically.
- There's usually a limit to how much you can vertically scale (hardware limit)



junior operator

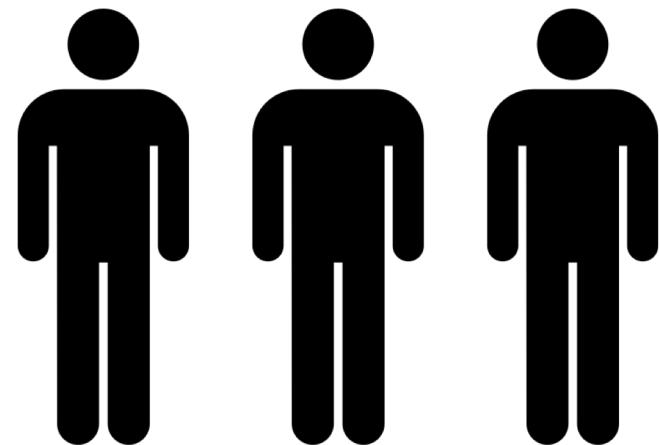
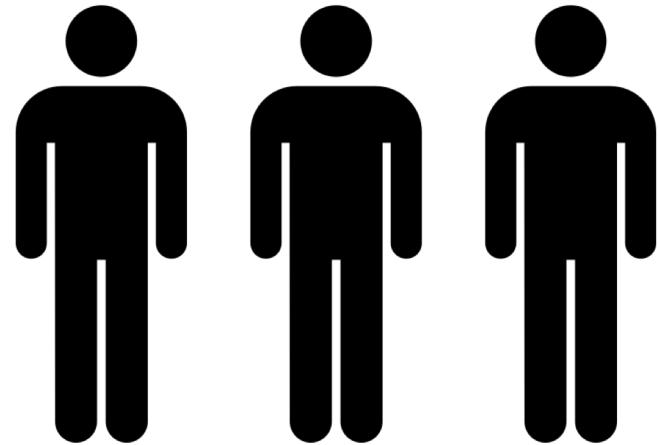


senior operator

# Horizontal Scalability

- Horizontal Scalability means increasing the number of instances / systems for your application
- Horizontal scaling implies distributed systems.
- This is very common for web applications / modern applications
- It's easy to horizontally scale thanks the cloud offerings such as Amazon EC2

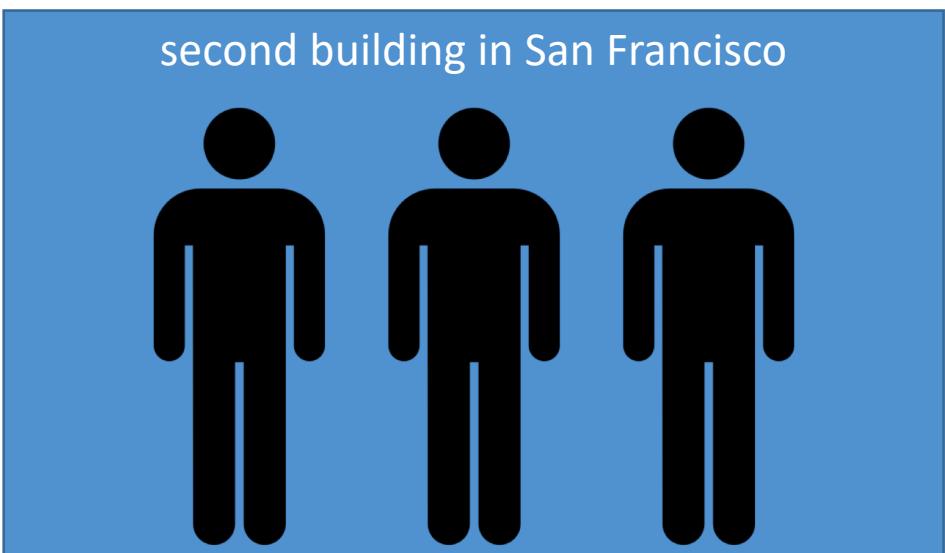
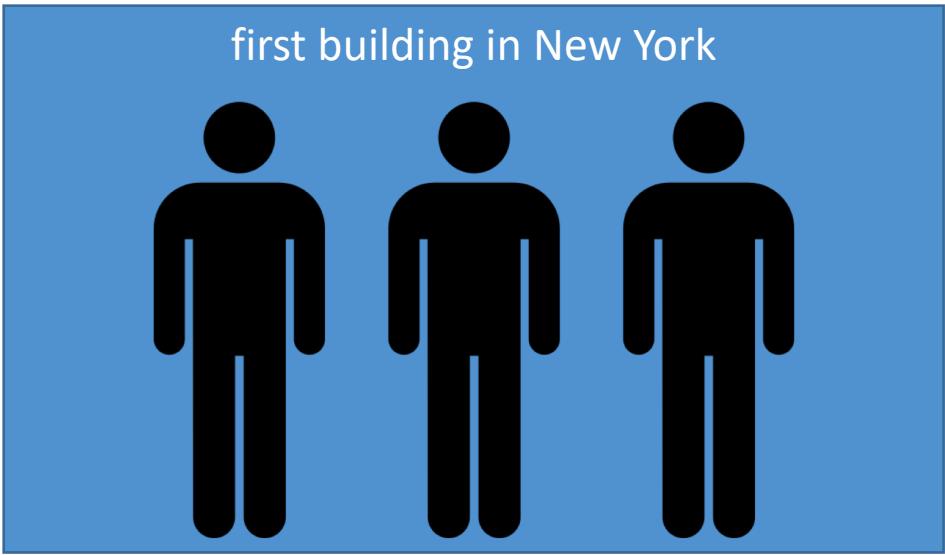
operator operator operator



operator operator operator

# High Availability

- High Availability usually goes hand in hand with horizontal scaling
- High availability means running your application / system in at least 2 data centers (== Availability Zones)
- The goal of high availability is to survive a data center loss
- The high availability can be passive (for RDS Multi AZ for example)
- The high availability can be active (for horizontal scaling)



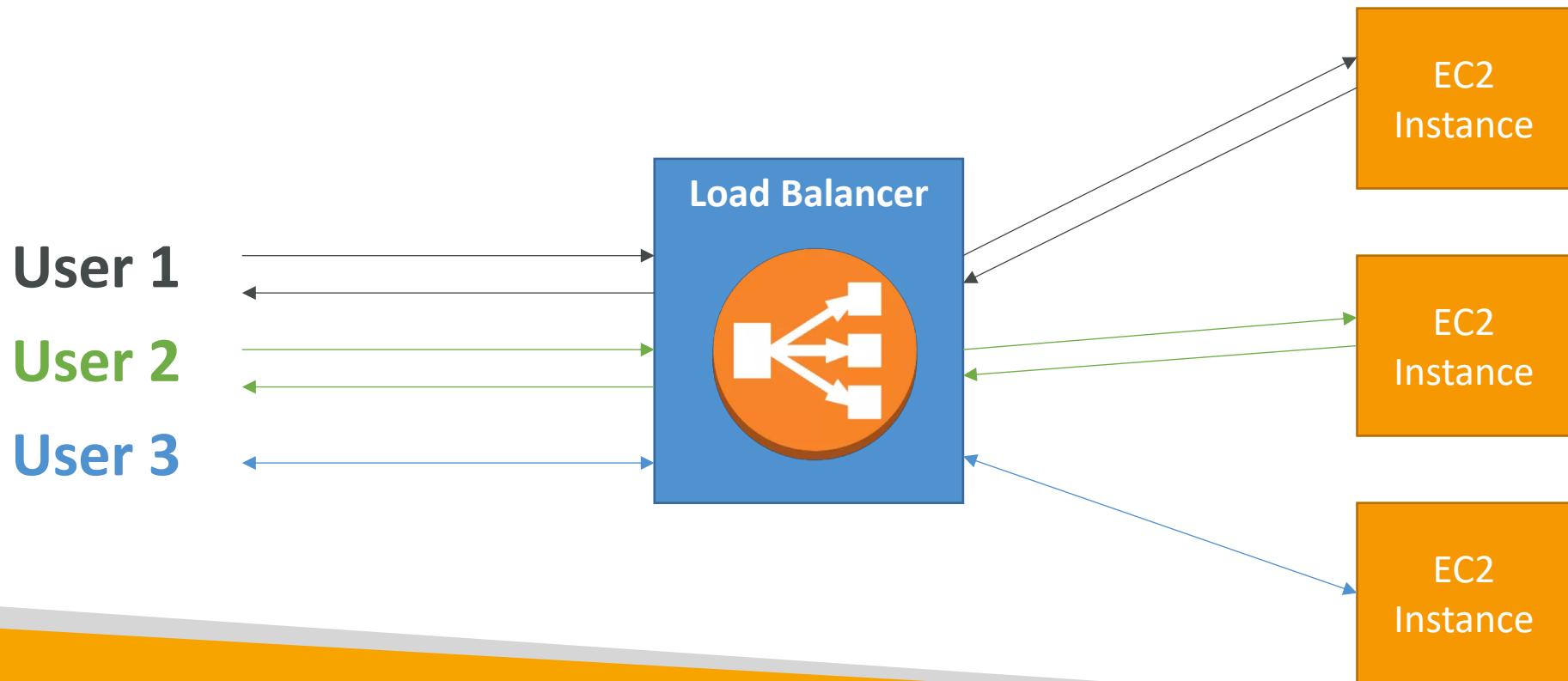
# High Availability & Scalability For EC2

- Vertical Scaling: Increase instance size (= scale up / down)
  - From: t2.nano - 0.5G of RAM, 1 vCPU
  - To: u-12tbl.metal – 12.3 TB of RAM, 448 vCPUs
- Horizontal Scaling: Increase number of instances (= scale out / in)
  - Auto Scaling Group
  - Load Balancer
- High Availability: Run instances for the same application across multi AZ
  - Auto Scaling Group multi AZ
  - Load Balancer multi AZ



# What is load balancing?

- Load balancers are servers that forward internet traffic to multiple servers (EC2 Instances) downstream.



# Why use a load balancer?

- Spread load across multiple downstream instances
- Expose a single point of access (DNS) to your application
- Seamlessly handle failures of downstream instances
- Do regular health checks to your instances
- Provide SSL termination (HTTPS) for your websites
- Enforce stickiness with cookies
- High availability across zones
- Separate public traffic from private traffic

# Why use an EC2 Load Balancer?

- An ELB (EC2 Load Balancer) is a **managed load balancer**
  - AWS guarantees that it will be working
  - AWS takes care of upgrades, maintenance, high availability
  - AWS provides only a few configuration knobs
- It costs less to setup your own load balancer but it will be a lot more effort on your end.
- It is integrated with many AWS offerings / services

# Types of load balancer on AWS

- AWS has **3 kinds of Load Balancers**
- Classic Load Balancer (v1 - old generation) - 2009
- Application Load Balancer (v2 - new generation) - 2016
- Network Load Balancer (v2 - new generation) - 2017
- Overall, it is recommended to use the newer / v2 generation load balancers as they provide more features
- You can setup **internal** (private) or **external** (public) ELBs

# Health Checks

- Health Checks are crucial for Load Balancers
- They enable the load balancer to know if instances it forwards traffic to are available to reply to requests
- The health check is done on a port and a route (/health is common)
- If the response is not 200 (OK), then the instance is unhealthy

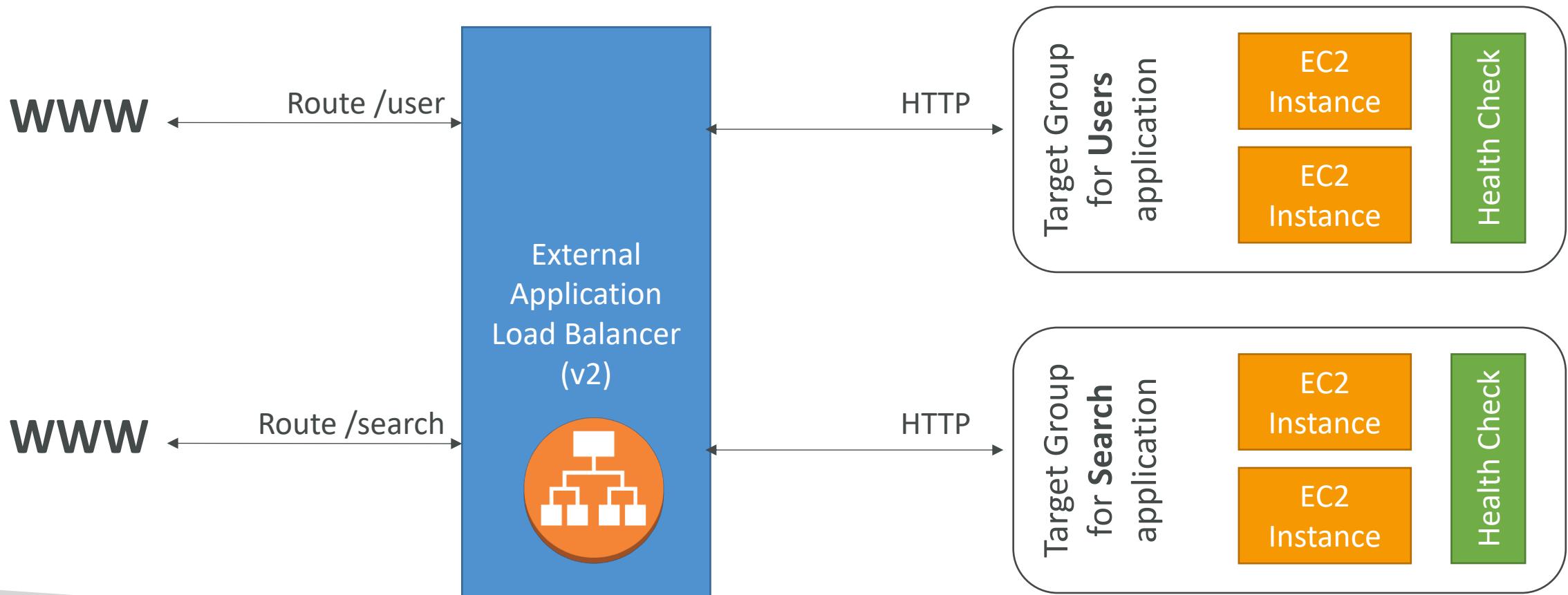


# Application Load Balancer (v2)

- Application load balancers (Layer 7) allow to do:
  - Load balancing to multiple HTTP applications across machines (target groups)
  - Load balancing to multiple applications on the same machine (ex: containers)
  - Load balancing based on route in URL
  - Load balancing based on hostname in URL
- Basically, they're awesome for micro services & container-based application (example: Docker & Amazon ECS)
- Has a port mapping feature to redirect to a dynamic port
- In comparison, we would need to create one Classic Load Balancer per application before. That was very expensive and inefficient!

# Application Load Balancer (v2)

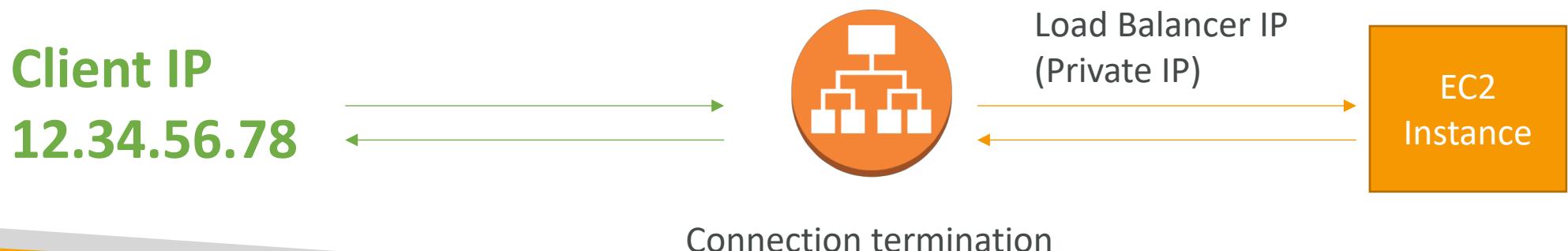
## HTTP Based Traffic



# Application Load Balancer v2

## Good to Know

- Stickiness can be enabled at the target group level
  - Same request goes to the same instance
  - Stickiness is directly generated by the ALB (not the application)
- ALB support HTTP/HTTPS & Websockets protocols
- The application servers don't see the IP of the client directly
  - The true IP of the client is inserted in the header **X-Forwarded-For**
  - We can also get Port (**X-Forwarded-Port**) and proto (**X-Forwarded-Proto**)

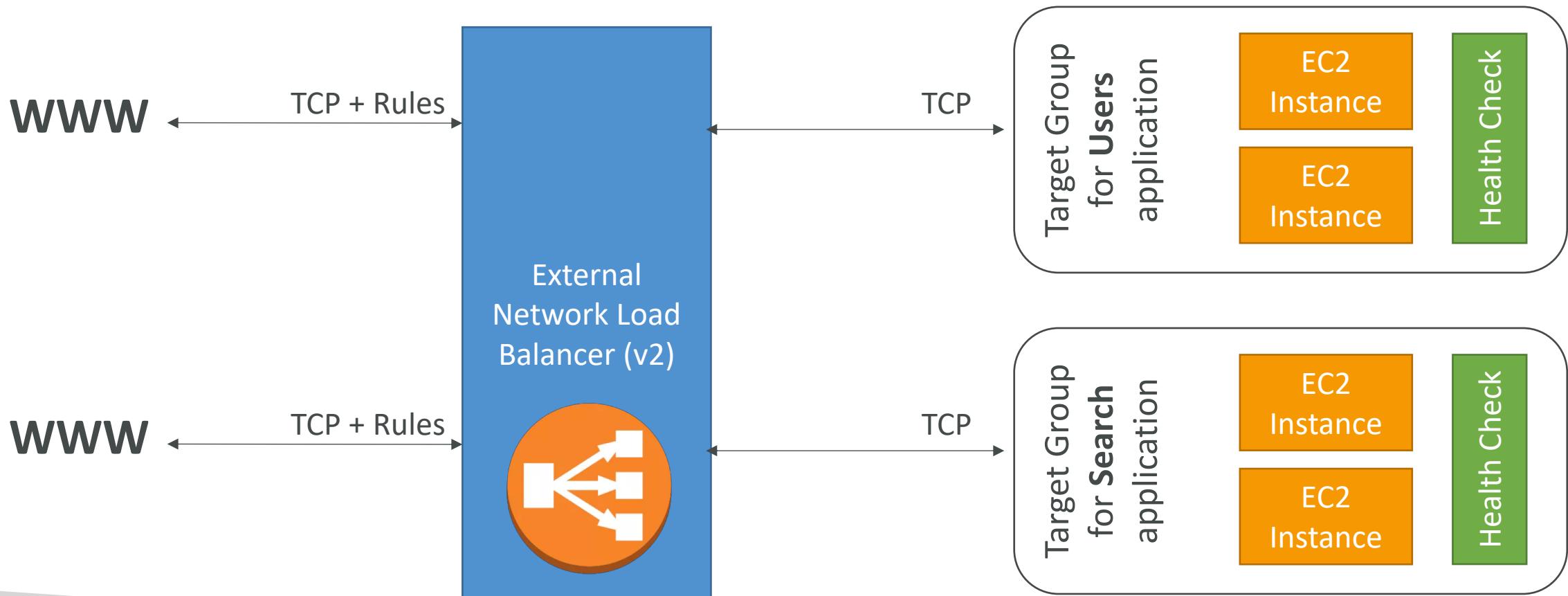


# Network Load Balancer (v2)

- Network load balancers (Layer 4) allow to do:
  - Forward TCP traffic to your instances
  - Handle millions of requests per second
  - Support for static IP or elastic IP
  - Less latency ~100 ms (vs 400 ms for ALB)
- Network Load Balancers are mostly used for extreme performance and should not be the default load balancer you choose
- Overall, the creation process is the same as Application Load Balancers

# Network Load Balancer (v2)

## TCP Based Traffic



# Load Balancer Good to Know

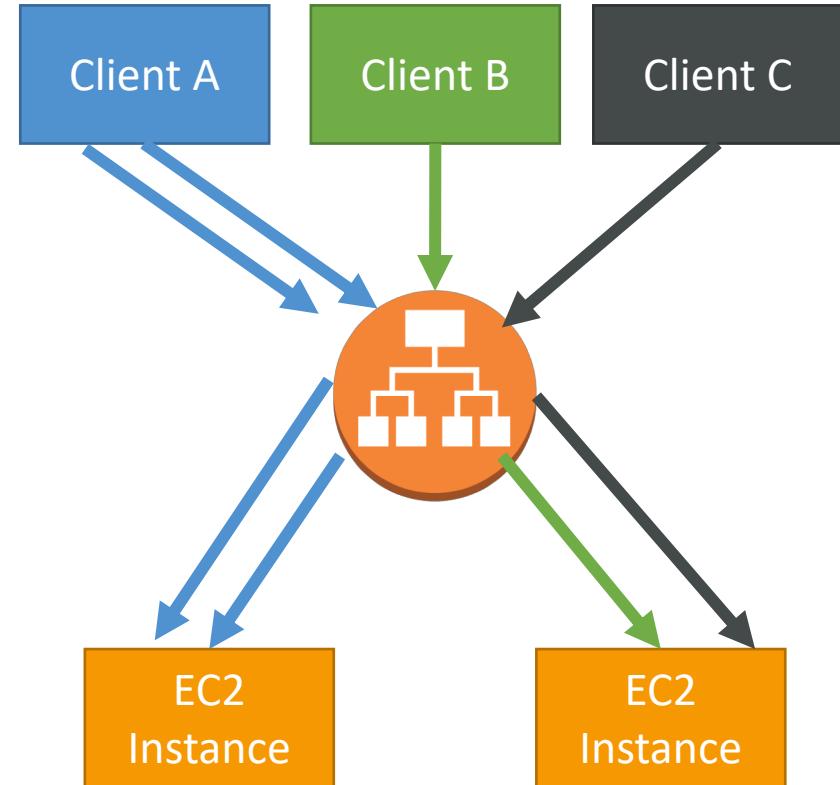
- Classic Load Balancers are Deprecated
  - Application Load Balancers for HTTP / HTTPS & Websocket
  - Network Load Balancer for TCP
- CLB and ALB support SSL certificates and provide SSL termination
- All Load Balancers have health check capability
- ALB can route on based on hostname / path
- ALB is a great fit with ECS (Docker)

# Load Balancer Good to Know

- Any Load Balancer (CLB, ALB, NLB) has a static host name. Do not resolve and use underlying IP
- LBs can scale but not instantaneously – contact AWS for a “warm-up”
- NLB directly see the client IP
- 4xx errors are client induced errors
- 5xx errors are application induced errors
  - Load Balancer Errors 503 means at capacity or no registered target
- If the LB can't connect to your application, check your security groups!

# Load Balancer Stickiness

- It is possible to implement stickiness so that the same client is always redirected to the same instance behind a load balancer
- This works for Classic Load Balancers & Application Load Balancers
- The “cookie” used for stickiness has an expiration date you control
- Use case: make sure the user doesn’t lose his session data
- Enabling stickiness may bring imbalance to the load over the backend EC2 instances



# Load Balancer for SysOps

- Application Load Balancer:
  - Layer 7 (HTTP, HTTPS, Websocket)
  - URL based routing (hostname or path)
  - Does not support static IP, but has a fixed DNS
  - Provide SSL termination
- Network Load Balancer:
  - Layer 4 (TCP)
  - No pre-warming needed
  - 1 static IP per subnet
  - No SSL termination (SSL must be enabled by the application itself)
- Exam tip: Chain an NLB and a ALB to “give” ALB a fixed IP

# Load Balancer Pre-warming

- ELB scale gradually to traffic
- ELB may fail in case of sudden spike of traffic (10x traffic)
- If you expect high traffic (Christmas season), open a support ticket with AWS to pre-warm your ELB
  - Duration of traffic
  - Expected requests per second
  - Size of request (in KB)

# Load Balancer Error codes

- Successful request : Code 200.
- Unsuccessful at client side : 4XX code.
  - Error 400 : Bad Request
  - Error 401 : Unauthorized
  - Error 403 : Forbidden
  - Error 460 : Client closed connection.
  - Error 463 : X-Forwarded For header with >30 IP (Similar to malformed request).
- Unsuccessful at server side : 5xx code.
  - An error 500 / Internal server error would mean some error on the ELB itself.
  - Error 502 : Bad Gateway
  - **An error 503 / Service Unavailable**
  - Error 504 / Gateway timeout : probably an issue within the server.
  - Error 561 : Unauthorized

# Supporting SSL for Old Browsers

- Common question is: how do we support Legacy Browsers that has an old TLS (such as TLS 1.0) ?
- Answer: change the policy to allow for weaker cipher (e.g. DES-CBC3-SHA for TLS 1.0)
- Note: only a very small % of the internet uses TLS 1.0
- Elastic Load Balancing provides the following security policies for Application Load Balancers:
  - ELBSecurityPolicy-2016-08
  - ELBSecurityPolicy-FS-2018-06
  - ELBSecurityPolicy-TLS-1-2-2017-01
  - ELBSecurityPolicy-TLS-1-2-Ext-2018-06
  - ELBSecurityPolicy-TLS-1-1-2017-01
  - ELBSecurityPolicy-2015-05
  - ELBSecurityPolicy-TLS-1-0-2015-04 <= allows for the weaker cipher
- See here for more information:  
<https://docs.aws.amazon.com/elasticloadbalancing/latest/application/create-https-listener.html#describe-ssl-policies>

# Load Balancer common troubleshooting

- Check Security Groups
- Check Health Checks
- Sticky sessions may bring “imbalance” on the load balancing side
- For Multi-AZ, make sure cross zone balancing is enabled
- Use **Internal** load balancer for private applications that don’t need a public access
- Enable Deletion Protection to prevent against accidental deletes

# Load Balancers Monitoring

- All Load Balancer metrics are directly pushed to CloudWatch metrics
- BackendConnectionErrors
- HealthyHostCount / UnHealthyHostCount
- HTTPCode\_Backend\_2XX: Successful request.
- HTTPCode\_Backend\_3XX, redirected request
- HTTPCode\_ELB\_4XX: Client error codes
- HTTPCode\_ELB\_5XX: Server error codes generated by the load balancer.
- Latency
- RequestCount
- **SurgeQueueLength:** The total number of requests (HTTP Listener) or connections (TCP listener) that are pending routing to a healthy instance. Help to scale out ASG. Max value is 1024
- **SpilloverCount:** The total number of requests that were rejected because the surge queue is full.

# Load Balancers Access Logs

- Access logs from Load Balancers can be stored in S3 and contain:
  - Time
  - Client IP address
  - Latencies
  - Request paths
  - Server response
  - Trace Id
- Only pay for the S3 storage
- Helpful for compliance reason
- Helpful for keeping access data even after ELB or EC2 instances are terminated
- Access Logs are already encrypted

# Application Load Balancer Request Tracing

- Request tracing – Each HTTP request has an added custom header 'X-Amzn-Trace-Id'

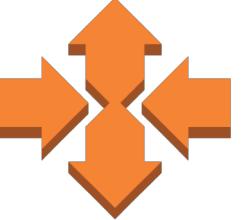
- Example:

X-Amzn-Trace-Id: Root=1-67891233-abcdef012345678912345678

- This is useful in logs / distributed tracing platform to track a single request
- Application Load Balancer is not (yet) integrated with X-Ray

# Load Balancer troubleshooting using metrics

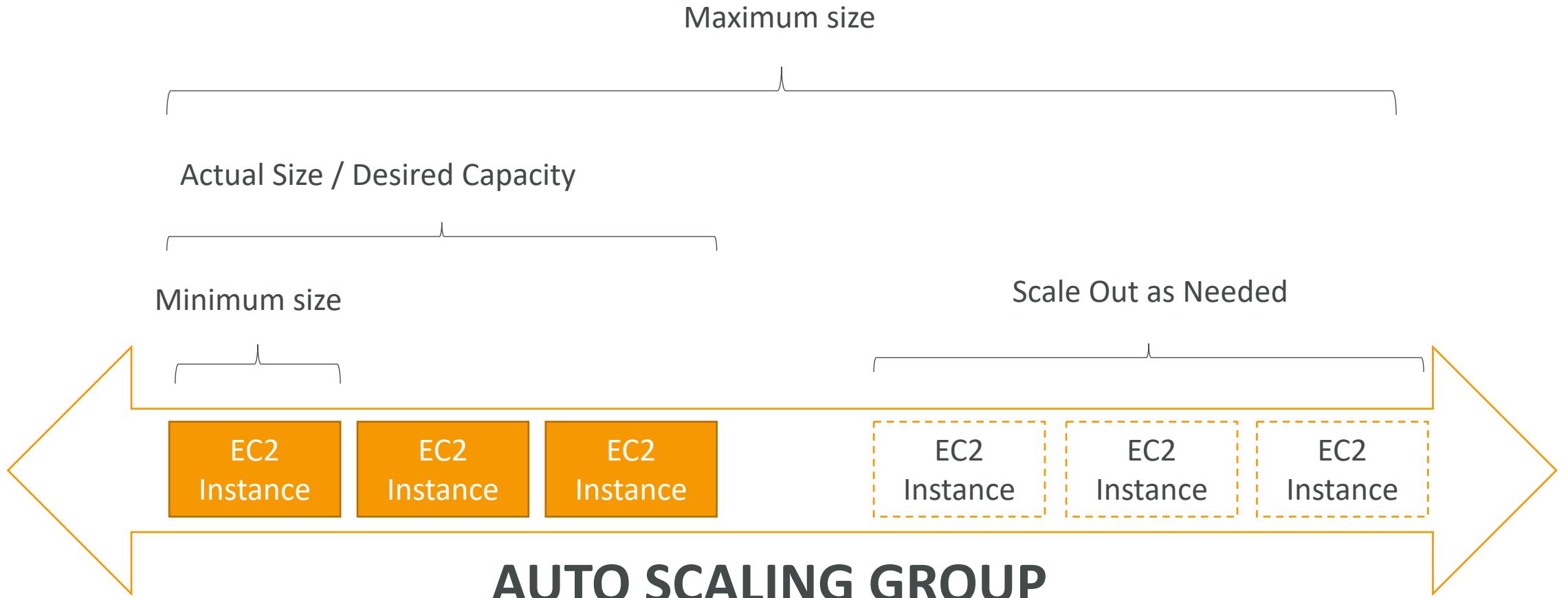
- **HTTP 400: BAD\_REQUEST =>** The client sent a malformed request that does not meet HTTP specifications.
- **HTTP 503: Service Unavailable =>** Ensure that you have healthy instances in every Availability Zone that your load balancer is configured to respond in. Look for HealthyHostCount in CloudWatch
- **HTTP 504: Gateway Timeout =>** Check if keep-alive settings on your EC2 instances are enabled and make sure that the keep-alive timeout is greater than the idle timeout settings of load balancer.
- Set alarms & look at the documentation for troubleshooting:  
<https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-error-message.html>



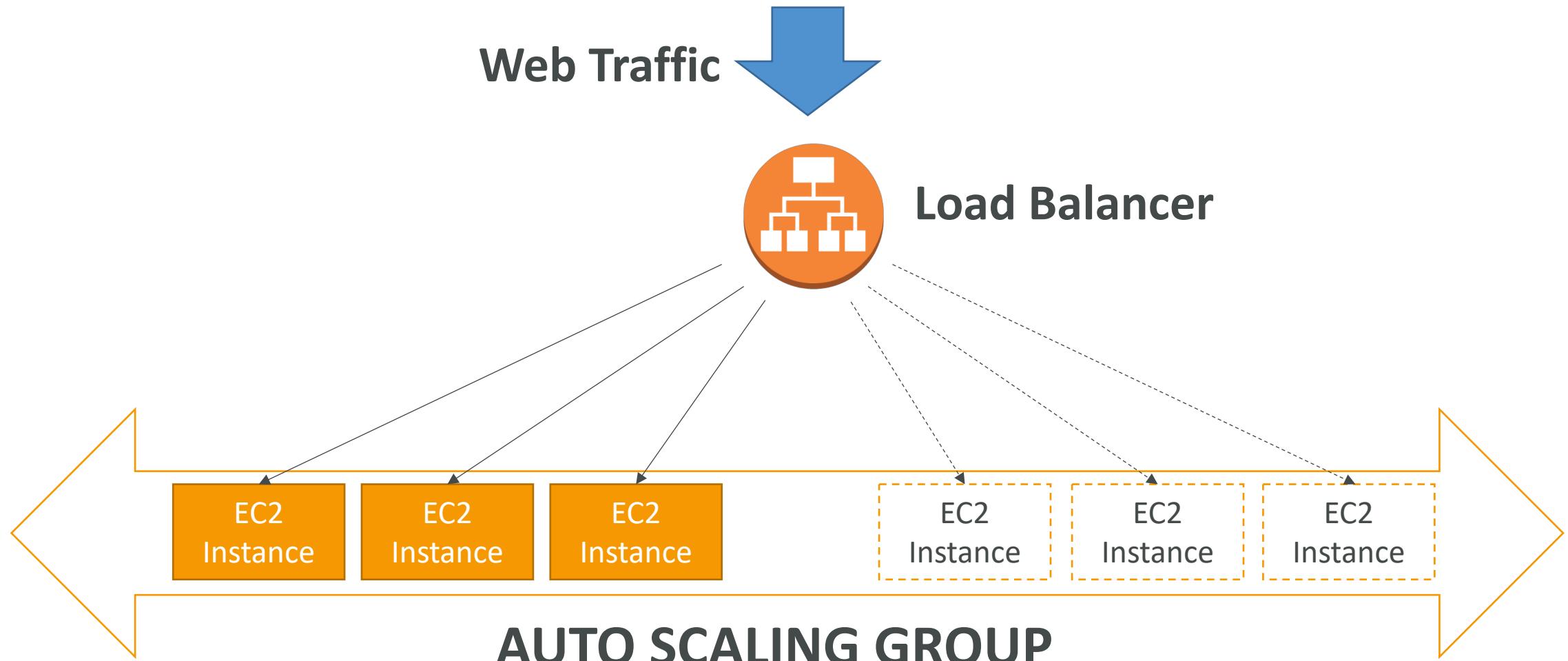
# What's an Auto Scaling Group?

- In real-life, the load on your websites and application can change
- In the cloud, you can create and get rid of servers very quickly
- The goal of an Auto Scaling Group (ASG) is to:
  - Scale out (add EC2 instances) to match an increased load
  - Scale in (remove EC2 instances) to match a decreased load
  - Ensure we have a minimum and a maximum number of machines running
  - Automatically Register new instances to a load balancer

# Auto Scaling Group in AWS



# Auto Scaling Group in AWS With Load Balancer

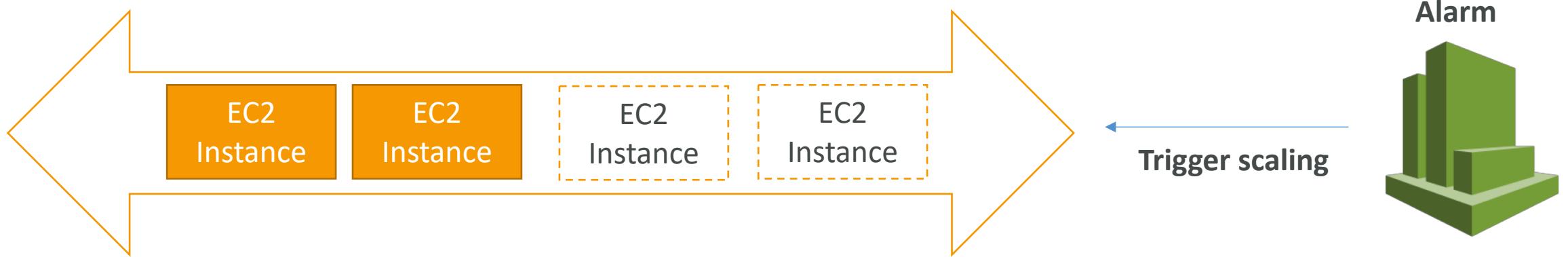


# ASGs have the following attributes

- A launch configuration
  - AMI + Instance Type
  - EC2 User Data
  - EBS Volumes
  - Security Groups
  - SSH Key Pair
- Min Size / Max Size / Initial Capacity
- Network + Subnets Information
- Load Balancer Information
- Scaling Policies

# Auto Scaling Alarms

- It is possible to scale an ASG based on CloudWatch alarms
- An Alarm monitors a metric (such as Average CPU)
- Metrics are computed for the overall ASG instances
- Based on the alarm:
  - We can create scale-out policies (increase the number of instances)
  - We can create scale-in policies (decrease the number of instances)



# Auto Scaling New Rules

- It is now possible to define "better" auto scaling rules that are directly managed by EC2
  - Target Average CPU Usage
  - Number of requests on the ELB per instance
  - Average Network In
  - Average Network Out
- These rules are easier to set up and can make more sense

# Auto Scaling Custom Metric

- We can auto scale based on a custom metric (ex: number of connected users)
- 1. Send custom metric from application on EC2 to CloudWatch (PutMetric API)
- 2. Create CloudWatch alarm to react to low / high values
- 3. Use the CloudWatch alarm as the scaling policy for ASG

# ASG Brain Dump

- Scaling policies can be on CPU, Network... and can even be on custom metrics or based on a schedule (if you know your visitors patterns)
- ASGs use Launch configurations and you update an ASG by providing a new launch configuration
- IAM roles attached to an ASG will get assigned to EC2 instances
- ASG are free. You pay for the underlying resources being launched
- Having instances under an ASG means that if they get terminated for whatever reason, the ASG will restart them. Extra safety!
- ASG can terminate instances marked as unhealthy by an LB (and hence replace them)

# Scaling Processes in ASG

- Launch: Add a new EC2 to the group, increasing the capacity
- Terminate: Removes an EC2 instance from the group, decreasing its capacity.
- HealthCheck: Checks the health of the instances
- ReplaceUnhealthy: Terminate unhealthy instances and re-create them
- AZRebalance: Balancer the number of EC2 instances across AZ
- AlarmNotification: Accept notification from CloudWatch
- ScheduledActions: Performs scheduled actions that you create.
- AddToLoadBalancer: Adds instances to the load balancer or target group
- We can suspend these processes!

# Note on AZRebalance

- AZRebalance = launch new instance then terminate old instance
- If you suspend the Launch process
  - AZRebalance won't launch instances
  - AZRebalance won't terminate instances
- If you suspend the Terminate process
  - The ASG can grow up to 10% of this size (it's allowed during rebalances)
  - The ASG could remain at the increased capacity as it can't terminate instances

# ASG for SysOps

- To make sure you have high availability, means you have least 2 instances running across 2 AZ in your ASG (must configure multi AZ ASG)
- Health checks available:
  - EC2 Status Checks
  - ELB Health Checks
- ASG will launch a new instance after terminating an unhealthy one
- ASG will not reboot unhealthy hosts for you
- Good to know CLI:
  - set-instance-health
  - terminate-instance-in-auto-scaling-group

# Troubleshooting ASG issues

- <number of instances> instance(s) are already running. Launching EC2 instance failed.
  - The Auto Scaling group has reached the limit set by the DesiredCapacity parameter. Update your Auto Scaling group by providing a new value for the desired capacity.
- Launching EC2 instances is failing:
  - The security group does not exist. SG might have been deleted
  - The key pair does not exist. The key pair might have been deleted
- If the ASG fails to launch an instance for over 24 hours, it will automatically suspend the processes (administration suspension)

# CloudWatch Metrics for ASG

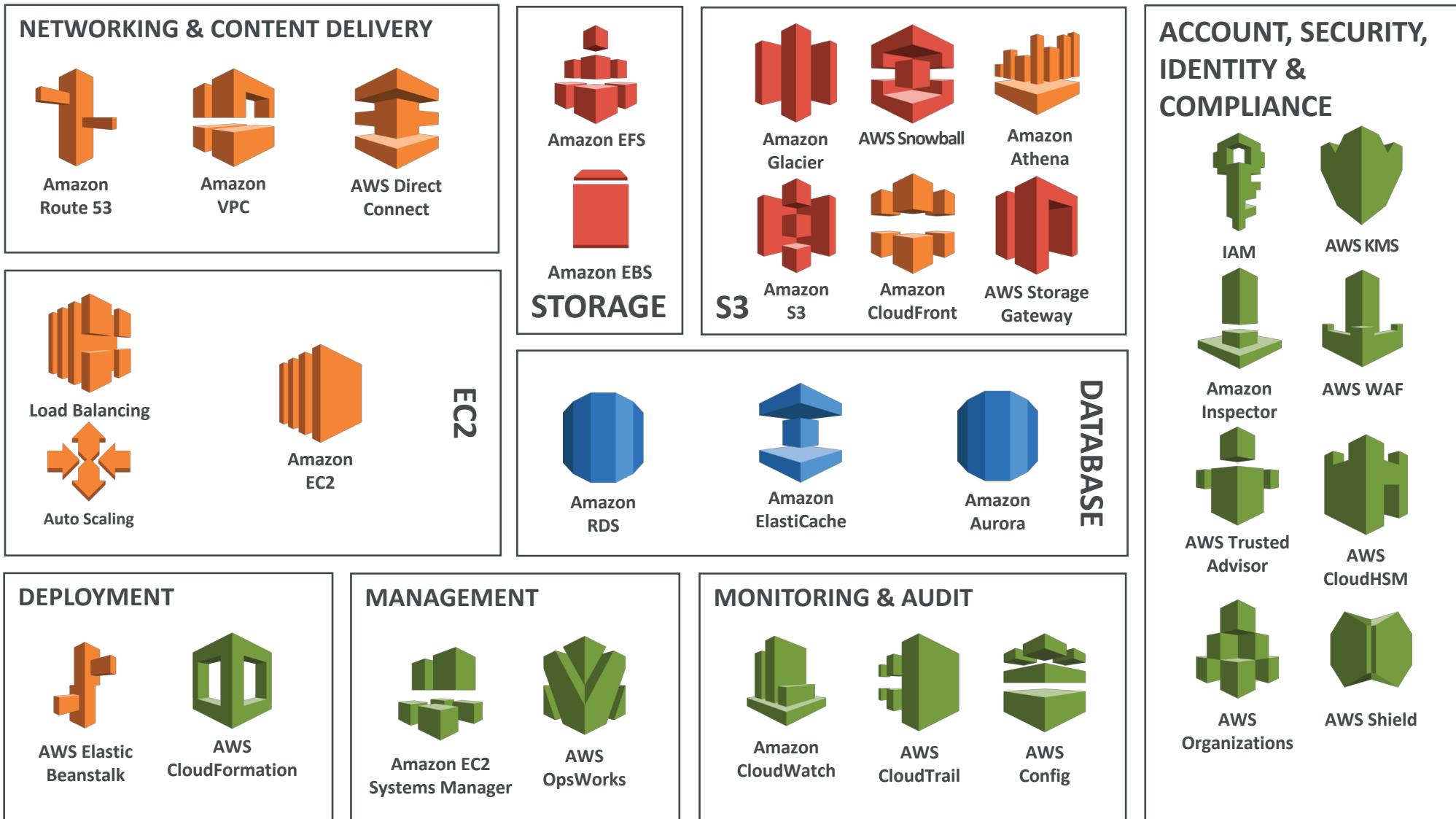
- The following metrics are available for ASG (opt-in):
  - GroupMinSize
  - GroupMaxSize
  - GroupDesiredCapacity
  - GroupInServiceInstances
  - GroupPendingInstances
  - GroupStandbyInstances
  - GroupTerminatingInstances
  - GroupTotalInstances
- You should enable metric collection to see these metrics
- Metrics are collected every 1 minute

# CloudWatch Metrics for ASG

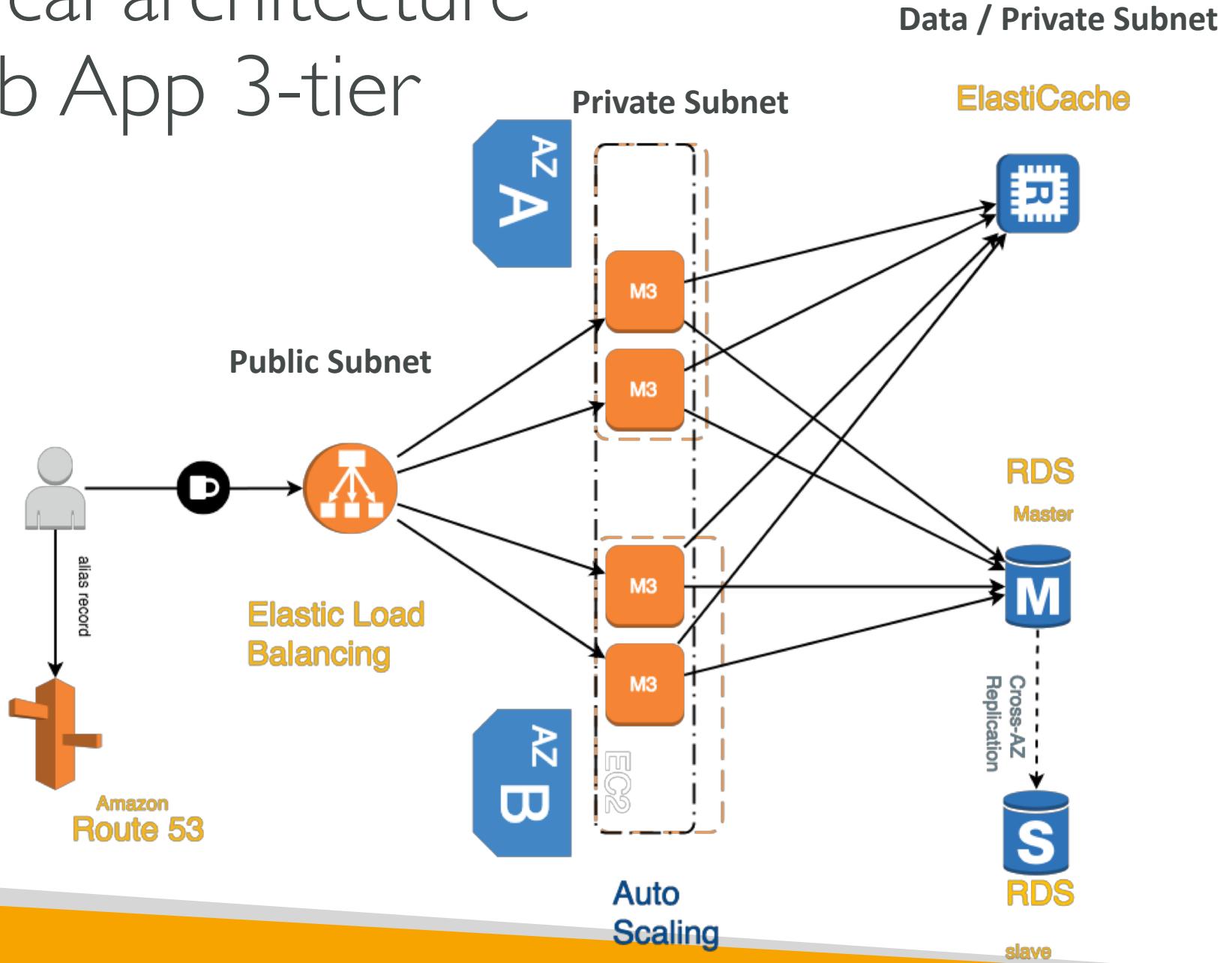
- You can also monitor the underlying EC2
- Basic monitoring: 5 minutes granularity
- Detailed monitoring: 1 minute granularity

# Deployment Automation

Elastic Beanstalk



# Typical architecture Web App 3-tier



# Elastic Beanstalk

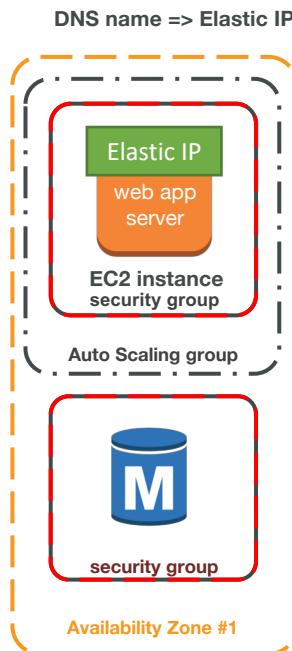
- Managed service
  - Instance configuration / OS is handled by Beanstalk
  - Deployment strategy is configurable but performed by Elastic Beanstalk
- Just the application code is the responsibility of the developer
- Beanstalk = deploy the application without managing the infrastructure
- Three architecture models:
  - Single Instance deployment: good for dev
  - LB + ASG: great for production or pre-production web applications
  - ASG only: great for non-web apps in production (workers, etc..)

# Elastic Beanstalk

- Support for many platforms:
  - Go
  - Java SE
  - Java with Tomcat
  - .NET on Windows Server with IIS
  - Node.js
  - PHP
  - Python
  - Ruby
  - Packer Builder
- Single Container Docker
- Multicontainer Docker
- Preconfigured Docker
- If not supported, you can write your custom platform (advanced)

# Elastic Beanstalk Deployment Modes

## Single Instance Great for dev



## High Availability with Load Balancer Great for prod



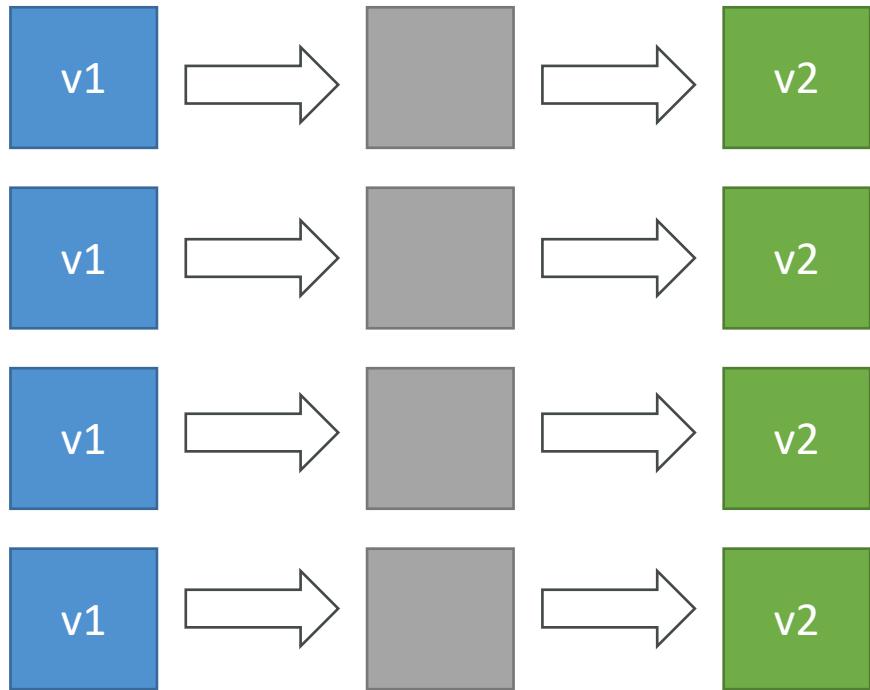
# Beanstalk Deployment Options for Updates

- **All at once (deploy all in one go)** – fastest, but instances aren't available to serve traffic for a bit (downtime)
- **Rolling**: update a few instances at a time (bucket), and then move onto the next bucket once the first bucket is healthy
- **Rolling with additional batches**: like rolling, but spins up new instances to move the batch (so that the old application is still available)
- **Immutable**: spins up new instances in a new ASG, deploys version to these instances, and then swaps all the instances when everything is healthy

# Elastic Beanstalk Deployment

## All at once

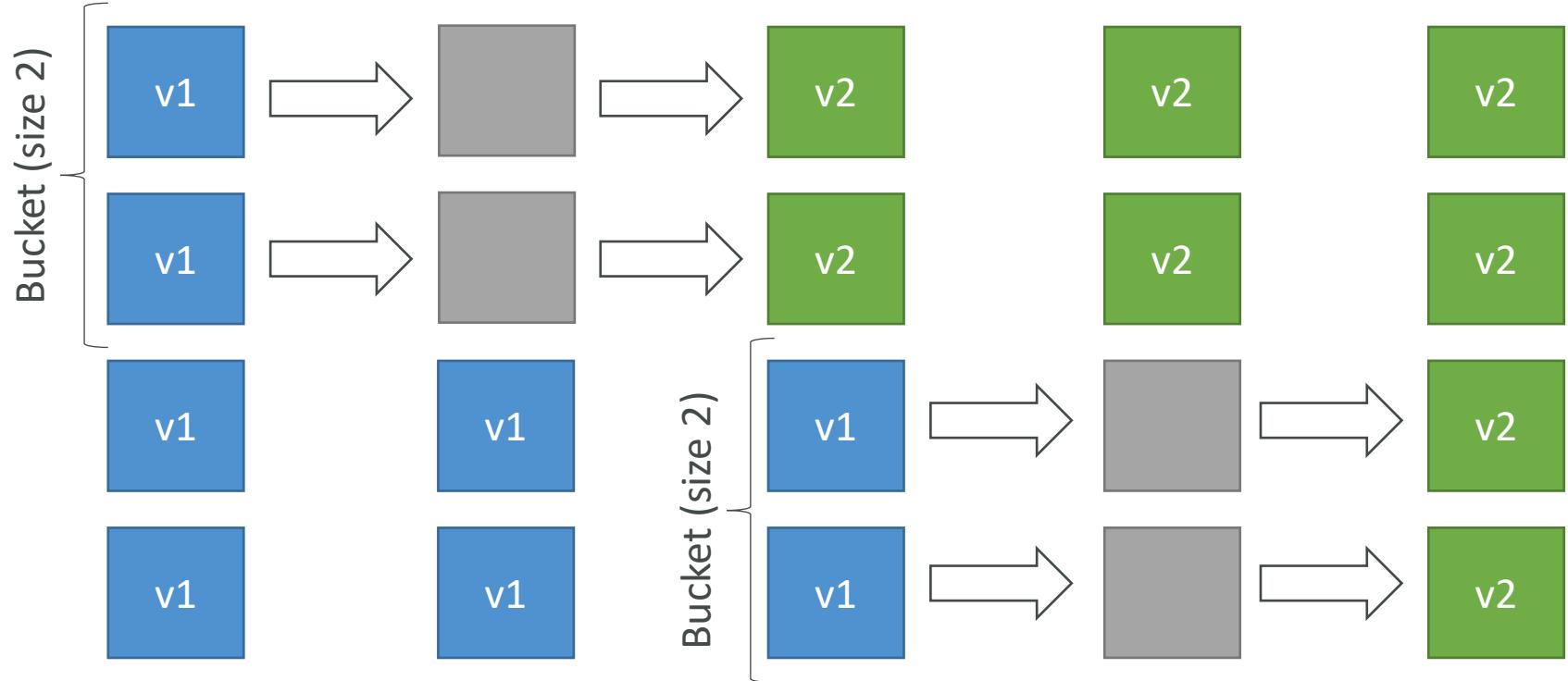
- Fastest deployment
- Application has downtime
- Great for quick iterations in development environment
- No additional cost



# Elastic Beanstalk Deployment

## Rolling

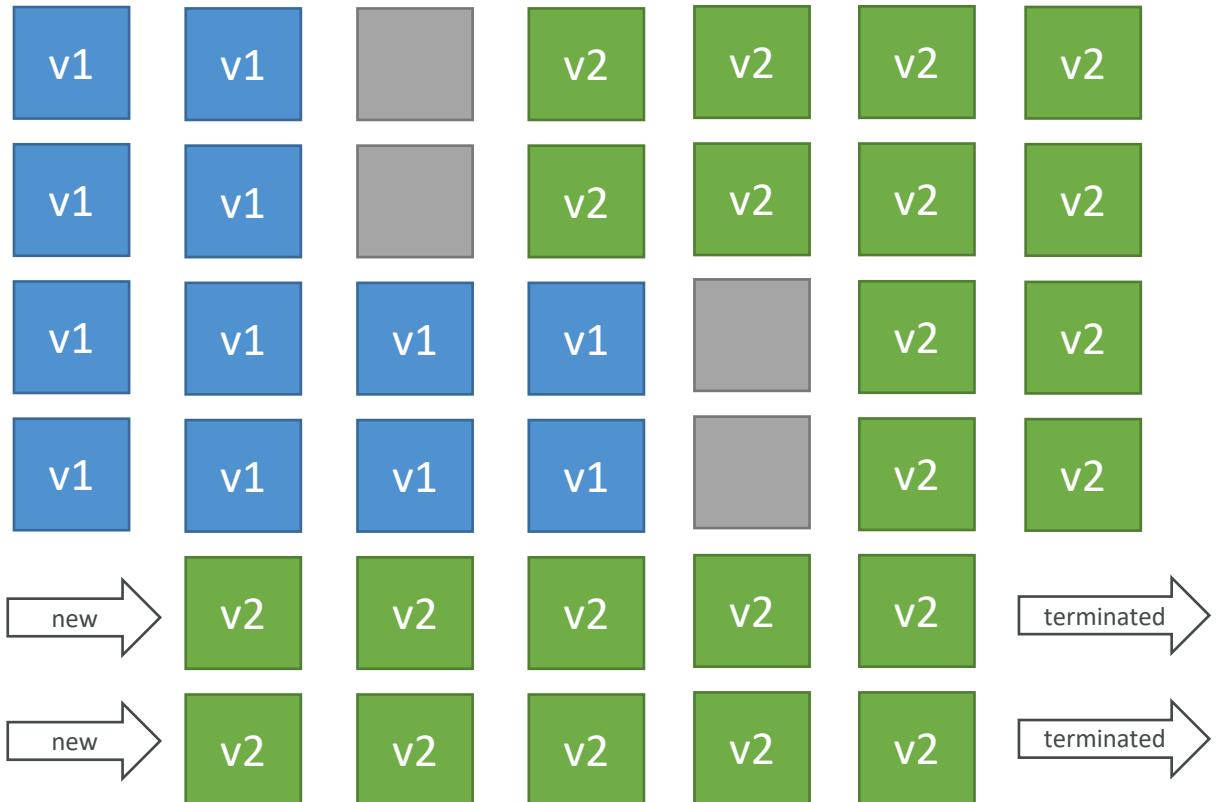
- Application is running below capacity
- Can set the bucket size
- Application is running both versions simultaneously
- No additional cost
- Long deployment



# Elastic Beanstalk Deployment

## Rolling with additional batches

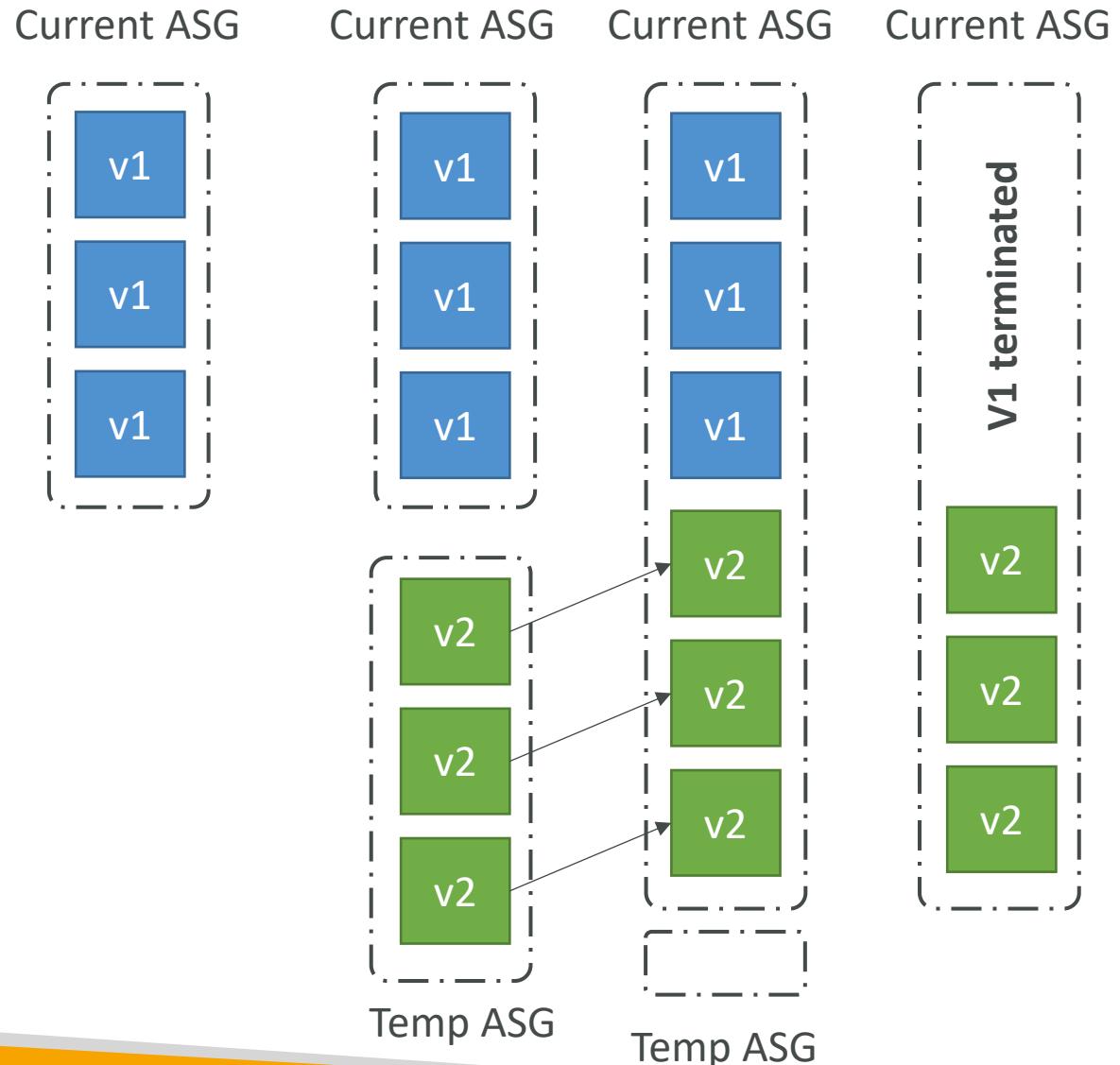
- Application is running at capacity
- Can set the bucket size
- Application is running both versions simultaneously
- Small additional cost
- Additional batch is removed at the end of the deployment
- Longer deployment
- Good for prod



# Elastic Beanstalk Deployment

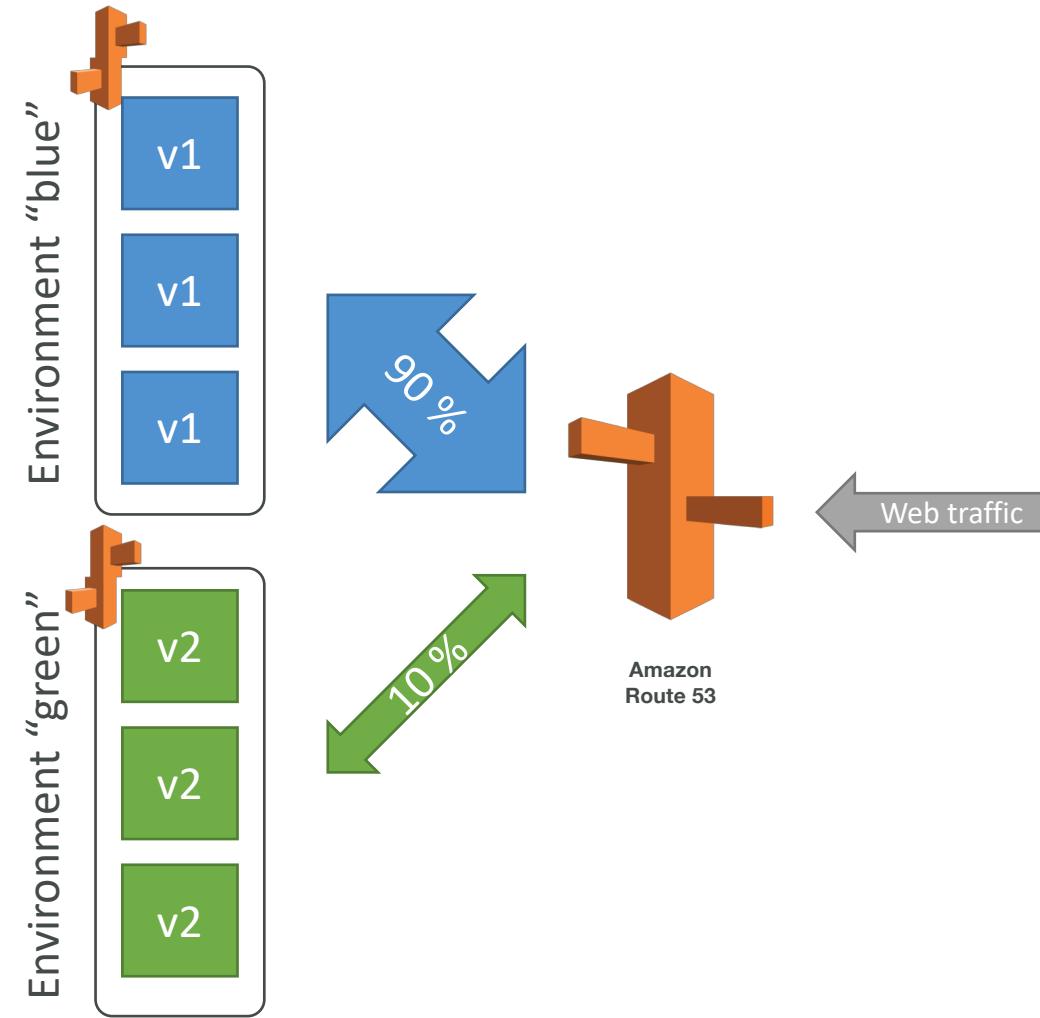
## Immutable

- Zero downtime
- New Code is deployed to new instances on a temporary ASG
- High cost, double capacity
- Longest deployment
- Quick rollback in case of failures  
(just terminate new ASG)
- Great for prod



# Elastic Beanstalk Deployment Blue / Green

- Not a “direct feature” of Elastic Beanstalk
- Zero downtime and release facility
- Create a new “stage” environment and deploy v2 there
- The new environment (green) can be validated independently and roll back if issues
- Route 53 can be setup using weighted policies to redirect a little bit of traffic to the stage environment
- Using Beanstalk, “swap URLs” when done with the environment test



# Elastic Beanstalk Deployment Summary from AWS Doc

- <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.deploy-existing-version.html>

Deployment Methods							
Method	Impact of Failed Deployment	Deploy Time	Zero Downtime	No DNS Change	Rollback Process	Code Deployed To	
All at once	Downtime	⊕	X	✓	Manual Redeploy	Existing instances	
Rolling	Single batch out of service; any successful batches prior to failure running new application version	⊕ ⊕ †	✓	✓	Manual Redeploy	Existing instances	
Rolling with additional batch	Minimal if first batch fails, otherwise, similar to Rolling	⊕ ⊕ ⊕ †	✓	✓	Manual Redeploy	New and existing instances	
Immutable	Minimal	⊕ ⊕ ⊕ ⊕	✓	✓	Terminate New Instances	New instances	
Blue/green	Minimal	⊕ ⊕ ⊕ ⊕	✓	X	Swap URL	New instances	

# Beanstalk for SysOps

- Beanstalk can put applications logs into CloudWatch Logs
- You manage the application, AWS will manage the underlying infrastructure
- Know the different deployment modes for your application
- Custom domain: Route 53 ALIAS or CNAME on top of Beanstalk URL
- You are not responsible for patching the runtimes (Node.js, PHP, etc...)
- Questions are very basic compared to Developer Exam

# How Beanstalk deploys applications

## Ex: Rolling

1. EC2 has a base AMI (can configure)
2. EC2 gets the new code of the app
3. EC2 resolves the app dependencies (can take a while)
4. Apps get swapped on the EC2 instance

Resolving dependencies can take a long time!

We can use Golden AMI to fix the problem

# Golden AMI

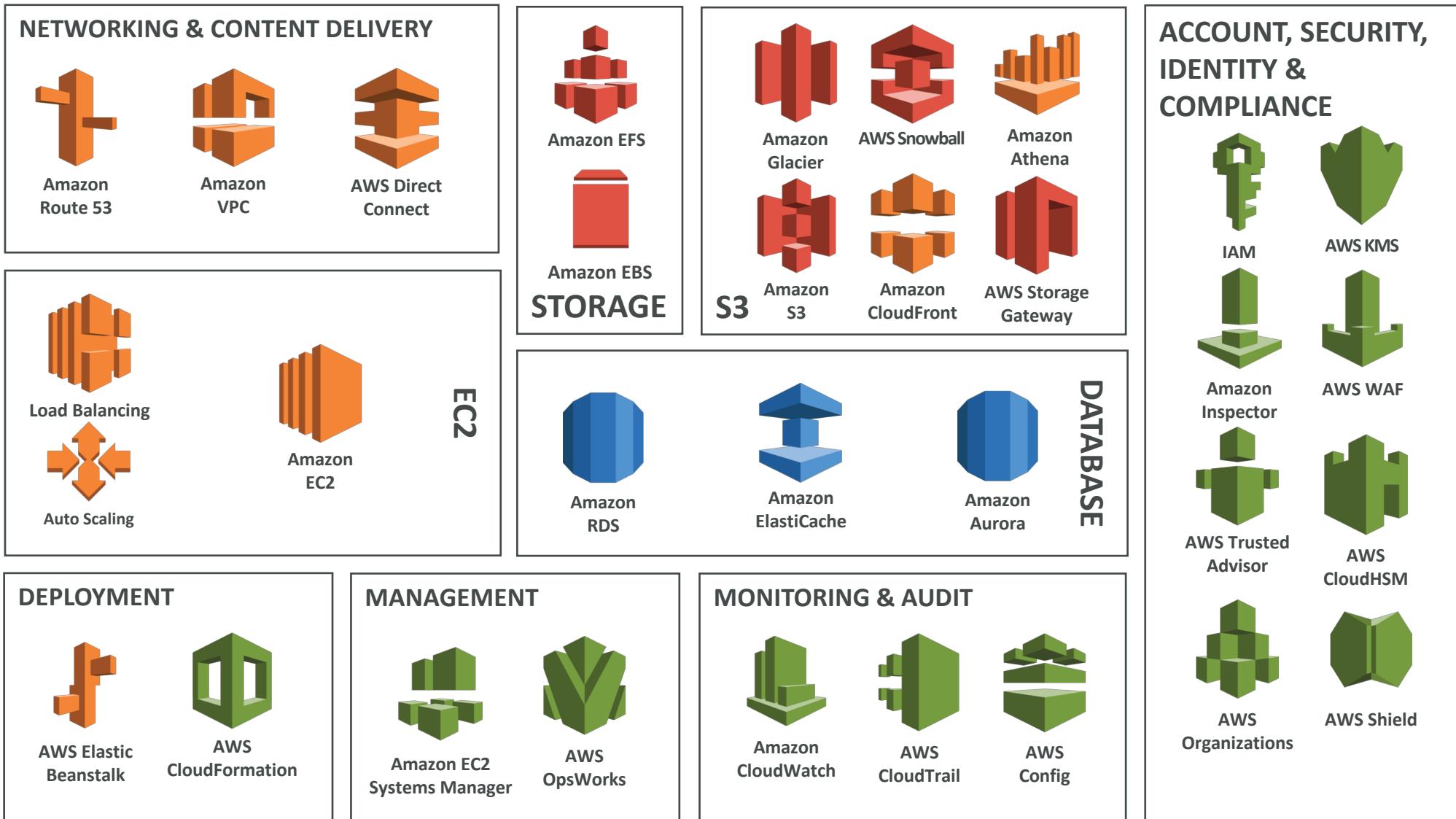
- If your application has a lot of application or OS dependencies, and you want to deploy as quickly as possible, you should create a **Golden AMI**:
- Golden AMI = standardized company-specific AMI with:
  - Package OS dependencies
  - Package App dependencies
  - Package company-wide software
- By using a Golden AMI to deploy to Beanstalk (in combination of blue/green new ASG deployment), our application won't need to resolve dependencies or a long time to configure!

# Troubleshooting Beanstalk

- If the health of your environment changes to red, try the following :
  - Review environment events
  - Pull logs to view recent log file entries
  - Roll back to a previous, working version of the application
- When accessing external resources, make sure the security groups are correctly configured
- In case of command timeouts, you can increase the deployment timeout

# AWS CloudFormation

Managing your infrastructure as code



# CloudFormation Section

- Very important exam topic!
- Basics lectures included from the developer course
- Advanced SysOps lectures added afterwards
  - Troubleshooting
  - Advanced CloudFormation options

# Infrastructure as Code

- Currently, we have been doing a lot of manual work
- All this manual work will be very tough to reproduce:
  - In another region
  - in another AWS account
  - Within the same region if everything was deleted
- Wouldn't it be great, if all our infrastructure was... code?
- That code would be deployed and create / update / delete our infrastructure

# What is CloudFormation



- CloudFormation is a declarative way of outlining your AWS Infrastructure, for any resources (most of them are supported).
- For example, within a CloudFormation template, you say:
  - I want a security group
  - I want two EC2 machines using this security group
  - I want two Elastic IPs for these EC2 machines
  - I want an S3 bucket
  - I want a load balancer (ELB) in front of these machines
- Then CloudFormation creates those for you, in the **right order**, with the **exact configuration** that you specify

# Benefits of AWS CloudFormation (1/2)

- Infrastructure as code
  - No resources are manually created, which is excellent for control
  - The code can be version controlled for example using git
  - Changes to the infrastructure are reviewed through code
- Cost
  - Each resources within the stack is tagged with an identifier so you can easily see how much a stack costs you
  - You can estimate the costs of your resources using the CloudFormation template
  - Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely

# Benefits of AWS CloudFormation (2/2)

- Productivity
  - Ability to destroy and re-create an infrastructure on the cloud on the fly
  - Automated generation of Diagram for your templates!
  - Declarative programming (no need to figure out ordering and orchestration)
- Separation of concern: create many stacks for many apps, and many layers. Ex:
  - VPC stacks
  - Network stacks
  - App stacks
- Don't re-invent the wheel
  - Leverage existing templates on the web!
  - Leverage the documentation

# How CloudFormation Works

- Templates have to be uploaded in S3 and then referenced in CloudFormation
- To update a template, we can't edit previous ones. We have to re-upload a new version of the template to AWS
- Stacks are identified by a name
- Deleting a stack deletes every single artifact that was created by CloudFormation.

# Deploying CloudFormation templates

- Manual way:
  - Editing templates in the CloudFormation Designer
  - Using the console to input parameters, etc
- Automated way:
  - Editing templates in a YAML file
  - Using the AWS CLI (Command Line Interface) to deploy the templates
  - Recommended way when you fully want to automate your flow

# CloudFormation Building Blocks

Templates components (one course section for each):

1. Resources: your AWS resources declared in the template (**MANDATORY**)
2. Parameters: the dynamic inputs for your template
3. Mappings: the static variables for your template
4. Outputs: References to what has been created
5. Conditionals: List of conditions to perform resource creation
6. Metadata

Templates helpers:

1. References
2. Functions

Note:

# This is an introduction to CloudFormation

- It can take over 3 hours to properly learn and master CloudFormation
- This section is meant so you get a good idea of how it works
- We'll be slightly less hands-on than in other sections
  
- We'll learn everything we need to answer questions for the exam
- The exam does not require you to actually write CloudFormation
- The exam expects you to understand how to read CloudFormation

# Introductory Example

- We're going to create a simple EC2 instance.
  - Then we're going to create to add an Elastic IP to it
  - And we're going to add two security groups to it
  - For now, forget about the code syntax.
  - We'll look at the structure of the files later on
- 
- We'll see how in no-time, we are able to get started with CloudFormation!



# YAML Crash Course

```
1  invoice:      34843
2  date   :     2001-01-23
3  bill-to:
4    given  :   Chris
5    family :  Dumars
6    address:
7      lines: |
8        458 Walkman Dr.
9        Suite #292
10       city   : Royal Oak
11       state  : MI
12       postal : 48046
13 product:
14   - sku        : BL394D
15     quantity   : 4
16     description: Basketball
17     price      : 450.00
18   - sku        : BL4438H
19     quantity   : 1
20     description: Super Hoop
21     price      : 2392.00
```

- YAML and JSON are the languages you can use for CloudFormation.
  - JSON is horrible for CF
  - YAML is great in so many ways
  - Let's learn a bit about it!
- 
- Key value Pairs
  - Nested objects
  - Support Arrays
  - Multi line strings
  - Can include comments!

# What are resources?

- Resources are the core of your CloudFormation template (MANDATORY)
- They represent the different AWS Components that will be created and configured
- Resources are declared and can reference each other
- AWS figures out creation, updates and deletes of resources for us
- There are over 224 types of resources (!)
- Resource types identifiers are of the form:

**AWS::aws-product-name::data-type-name**

# How do I find resources documentation?

- I can't teach you all of the 224 resources, but I can teach you how to learn how to use them.
- All the resources can be found here:  
<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html>
- Then, we just read the docs ☺
- Example here (for an EC2 instance):  
<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html>

# Analysis of CloudFormation Template

- Going back to the example of the introductory section, let's learn why it was written this way.
- Relevant documentation can be found here:
  - <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html>
  - <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-security-group.html>
  - <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-eip.html>

# FAQ for resources

- Can I create a dynamic amount of resources?
  - No, you can't. Everything in the CloudFormation template has to be declared. You can't perform code generation there
- Is every AWS Service supported?
  - Almost. Only a select few niches are not there yet
  - You can work around that using AWS Lambda Custom Resources

# What are parameters?

- Parameters are a way to provide inputs to your AWS CloudFormation template
- They're important to know about if:
  - You want to reuse your templates across the company
  - Some inputs can not be determined ahead of time
- Parameters are extremely powerful, controlled, and can prevent errors from happening in your templates thanks to types.

# When should you use a parameter?

- Ask yourself this:
  - Is this CloudFormation resource configuration likely to change in the future?
  - If so, make it a parameter.
- You won't have to re-upload a template to change its content ☺

**Parameters:**

**SecurityGroupDescription:**

**Description:** Security Group Description  
**(Simple parameter)**

**Type:** String

# Parameters Settings

Parameters can be controlled by all these settings:

- **Type:**
  - String
  - Number
  - CommaDelimitedList
  - List<Type>
  - AWS Parameter (to help catch invalid values – match against existing values in the AWS Account)
- **Description**
- **Constraints**
  - ConstraintDescription (String)
  - Min/MaxLength
  - Min/MaxValue
  - Defaults
  - AllowedValues (array)
  - AllowedPattern (regexp)
  - NoEcho (Boolean)

# How to Reference a Parameter

- The `Fn::Ref` function can be leveraged to reference parameters
- Parameters can be used anywhere in a template.
- The shorthand for this in YAML is `!Ref`
- The function can also reference other elements within the template

```
DbSubnet1:  
  Type: AWS::EC2::Subnet  
  Properties:  
    VpcId: !Ref MyVPC
```

# Concept: Pseudo Parameters

- AWS offers us pseudo parameters in any CloudFormation template.
- These can be used at any time and are enabled by default

Reference Value	Example Return Value
AWS::AccountId	1234567890
AWS::NotificationARNs	[arn:aws:sns:us-east-1:123456789012:MyTopic]
AWS::NoValue	Does not return a value.
AWS::Region	us-east-2
AWS::StackId	arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/1c2fa620-982a-11e3-aff7-50e2416294e0
AWS::StackName	MyStack

# What are mappings?

- Mappings are fixed variables within your CloudFormation Template.
- They're very handy to differentiate between different environments (dev vs prod), regions (AWS regions), AMI types, etc
- All the values are hardcoded within the template
- Example:

```
Mappings:  
  Mapping01:  
    Key01:  
      Name: Value01  
    Key02:  
      Name: Value02  
    Key03:  
      Name: Value03
```

```
RegionMap:  
  us-east-1:  
    "32": "ami-6411e20d"  
    "64": "ami-7a11e213"  
  us-west-1:  
    "32": "ami-c9c7978c"  
    "64": "ami-cfc7978a"  
  eu-west-1:  
    "32": "ami-37c2f643"  
    "64": "ami-31c2f645"
```

# When would you use mappings vs parameters ?

- Mappings are great when you know in advance all the values that can be taken and that they can be deduced from variables such as
  - Region
  - Availability Zone
  - AWS Account
  - Environment (dev vs prod)
  - Etc...
- They allow safer control over the template.
- Use parameters when the values are really user specific

# Fn::FindInMap

## Accessing Mapping Values

- We use **Fn::FindInMap** to return a named value from a specific key
- **!FindInMap [ MapName, TopLevelKey, SecondLevelKey ]**

```
AWSTemplateFormatVersion: "2010-09-09"
Mappings:
  RegionMap:
    us-east-1:
      "32": "ami-6411e20d"
      "64": "ami-7a11e213"
    us-west-1:
      "32": "ami-c9c7978c"
      "64": "ami-cfc7978a"
    eu-west-1:
      "32": "ami-37c2f643"
      "64": "ami-31c2f645"
    ap-southeast-1:
      "32": "ami-66f28c34"
      "64": "ami-60f28c32"
    ap-northeast-1:
      "32": "ami-9c03a89d"
      "64": "ami-a003a8a1"
Resources:
  myEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", 32]
      InstanceType: m1.small
```

# What are outputs?

- The Outputs section declares *optional* outputs values that we can import into other stacks (if you export them first)!
- You can also view the outputs in the AWS Console or in using the AWS CLI
- They're very useful for example if you define a network CloudFormation, and output the variables such as VPC ID and your Subnet IDs
- It's the best way to perform some collaboration cross stack, as you let expert handle their own part of the stack
- You can't delete a CloudFormation Stack if its outputs are being referenced by another CloudFormation stack

# Outputs Example

- Creating a SSH Security Group as part of one template
- We create an output that references that security group

**Outputs:**

**StackSSHSecurityGroup:**

**Description:** The SSH Security Group for our Company

**Value:** !Ref MyCompanyWideSSHSecurityGroup

**Export:**

**Name:** SSHSecurityGroup

# Cross Stack Reference

- We then create a second template that leverages that security group
- For this, we use the **Fn::ImportValue** function
- You can't delete the underlying stack until all the references are deleted too.

```
Resources:  
  MySecureInstance:  
    Type: AWS::EC2::Instance  
    Properties:  
      AvailabilityZone: us-east-1a  
      ImageId: ami-a4c7edb2  
      InstanceType: t2.micro  
      SecurityGroups:  
        - !ImportValue SSHSecurityGroup
```

# What are conditions used for?

- Conditions are used to control the creation of resources or outputs based on a condition.
- Conditions can be whatever you want them to be, but common ones are:
  - Environment (dev / test / prod)
  - AWS Region
  - Any parameter value
- Each condition can reference another condition, parameter value or mapping

# How to define a condition?

## Conditions:

```
| CreateProdResources: !Equals [ !Ref EnvType, prod ]
```

- The logical ID is for you to choose. It's how you name condition
- The intrinsic function (logical) can be any of the following:
  - Fn::And
  - Fn::Equals
  - Fn::If
  - Fn::Not
  - Fn::Or

# Using a Condition

- Conditions can be applied to resources / outputs / etc...

```
Resources:
```

```
  MountPoint:
```

```
    Type: "AWS::EC2::VolumeAttachment"
```

```
    Condition: CreateProdResources
```

# CloudFormation

## Must Know Intrinsic Functions

- Ref
- Fn::GetAtt
- Fn::FindInMap
- Fn::ImportValue
- Fn::Join
- Fn::Sub
- Condition Functions (Fn::If, Fn::Not, Fn::Equals, etc...)

# Fn::Ref

- The Fn::Ref function can be leveraged to reference
  - Parameters => returns the value of the parameter
  - Resources => returns the physical ID of the underlying resource (ex: EC2 ID)
- The shorthand for this in YAML is !Ref

```
DbSubnet1:  
  Type: AWS::EC2::Subnet  
  Properties:  
    VpcId: !Ref MyVPC
```

# Fn::GetAtt

- Attributes are attached to any resources you create
- To know the attributes of your resources, the best place to look at is the documentation.
- For example: the AZ of an EC2 machine!

**Resources:**

**EC2Instance:**

**Type:** "AWS::EC2::Instance"

**Properties:**

**ImageId:** ami-1234567

**InstanceType:** t2.micro

**NewVolume:**

**Type:** "AWS::EC2::Volume"

**Condition:** CreateProdResources

**Properties:**

**Size:** 100

**AvailabilityZone:**

**!GetAtt** EC2Instance.AvailabilityZone

# Fn::FindInMap

## Accessing Mapping Values

- We use **Fn::FindInMap** to return a named value from a specific key
- **!FindInMap [ MapName, TopLevelKey, SecondLevelKey ]**

```
AWSTemplateFormatVersion: "2010-09-09"
Mappings:
  RegionMap:
    us-east-1:
      "32": "ami-6411e20d"
      "64": "ami-7a11e213"
    us-west-1:
      "32": "ami-c9c7978c"
      "64": "ami-cfc7978a"
    eu-west-1:
      "32": "ami-37c2f643"
      "64": "ami-31c2f645"
    ap-southeast-1:
      "32": "ami-66f28c34"
      "64": "ami-60f28c32"
    ap-northeast-1:
      "32": "ami-9c03a89d"
      "64": "ami-a003a8a1"
Resources:
  myEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", 32]
      InstanceType: m1.small
```

# Fn::ImportValue

- Import values that are exported in other templates
- For this, we use the **Fn::ImportValue** function

```
Resources:  
  MySecureInstance:  
    Type: AWS::EC2::Instance  
    Properties:  
      AvailabilityZone: us-east-1a  
      ImageId: ami-a4c7edb2  
      InstanceType: t2.micro  
      SecurityGroups:  
        - !ImportValue SSHSecurityGroup
```

# Fn::Join

- Join values with a delimiter

```
!Join [ delimiter, [ comma-delimited list of values ] ]
```

- This creates “a:b:c”

```
!Join [ ":", [ a, b, c ] ]
```

# Function Fn::Sub

- Fn::Sub, or !Sub as a shorthand, is used to substitute variables from a text. It's a very handy function that will allow you to fully customize your templates.
- For example, you can combine Fn::Sub with References or AWS Pseudo variables!
- String must contain \${VariableName} and will substitute them

```
!Sub
- String
- { Var1Name: Var1Value, Var2Name: Var2Value }
```

```
!Sub String
```

# Condition Functions

**Conditions:**

```
| CreateProdResources: !Equals [ !Ref EnvType, prod ]
```

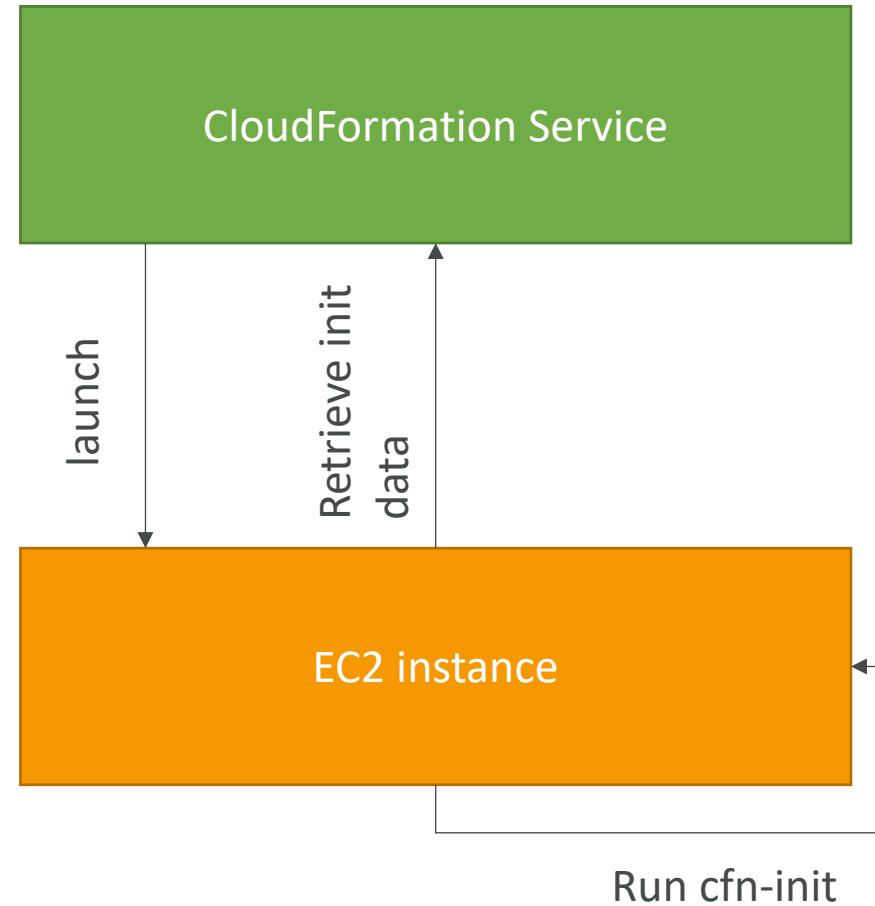
- The logical ID is for you to choose. It's how you name condition
- The intrinsic function (logical) can be any of the following:
  - Fn::And
  - Fn::Equals
  - Fn::If
  - Fn::Not
  - Fn::Or

# User Data in EC2 for CloudFormation

- We can have user data at EC2 instance launch through the console
- We can also include it in CloudFormation
- The important thing to pass is the entire script through the function Fn::Base64
- Good to know: user data script log is in /var/log/cloud-init-output.log
- Let's see how to do this in CloudFormation

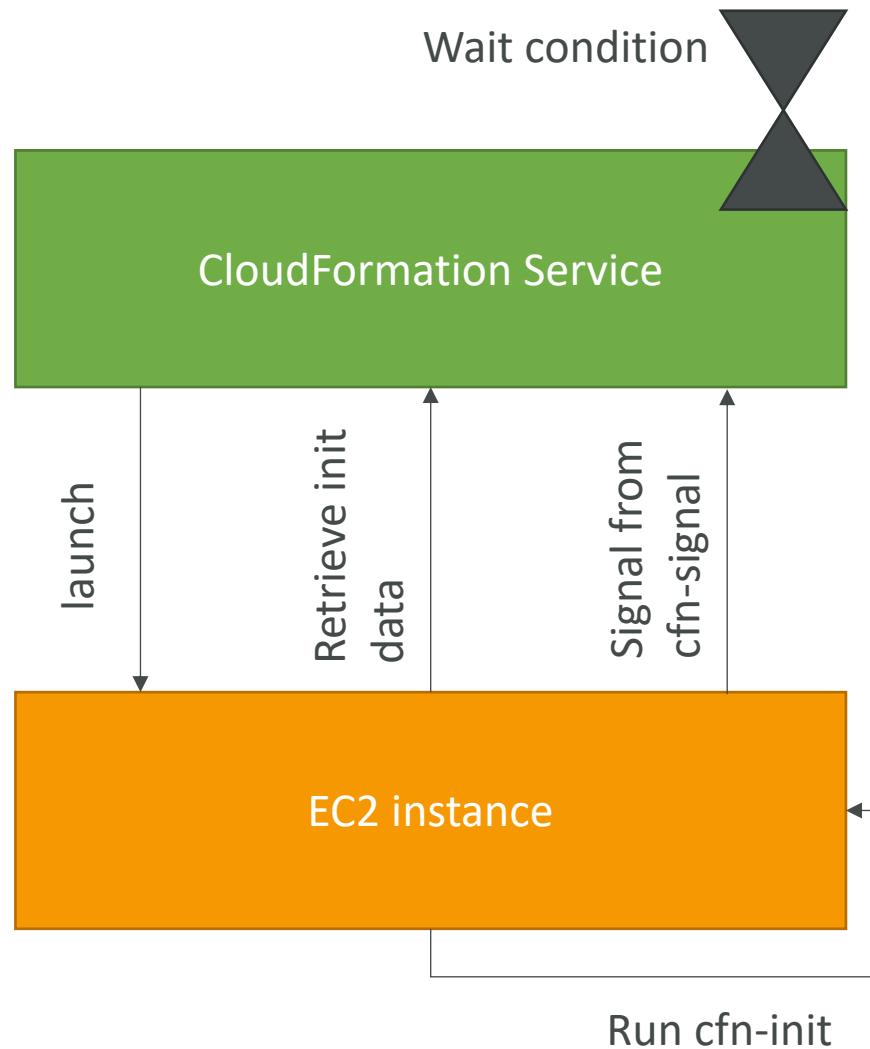
# cfn-init

- AWS::CloudFormation::Init must be in the Metadata of a resource
- With the cfn-init script, it helps make complex EC2 configurations readable
- The EC2 instance will query the CloudFormation service to get init data
- Logs go to `/var/log/cfn-init.log`
- Let's see how it works through a sample CloudFormation



# cfn-signal & wait conditions

- We still don't know how to tell CloudFormation that the EC2 instance got properly configured after a **cfn-init**
- For this, we can use the **cfn-signal** script!
  - We run cfn-signal right after cfn-init
  - Tell CloudFormation service to keep on going or fail
- We need to define **WaitCondition**:
  - Block the template until it receives a signal from cfn-signal
  - We attach a **CreationPolicy** (also works on EC2, ASG)



# Wait Condition Didn't Receive the Required Number of Signals from an Amazon EC2 Instance

- Ensure that the AMI you're using has the AWS CloudFormation helper scripts installed. If the AMI doesn't include the helper scripts, you can also download them to your instance.
- Verify that the cfn-init & cfn-signal command was successfully run on the instance. You can view logs, such as /var/log/cloud-init.log or /var/log/cfn-init.log, to help you debug the instance launch.
- You can retrieve the logs by logging in to your instance, but you must disable rollback on failure or else AWS CloudFormation deletes the instance after your stack fails to create.
- Verify that the instance has a connection to the Internet. If the instance is in a VPC, the instance should be able to connect to the Internet through a NAT device if it's in a private subnet or through an Internet gateway if it's in a public subnet.
- For example, run: curl -I <https://aws.amazon.com>

# Rollbacks on failures

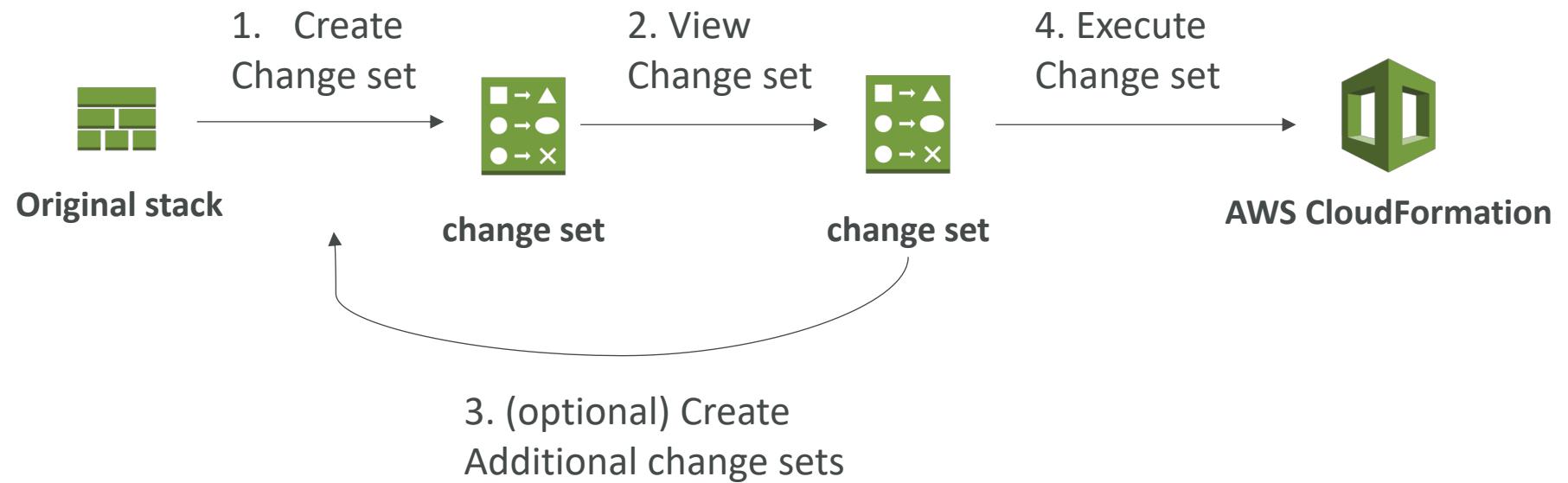
- Stack Creation Fails: (CreateStack API)
  - Default: everything rolls back (gets deleted). We can look at the log  
**OnFailure=ROLLBACK**
  - Troubleshoot: Option to disable rollback and manually troubleshoot  
**OnFailure=DO NOTHING**
  - Delete: get rid of the stack entirely, do not keep anything  
**OnFailure=DELETE**
- Stack Update Fails: (UpdateStack API)
  - The stack automatically rolls back to the previous known working state
  - Ability to see in the log what happened and error messages

# Nested stacks

- Nested stacks are stacks as part of other stacks
- They allow you to isolate repeated patterns / common components in separate stacks and call them from other stacks
- Example:
  - Load Balancer configuration that is re-used
  - Security Group that is re-used
- Nested stacks are considered best practice
- To update a nested stack, always update the parent (root stack)

# ChangeSets

- When you update a stack, you need to know what changes before it happens for greater confidence
- ChangeSets won't say if the update will be successful



From: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-changesets.html>

# Retaining Data on Deletes

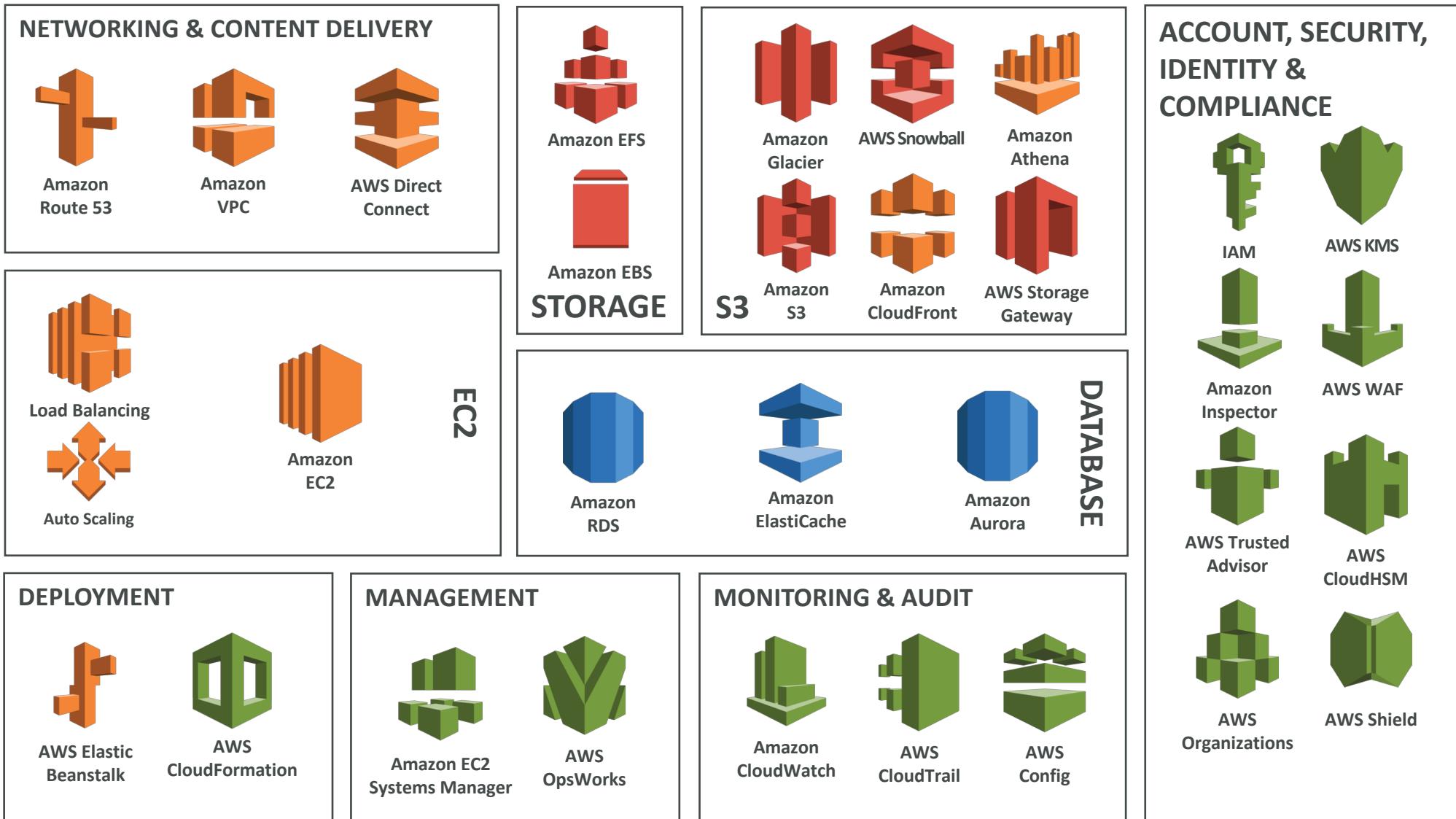
- You can put a `DeletionPolicy` on any resource to control what happens when the CloudFormation template is deleted
- `DeletionPolicy=Retain`:
  - Specify on resources to preserve / backup in case of CloudFormation deletes
  - To keep a resource, specify `Retain` (works for any resource / nested stack)
- `DeletionPolicy=Snapshot`:
  - EBS Volume, ElastiCache Cluster, ElastiCache ReplicationGroup
  - RDS DBInstance, RDS DBCluster, Redshift Cluster
- `DeletePolicy=Delete` (default behavior):
  - Note: for `AWS::RDS::DBCluster` resources, the default policy is Snapshot
  - Note: to delete an S3 bucket, you need to first empty the bucket of its content

# Termination Protection on Stacks

- To prevent accidental deletes of CloudFormation templates, use TerminationProtection
- Let's see this quickly!

# EC2 Storage and Data Management

EBS, Instance Store & EFS



# EBS & EFS Section

- EBS and EFS in depth
- Performance
- Troubleshooting
- Operations
- Monitoring

# What's an EBS Volume?

- An EC2 machine loses its root volume (main drive) when it is manually terminated.
- Unexpected terminations might happen from time to time (AWS would email you)
- Sometimes, you need a way to store your instance data somewhere
- An **EBS (Elastic Block Store) Volume** is a **network** drive you can attach to your instances while they run
- It allows your instances to persist data

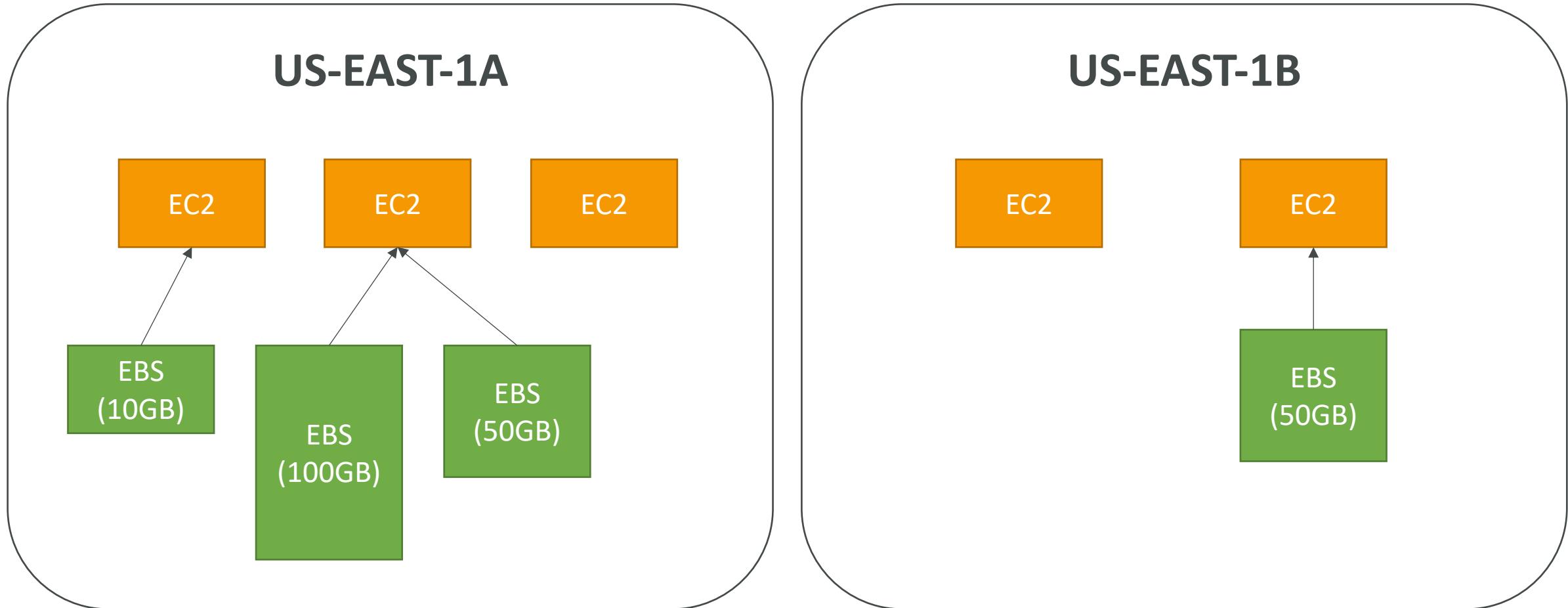


Amazon EBS

# EBS Volume

- It's a network drive (i.e. not a physical drive)
  - It uses the network to communicate the instance, which means there might be a bit of latency
  - It can be detached from an EC2 instance and attached to another one quickly
- It's locked to an Availability Zone (AZ)
  - An EBS Volume in us-east-1a cannot be attached to us-east-1b
  - To move a volume across, you first need to snapshot it
- Have a provisioned capacity (size in GBs, and IOPS)
  - You get billed for all the provisioned capacity
  - You can increase the capacity of the drive over time

# EBS Volume Example



# EBS Volume Types

- EBS Volumes come in 4 types
  - **GP2 (SSD)**: General purpose SSD volume that balances price and performance for a wide variety of workloads
  - **IO1 (SSD)**: Highest-performance SSD volume for mission-critical low-latency or high-throughput workloads
  - **ST1 (HDD)**: Low cost HDD volume designed for frequently accessed, throughput-intensive workloads
  - **SCI (HDD)**: Lowest cost HDD volume designed for less frequently accessed workloads
- EBS Volumes are characterized in Size | Throughput | IOPS (I/O Ops Per Sec)
- When in doubt always consult the AWS documentation – it's good!
- Only GP2 and IO1 can be used as boot volumes

# EBS Volume Types Use cases

## GP2 (from AWS doc)

- Recommended for most workloads
  - System boot volumes
  - Virtual desktops
  - Low-latency interactive apps
  - Development and test environments
- 
- 1 GiB - 16 TiB
  - Small gp2 volumes can burst IOPS to 3000
  - Max IOPS is 16,000...
  - 3 IOPS per GB, means at 5,334GB we are at the max IOPS

# EBS Volume Types Use cases

## IO1 (from AWS doc)

- Critical business applications that require sustained IOPS performance, or more than 16,000 IOPS per volume (gp2 limit)
  - Large database workloads, such as:
  - MongoDB, Cassandra, Microsoft SQL Server, MySQL, PostgreSQL, Oracle
- 
- 4 GiB - 16 TiB
  - IOPS is provisioned (PIOPS) – MIN 100 - MAX 64,000 (Nitro instances) else MAX 32,000 (other instances)
  - The maximum ratio of provisioned IOPS to requested volume size (in GiB) is 50:1

# EBS Volume Types Use cases

## ST1 (from AWS doc)

- Streaming workloads requiring consistent, fast throughput at a low price.
  - Big data, Data warehouses, Log processing
  - Apache Kafka
  - Cannot be a boot volume
- 
- 500 GiB - 16 TiB
  - Max IOPS is 500
  - Max throughput of 500 MiB/s – can burst

# EBS Volume Types Use cases

## SCI (from AWS doc)

- Throughput-oriented storage for large volumes of data that is infrequently accessed
  - Scenarios where the lowest storage cost is important
  - Cannot be a boot volume
- 
- 500 GiB - 16 TiB
  - Max IOPS is 250
  - Max throughput of 250 MiB/s – can burst

# Gp2 volumes I/O burst

- If your gp2 volume is less than 1000 GiB (means IOPS less than 3000) it can “burst” to 3000 IOPS performance
- This is a similar concept to t2 instances with their CPU
- You accumulate “burst credit over time”, which allows your volume to have good performance when needed
- The bigger the volume the faster you fill up your “burst credit balance”
- What happens if I empty my I/O credit balance?
  - The maximum I/O you get becomes the baseline you paid for
  - If you see the balance being 0 all the time, increase the gp2 volume or switch to io1
  - Use CloudWatch to monitoring the I/O credit balance
- Note: burst concept also applies to st1 or sc1 (for increase in throughput)

# Computing MB/s based on IOPS

- gp2:
  - Throughput in MiB/s = (Volume size in GiB) × (IOPS per GiB) × (I/O size in KiB)
  - ex: 300 I/O operations per second \* 256 KiB per I/O operation = 75 MiB/s
  - Limit to a max of 250 MiB/s (means volume  $\geq$  334 GiB won't increase throughput)
- io1:
  - Throughput in MiB/s = (Provisioned IOPS) × (I/O size in KiB)
  - The throughput limit of io1 volumes is 256 KiB/s for each IOPS provisioned
  - Limit to a max of 500 MiB/s (at 32,000 IOPS) and 1000 MiB/s (at 64,000 IOPS)

# EBS Volume Resizing

- Feb 2017: You can **resize** the EBS volumes
- You can only increase the EBS volumes:
  - Size (any volume type)
  - IOPS (only in IO1)
- After resizing an EBS volume, you need to repartition your drive
- After increasing the size, it's possible for the volume to be in a long time in the “optimisation” phase. The volume is still usable

# EBS Snapshots

- Incremental – only backup changed blocks
- EBS backups use IO and you shouldn't run them while your application is handling a lot of traffic
- Snapshots will be stored in S3 (but you won't directly see them)
- Not necessary to detach volume to do snapshot, but recommended
- Max 100,000 snapshots
- Can copy snapshots across AZ or Region
- Can make Image (AMI) from Snapshot
- EBS volumes restored by snapshots need to be pre-warmed (using fio or dd command to read the entire volume)
- Snapshots can be automated using Amazon Data Lifecycle Manager

# EBS Migration

- EBS Volumes are only locked to a specific AZ
- To migrate it to a different AZ (or region):
  - Snapshot the volume
  - (optional) Copy the volume to a different region
  - Create a volume from the snapshot in the AZ of your choice
- Let's practice!

# EBS Encryption

- When you create an encrypted EBS volume, you get the following:
  - Data at rest is encrypted inside the volume
  - All the data in flight moving between the instance and the volume is encrypted
  - All snapshots are encrypted
  - All volumes created from the snapshot
- Encryption and decryption are handled transparently (you have nothing to do)
- Encryption has a minimal impact on latency
- EBS Encryption leverages keys from KMS (AES-256)
- Copying an unencrypted snapshot allows encryption
- Snapshots of encrypted volumes are encrypted

# Encryption: encrypt an unencrypted EBS volume

- Create an EBS snapshot of the volume
- Encrypt the EBS snapshot ( using copy )
- Create new ebs volume from the snapshot ( the volume will also be encrypted )
- Now you can attach the encrypted volume to the original instance

# EBS vs Instance Store

- Some instances do not come with Root EBS volumes
- Instead, they come with “Instance Store” (= ephemeral storage)
- Instance store is physically attached to the machine (EBS is a network drive)
- Pros:
  - Better I/O performance
  - Good for buffer / cache / scratch data / temporary content
  - Data survives reboots
- Cons:
  - On stop or termination, the instance store is lost
  - You can't resize the instance store
  - Backups must be operated by the user

# EBS for SysOps

- If you plan to use the root volume of an instance after its terminated:
  - Set the Delete on Termination flag to "No"
  - You can see this option when creating the EC2 instance.
- If you use EBS for high performance, use EBS-optimized instance types
- If an EBS volume is unused, you still pay for it
- For cost saving over a long period, it can be cheaper to snapshot a volume and restore it later if unused

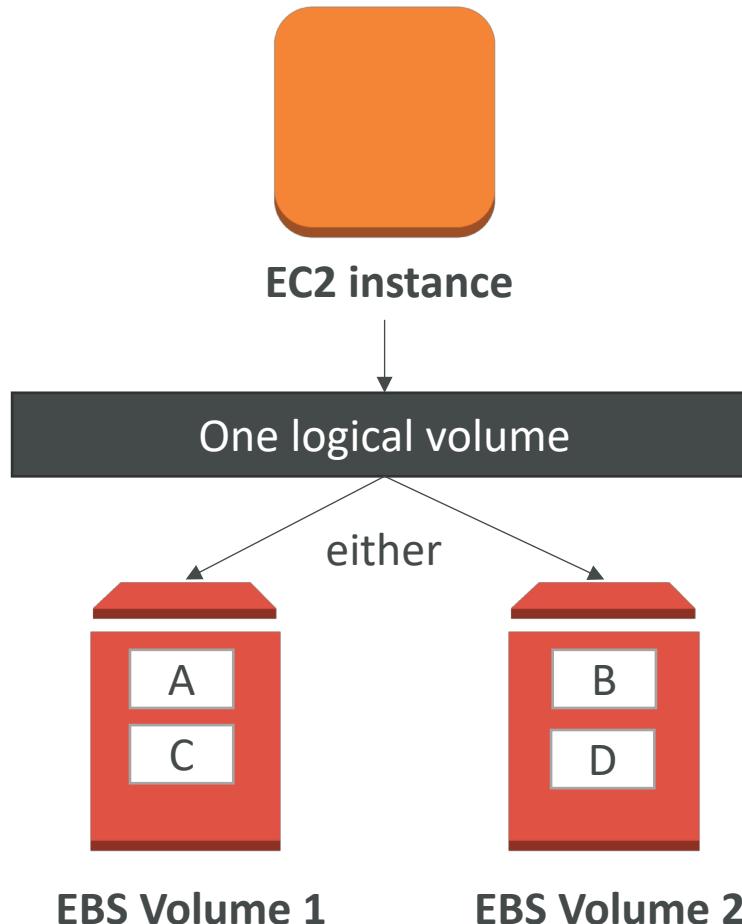
# EBS Troubleshooting

- High wait time or slow response for SSD => increase IOPS
- EC2 won't start with EBS volume as root: make sure volume names are properly mapped (/dev/xvdb instead of /dev/xvda for example)
- After increasing a volume size, you still need to repartition to use the incremental storage (xfs\_growfs for example)

# EBS RAID Options

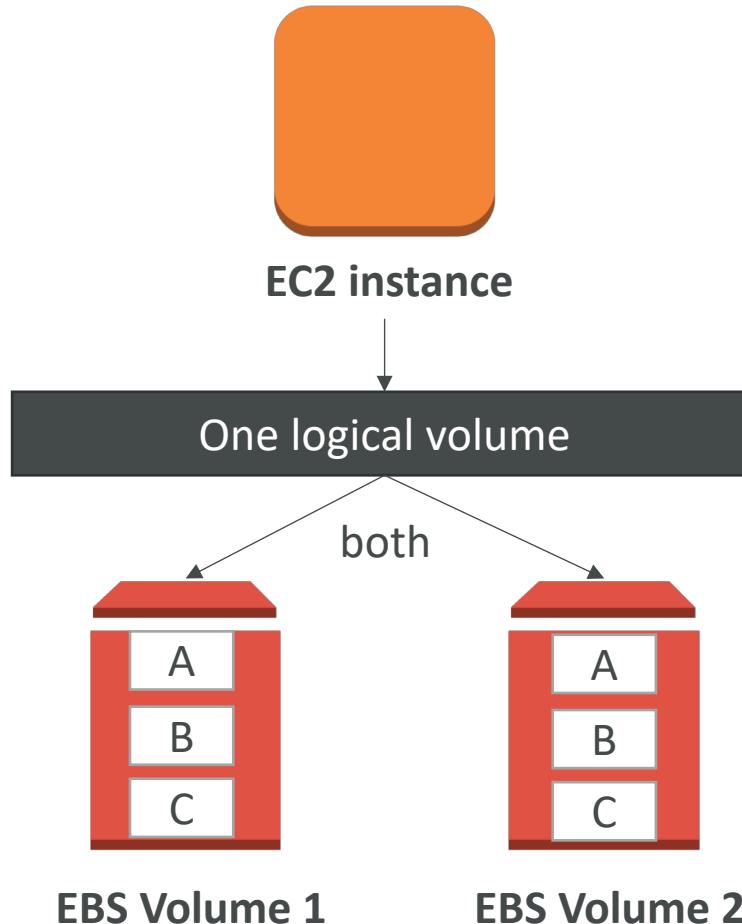
- EBS is already redundant storage (replicated within an AZ)
- But what if you want to increase IOPS to say 100 000 IOPS?
- What if you want to mirror your EBS volumes?
- You would mount volumes in parallel in RAID settings!
- RAID is possible as long as your OS supports it
- Some RAID options are:
  - RAID 0
  - RAID 1
  - RAID 5 (not recommended for EBS – see documentation)
  - RAID 6 (not recommended for EBS – see documentation)
- We'll explore RAID 0 and RAID 1

# RAID 0 (increase performance)



- Combining 2 or more volumes and getting the total disk space and I/O
- But one disk fails, all the data is failed
- Use cases would be:
  - An application that needs a lot of IOPS and doesn't need fault-tolerance
  - A database that has replication already built-in
- Using this, we can have a very big disk with a lot of IOPS
- For example
  - two 500 GiB Amazon EBS io1 volumes with 4,000 provisioned IOPS each will create a...
  - 1000 GiB RAID 0 array with an available bandwidth of 8,000 IOPS and 1,000 MB/s of throughput

# RAID I (increase fault tolerance)



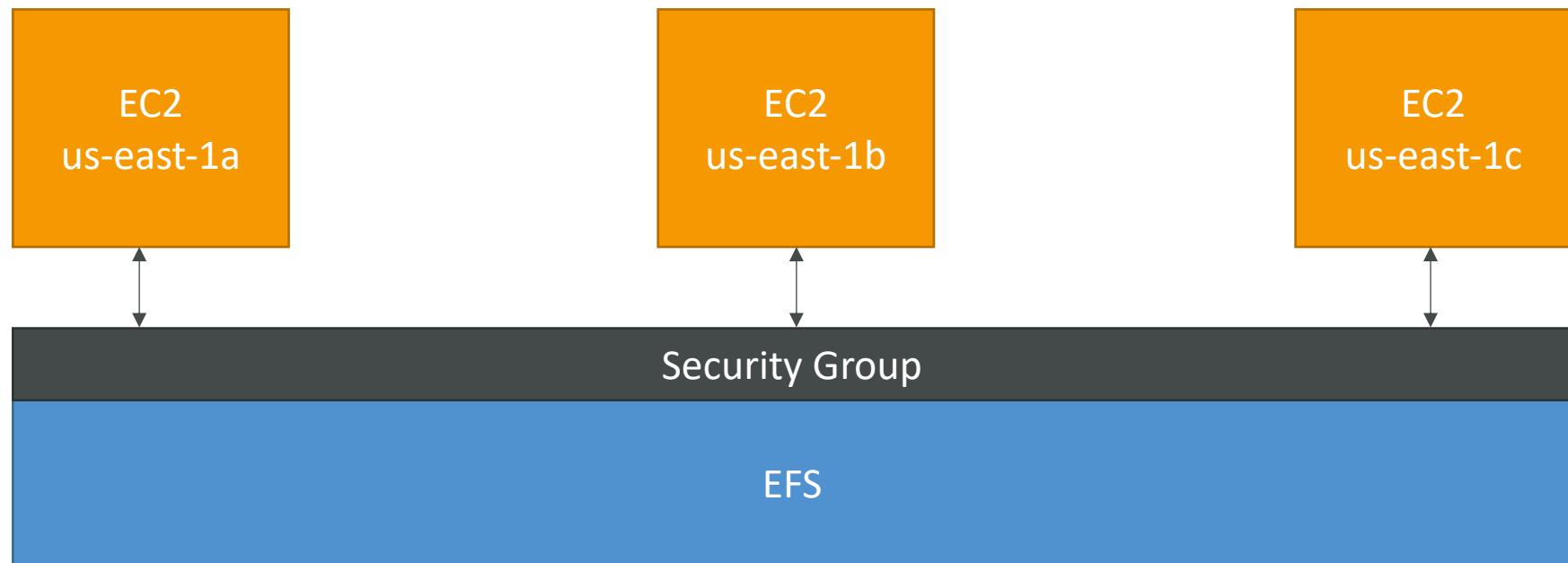
- RAID I = Mirroring a volume to another
- If one disk fails, our logical volume is still working
- We have to send the data to two EBS volume at the same time (2x network)
- Use case:
  - Application that need increase volume fault tolerance
  - Application where you need to service disks
- For example:
  - two 500 GiB Amazon EBS io1 volumes with 4,000 provisioned IOPS each will create a...
  - 500 GiB RAID I array with an available bandwidth of 4,000 IOPS and 500 MB/s of throughput

# CloudWatch and EBS

- Important EBS Volume CloudWatch metrics:
  - **VolumeldleTime**: number of seconds when no read / write is submitted
  - **VolumeQueueLength**: number of operations waiting to be executed. High number means probably an IOPS or application issue
  - **BurstBalance**: if it becomes 0, we need a volume with more IOPS
- gp2 volume types: 5 minutes interval
- io1 volume types: 1 minute interval
- EBS volumes have status check:
  - Ok - The volume is performing good.
  - Warning - Performance below expected.
  - Impaired - Stalled, performance severely degraded.
  - Insufficient-data - metric data collection in progress.

# EFS – Elastic File System

- Managed NFS (network file system) that can be mounted on many EC2
- EFS works with EC2 instances in multi-AZ
- Highly available, scalable, expensive (3x gp2), pay per use

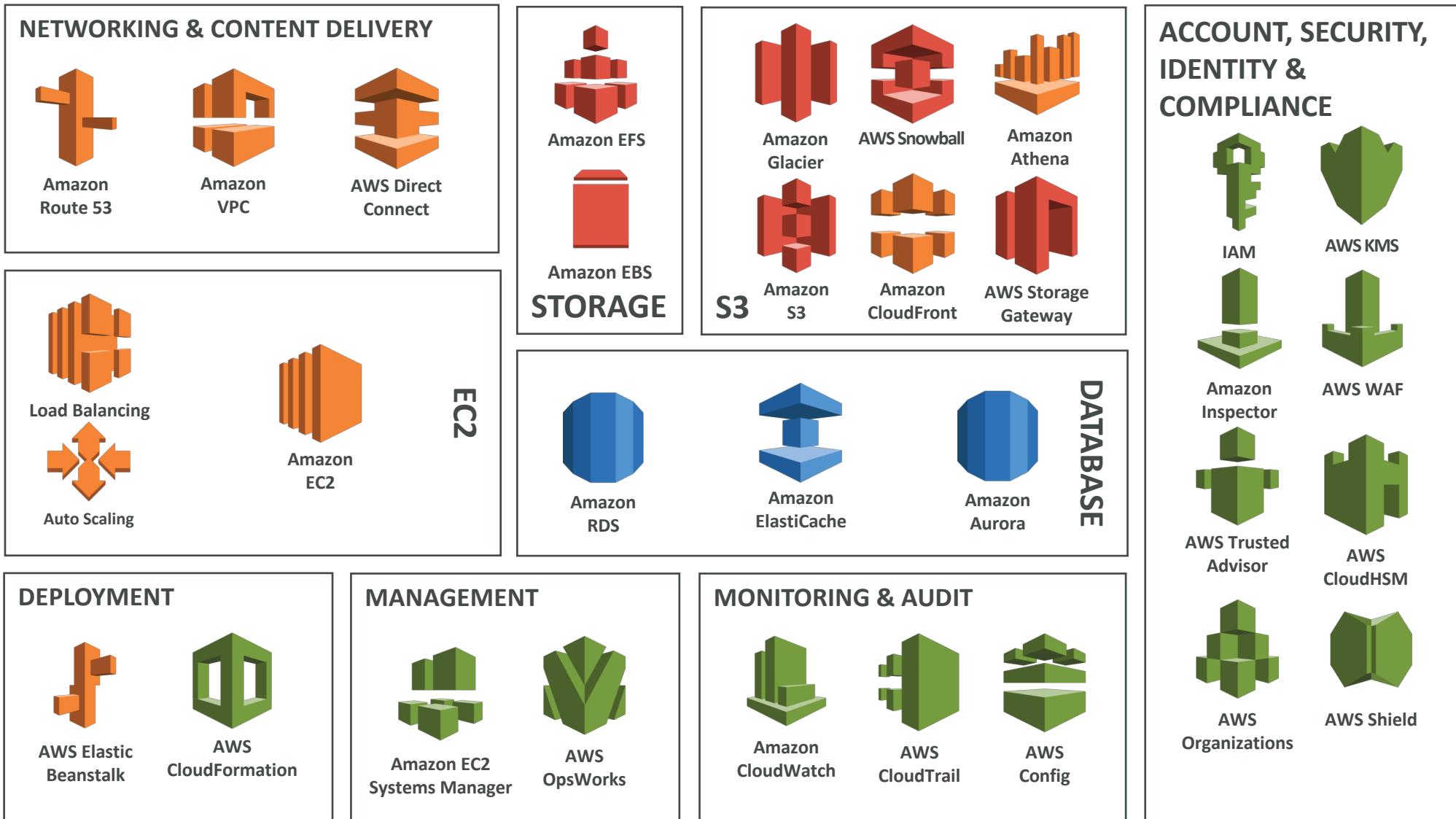


# EFS – Elastic File System

- Use cases: content management, web serving, data sharing, Wordpress
- Uses NFSv4.1 protocol
- Uses security group to control access to EFS
- Compatible with Linux based AMI (not Windows)
- Performance mode:
  - General purpose (default)
  - Max I/O – used when thousands of EC2 are using the EFS
- EFS file sync to sync from on-premise file system to EFS
- Backup EFS-to-EFS (incremental – can choose frequency)
- Encryption at rest using KMS

# S3 Storage and Data Management

S3, Glacier, Athena, Snowball, Snowball Edge and Storage Gateway

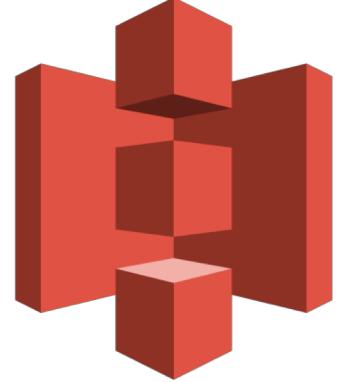


# S3 Section

- Two parts:
- 1<sup>st</sup> part: S3 Fundamentals (from developer course)
  - Bucket & Objects
  - Versioning
  - Security – Encryption & Bucket Policies
  - Websites
  - CORS
  - Consistency Model
- 2<sup>nd</sup> part: S3 for SysOps

# Section introduction

- Amazon S3 is one of the main building blocks of AWS
  - It's advertised as "infinitely scaling" storage
  - It's widely popular and deserves its own section
- 
- Many websites use AWS S3 as a backbone
  - Many AWS services uses AWS S3 as an integration as well
- 
- We'll have a step-by-step approach to S3



# AWS S3 Overview - Buckets

- Amazon S3 allows people to store objects (files) in “buckets” (directories)
- Buckets must have a **globally unique name**
- Buckets are defined at the region level
- Naming convention
  - No uppercase
  - No underscore
  - 3-63 characters long
  - Not an IP
  - Must start with lowercase letter or number



# AWS S3 Overview - Objects

- Objects (files) have a Key. The key is the **FULL** path:
  - <my\_bucket>/[my\\_file.txt](#)
  - <my\_bucket>/[my\\_folder1/another\\_folder/my\\_file.txt](#)
- There's no concept of "directories" within buckets (although the UI will trick you to think otherwise)
- Just keys with very long names that contain slashes ("")
- Object Values are the content of the body:
  - Max Size is 5TB
  - If uploading more than 5GB, must use "multi-part upload"
- Metadata (list of text key / value pairs – system or user metadata)
- Tags (Unicode key / value pair – up to 10) – useful for security / lifecycle
- Version ID (if versioning is enabled)



# AWS S3 - Versioning



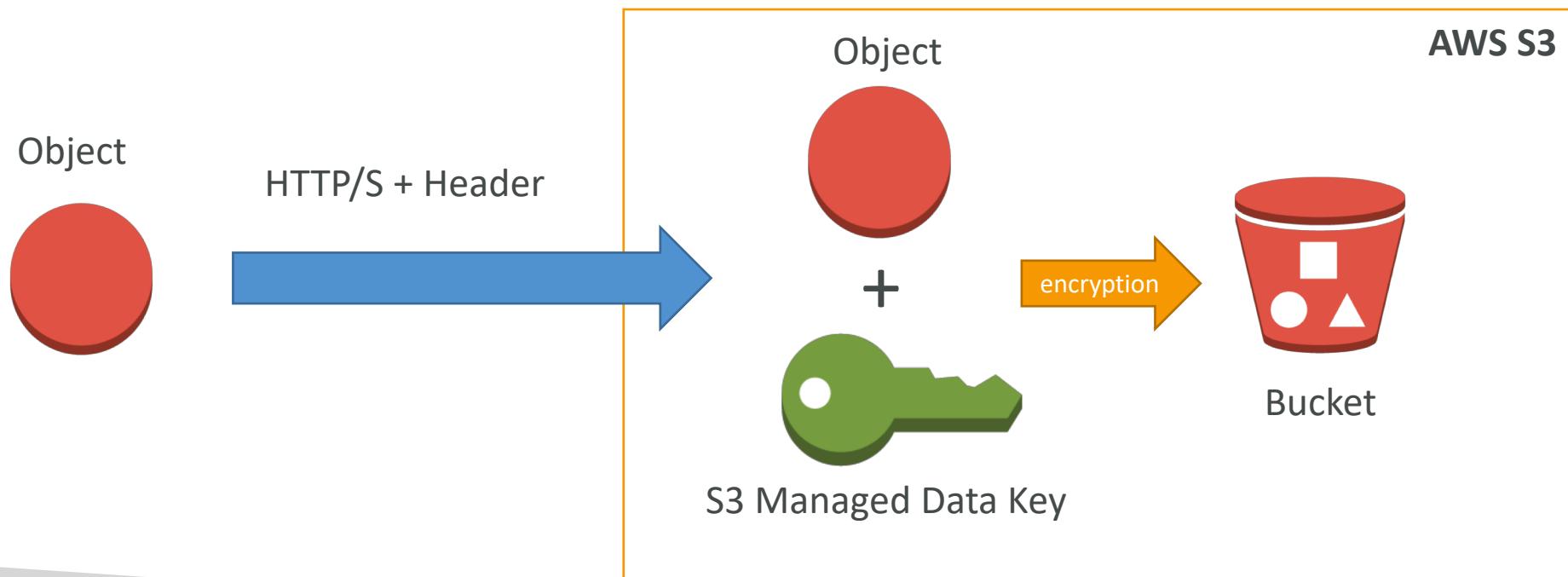
- You can version your files in AWS S3
- It is enabled at the **bucket level**
- Same key overwrite will increment the “version”: 1, 2, 3....
- It is best practice to version your buckets
  - Protect against unintended deletes (ability to restore a version)
  - Easy roll back to previous version
- Any file that is not versioned prior to enabling versioning will have version “null”

# S3 Encryption for Objects

- There are 4 methods of encrypting objects in S3
  - SSE-S3: encrypts S3 objects using keys handled & managed by AWS
  - SSE-KMS: leverage AWS Key Management Service to manage encryption keys
  - SSE-C: when you want to manage your own encryption keys
  - Client Side Encryption
- It's important to understand which ones are adapted to which situation for the exam

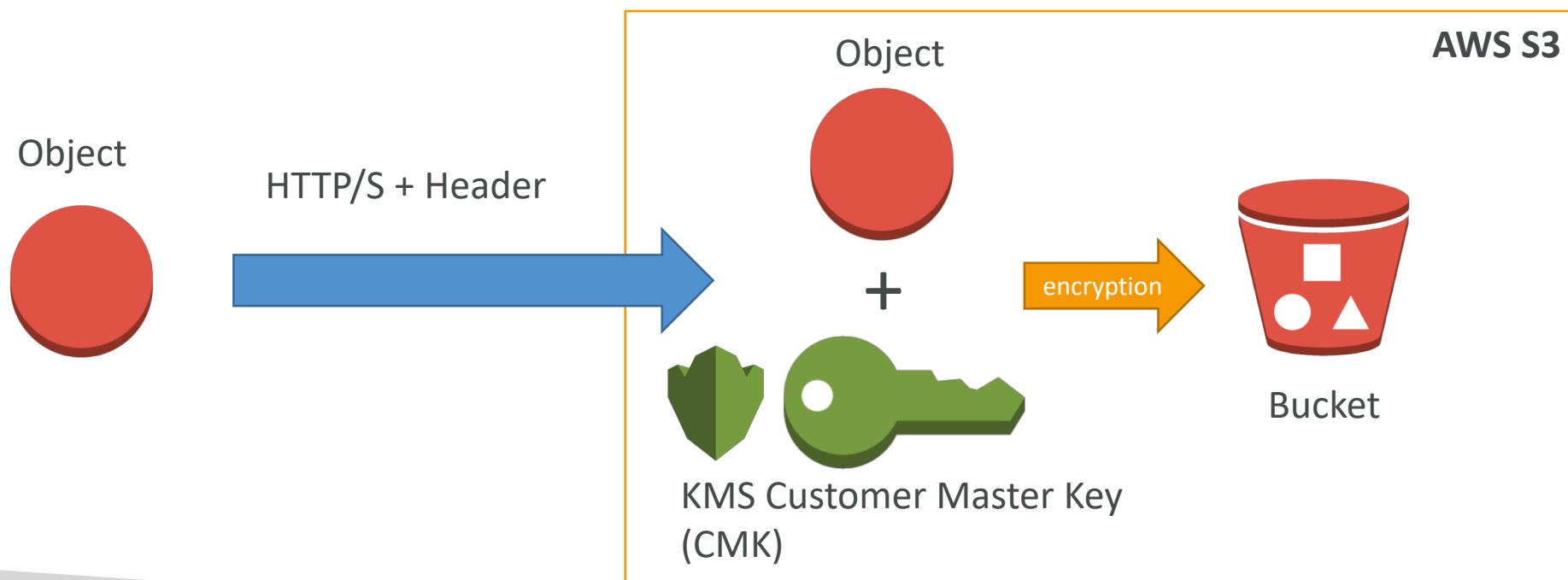
# SSE-S3

- SSE-S3: encryption using keys handled & managed by AWS S3
- Object is encrypted server side
- AES-256 encryption type
- Must set header: "x-amz-server-side-encryption": "AES256"



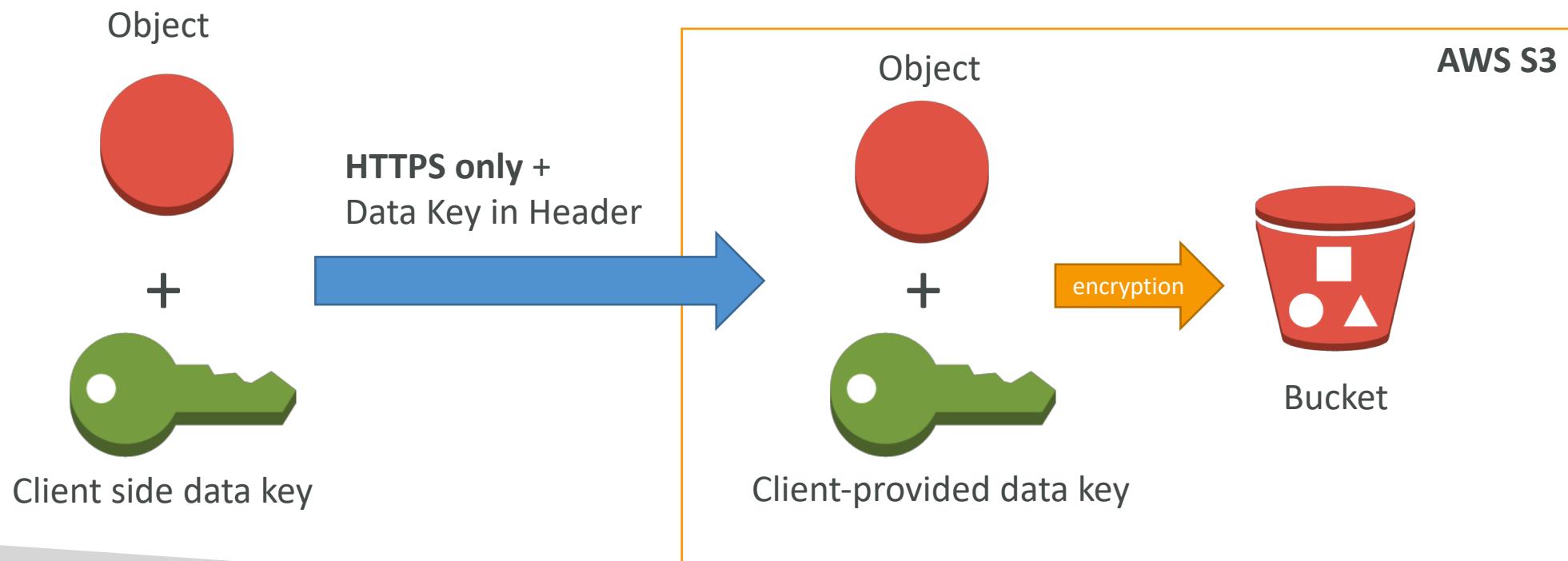
# SSE-KMS

- SSE-KMS: encryption using keys handled & managed by KMS
- KMS Advantages: user control + audit trail
- Object is encrypted server side
- Must set header: "x-amz-server-side-encryption": "aws:kms"



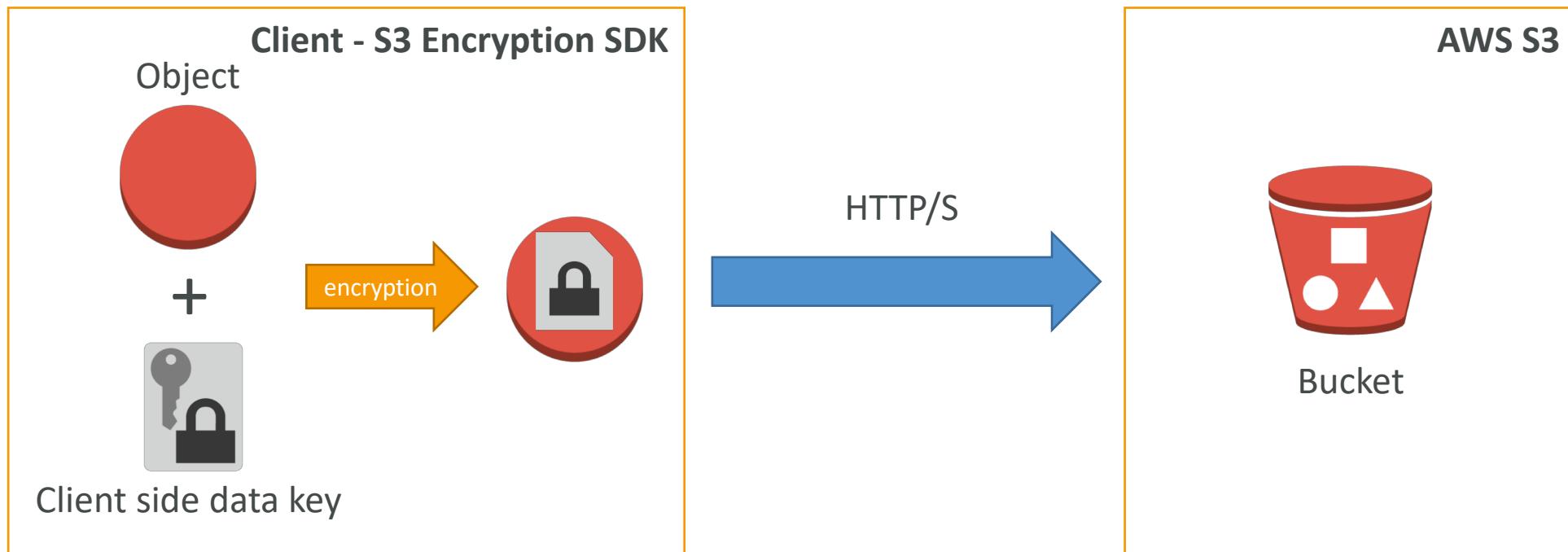
# SSE-C

- SSE-C: server-side encryption using data keys fully managed by the customer outside of AWS
- Amazon S3 does not store the encryption key you provide
- **HTTPS must be used**
- Encryption key must provided in HTTP headers, for every HTTP request made



# Client Side Encryption

- Client library such as the Amazon S3 Encryption Client
- Clients must encrypt data themselves before sending to S3
- Clients must decrypt data themselves when retrieving from S3
- Customer fully manages the keys and encryption cycle



# Encryption in transit (SSL)

- AWS S3 exposes:
  - HTTP endpoint: non encrypted
  - HTTPS endpoint: encryption in flight
- You're free to use the endpoint you want, but HTTPS is recommended
- HTTPS is mandatory for SSE-C
- Encryption in flight is also called SSL / TLS

# S3 Security

- User based
  - IAM policies - which API calls should be allowed for a specific user from IAM console
- Resource Based
  - Bucket Policies - bucket wide rules from the S3 console - allows cross account
  - Object Access Control List (ACL) – finer grain
  - Bucket Access Control List (ACL) – less common

# S3 Bucket Policies

- JSON based policies
  - Resources: buckets and objects
  - Actions: Set of API to Allow or Deny
  - Effect: Allow / Deny
  - Principal: The account or user to apply the policy to
- Use S3 bucket for policy to:
  - Grant public access to the bucket
  - Force objects to be encrypted at upload
  - Grant access to another account (Cross Account)

# S3 Security - Other

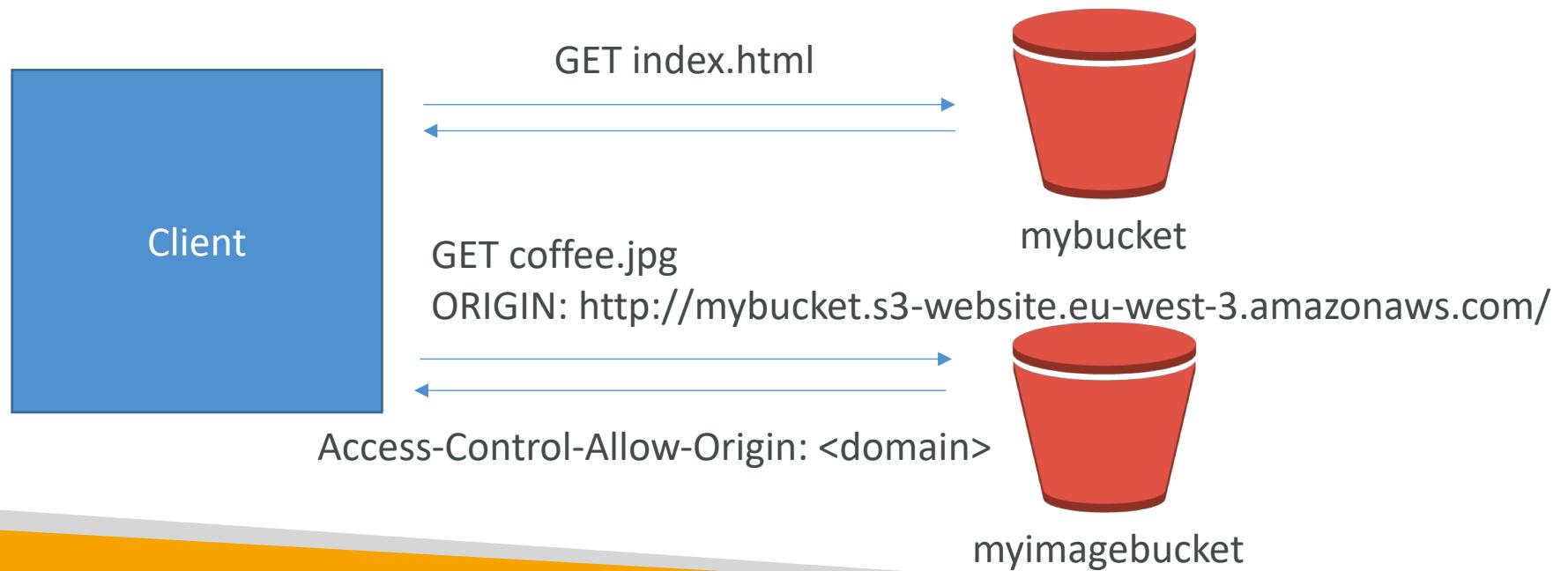
- Networking:
  - Supports VPC Endpoints (for instances in VPC without www internet)
- Logging and Audit:
  - S3 access logs can be stored in other S3 bucket
  - API calls can be logged in AWS CloudTrail
- User Security:
  - MFA (multi factor authentication) can be required in versioned buckets to delete objects
  - Signed URLs: URLs that are valid only for a limited time (ex: premium video service for logged in users)

# S3 Websites

- S3 can host static websites and have them accessible on the www
- The website URL will be:
  - <bucket-name>.s3-website-<AWS-region>.amazonaws.com
  - OR
  - <bucket-name>.s3-website.<AWS-region>.amazonaws.com
- If you get a 403 (Forbidden) error, make sure the bucket policy allows public reads!

# S3 CORS

- If you request data from another S3 bucket, you need to enable CORS
- Cross Origin Resource Sharing allows you to limit the number of websites that can request your files in S3 (and limit your costs)
- It's a popular exam question

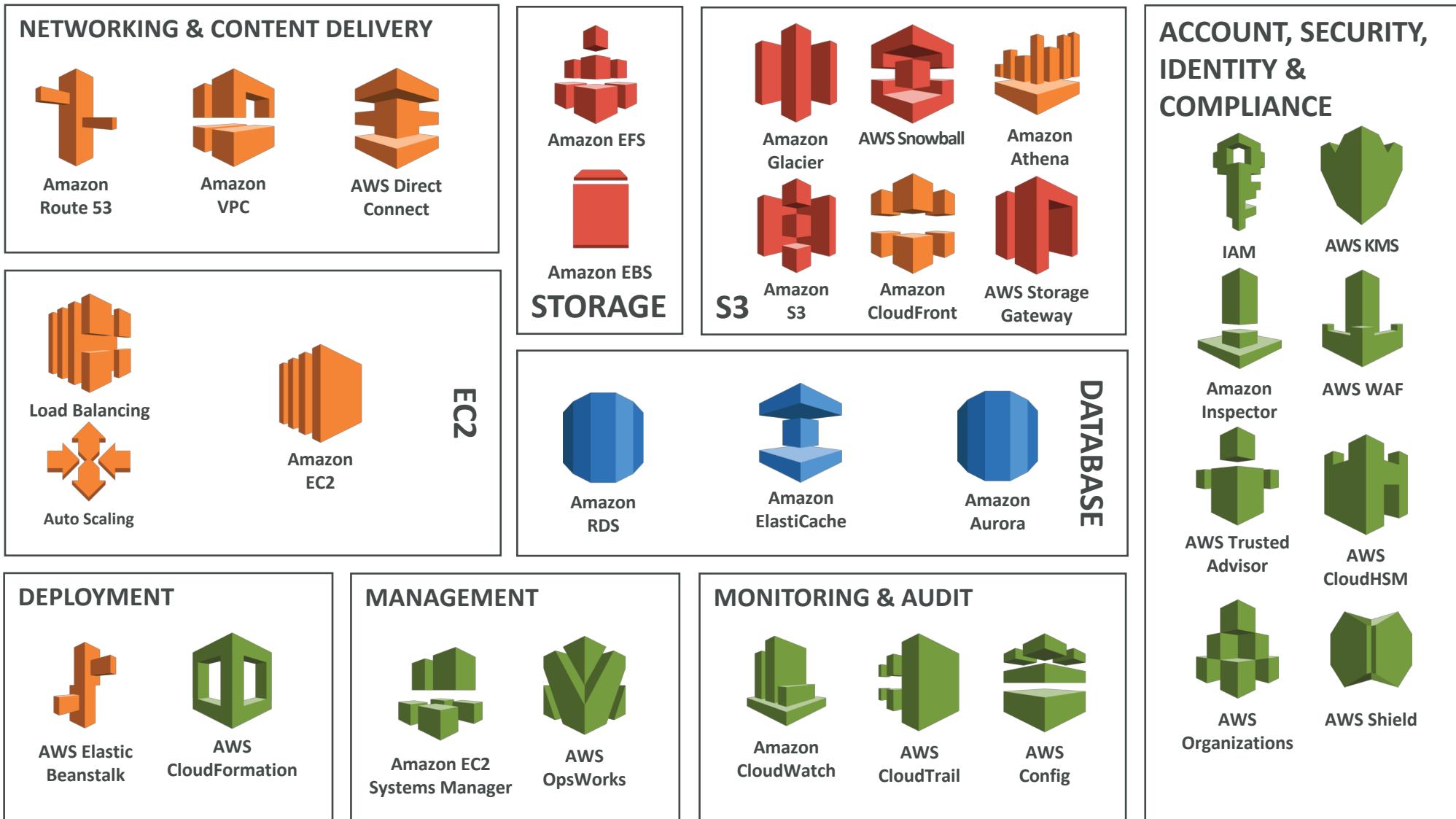


# AWS S3 - Consistency Model

- Read after write consistency for PUTS of new objects
  - As soon as an object is written, we can retrieve it  
ex: (PUT 200 -> GET 200)
  - This is true, **except** if we did a GET before to see if the object existed  
ex: (GET 404 -> PUT 200 -> GET 404) – eventually consistent
- Eventual Consistency for DELETES and PUTS of existing objects
  - If we read an object after updating, we might get the older version  
ex: (PUT 200 -> PUT 200 -> GET 200 (might be older version))
  - If we delete an object, we might still be able to retrieve it for a short time  
ex: (DELETE 200 -> GET 200)

# S3 Storage and Data Management For SysOps

S3, Glacier, Athena, Snowball, Snowball Edge and Storage Gateway



# S3 for SysOps

- Versioning
- MFA-Delete
- Default Encryption
- Logs
- Cross Region Replication
- Pre-Signed URLs
- CloudFront
- Inventory
- Storage Tiers
- Lifecycle Rules
- Analytics
- Glacier
- Snowball
- Storage Gateway
- Athena

# S3 Versioning for SysOps

- S3 Versioning creates a new version each time you change a file
- That includes when you encrypt a file!
- It's a nice way to get protected against hackers wanting to encrypt our data
  
- Deleting a file in the S3 bucket just adds a delete marker on the versioning
- To delete a bucket, you need to remove all the file version within it

# S3 MFA-Delete

- MFA (multi factor authentication) forces user to generate a code on a device (usually a mobile phone or hardware) before doing important operations on S3
- To use MFA-Delete, enable Versioning on the S3 bucket
- You will need MFA to
  - permanently delete an object version
  - suspend versioning on the bucket
- You won't need MFA for
  - enabling versioning
  - listing deleted versions
- Only the bucket owner (root account) can enable/disable MFA-Delete
- MFA-Delete currently can only be enabled using the CLI

# S3 Default Encryption vs Bucket Policies

- The old way to enable default encryption was to use a bucket policy and refuse any HTTP command without the proper headers:

```
{  
    "Version": "2012-10-17",  
    "Id": "PutObjPolicy",  
    "Statement": [  
        {  
            "Sid": "DenyIncorrectEncryptionHeader",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::<bucket_name>/*",  
            "Condition": {  
                "StringNotEquals": {  
                    "s3:x-amz-server-side-encryption": "AES256"  
                }  
            }  
        }  
    ],  
}.
```

```
{  
    "Sid": "DenyUnEncryptedObjectUploads",  
    "Effect": "Deny",  
    "Principal": "*",  
    "Action": "s3:PutObject",  
    "Resource": "arn:aws:s3:::<bucket_name>/*",  
    "Condition": {  
        "Null": {  
            "s3:x-amz-server-side-encryption": true  
        }  
    }  
}
```

- The new way is to use the “default encryption” option in S3
- Note: Bucket Policies are evaluated before “default encryption”

# S3 Access Logs

- For audit purpose, you may want to log all access to S3 buckets
- Any request made to S3, from any account, authorized or denied, will be logged into another S3 bucket
- That data can be analyzed using data analysis tools...
- Or Amazon Athena as we'll see later in this section!
- The log format is at:  
<https://docs.aws.amazon.com/AmazonS3/latest/dev/LogFileFormat.html>



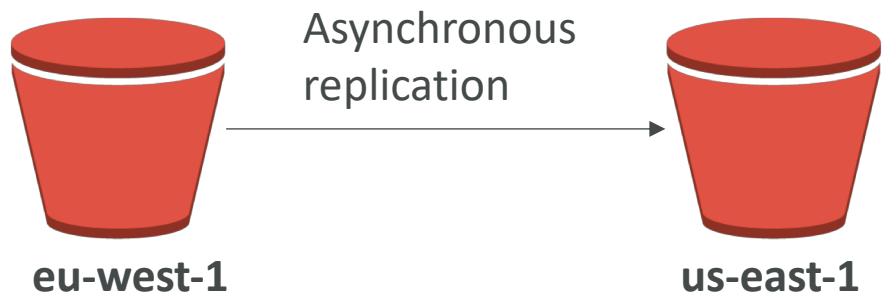
# S3 Bucket Policies Lab

- Let's analyze the policies at:

<https://docs.aws.amazon.com/AmazonS3/latest/dev/example-bucket-policies.html>

# S3 Cross Region Replication

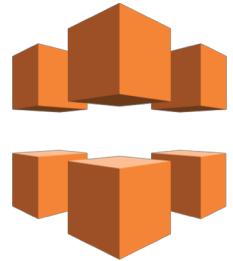
- Must enable versioning (source and destination)
- Buckets must be in different AWS regions
- Can be in different accounts
- Copying is asynchronous
- Must give proper IAM permissions to S3
- Use cases: compliance, lower latency access, replication across accounts



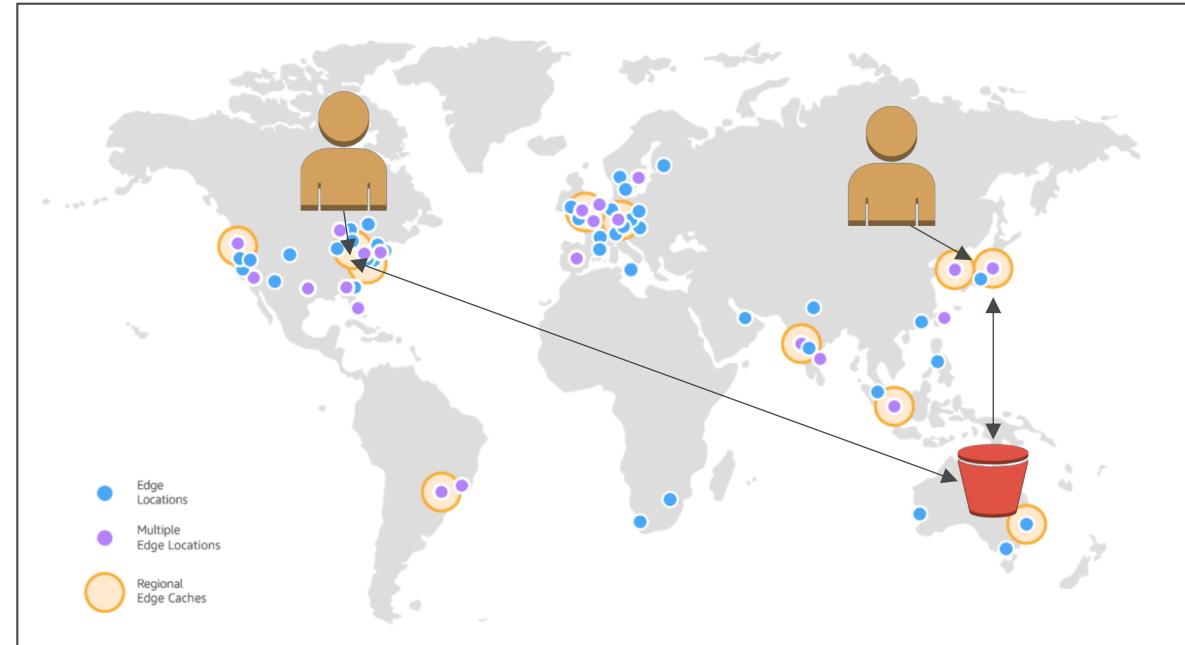
# S3 pre-signed URLs

- Can generate pre-signed URLs using SDK or CLI
  - For downloads (easy, can use the CLI)
  - For uploads (harder, must use the SDK)
- Valid for a default of 3600 seconds, can change timeout with --expires-in [TIME\_BY\_SECONDS] argument
- Users given a pre-signed URL inherit the permissions of the person who generated the URL for GET / PUT
- Examples :
  - Allow only logged-in users to download a premium video on your S3 bucket
  - Allow an ever changing list of users to download files by generating URLs dynamically
  - Allow temporarily a user to upload a file to a precise location in our bucket

# AWS CloudFront



- Content Delivery Network (CDN)
- Improves read performance, content is cached at the edge
- 136 Point of Presence globally (edge locations)
- Popular with S3 but works with EC2, Load Balancing
- Can help protect against network attacks
- Can provide SSL encryption (HTTPS) at the edge using ACM
- CloudFront can use SSL encryption (HTTPS) to talk to your applications
- Support RTMP Protocol (videos / media)



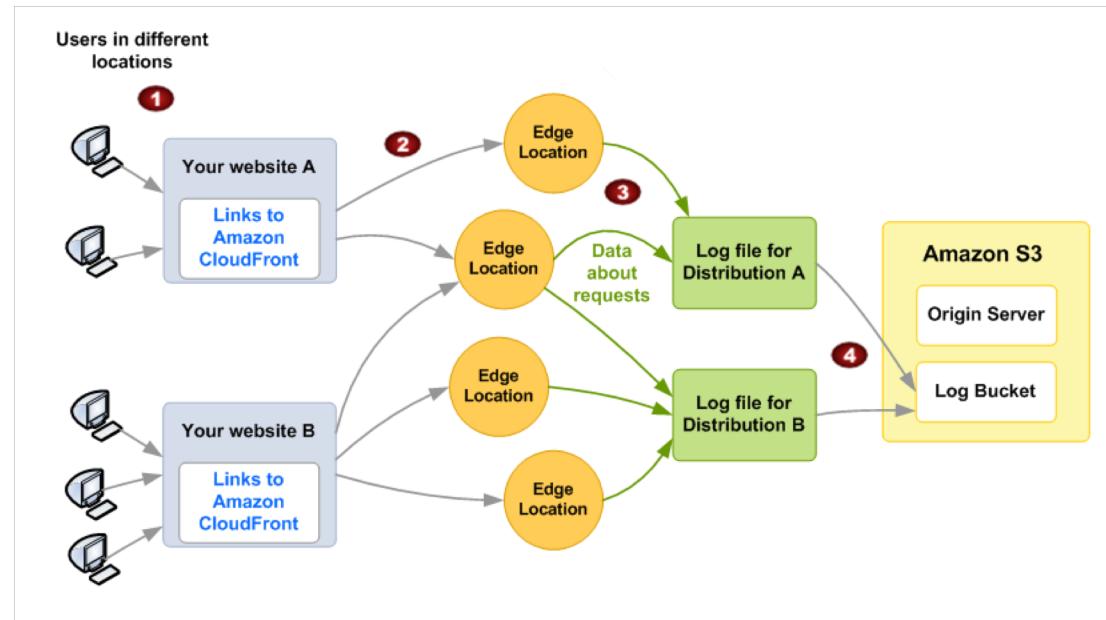
Source: <https://aws.amazon.com/cloudfront/features/?nc=sn&loc=2>

# AWS CloudFront Hands On

- We'll create an S3 bucket
- We'll create a CloudFront distribution
- We'll create an Origin Access Identity
- We'll limit the S3 bucket to be accessed only using this identity

# CloudFront Access Logs

- CloudFront access logs: logs every request made to CloudFront into a logging S3 bucket



# CloudFront Reports

- It's possible to generate reports on:
  - Cache Statistics Report
  - Popular Objects Report
  - Top Referrers Report
  - Usage Reports
  - Viewers Report
- These reports are based on the data from the Access Logs.

# CloudFront troubleshooting

- CloudFront caches HTTP 4xx and 5xx status codes returned by S3 ( or the origin server)
- 4xx error code indicates that user doesn't have access to the underlying bucket (403) or the object user is requesting is not found (404)
- 5xx error codes indicates Gateway issues

# S3 Inventory

- Amazon S3 inventory helps manage your storage
- Audit and report on the replication and encryption status of your objects
- Use cases:
  - Business
  - Compliance
  - Regulatory needs.
- You can query all the data using Amazon Athena, Redshift, Presto, Hive, Spark...
- You can setup multiple inventories
- Data goes from a source bucket to a target bucket (need to setup policy)

# S3 Storage Tiers

- Amazon S3 Standard - General Purpose
- Amazon S3 Standard-Infrequent Access (IA)
- Amazon S3 One Zone-Infrequent Access
- Amazon S3 Reduced Redundancy Storage (deprecated)
- Amazon S3 Intelligent Tiering (new!)
- Amazon Glacier (dedicated lecture)

# S3 Standard – General Purpose

- High durability (99.99999999%) of objects across multiple AZ
- If you store 10,000,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000 years
- 99.99% Availability over a given year
- Sustain 2 concurrent facility failures
  
- Use Cases: Big Data analytics, mobile & gaming applications, content distribution...

# S3 Reduced Redundancy Storage (RRS) - DEPRECATED

- Designed to provide 99.99% durability
- 99.99% availability of objects over a given year
- Designed to sustain the loss of data in a single facility
  
- Use cases: noncritical, reproducible data at lower levels of redundancy than Amazon S3's standard storage (thumbnails, transcoded media, processed data that can be reproduced)

# S3 Standard – Infrequent Access (IA)

- Suitable for data that is less frequently accessed, but requires rapid access when needed
- High durability (99.99999999%) of objects across multiple AZs
- 99.99% Availability
- Low cost compared to Amazon S3 Standard
- Sustain 2 concurrent facility failures
- Use Cases: As a data store for disaster recovery, backups...

# S3 One Zone - Infrequent Access (IA)

- Same as IA but data is stored in a single AZ
- High durability (99.99999999%) of objects in a single AZ; data lost when AZ is destroyed
- 99.95% Availability
- Low latency and high throughput performance
- Supports SSL for data at transit and encryption at rest
- Low cost compared to IA (by 20%)
- Use Cases: Storing secondary backup copies of on-premise data, or storing data you can recreate

# S3 Intelligent Tiering (new!)

- Probably not at the exam (yet!)
- Same low latency and high throughput performance of S3 Standard
- Small monthly monitoring and auto-tiering fee
- Automatically moves objects between two access tiers based on changing access patterns
- Designed for durability of 99.999999999% of objects across multiple Availability Zones
- Resilient against events that impact an entire Availability Zone
- Designed for 99.9% availability over a given year

# S3 Storage Tiers Comparison

	Standard	Reduced Redundancy Storage	Standard - Infrequent Access	One - Infrequent Access	S3 Intelligent-Tiering
Durability	99.99999999%	99.99%	99.99999999%	99.99999999%	99.99999999%
Availability	99.99%	99.99%	99.9%	99.5%	99.90%
AZ	≥3	≥2	≥3	1	≥3
Concurrent facility fault tolerance	2	1	2	0	1

Frequently accessed      Infrequently accessed      Intelligent (new!)

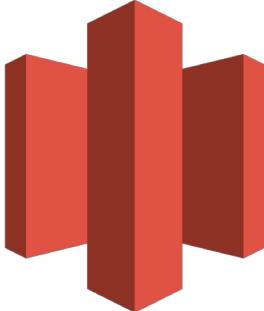
# S3 Lifecycle Rules

- Set of rules to move data between different tiers, to save storage cost
- Example: General Purpose => Infrequent Access => Glacier
- **Transition actions:** It defines when objects are transitioned to another storage class.  
Eg: We can choose to move objects to Standard IA class 60 days after you created them or can move to Glacier for archiving after 6 months
- **Expiration actions:** Helps to configure objects to expire after a certain time period. S3 deletes expired objects on our behalf  
Eg: Access log files can be set to delete after a specified period of time
- Can be used to delete incomplete multi-part uploads!

# S3 Analytics – Storage Class Analysis

- You can setup analytics to help determine when to transition objects from Standard to Standard\_IA
- Does not work for ONEZONE\_IA or GLACIER
- Report is updated on a daily basis
- Takes about 24h to 48h hours to first start
- This helps you put together Lifecycle Rules!

# Glacier



- Low cost object storage meant for archiving / backup
- Data is retained for the longer term (10s of years)
- Alternative to on-premise magnetic tape storage
- Average annual durability is 99.999999999%
- Cost per storage per month (\$0.004 / GB) + retrieval cost
- Each item in Glacier is called “Archive” (up to 40TB)
- Archives are stored in “Vaults”
- Exam tip: archival from S3 after XXX days => use Glacier

# Glacier Operations

- 3 Glacier operations:
  - Upload - Single operation or by parts (MultiPart upload) for larger archives
  - Download - First initiate a retrieval job for the particular archive, Glacier then prepares it for download. User then has a limited time to download the data from staging server
  - Delete - Use Glacier Rest API or AWS SDKs by specifying archive ID
- Restore links have an expiry date
- 3 retrieval options:
  - Expedited (1 to 5 minutes retrieval) – \$0.03 per GB and \$0.01 per request
  - Standard (3 to 5 hours) - \$0.01 per GB and 0.05 per 1000 requests
  - Bulk (5 to 12 hours) - \$0.0025 per GB and \$0.025 per 1000 requests

# Glacier - Vault Policies & Vault Lock

- Vault is a collection of archives
- Each Vault has:
  - ONE vault access policy
  - ONE vault lock policy
- Vault Policies are written in JSON
- Vault Access Policy is similar to bucket policy (restrict user / account permissions)
- Vault Lock Policy is a policy you lock, for regulatory and compliance requirements.
  - The policy is immutable, it can never be changed (that's why it's call LOCK)
  - Example 1: forbid deleting an archive if less than 1 year old
  - Example 2: implement WORM policy (write once read many)

# Snowball

- Physical data transport solution that helps moving TBs or PBs of data in or out of AWS
- Alternative to moving data over the network (and paying network fees)
- Secure, tamper resistant, uses KMS 256 bit encryption
- Tracking using SNS and text messages. E-ink shipping label
- Pay per data transfer job
- Use cases: large data cloud migrations, DC decommission, disaster recovery
- If it takes more than a week to transfer over the network, use Snowball devices!



# Snowball Process

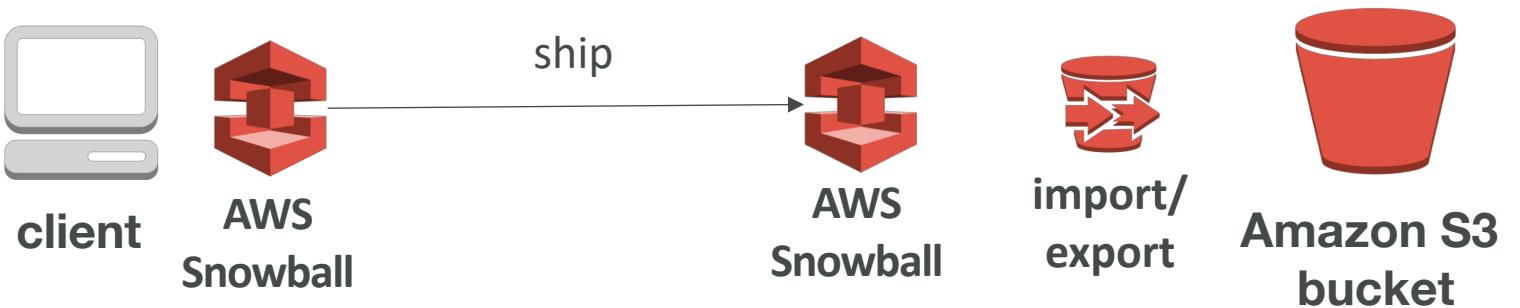
1. Request snowball devices from the AWS console for delivery
2. Install the snowball client on your servers
3. Connect the snowball to your servers and copy files using the client
4. Ship back the device when you're done (goes to the right AWS facility)
5. Data will be loaded into an S3 bucket
6. Snowball is completely wiped
7. Tracking is done using SNS, text messages and the AWS console

# Diagrams

- Direct upload to S3:



- With snowball



# Snowball Edge

- Snowball Edges add computational capability to the device
- 100 TB capacity with either:
  - Storage optimized – 24 vCPU
  - Compute optimized – 52 vCPU & optional GPU
- Supports a custom EC2 AMI so you can perform processing on the go
- Supports custom Lambda functions
- Very useful to pre-process the data while moving
- Use case: data migration, image collation, IoT capture, machine learning



# AWS Snowmobile



- Transfer exabytes of data (1 EB = 1,000 PB = 1,000,000 TBs)
- Each Snowmobile has 100 PB of capacity (use multiple in parallel)
- Better than Snowball if you transfer more than 10 PB

# Hybrid Cloud for Storage

- AWS is pushing for "hybrid cloud"
  - Part of your infrastructure is on the cloud
  - Part of your infrastructure is on-premise
- This can be due to
  - Long cloud migrations
  - Security requirements
  - Compliance requirements
  - IT strategy
- S3 is a proprietary storage technology (unlike EFS / NFS), so how do you expose the S3 data on-premise?
- AWS Storage Gateway!

# AWS Storage Cloud Native Options

## BLOCK

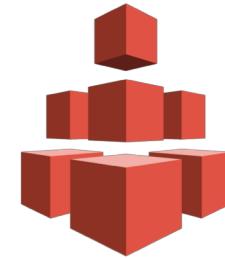


Amazon EBS



EC2 Instance  
Store

## FILE

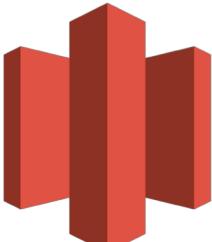


Amazon EFS

## OBJECT



S3



Glacier

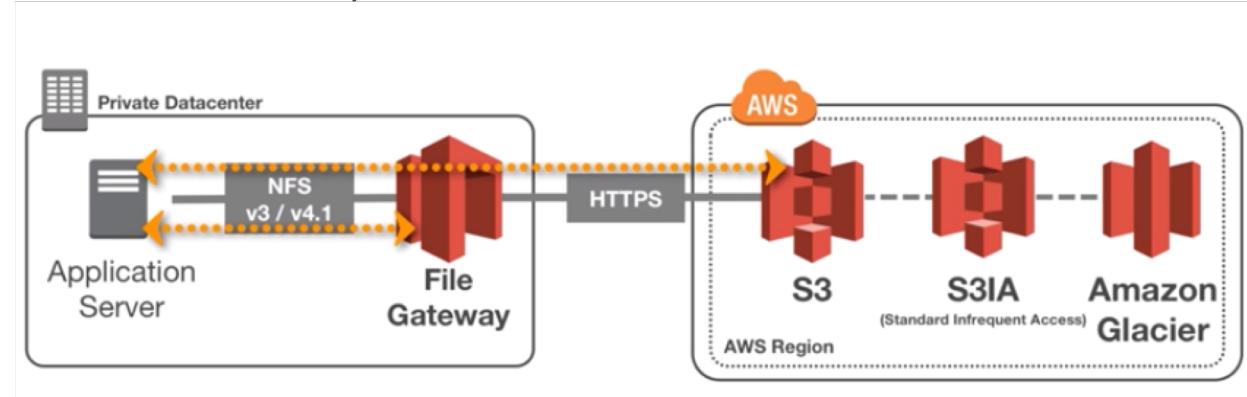
# AWS Storage Gateway

- Bridge between on-premise data and cloud data in S3
- Use cases: disaster recovery, backup & restore, tiered storage
- 3 types of Storage Gateway:
  - File Gateway
  - Volume Gateway
  - Tape Gateway
- Exam Tip: You need to know the differences between all 3!



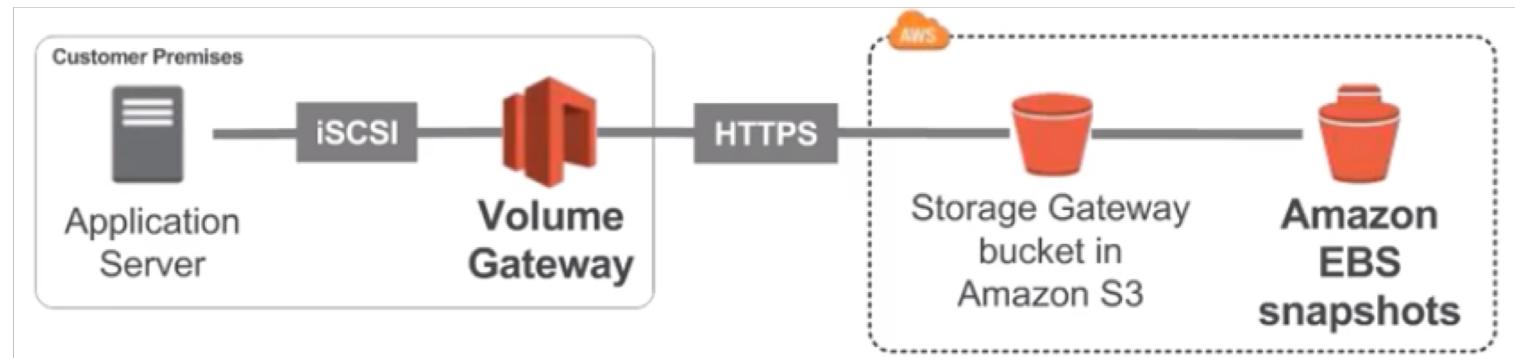
# File Gateway

- Configured S3 buckets are accessible using the NFS and SMB protocol
- Supports S3 standard, S3 IA, S3 One Zone IA
- Bucket access using IAM roles for each File Gateway
- Most recently used data is cached in the file gateway
- Can be mounted on many servers



# Volume Gateway

- Block storage using iSCSI protocol backed by S3
- Backed by EBS snapshots which can help restore on-premise volumes!
- **Cached volumes:** low latency access to most recent data
- **Stored volumes:** entire dataset is on premise, scheduled backups to S3



# Tape Gateway

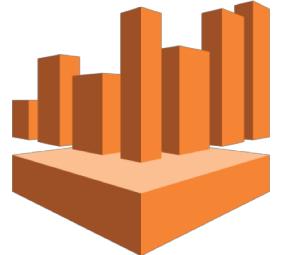
- Some companies have backup processes using physical tapes (!)
- With Tape Gateway, companies use the same processes but in the cloud
- Virtual Tape Library (VTL) backed by Amazon S3 and Glacier
- Back up data using existing tape-based processes (and iSCSI interface)
- Works with leading backup software vendors



# AWS Storage Gateway Summary

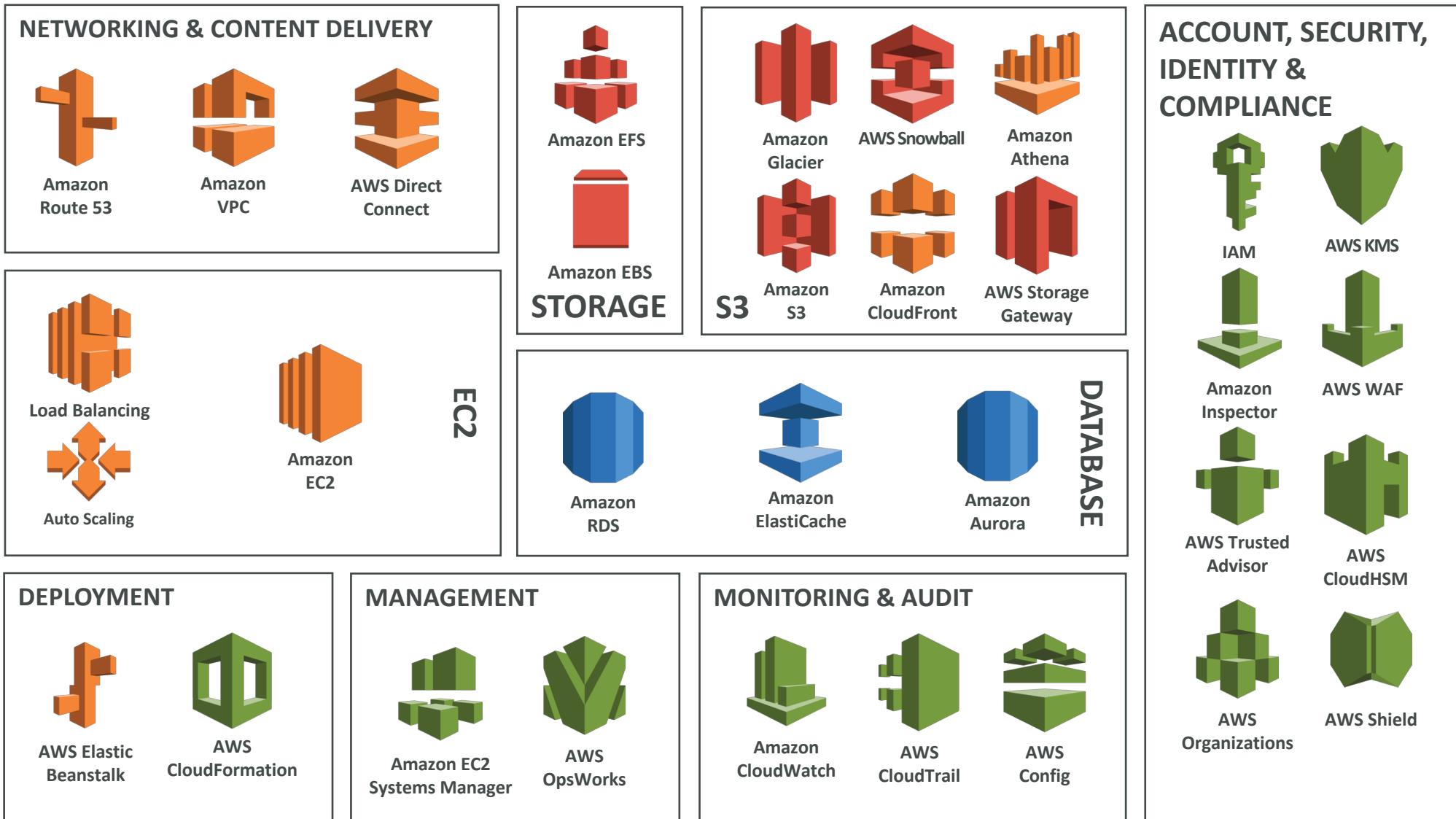
- Exam tip: Read the question well, it will hint at which gateway to use
- On premise data to the cloud => Storage Gateway
- File access / NFS => File Gateway  
(backed by S3)
- Volumes / Block Storage / iSCSI => Volume gateway  
(backed by S3 with EBS snapshots)
- VTL Tape solution / Backup with iSCSI => Tape Gateway  
(backed by S3 and Glacier)

# AWS Athena



- Serverless service to perform analytics **directly against S3 files**
- Uses SQL language to query the files
- Has a JDBC / ODBC driver
- Charged per query and amount of data scanned
- Supports CSV, JSON, ORC, Avro, and Parquet (built on Presto)
- Use cases: Business intelligence / analytics / reporting, analyze & query VPC Flow Logs, ELB Logs, CloudTrail trails, etc...
- Exam Tip: Analyze data directly on S3 => use Athena

# Databases



# Databases Section

- RDS in Depth
  - Basics refresher lectures
  - Multi AZ vs Read Replicas
  - Parameter Groups
  - Backups & Snapshots
  - Security
  - API
  - CloudWatch and Performance Insights
- Aurora in depth
- ElastiCache refresher

# AWS RDS Overview



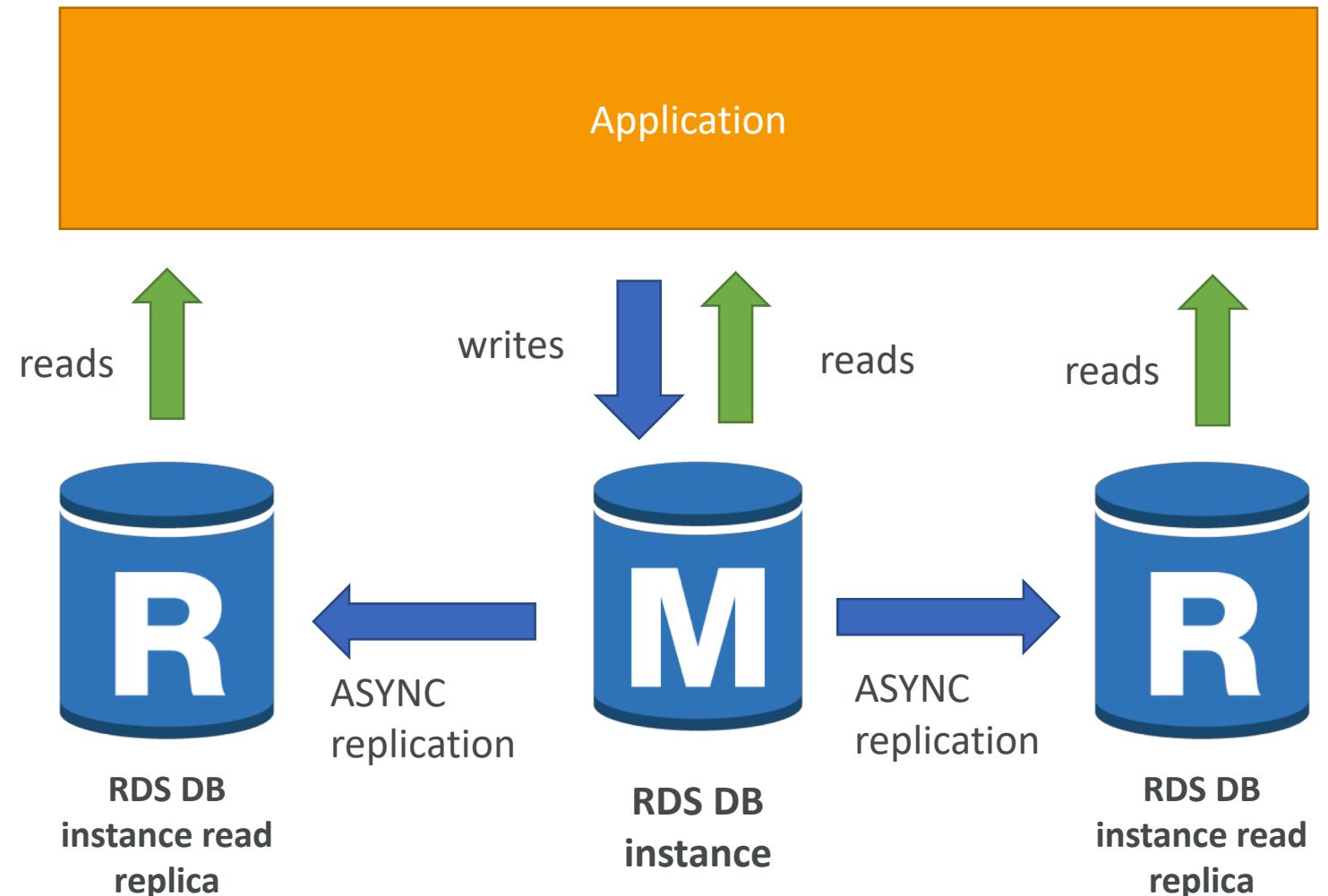
- RDS stands for Relational Database Service
- It's a managed DB service for DB use SQL as a query language.
- It allows you to create databases in the cloud that are managed by AWS
  - Postgres
  - Oracle
  - MySQL
  - MariaDB
  - Oracle
  - Microsoft SQL Server
  - Aurora (AWS Proprietary database)

# Advantage over using RDS versus deploying DB on EC2

- Managed service:
- OS patching level
- Continuous backups and restore to specific timestamp (Point in Time Restore)!
- Monitoring dashboards
- Read replicas for improved read performance
- Multi AZ setup for DR (Disaster Recovery)
- Maintenance windows for upgrades
- Scaling capability (vertical and horizontal)
- BUT you can't SSH into your instances

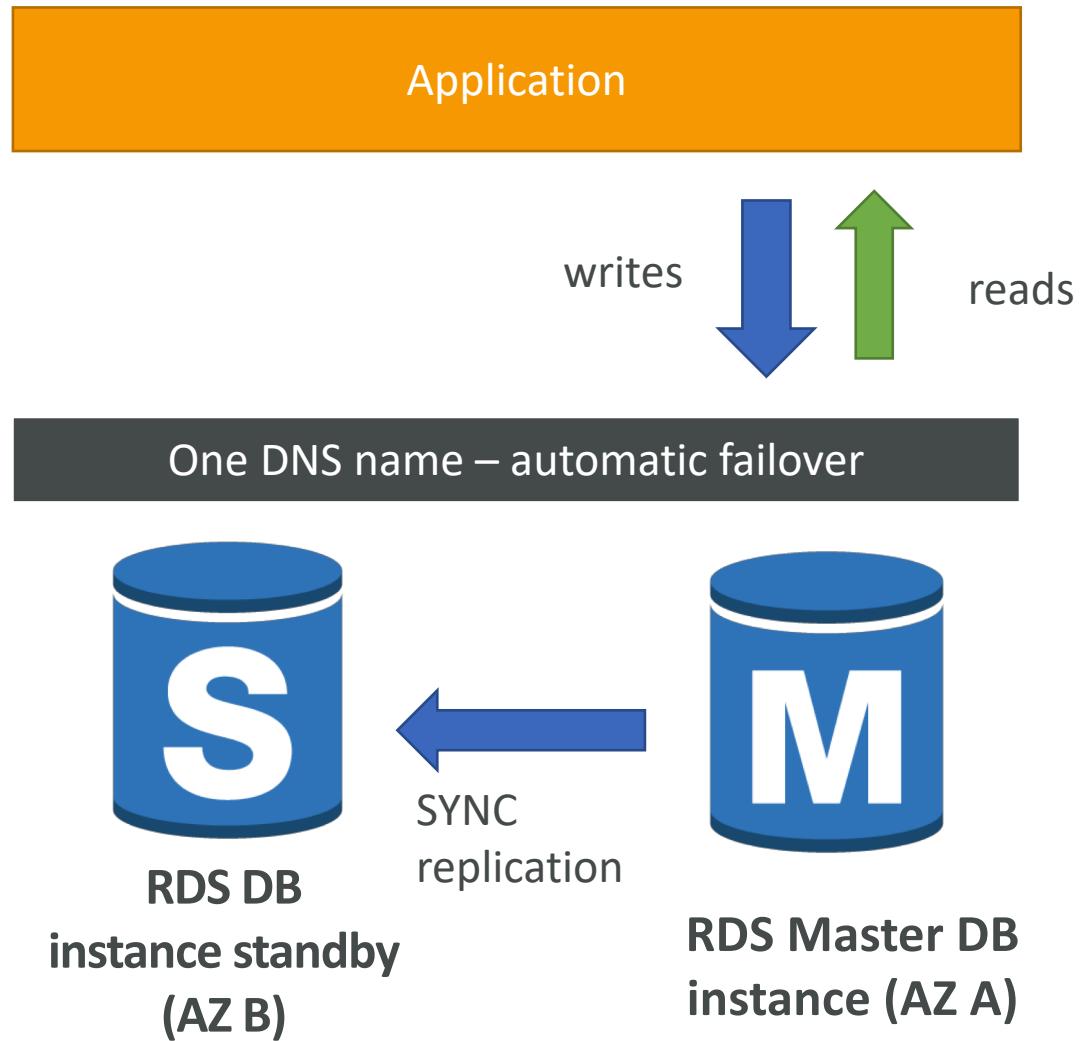
# RDS Read Replicas for read scalability

- Up to 5 Read Replicas
- Within AZ, Cross AZ or Cross Region
- Replication is **ASYNC**, so reads are eventually consistent
- Replicas can be promoted to their own DB
- Applications must update the connection string to leverage read replicas



# RDS Multi AZ (Disaster Recovery)

- SYNC replication
- One DNS name – automatic app failover to standby
- Increase availability
- Failover in case of loss of AZ, loss of network, instance or storage failure
- No manual intervention in apps
- Not used for scaling



# RDS Backups

- Backups are automatically enabled in RDS
- Automated backups:
  - Daily full snapshot of the database
  - Capture transaction logs in real time
  - => ability to restore to any point in time
  - 7 days retention (can be increased to 35 days)
- DB Snapshots:
  - Manually triggered by the user
  - Retention of backup for as long as you want

# RDS Encryption

- Encryption at rest capability with AWS KMS - AES-256 encryption
- SSL certificates to encrypt data to RDS in flight
- To enforce SSL:
  - PostgreSQL: `rds.force_ssl=1` in the AWS RDS Console (Parameter Groups)
  - MySQL: Within the DB:  
`GRANT USAGE ON *.* TO 'mysqluser'@'%' REQUIRE SSL;`
- To connect using SSL:
  - Provide the SSL Trust certificate (can be download from AWS)
  - Provide SSL options when connecting to database

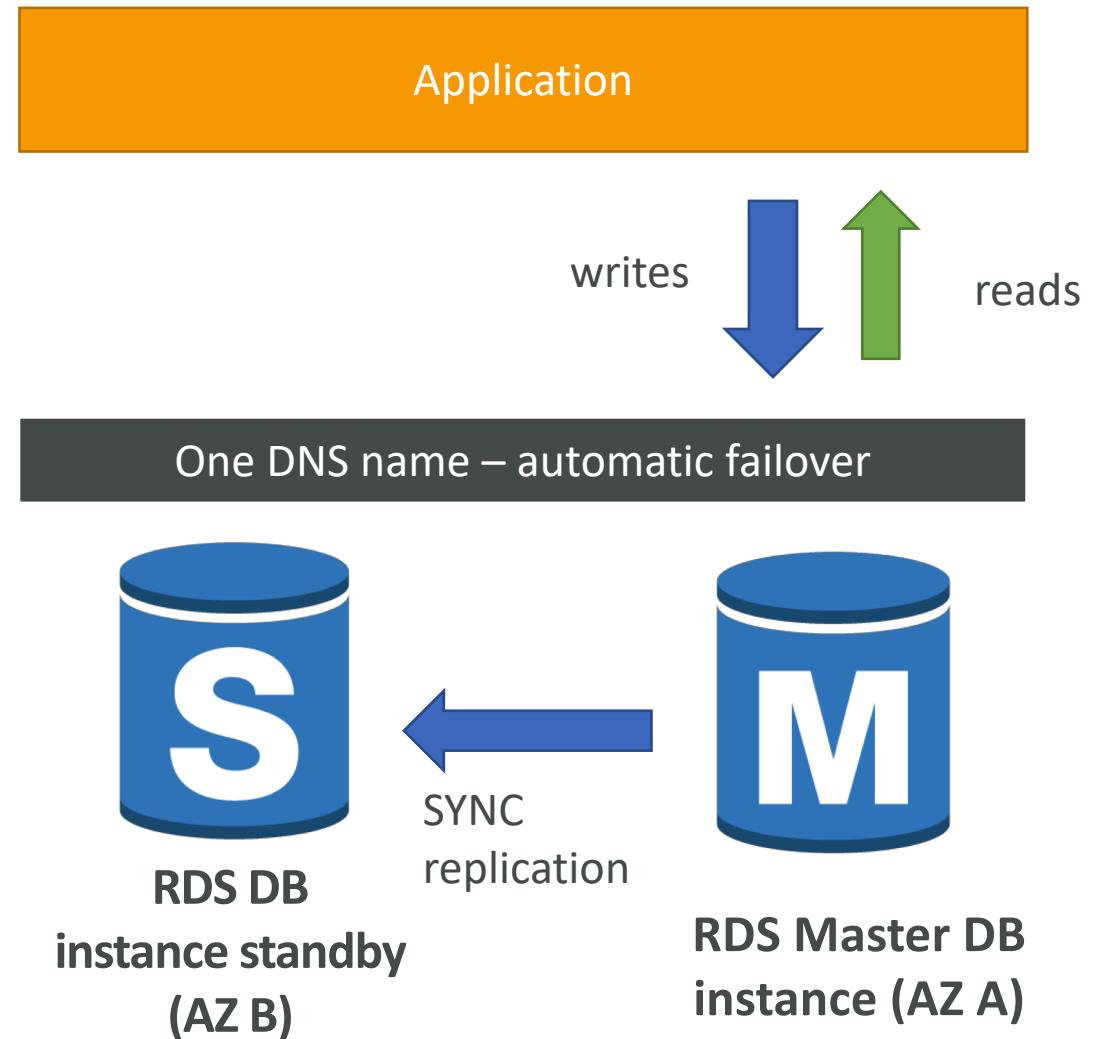
# RDS Security

- RDS databases are usually deployed within a private subnet, not in a public one
- RDS Security works by leveraging security groups (the same concept as for EC2 instances) – it controls who can **communicate** with RDS
- IAM policies help control who can **manage** AWS RDS
- Traditional Username and Password can be used to **login** to the database
- IAM users can now be used too (for MySQL / Aurora – **NEW!**)

# RDS vs Aurora

- Aurora is a proprietary technology from AWS (not open sourced)
- Postgres and MySQL are both supported as Aurora DB (that means your drivers will work as if Aurora was a Postgres or MySQL database)
- Aurora is “AWS cloud optimized” and claims 5x performance improvement over MySQL on RDS, over 3x the performance of Postgres on RDS
- Aurora storage automatically grows in increments of 10GB, up to 64 TB.
- Aurora can have 15 replicas while MySQL has 5, and the replication process is faster (sub 10 ms replica lag)
- Failover in Aurora is instantaneous. It’s HA native.
- Aurora costs more than RDS (20% more) – but is more efficient

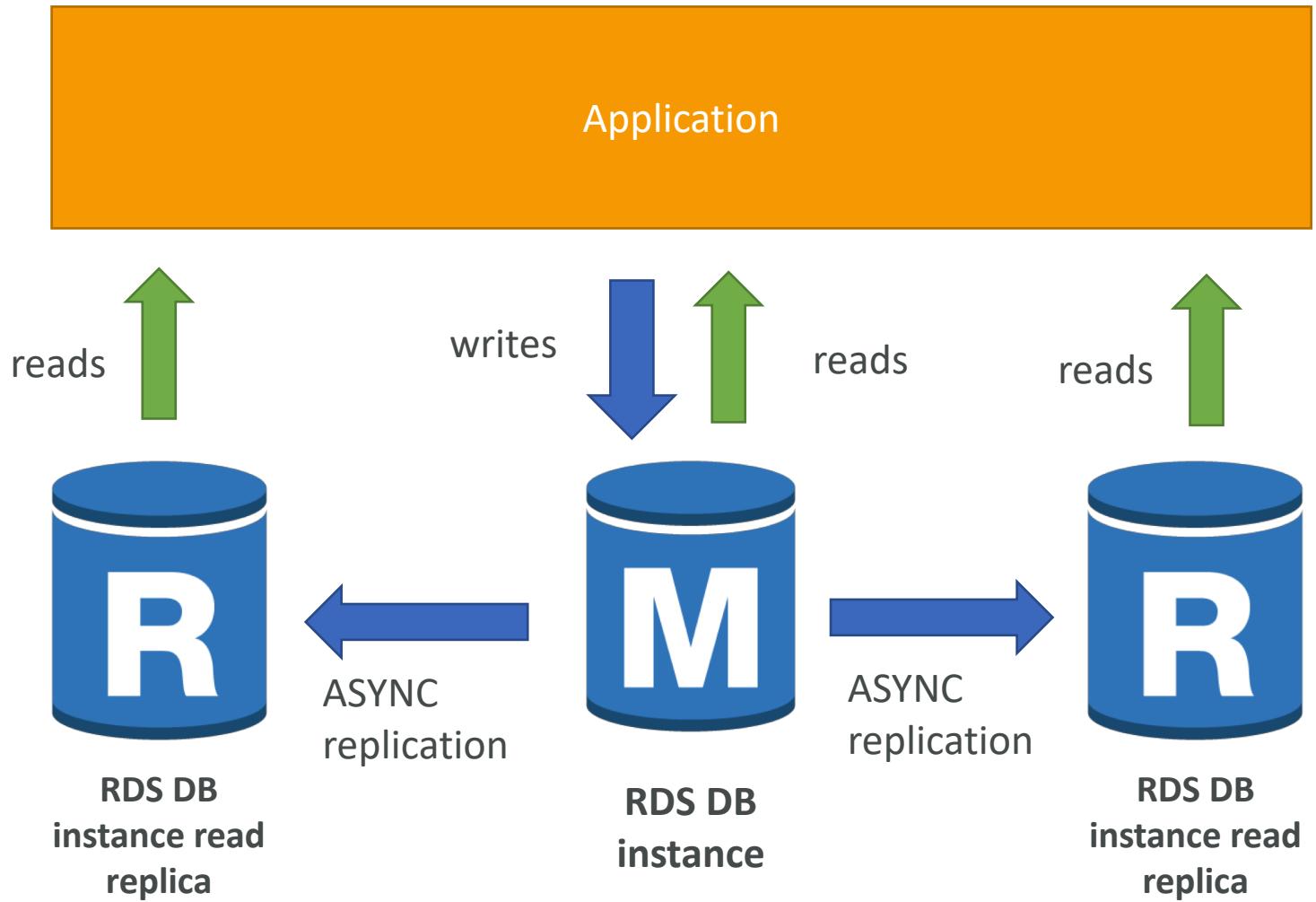
# RDS Multi AZ



# RDS Multi AZ in depth

- Multi AZ is not used to support the reads
- The failover happens only in the following conditions:
  - The primary DB instance fails
  - An Availability Zone outage
  - The DB instance server type is changed
  - The operating system of the DB instance is undergoing software patching.
  - A manual failover of the DB instance was initiated using Reboot with failover.
- No failover for DB operations: long-running queries, deadlocks or database corruption errors.
- Endpoint is the same after failover (no URL change in application needed)
  
- Lower maintenance impact it happens on the standby, which is then promoted to master
- Backups are created from the standby
- Multi AZ is only within a single region, not cross region. Region outages impact availability

# RDS Read Replicas



# RDS Read Replicas in depth

- Read Replicas help scaling read traffic
- A Read Replica can be promoted as a standalone database (manually)
- Read Replicas can be within AZ, Cross AZ or Cross Region.
- Each Read Replica has its own DNS endpoint
- You can have Read Replicas of Read Replicas.
- Read Replicas can be Multi-AZ.
- Read Replicas help with Disaster Recovery (DR) by using a cross region RR
- RDS Read Replicas are not supported for Oracle
- Read Replicas can be used to run BI / Analytics Reports for example

# DB Parameter Groups

- You can configure the DB engine using Parameter Groups
- Dynamic parameters are applied immediately
- Static parameters are applied after instance reboot
- You can modify parameter group associated with a DB (must reboot)
- See documentation for list of parameters for a DB technology
- **Must-know parameter:**
  - PostgreSQL / SQL Server: `rds.force_ssl=1` => force SSL connections
  - Reminder: for SSL on MySQL / MariaDB, you must run:  
`GRANT SELECT ON mydatabase.* TO 'myuser'@'%' IDENTIFIED BY '....' REQUIRE SSL;`

# RDS Backup vs Snapshots

## BACKUPS

- Backups are “continuous” and allow point in time recovery
- Backups happen during maintenance windows
- When you delete a DB instance, you can retain automated backups.
- Backups have a retention period you set between 0 and 35 days

## SNAPSHOTS

- Snapshots takes IO operations and can stop the database from seconds to minutes
- Snapshots taken on Multi AZ DB don’t impact the master – just the standby
- Snapshots are incremental after the first snapshot (which is full)
- You can copy & share DB Snapshots
- Manual Snapshots don’t expire
- You can take a “final snapshot” when you delete your DB

# RDS Security for SysOps

- Encryption at rest:
  - Is done only when you first create the DB instance
  - or: unencrypted DB => snapshot => copy snapshot as encrypted => create DB from snapshot
- Your responsibility:
  - Check the ports / IP / security group inbound rules in DB's SG
  - In-database user creation and permissions
  - Creating a database with or without public access
  - Ensure parameter groups or DB is configured to only allow SSL connections
- AWS responsibility:
  - No SSH access
  - No manual DB patching
  - No manual OS patching
  - No way to audit the underlying instance

# RDS API for SysOps

- `DescribeDBInstances` API –
  - Helps to get a list of all the DB Instances you have deployed including Read Replicas
  - Helps to get the DB version
- `CreateDBSnapshot` API – Make a snapshot of a DB
- `DescribeEvents` API - Helps to return information about events related to your DB Instance
- `RebootDBInstance` API - Helps to initiate a ‘forced’ failover by rebooting DB instance

# RDS with CloudWatch

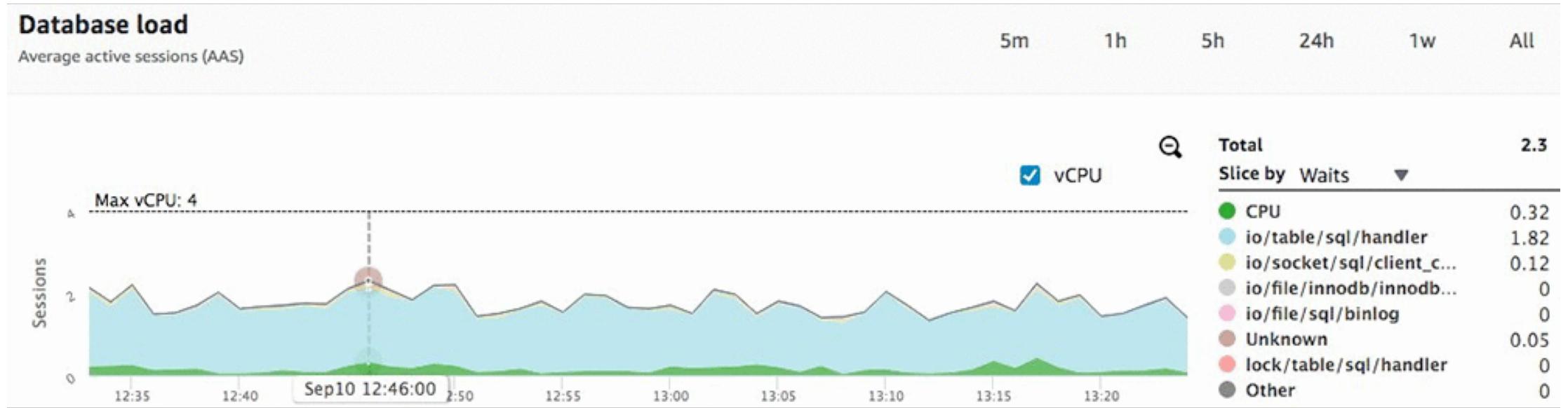
- CloudWatch metrics associated with RDS (gathered from the hypervisor):
  - DatabaseConnections
  - SwapUsage
  - ReadIOPS / WriteIOPS
  - ReadLatency / WriteLatency
  - ReadThroughPut / WriteThroughPut
  - DiskQueueDepth
  - FreeStorageSpace
- Enhanced Monitoring (gathered from an agent on the DB instance)
  - Useful when you need to see how different processes or threads use the CPU
  - Access to over 50 new CPU, memory, file system, and disk I/O metrics

# RDS Performance Insights

- Visualize your database performance and analyze any issues that affect it
- With the Performance Insights dashboard, you can visualize the database load and filter the load:
  - By Waits => find the resource that is the bottleneck (CPU, IO, lock, etc...)
  - By SQL statements => find the SQL statement that is the problem
  - By Hosts => find the server that is using the most our DB
  - By Users => find the user that is using the most our DB
- DBLoad = the number of active sessions for the DB engine
- You can view the SQL queries that are putting load on your database

# Performance Insights Screenshots

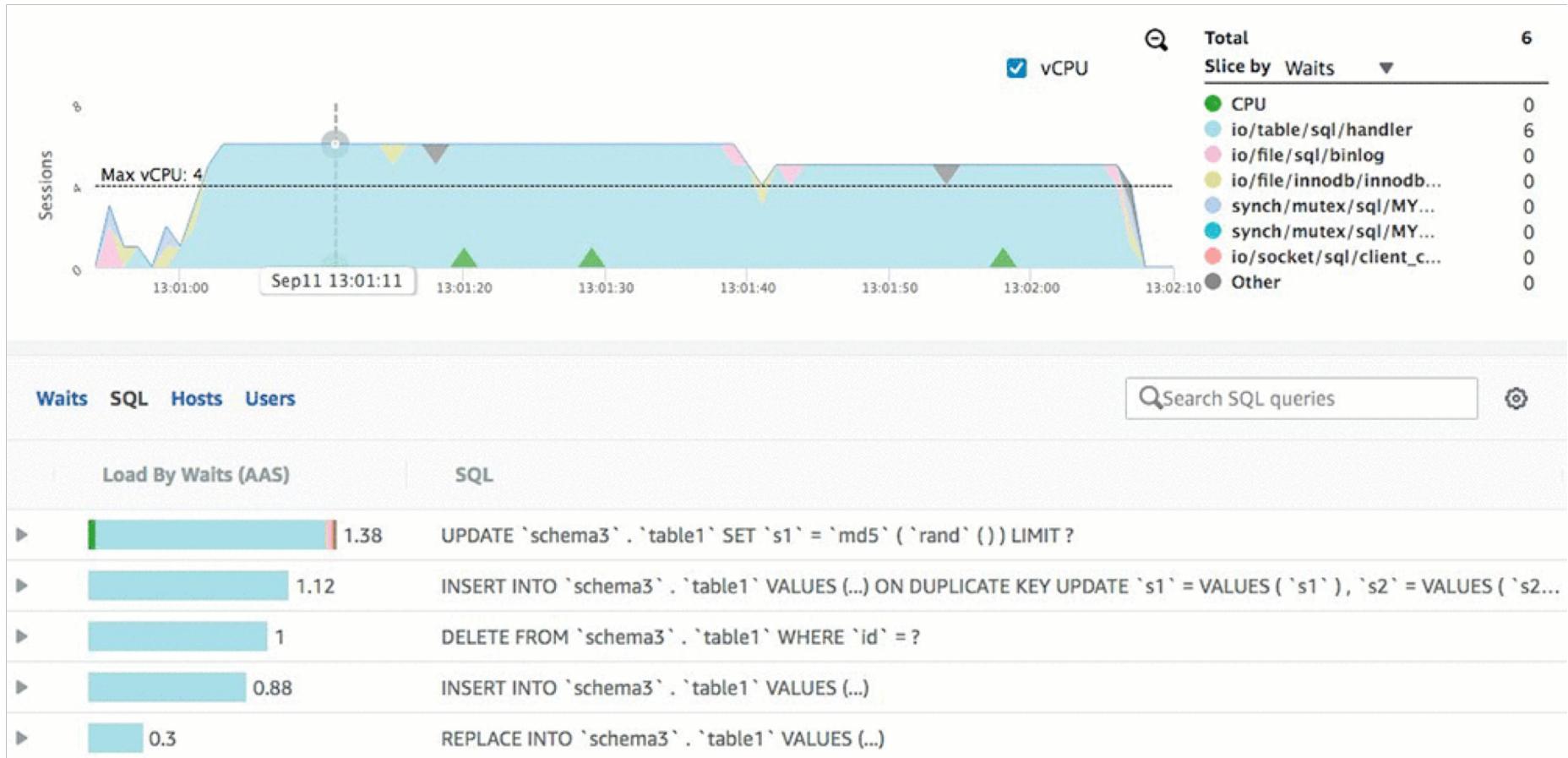
## DB Waits



From: <https://aws.amazon.com/blogs/database/tuning-amazon-rds-for-mysql-with-performance-insights/>

# Performance Insights Screenshots

## SQL



From: <https://aws.amazon.com/blogs/database/tuning-amazon-rds-for-mysql-with-performance-insights/>

# Performance Insights Screenshots

## Users



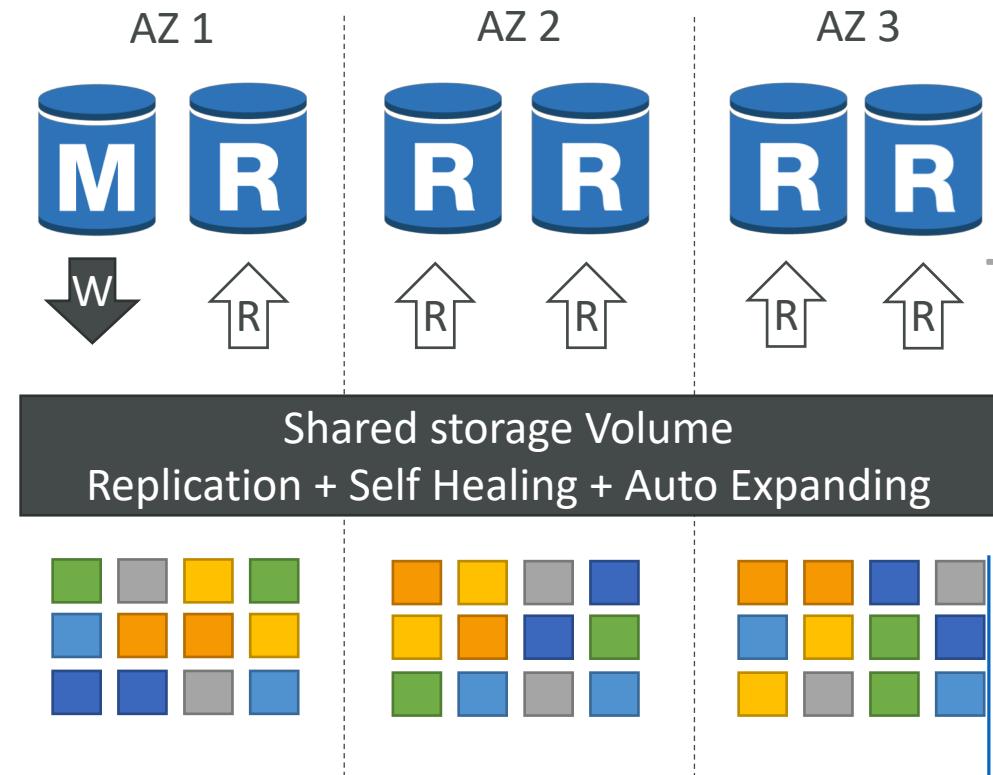
From: <https://aws.amazon.com/blogs/database/tuning-amazon-rds-for-mysql-with-performance-insights/>

# Amazon Aurora

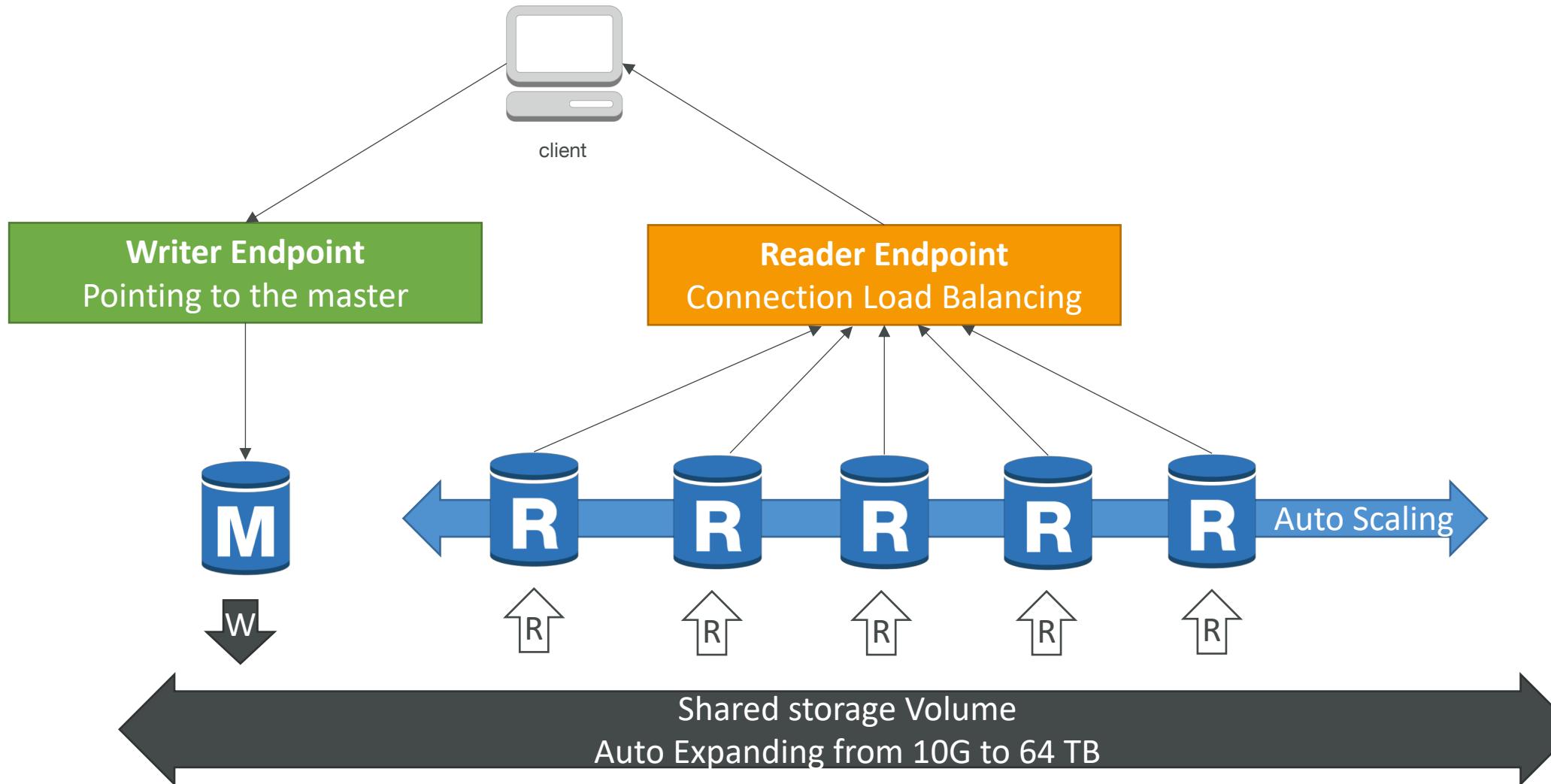
- Aurora is a proprietary technology from AWS (not open sourced)
- Postgres and MySQL are both supported as Aurora DB (that means your drivers will work as if Aurora was a Postgres or MySQL database)
- Aurora is “AWS cloud optimized” and claims 5x performance improvement over MySQL on RDS, over 3x the performance of Postgres on RDS
- Aurora storage automatically grows in increments of 10GB, up to 64 TB.
- Aurora can have 15 replicas while MySQL has 5, and the replication process is faster (sub 10 ms replica lag)
- Failover in Aurora is instantaneous. It’s HA native.
- Aurora costs more than RDS (20% more) – but is more efficient

# Aurora High Availability and Read Scaling

- 6 copies of your data across 3 AZ:
  - 4 copies out of 6 needed for writes
  - 3 copies out of 6 need for reads
  - Self healing with peer-to-peer replication
  - Storage is striped across 100s of volumes
- One Aurora Instance takes writes (master)
- Automated failover for master in less than 30 seconds
- Master + up to 15 Aurora Read Replicas serve reads
- Support for Cross Region Replication



# Aurora DB Cluster



# Features of Aurora

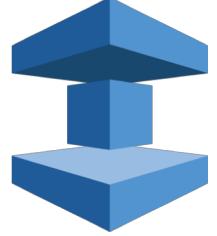
- Automatic fail-over
- Backup and Recovery
- Isolation and security
- Industry compliance
- Push-button scaling
- Automated Patching with Zero Downtime
- Advanced Monitoring
- Routine Maintenance
- Backtrack: restore data at any point of time without using backups

# Aurora Security

- Encryption at rest using KMS
- Automated backups, snapshots and replicas are also encrypted
- Encryption in flight using SSL (same process as MySQL or Postgres)
- Authentication using IAM
- You are responsible for protecting the instance with security groups
- You can't SSH

# Aurora Serverless

- No need to choose an instance size
- Only supports MySQL 5.6 (as of Jan 2019) & Postgres (beta)
- Helpful when you can't predict the workload
- DB cluster starts, shutdown and scales automatically based on CPU / connections
- Can migrate from Aurora Cluster to Aurora Serverless and vice versa
- Aurora Serverless usage is measured in ACU (Aurora Capacity Units)
- Billed in 5 minutes increment of ACU
- Note: some features of Aurora aren't supported in Serverless mode, be sure to check the documentation if you plan on using it



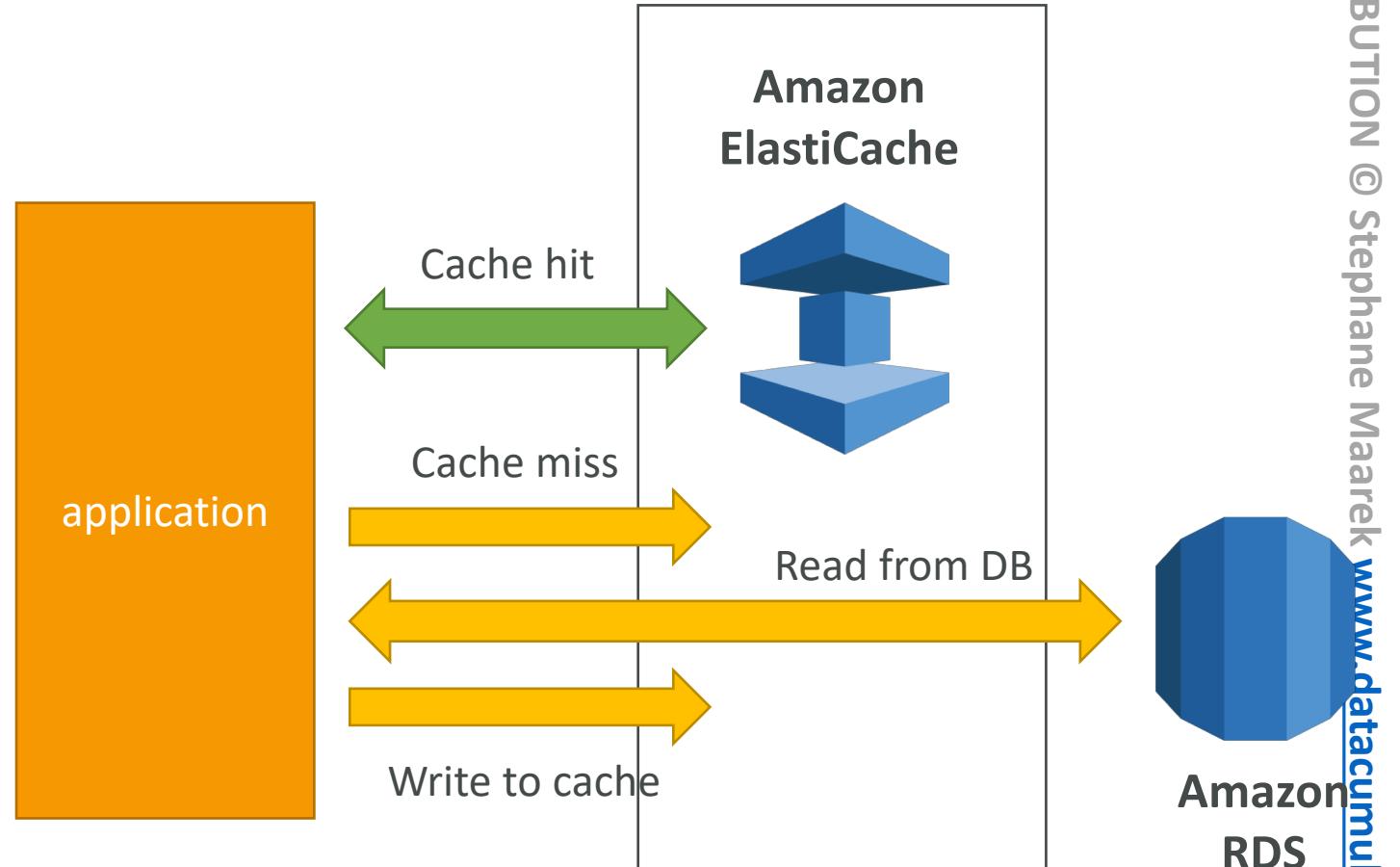
# AWS ElastiCache Overview

- The same way RDS is to get managed Relational Databases...
- ElastiCache is to get managed Redis or Memcached
- Caches are in-memory databases with really high performance, low latency
- Helps reduce load off of databases for read intensive workloads
- Helps make your application stateless
- Write Scaling using sharding
- Read Scaling using Read Replicas
- Multi AZ with Failover Capability
- AWS takes care of OS maintenance / patching, optimizations, setup, configuration, monitoring, failure recovery and backups

# ElastiCache

## Solution Architecture - DB Cache

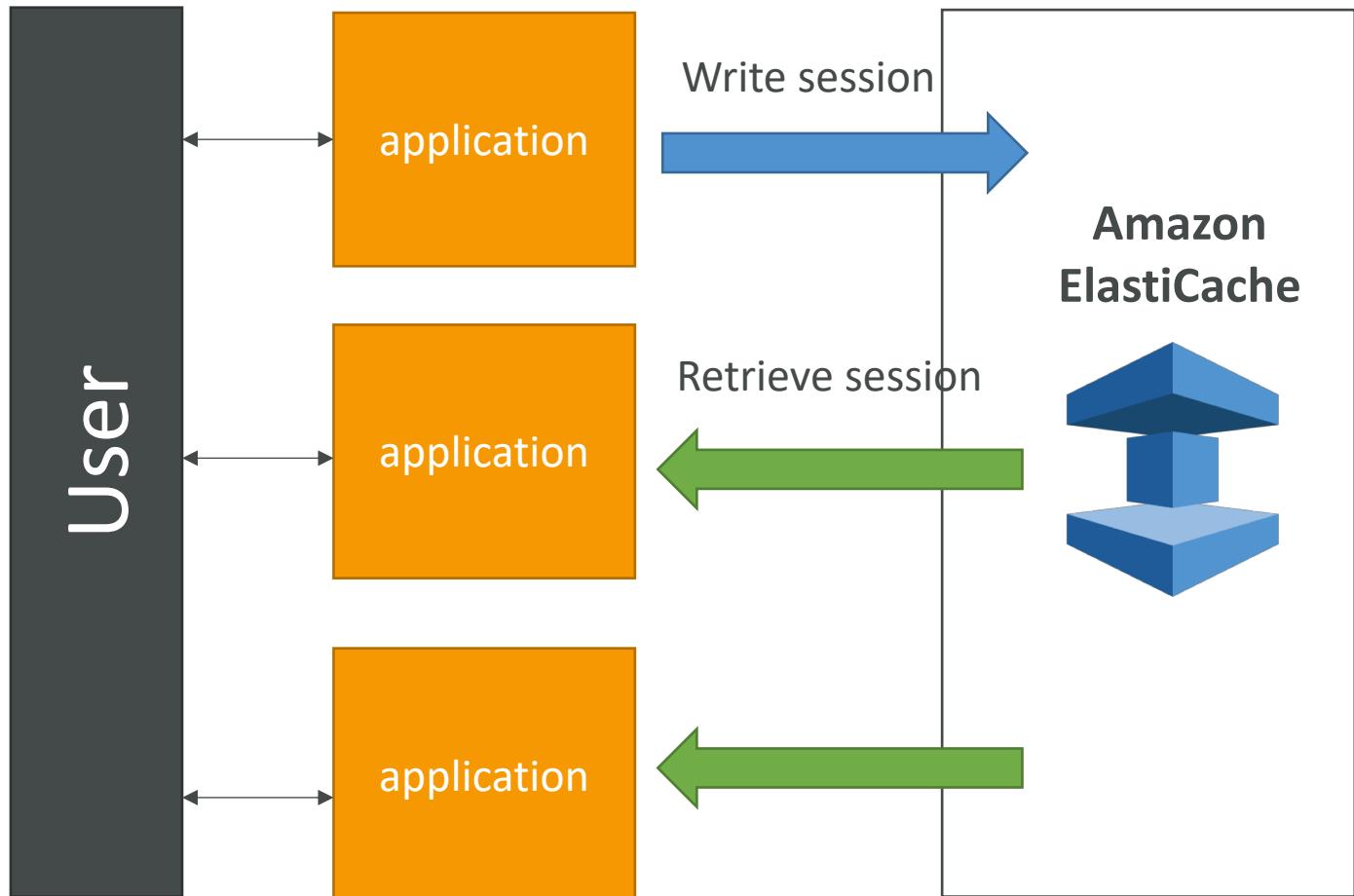
- Applications queries ElastiCache, if not available, get from RDS and store in ElastiCache.
- Helps relieve load in RDS
- Cache must have an invalidation strategy to make sure only the most current data is used in there.



# ElastiCache

## Solution Architecture – User Session Store

- User logs into any of the application
- The application writes the session data into ElastiCache
- The user hits another instance of our application
- The instance retrieves the data and the user is already logged in



# Redis Overview

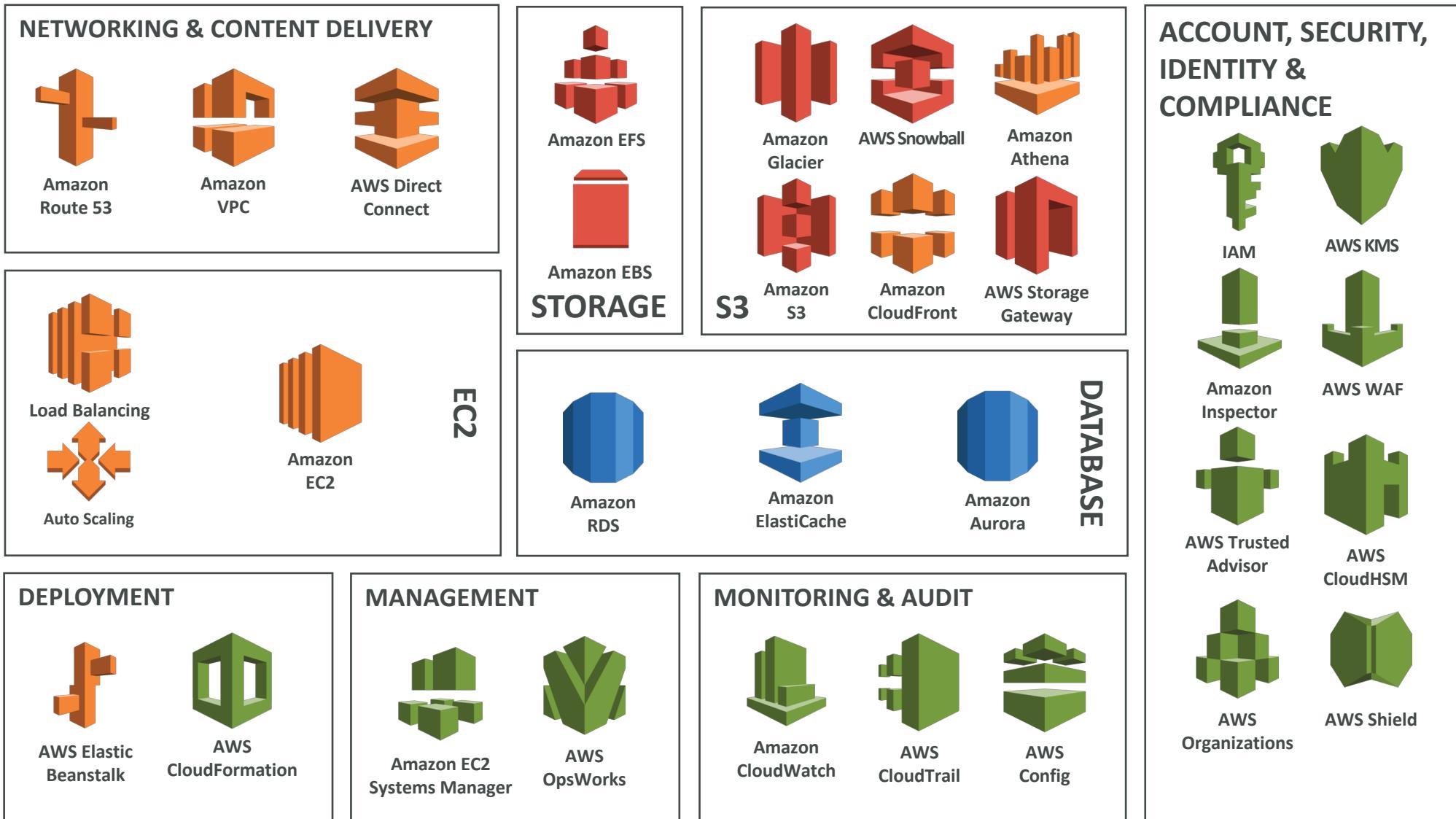
- Redis is an in-memory key-value store
- Super low latency (sub ms)
- Cache survive reboots by default (it's called persistence)
- Great to host
  - User sessions
  - Leaderboard (for gaming)
  - Distributed states
  - Relieve pressure on databases (such as RDS)
  - Pub / Sub capability for messaging
- Multi AZ with Automatic Failover for disaster recovery if you don't want to lose your cache data
- Support for Read Replicas

# Memcached Overview

- Memcached is an in-memory object store
- Cache doesn't survive reboots
- Use cases:
  - Quick retrieval of objects from memory
  - Cache often accessed objects
- Overall, Redis has largely grown in popularity and has better feature sets than Memcached.
- I would personally only use Redis for caching needs.
- AWS exam wouldn't ask if Redis or Memcached is better. Just "ElastiCache" in general

# AWS Monitoring, Audit and Performance

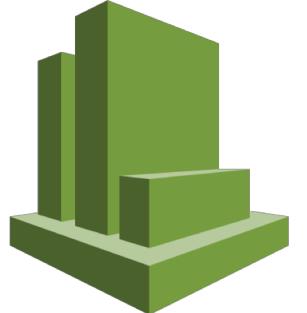
CloudWatch, CloudTrail & AWS Config



# Introduction to Monitoring in AWS

- AWS CloudWatch:
  - Metrics: Collect and track key metrics
  - Logs: Collect, monitor, analyze and store log files
  - Events: Send notifications when certain events happen in your AWS
  - Alarms: React in real-time to metrics / events
- AWS CloudTrail:
  - Internal monitoring of API calls being made
  - Audit changes to AWS Resources by your users
- AWS Config:
  - Track configuration of resources over time
  - Evaluate configuration compliance based on rules

# AWS CloudWatch Metrics



- CloudWatch provides metrics for every services in AWS
- **Metric** is a variable to monitor (CPUUtilization, NetworkIn...)
- Metrics belong to **namespaces**
- Dimension is an attribute of a metric (instance id, environment, etc....).
- Up to 10 dimensions per metric
- Metrics have **timestamps**
- Can create CloudWatch dashboards of metrics

# AWS CloudWatch EC2 Detailed monitoring

- EC2 instance metrics have metrics “every 5 minutes”
- With detailed monitoring (for a cost), you get data “every 1 minute”
- Use detailed monitoring if you want to more prompt scale your ASG!
- The AWS Free Tier allows us to have 10 detailed monitoring metrics
- Note: EC2 Memory usage is by default not pushed (must be pushed from inside the instance as a custom metric)

# AWS CloudWatch Custom Metrics

- Possibility to define and send your own custom metrics to CloudWatch
- Ability to use dimensions (attributes) to segment metrics
  - Instance.id
  - Environment.name
- Metric resolution:
  - Standard: 1 minute
  - High Resolution: up to 1 second (**StorageResolution** API parameter) – Higher cost
- Use API call **PutMetricData**
- Use exponential back off in case of throttle errors

# CloudWatch Dashboards

- Great way to setup dashboards for quick access to keys metrics
- Dashboards are global
- Dashboards can include graphs from different regions
- You can change the time zone & time range of the dashboards
- You can setup automatic refresh (10s, 1m, 2m, 5m, 15m)
- Pricing:
  - 3 dashboards (up to 50 metrics) for free
  - \$3/dashboard/month afterwards

# AWS CloudWatch Logs

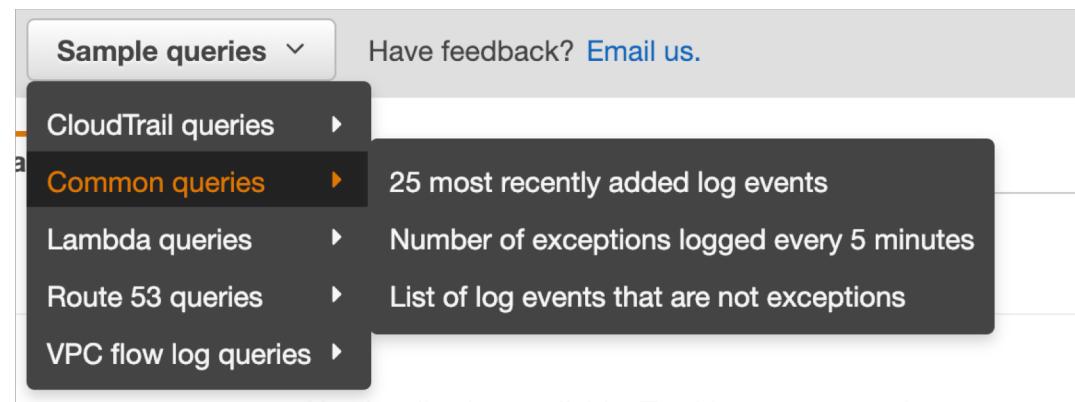
- Applications can send logs to CloudWatch using the SDK
- CloudWatch can collect log from:
  - Elastic Beanstalk: collection of logs from application
  - ECS: collection from containers
  - AWS Lambda: collection from function logs
  - VPC Flow Logs: VPC specific logs
  - API Gateway
  - CloudTrail based on filter
  - CloudWatch log agents: for example on EC2 machines
  - Route53: Log DNS queries
- CloudWatch Logs can go to:
  - Batch exporter to S3 for archival
  - Stream to ElasticSearch cluster for further analytics

# AWS CloudWatch Logs

- Logs storage architecture:
  - Log groups: arbitrary name, usually representing an application
  - Log stream: instances within application / log files / containers
- Can define log expiration policies (never expire, 30 days, etc..)
- Using the AWS CLI we can tail CloudWatch logs
- To send logs to CloudWatch, make sure IAM permissions are correct!
- Security: encryption of logs using KMS at the Group Level

# CloudWatch Logs Metric Filter & Insights

- CloudWatch Logs can use filter expressions
  - For example, find a specific IP inside of a log
  - Metric filters can be used to trigger alarms
- CloudWatch Logs Insights (new – Nov 2018) can be used to query logs and add queries to CloudWatch Dashboards



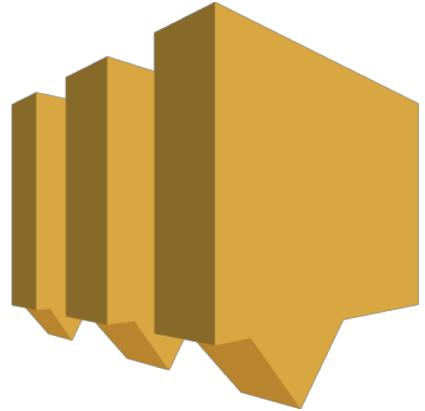
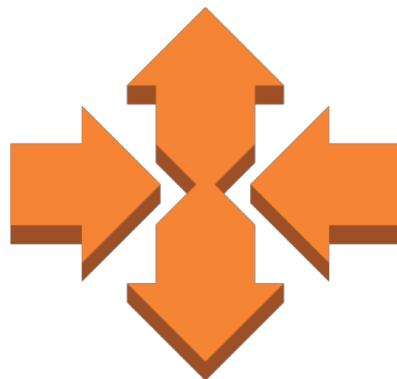
# AWS CloudWatch Alarms



- Alarms are used to trigger notifications for any metric
- Alarms can go to Auto Scaling, EC2 Actions, SNS notifications
- Various options (sampling, %, max, min, etc...)
- Alarm States:
  - OK
  - INSUFFICIENT\_DATA
  - ALARM
- Period:
  - Length of time in seconds to evaluate the metric
  - High resolution custom metrics: can only choose 10 sec or 30 sec

# CloudWatch Alarm Targets

- Stop, Terminate, Reboot, or Recover an EC2 Instance
- Trigger Auto Scaling Action
- Send notification to SNS (from which you can do pretty much anything)



# CloudWatch Alarm: good to know

- Alarms can be created based on CloudWatch Logs Metrics Filters
- CloudWatch doesn't test or validate the actions that is assigned
- To test alarms and notifications, set the alarm state to Alarm using CLI

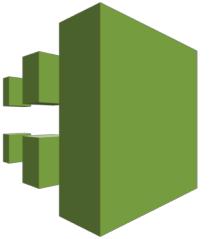
```
aws cloudwatch set-alarm-state --alarm-name "myalarm" --state-value  
ALARM --state-reason "testing purposes"
```

# AWS CloudWatch Events



- Source + Rule => Target
- Schedule: Cron jobs
- Event Pattern: Event rules to react to a service doing something
  - Ex: CodePipeline state changes!
- Triggers to Lambda functions, SQS/SNS/Kinesis Messages
- CloudWatch Event creates a small JSON document to give information about the change

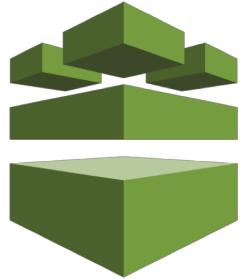
# AWS CloudTrail



- Provides governance, compliance and audit for your AWS Account
- CloudTrail is enabled by default!
- Get an history of events / API calls made within your AWS Account by:
  - Console
  - SDK
  - CLI
  - AWS Services
- Can put logs from CloudTrail into CloudWatch Logs
- If a resource is deleted in AWS, look into CloudTrail first!

# CloudTrail continued...

- CloudTrail shows the past 90 days of activity
- The default UI only shows “Create”, “Modify” or “Delete” events
- **CloudTrail Trail:**
  - Get a detailed list of all the events you choose
  - Ability to store these events in S3 for further analysis
  - Can be region specific or global
- CloudTrail Logs have SSE-S3 encryption when placed into S3
- Control access to S3 using IAM, Bucket Policy, etc...



# AWS Config

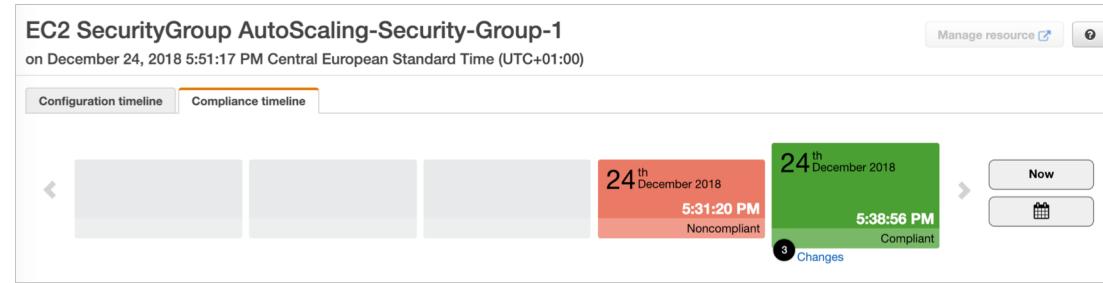
- Helps with auditing and compliance of your AWS resources
- Helps record configurations and changes over time
- Helps record compliance over time
- Possibility of storing AWS Config data into S3 (can be queried by Athena)
- Questions that can be solved by AWS Config:
  - Is there unrestricted SSH access to my security groups?
  - Do my buckets have any public access?
  - How has my ALB configuration changed over time?
- You can receive alerts (SNS notifications) for any changes
- AWS Config is a per-region service
- Can be aggregated across regions and accounts

# Config Rules

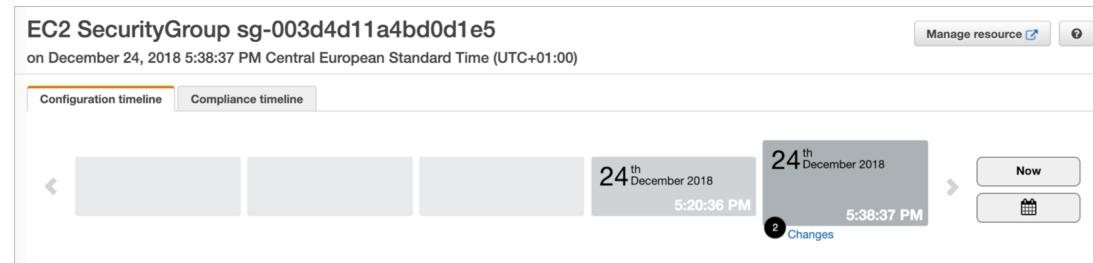
- Can use AWS managed config rules (over 75)
- Can make custom config rules (must be defined in AWS Lambda)
  - Evaluate if each EBS disk is of type gp2
  - Evaluate if each EC2 instance is t2.micro
- Rules can be evaluated triggered:
  - For each config change
  - And / or: at regular time intervals
- Pricing:
  - No free tier
  - ~\$2 USD per active rule per region per month (less after 10 rules)

# AWS Config Resource

- View compliance of a resource over time



- View configuration of a resource over time



- View CloudTrail API calls if enabled

# CloudWatch vs CloudTrail vs Config

- CloudWatch
  - Performance monitoring (metrics, CPU, network, etc...) & dashboards
  - Events & Alerting
  - Log Aggregation & Analysis
- CloudTrail
  - Record API calls made within your Account by everyone
  - Can define trails for specific resources
  - Global Service
- Config
  - Record configuration changes
  - Evaluate resources against compliance rules
  - Get timeline of changes and compliance

# For an Elastic Load Balancer

- CloudWatch:
  - Monitoring Incoming connections metric
  - Visualize error codes as a % over time
  - Make a dashboard to get an idea of your load balancer performance
- Config:
  - Track security group rules for the Load Balancer
  - Track configuration changes for the Load Balancer
  - Ensure an SSL certificate is always assigned to the Load Balancer (compliance)
- CloudTrail:
  - Track who made any changes to the Load Balancer with API calls

# AWS Account Management

Health Dashboards, AWS Organizations and Billing Console

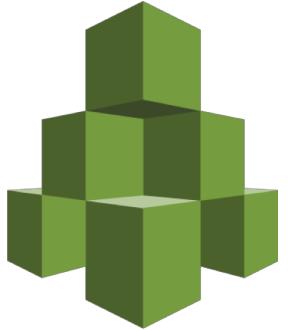
# AWS Status - Service Health Dashboard

- Shows all regions, all services health
- Shows historical information for each day
- Has an RSS feed you can subscribe to
- <https://status.aws.amazon.com/>

# AWS Personal Health Dashboard

- Global service
  - Show how AWS outages directly impact you
  - Shows impact on your resources
  - List issues and actions you can do to remediate them
- 
- <https://phd.aws.amazon.com/>

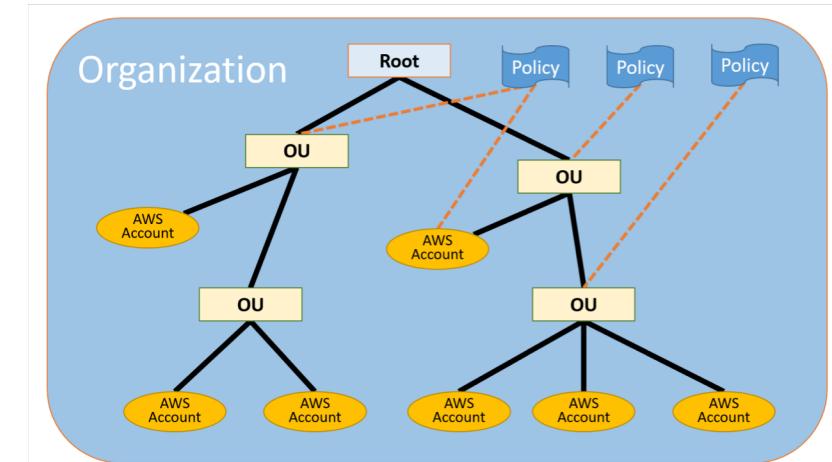
# AWS Organizations



- Global service
- Allows to manage multiple AWS accounts
- The main account is the master account – you can't change it
- Other accounts are member accounts
- Member accounts can only be part of one organization
- Consolidated Billing across all accounts - single payment method
- Pricing benefits from aggregated usage (volume discount for EC2, S3...)
- API is available to automate AWS account creation

# OU & Service Control Policies (SCPs)

- Organize accounts in Organizational Unit (OU)
  - Can be anything: dev / test / prod or Finance / HR / IT
  - Can nest OU within OU
- Apply Service Control Policies (SCPs) to OU
  - Permit / Deny access to AWS services
  - SCP has a similar syntax to IAM
  - It's a filter to IAM
- Helpful for sandbox account creation
- Helpful to separate dev and prod resources
- Helpful to only allow approved services



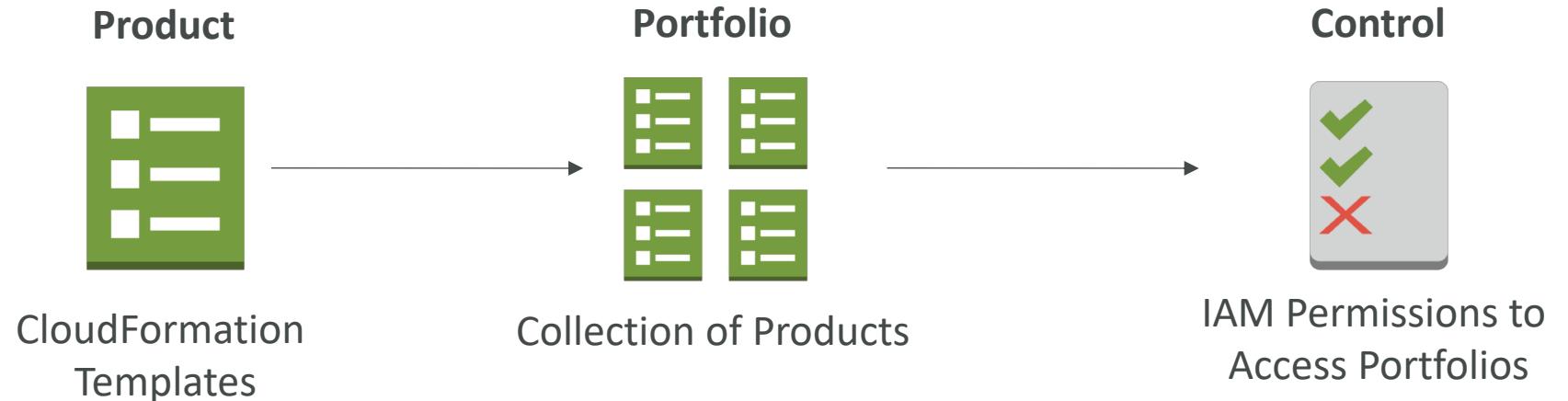


# AWS Service Catalog

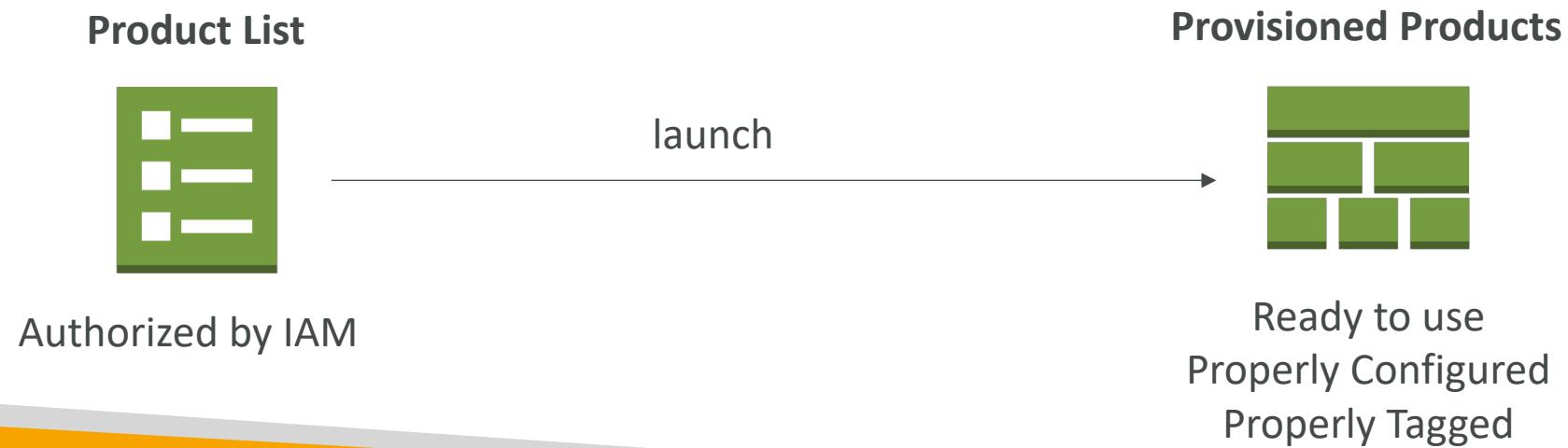
- Users that are new to AWS have too many options, and may create stacks that are not compliant / in line with the rest of the organization
- Some users just want a quick **self-service portal** to launch a set of authorized products pre-defined by admins
- Includes: virtual machines, databases, storage options, etc...
- Enter AWS Service Catalog!

# Service Catalog diagram

## ADMIN TASKS



## USER TASKS





# AWS Service Catalog

- Create and manage catalogs of IT services that are approved on AWS
- The “products” are CloudFormation templates
- Ex: Virtual machine images, Servers, Software, Databases, Regions, IP address ranges
- CloudFormation helps ensure consistency, and standardization by Admins
- They are assigned to Portfolios (teams)
- Teams are presented a self-service portal where they can launch the products
- All the deployed products are centrally managed deployed services
- Helps with governance, compliance, and consistency
- Can give user access to launching products without requiring deep AWS knowledge
- Integrations with “self-service portals” such as ServiceNow

# AWS Billing Alarms



- Billing data metric is stored in CloudWatch us-east-1
- Billing data are for overall **worldwide** AWS costs
- It's for actual cost, not for project costs
  
- Let's create a billing alarm together!

# AWS Cost Explorer

- A graphical tool to view and analyze your costs and usage
- Review charges and cost associated with your AWS account or org
- Forecast spending for the next 3 months
- Get recommendations for which EC2 Reserved Instances to purchase
- Access to default reports
- API to build custom cost management applications

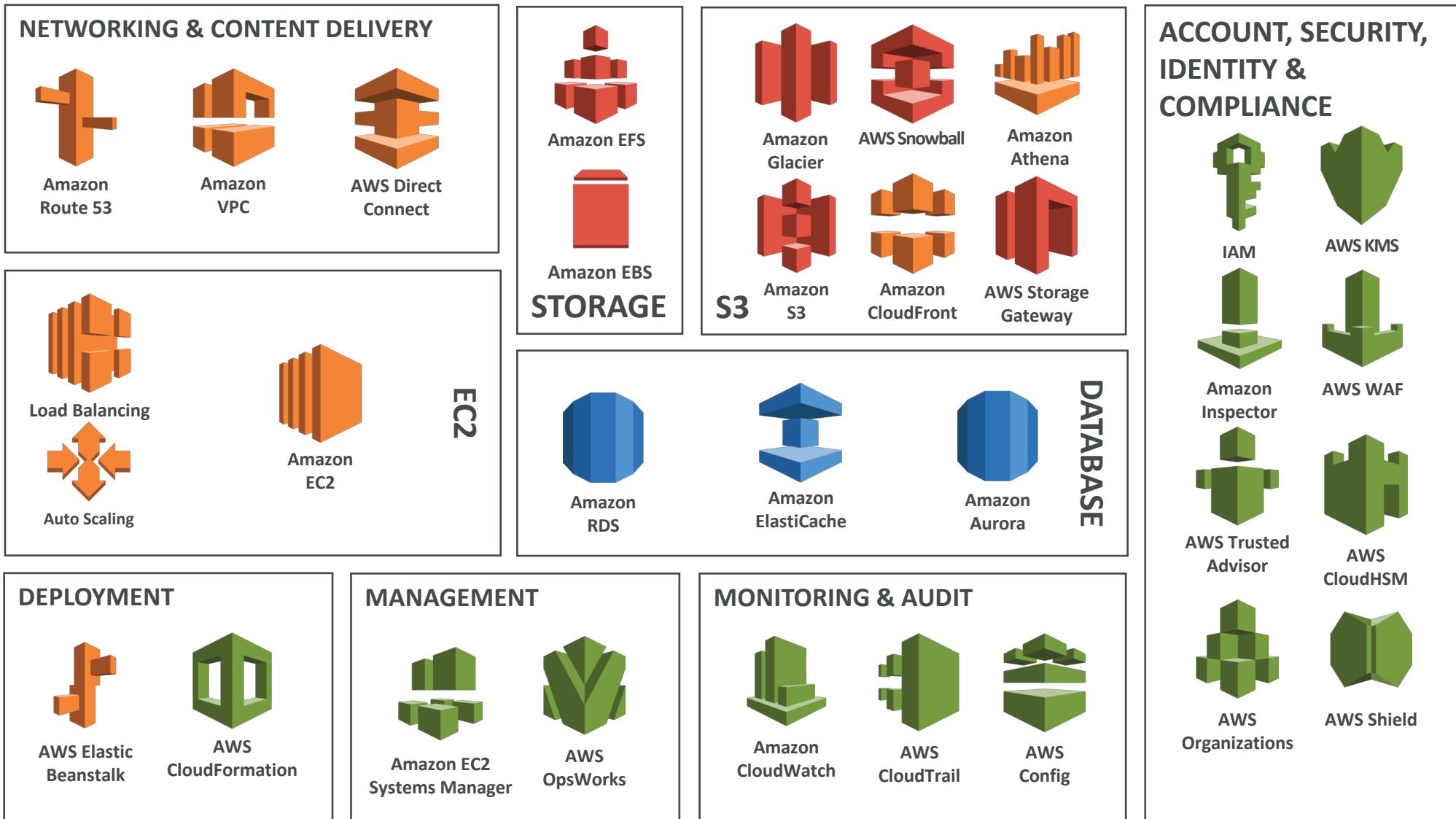
# AWS Budgets

- Create budget and send alarms when costs exceeds the budget
- 3 types of budgets: Usage, Cost, Reservation
- For Reserved Instances (RI)
  - Track utilization
  - Supports EC2, ElastiCache, RDS, Redshift
- Up to 5 SNS notifications per budget
- Can filter by: Service, Linked Account, Tag, Purchase Option, Instance Type, Region, Availability Zone, API Operation, etc...
- Same options as AWS Cost Explorer!
- 2 budgets are free, then \$0.02/day/budget

# AWS Cost Allocation Tags

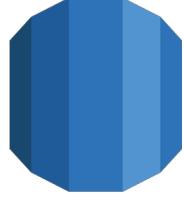
- With Tags we can track resources that relate to each other
- With Cost Allocation Tags we can enable detailed costing reports
- Just like Tags, but they show up as columns in Reports
- AWS Generated Cost Allocation Tags
  - Automatically applied to the resource you create
  - Starts with Prefix **aws:** (e.g. `aws: createdBy`)
  - They're not applied to resources created before the activation
- User tags
  - Defined by the user
  - Starts with Prefix **user:**
- Cost Allocation Tags just appear in the Billing Console
- Takes up to 24 hours for the tags to show up in the report

# Security and Compliance



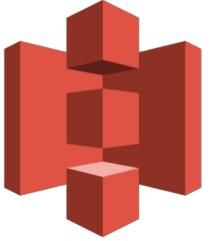
# AWS Shared Responsibility Model

- AWS responsibility - Security **of** the Cloud
  - Protecting infrastructure (hardware, software, facilities, and networking) that runs all of the AWS services
  - Managed services like S3, DynamoDB, RDS etc
- Customer responsibility - Security **in** the Cloud
  - For EC2 instance, customer is responsible for management of the guest OS (including security patches and updates), firewall & network configuration, IAM etc



# Example, for RDS

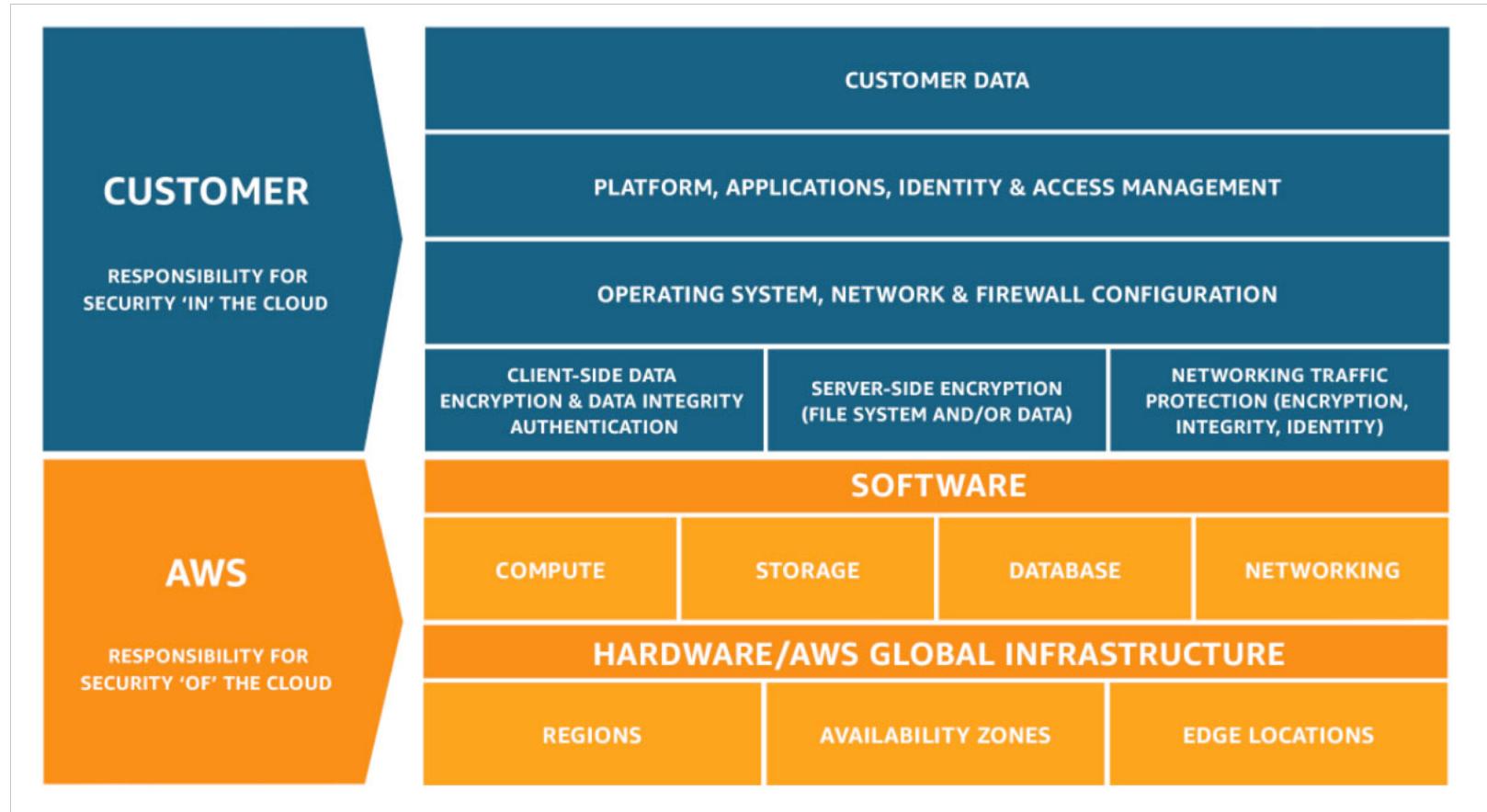
- AWS responsibility:
  - Manage the underlying EC2 instance, disable SSH access
  - Automated DB patching
  - Automated OS patching
  - Audit the underlying instance and disks & guarantee it functions
- Your responsibility:
  - Check the ports / IP / security group inbound rules in DB's SG
  - In-database user creation and permissions
  - Creating a database with or without public access
  - Ensure parameter groups or DB is configured to only allow SSL connections
  - Database encryption setting



# Example, for S3

- AWS responsibility:
  - Guarantee you get unlimited storage
  - Guarantee you get encryption
  - Ensure separation of the data between different customers
  - Ensure AWS employees can't access your data
- Your responsibility:
  - Bucket configuration
  - Bucket policy / public setting
  - IAM user and roles
  - Enabling encryption

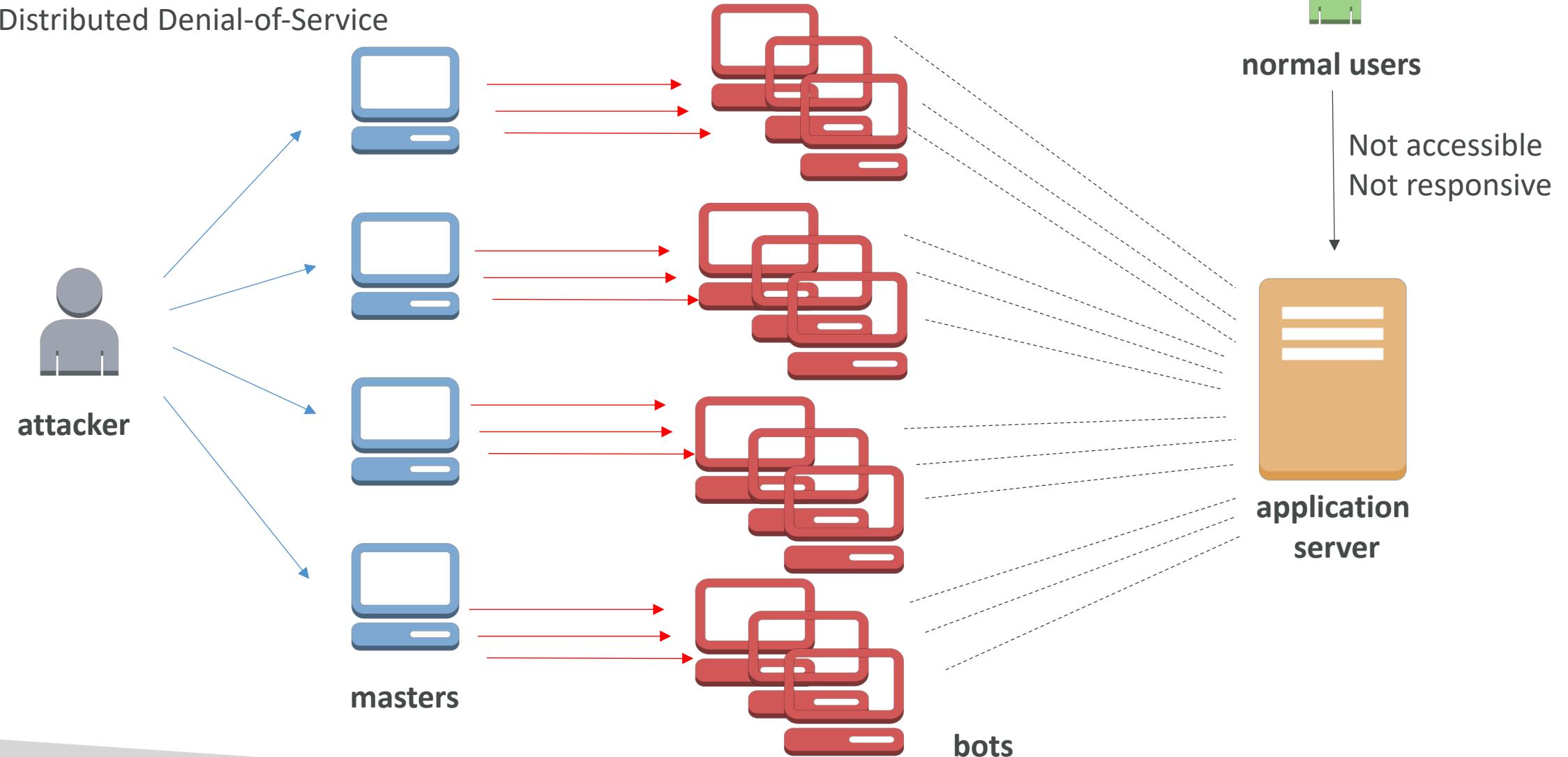
# Shared Responsibility Model diagram



<https://aws.amazon.com/compliance/shared-responsibility-model/>

# What's a DDOS\* Attack?

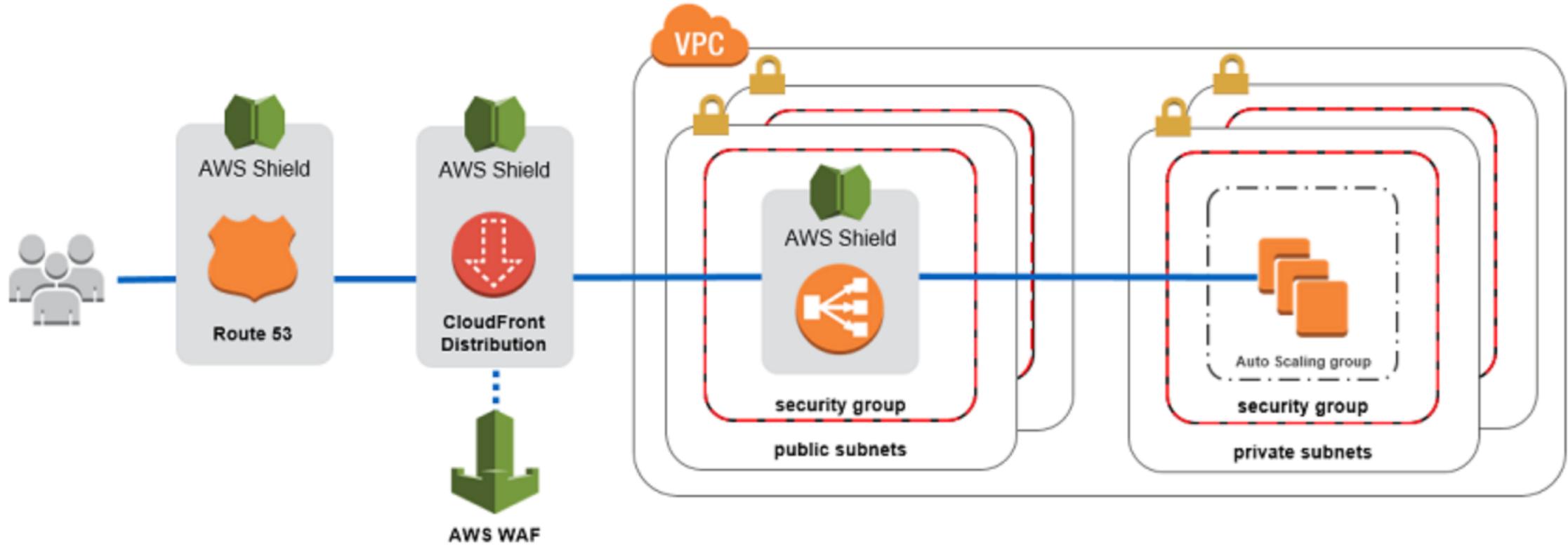
\*Distributed Denial-of-Service



# DDOS Protection on AWS

- AWS Shield Standard: protects against DDOS attack for your website and applications, for all customers at no additional costs
- AWS Shield Advanced: 24/7 premium DDoS protection
- AWS WAF: Filter specific requests based on rules
- CloudFront and Route 53:
  - Availability protection using global edge network
  - Combined with AWS Shield, provides attack mitigation at the edge
- Be ready to scale – leverage AWS Auto Scaling
- Separate static resources (S3 / CloudFront) from dynamic ones (EC2 / ALB)
- Read the whitepaper for details:  
[https://dl.awsstatic.com/whitepapers/Security/DDoS\\_White\\_Paper.pdf](https://dl.awsstatic.com/whitepapers/Security/DDoS_White_Paper.pdf)

# Sample Reference Architecture

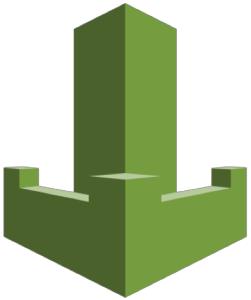


<https://aws.amazon.com/answers/networking/aws-ddos-attack-mitigation/>



# AWS Shield

- **AWS Shield Standard:**
  - Free service that is activated for every AWS customer
  - Provides protection from attacks such as SYN/UDP Floods, Reflection attacks and other layer 3/layer 4 attacks
- **AWS Shield Advanced:**
  - Optional DDoS mitigation service (\$3,000 per month per organization)
  - Protect against more sophisticated attack on CloudFront, Route 53, Classic, Application & Network Load Balancer (select regions), Elastic IP / EC2
  - **24/7 access to AWS DDoS response team (DRP)**
  - Protect against higher fees during usage spikes due to DDoS

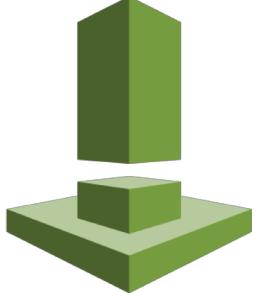


# AWS WAF – Web Application Firewall

- Protects your web applications from common web exploits
- Define customizable web security rules:
  - control which traffic to allow or block to your web applications
  - Rules can include: IP addresses, HTTP headers, HTTP body, or URI strings
  - Protects from common attack - SQL injection and Cross-Site Scripting (XSS)
  - Protect against bots, bad user agents, etc...
  - Size constraints
  - Geo match
- Deploy on CloudFront, Application Load Balancer or API Gateway
- Leverage existing marketplace of rules

# Penetration Testing on your AWS Cloud

- Permission is required for all penetration tests
- Request permissions only using AWS root credentials
- You must perform penetration testing yourself – no 3<sup>rd</sup> party
- <https://aws.amazon.com/forms/penetration-testing-request>
- For EC2, ELB, RDS, Aurora, CloudFront, API Gateway, Lambda, Lightsail
- Cannot test against nano / micro / small kind of EC2 instances
- Takes 2 business days to be approved



# AWS Inspector

- Only for EC2 instances
- Analyze against known vulnerabilities
- Analyze against unintended network accessibility
- AWS Inspector Agent must be installed on OS in EC2 instances
- Define template (rules package, duration, attributes, SNS topics)
- No own custom rules possible – only use AWS managed rules
- After the assessment, you get a report with a list of vulnerabilities

# What does AWS Inspector evaluate?

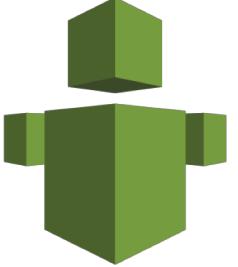
- Remember: only for EC2 instances
- For Network assessments:
  - Network Reachability
- For Host assessments:
  - Common Vulnerabilities and Exposures
  - Center for Internet Security (CIS) Benchmarks
  - Security Best Practices
  - Runtime Behavior Analysis

# Logging in AWS for security and compliance

- To help compliance requirements, AWS provides many service-specific security and audit logs
- Service Logs include:
  - CloudTrail trails - trace all API calls
  - Config Rules - for config & compliance over time
  - CloudWatch Logs - for full data retention
  - VPC Flow Logs - IP traffic within your VPC
  - ELB Access Logs - metadata of requests made to your load balancers
  - CloudFront Logs - web distribution access logs
  - WAF Logs - full logging of all requests analyzed by the service
- Logs can be analyzed using AWS Athena if they're stored in S3
- You should encrypt logs in S3, control access using IAM & Bucket Policies, MFA
- Move Logs to Glacier for cost savings
- Read whitepaper if interested at:  
[https://d0.awsstatic.com/whitepapers/compliance/AWS\\_Security\\_at\\_Scale\\_Logging\\_in\\_AWS\\_Whitepaper.pdf](https://d0.awsstatic.com/whitepapers/compliance/AWS_Security_at_Scale_Logging_in_AWS_Whitepaper.pdf)

# GuardDuty

- Intelligent Threat discovery to Protect AWS Account
- Uses Machine Learning algorithms, anomaly detection, 3<sup>rd</sup> party data
- One click to enable (30 days trial), no need to install software
- Input data includes:
  - CloudTrail Logs: unusual API calls, unauthorized deployments
  - VPC Flow Logs: unusual internal traffic, unusual IP address
  - DNS Logs: compromised EC2 instances sending encoded data within DNS queries
- Notifies you in case of findings
- Integration with AWS Lambda



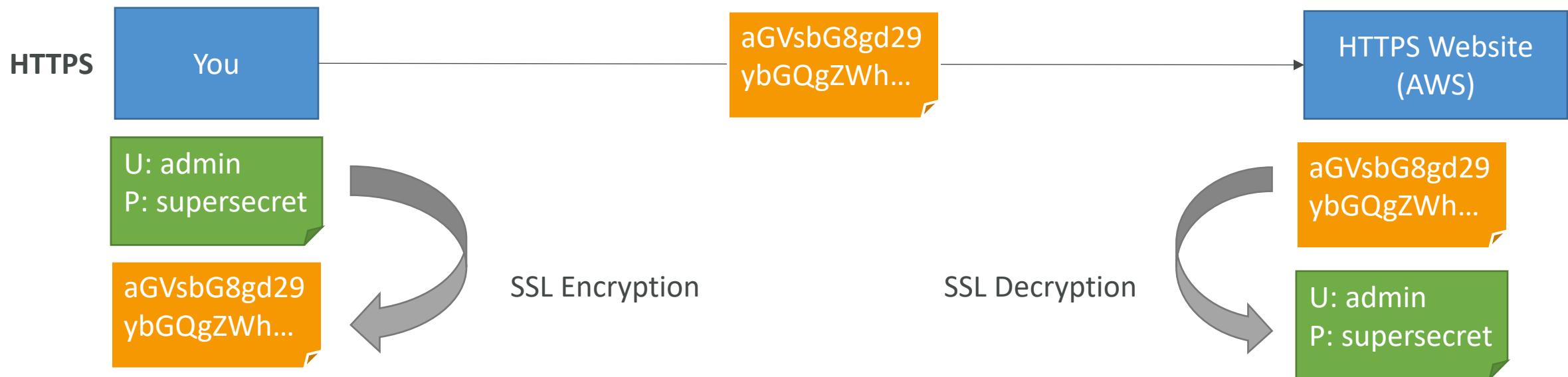
# Trusted Advisor

- No need to install anything – high level AWS account assessment
- Analyze your AWS accounts and provides recommendation:
  - Cost Optimization
  - Performance
  - Security
  - Fault Tolerance
  - Service Limits
- Core Checks and recommendations – all customers
- Can enable weekly email notification from the console
- Full Trusted Advisor – Available for Business & Enterprise support plans
  - Ability to set CloudWatch alarms when reaching limits

# Why encryption?

## Encryption in flight (SSL)

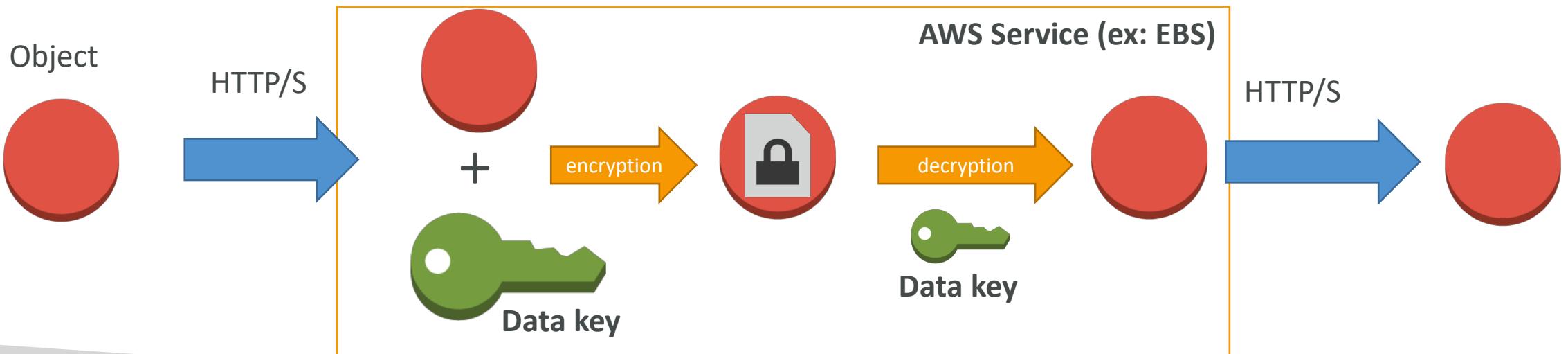
- Data is encrypted before sending and decrypted after receiving
- SSL certificates help with encryption (HTTPS)
- Encryption in flight ensures no MITM (man in the middle attack) can happen



# Why encryption?

## Server side encryption at rest

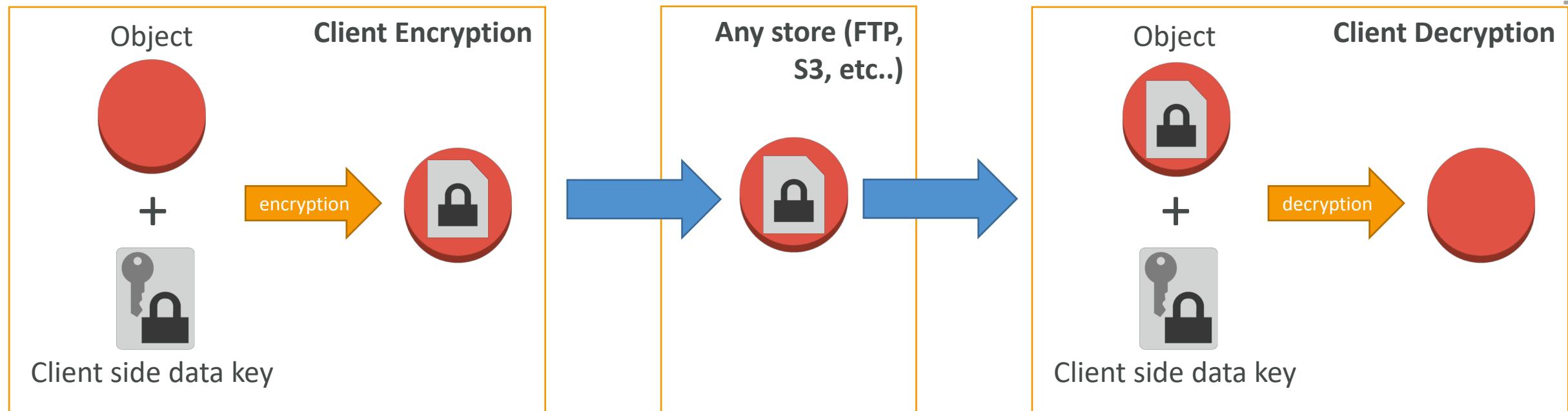
- Data is encrypted after being received by the server
- Data is decrypted before being sent
- It is stored in an encrypted form thanks to a key (usually a data key)
- The encryption / decryption keys must be managed somewhere and the server must have access to it



# Why encryption?

## Client side encryption

- Data is encrypted by the client and never decrypted by the server
- Data will be decrypted by a receiving client
- The server should not be able to decrypt the data
- Could leverage Envelope Encryption





# AWS KMS (Key Management Service)

- Anytime you hear “encryption” for an AWS service, it’s most likely KMS
- Easy way to control access to your data, AWS manages keys for us
- Fully integrated with IAM for authorization
- Seamlessly integrated into:
  - Amazon EBS: encrypt volumes
  - Amazon S3: Server side encryption of objects
  - Amazon Redshift: encryption of data
  - Amazon RDS: encryption of data
  - Amazon SSM: Parameter store
  - Etc...
- But you can also use the CLI / SDK

# AWS KMS 101

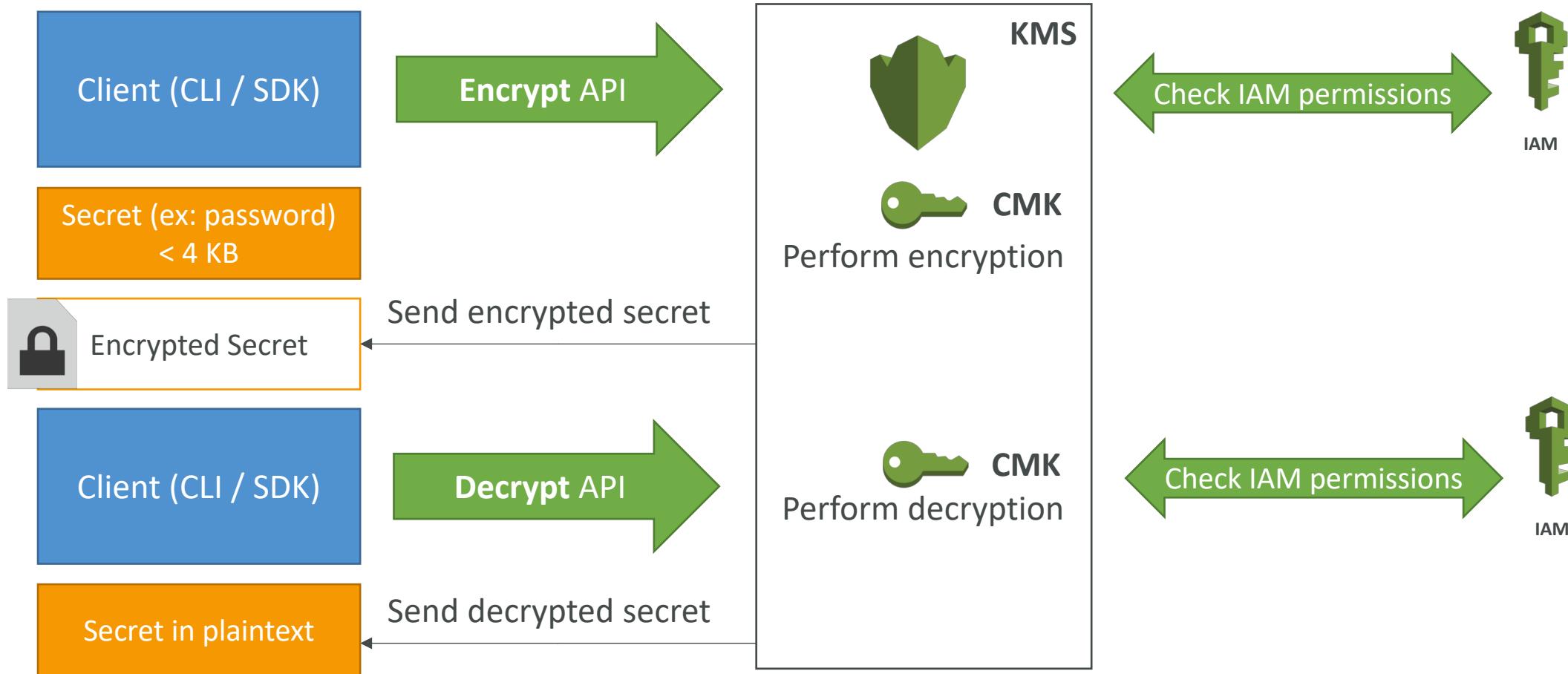
- Anytime you need to share sensitive information... use KMS
  - Database passwords
  - Credentials to external service
  - Private Key of SSL certificates
- The value in KMS is that the CMK used to encrypt data can never be retrieved by the user; and the CMK can be rotated for extra security
- **Never ever store your secrets in plaintext, especially in your code!**
- Encrypted secrets can be stored in the code / environment variables
- **KMS can only help in encrypting up to 4KB of data per call**
- If data > 4 KB, use envelope encryption
- To give access to KMS to someone:
  - Make sure the Key Policy allows the user
  - Make sure the IAM Policy allows the API calls

# AWS KMS (Key Management Service)

- Able to fully manage the keys & policies:
  - Create
  - Rotation policies
  - Disable
  - Enable
- Able to audit key usage (using CloudTrail)
- Three types of Customer Master Keys (CMK):
  - AWS Managed Service Default CMK: **free**
  - User Keys created in KMS: **\$1 / month**
  - User Keys imported (must be 256-bit symmetric key): **\$1 / month**
- + pay for API call to KMS (**\$0.03 / 10000 calls**)

# How does KMS work?

## API – Encrypt and Decrypt



# Encryption in AWS Services

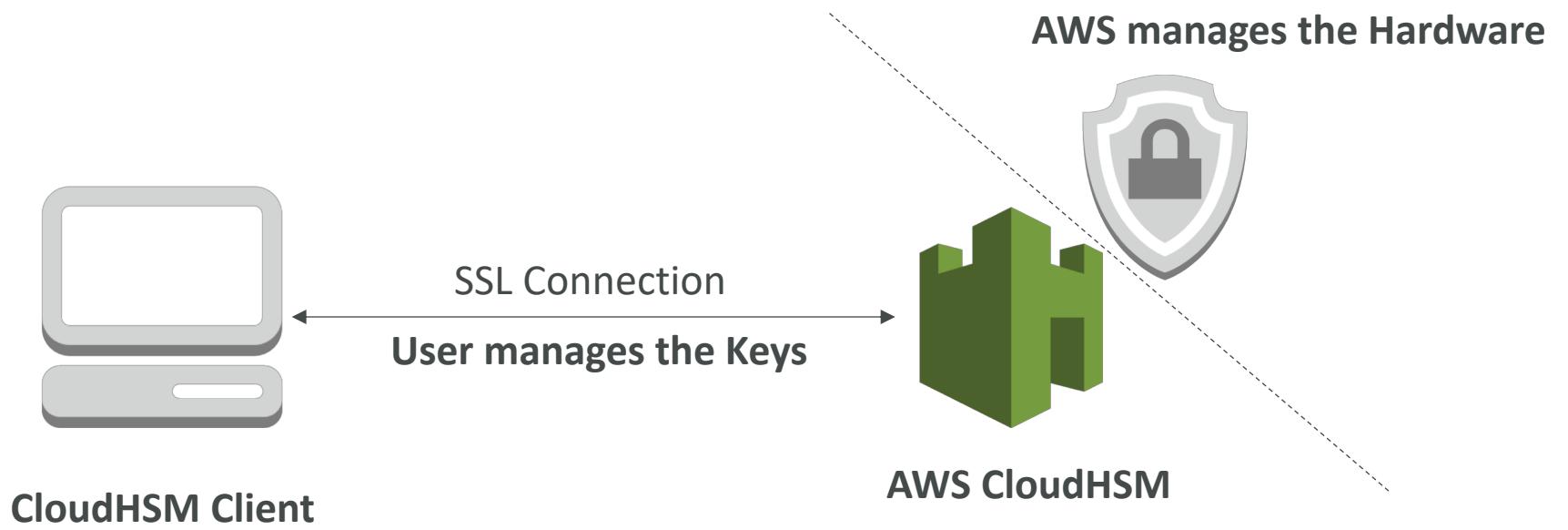
- Requires migration (through Snapshot / Backup):
  - EBS Volumes
  - RDS databases
  - ElastiCache
  - EFS network file system
- In-place encryption:
  - S3



# Cloud HSM

- KMS => AWS manages the software for encryption
- CloudHSM => AWS provisions encryption **hardware**
- Dedicated Hardware (HSM = Hardware Security Module)
- You manage your own encryption keys entirely (not AWS)
- The CloudHSM hardware device is tamper resistant
- FIPS 140-2 Level 3 compliance
- CloudHSM clusters are spread across multi AZ (HA)
- Supports both symmetric and **asymmetric** encryption (SSL/TLS keys)
- No free tier available
- Must use the CloudHSM Client Software

# CloudHSM Diagram



# CloudHSM vs KMS

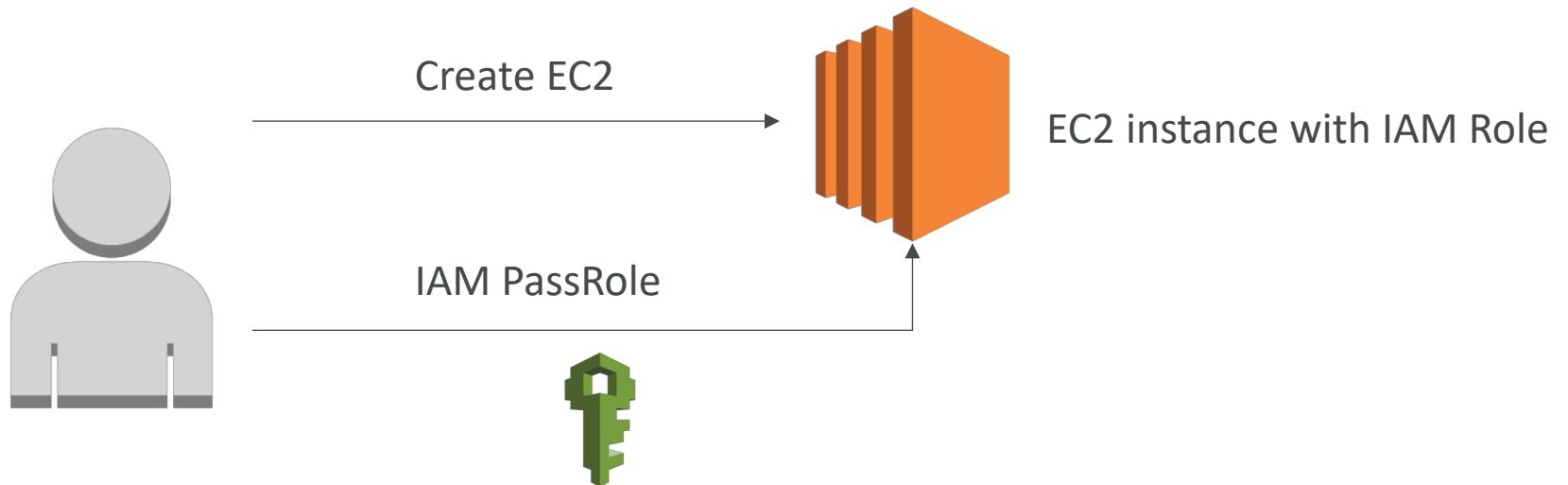
Feature	AWS KMS	AWS CloudHSM
Tenancy	Uses multi-tenant key storage	Single tenant key storage, dedicated to one customer
Keys	Keys owned and managed by AWS	Customer managed HSM
Encryption	Supports only symmetric key encryption	Supports both symmetric and asymmetric encryption
Cryptographic Acceleration	None	SSL/TLS Acceleration Oracle TDE Acceleration
Key Storage and Management	Accessible from multiple regions Centralized management from IAM	Deployed and managed from a customer VPC. Accessible and can be shared across VPCs using VPC peering
Free Tier Availability	Yes	No

# IAM + MFA (Multi Factor Authentication)

- MFA adds an added level of security while accessing your AWS account.
- AWS MFA accepts both virtual and hardware MFA devices.
- Virtual MFA device: Google Authenticator / Authy
- MFA for root user can be configured from IAM dashboard.
- MFA can also be configured from CLI.
- You can setup MFA for Individual Users
- **Credentials Report:**
  - A CSV report file on all the IAM users and credentials
  - This shows who all have enabled MFA

# IAM PassRole Option

- Example: When you create an EC2 instance, you assign a role to it
- In order to assign a role to an EC2 instance, you need **IAM:PassRole**



- IAM:PassRole can be used for any service where we assign roles (not only EC2)

# Policy Analysis: IAM PassRole

```
[{"Version": "2012-10-17", "Statement": [{"Sid": "Admin rights on EC2 service", "Action": "ec2:*", "Effect": "Allow", "Resource": "*"}, {"Sid": "Pass an IAM Role", "Action": ["iam:PassRole"], "Effect": "Allow", "Resource": "arn:aws:iam::<account-id>:role/Sample-EC2-Role"}]}
```

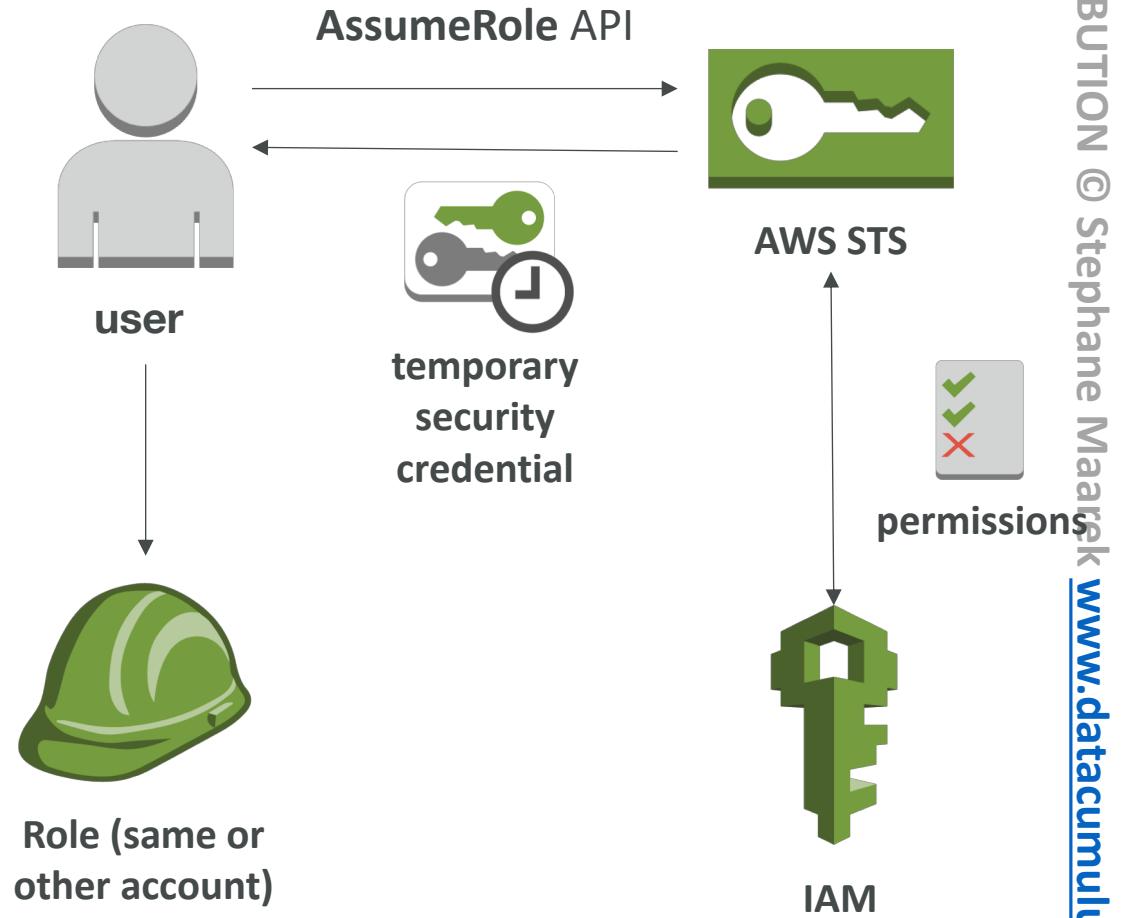
# AWS STS – Security Token Service



- Allows to grant limited and temporary access to AWS resources.
- Token is valid for up to one hour (must be refreshed)
- **Cross Account Access**
  - Allows users from one AWS account access resources in another
- **Federation (Active Directory)**
  - Provides a non-AWS user with temporary AWS access by linking users Active Directory credentials
  - Uses SAML (Security Assertion markup language)
  - Allows Single Sign On (SSO) which enables users to log in to AWS console without assigning IAM credentials
- **Federation with third party providers / Cognito**
  - Used mainly in web and mobile applications
  - Makes use of Facebook/Google/Amazon etc to federate them

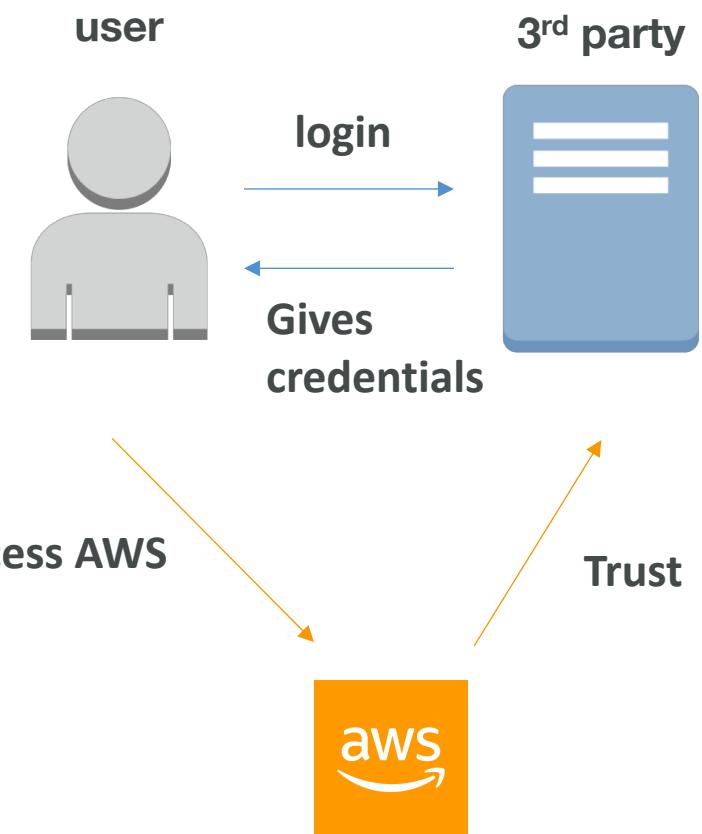
# Cross Account Access

- Define an IAM Role for another account to access
- Define which accounts can access this IAM Role
- Use AWS STS (Security Token Service) to retrieve credentials and impersonate the IAM Role you have access to (`AssumeRole API`)
- Temporary credentials can be valid between 15 minutes to 1 hour



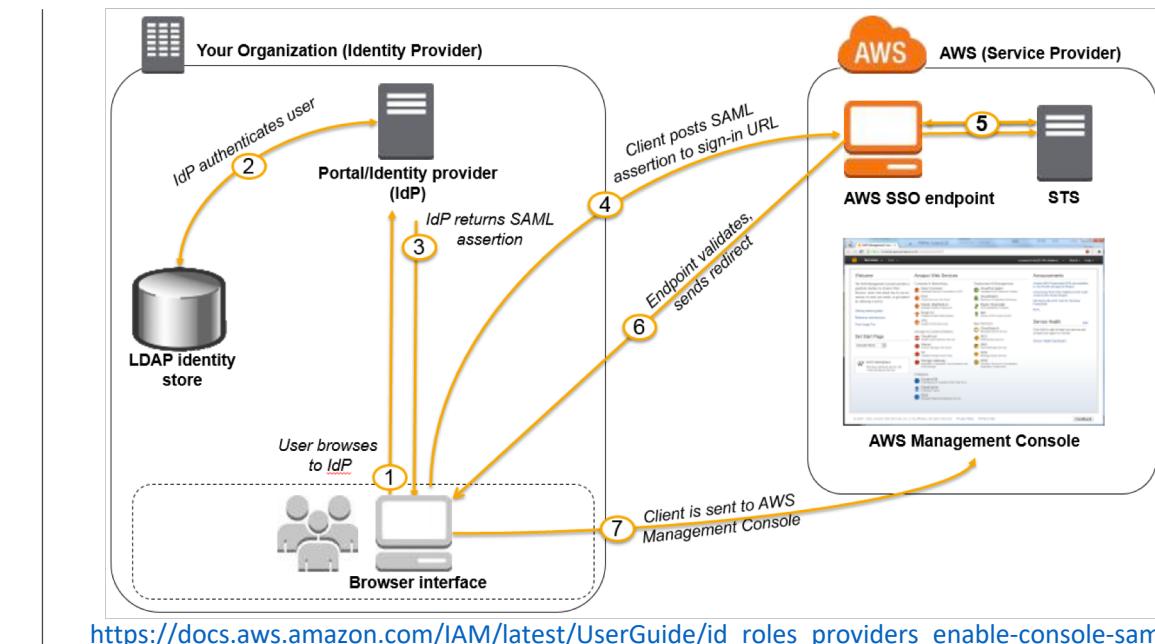
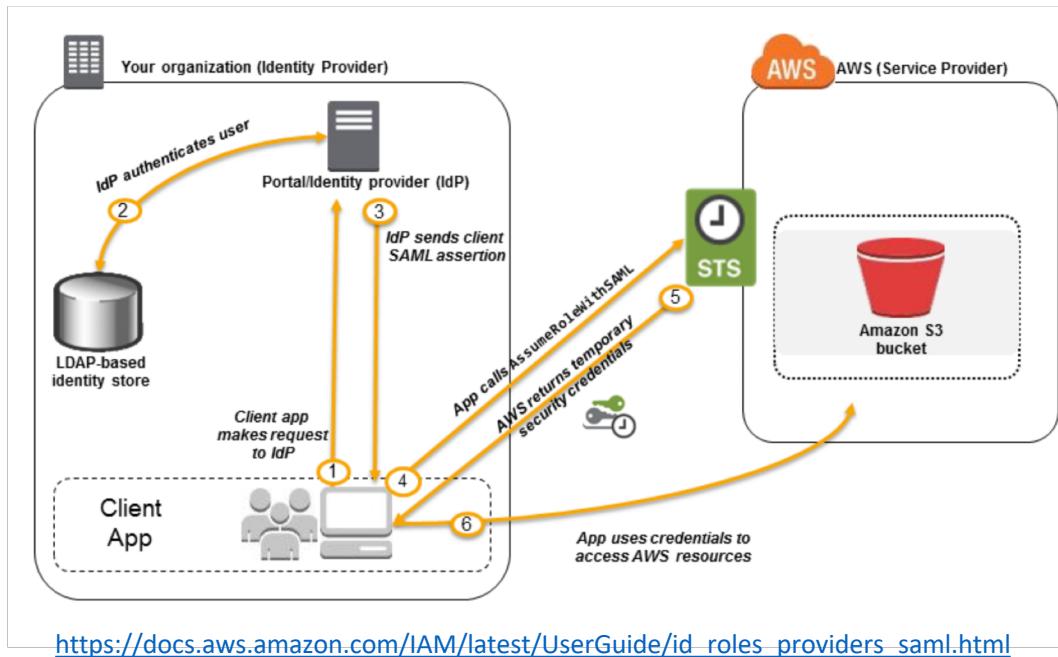
# What's Identity Federation?

- Federation lets users outside of AWS to assume temporary role for accessing AWS resources.
- These users assume identity provided access role.
- **Federation assumes a form of 3rd party authentication**
  - LDAP
  - Microsoft Active Directory ( $\sim$ = SAML)
  - Single Sign On
  - Open ID
  - Cognito
- **Using federation, you don't need to create IAM users (user management is outside of AWS)**



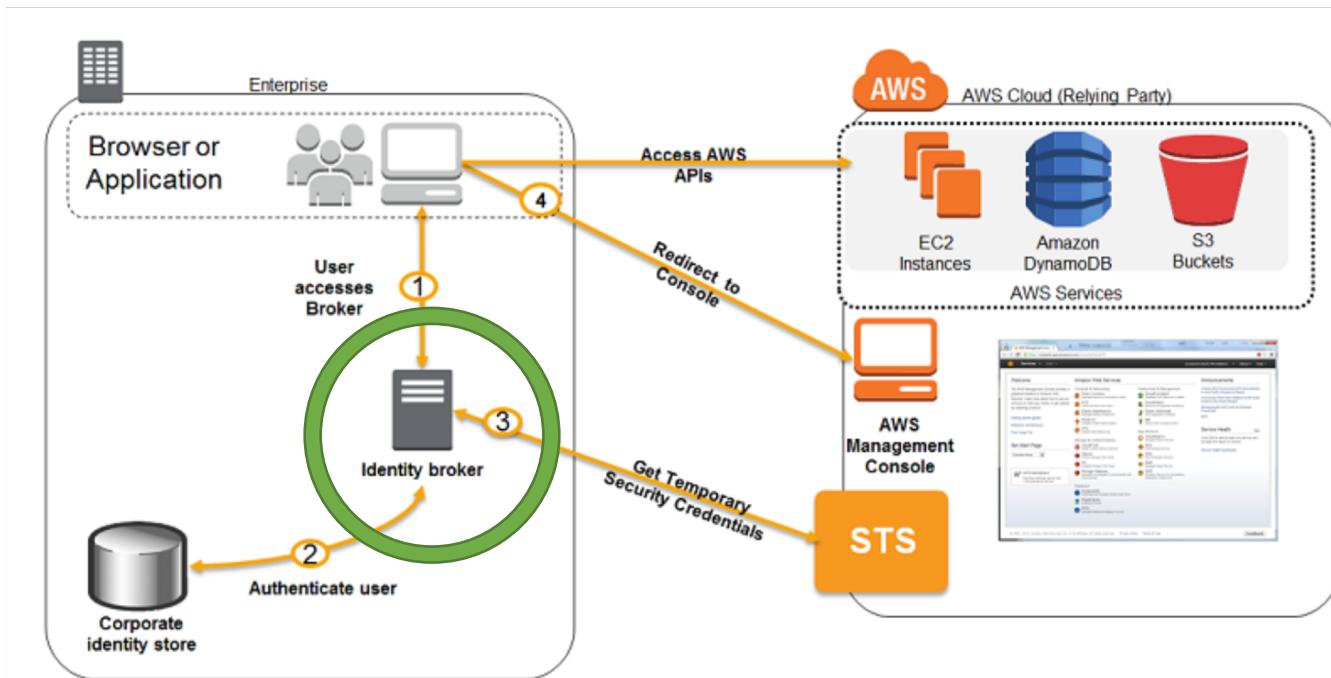
# SAML Federation For Enterprises

- To integrate Active Directory / ADFS with AWS (or any SAML 2.0)
- Provides access to AWS Console or CLI (through temporary creds)
- No need to create an IAM user for each of your employees



# Custom Identity Broker Application For Enterprises

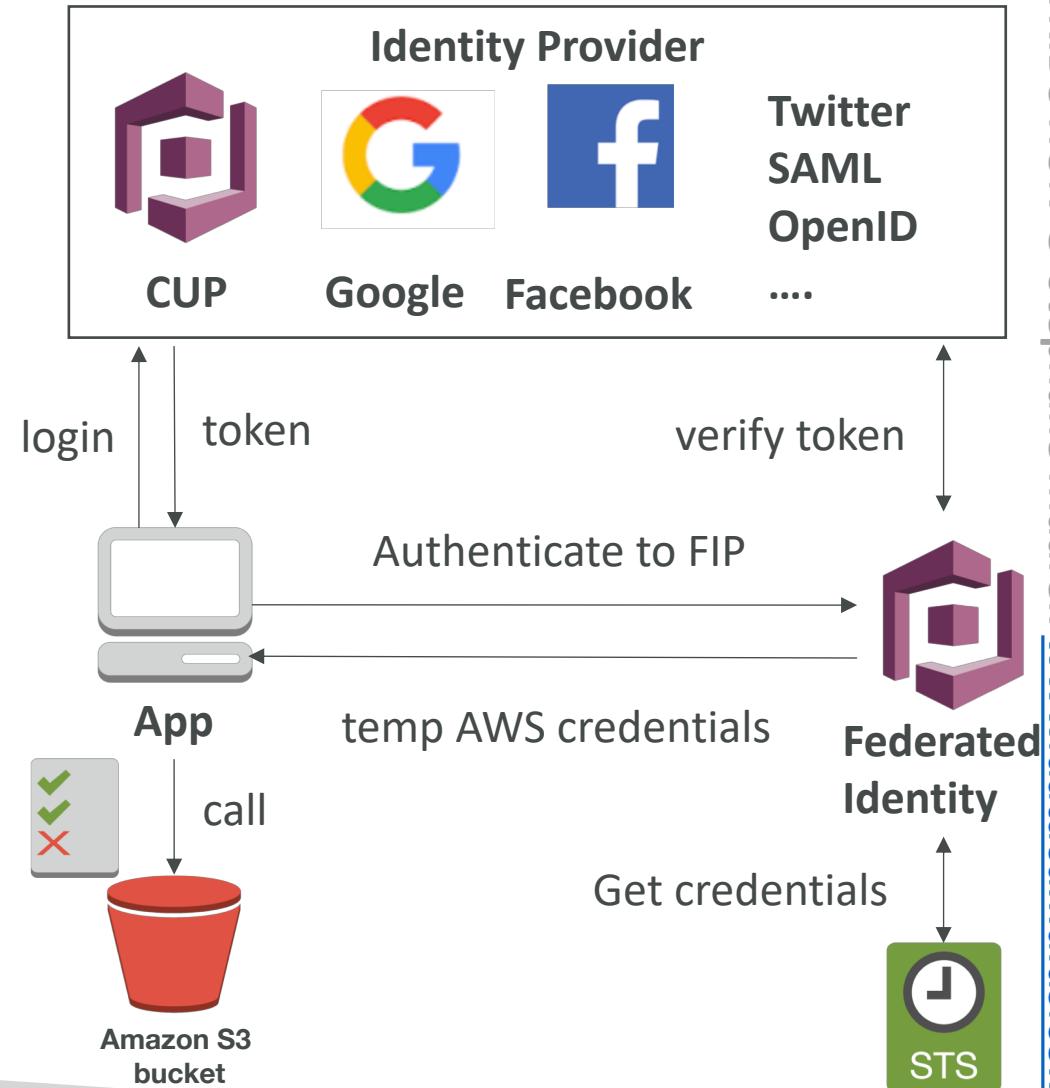
- Use only if identity provider is not compatible with SAML 2.0
- The identity broker must determine the appropriate IAM policy



[https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles-common-scenarios\\_federated-users.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles-common-scenarios_federated-users.html)

# AWS Cognito - Federated Identity Pools For Public Applications

- Goal:
  - Provide direct access to AWS Resources from the Client Side
- How:
  - Log in to federated identity provider – or remain anonymous
  - Get temporary AWS credentials back from the Federated Identity Pool
  - These credentials come with a pre-defined IAM policy stating their permissions
- Example:
  - provide (temporary) access to write to S3 bucket using Facebook Login
- Note:
  - Web Identity Federation is an alternative to using Cognito but AWS recommends against it



# JSON Policy Lab

- IAM:
  - [https://docs.aws.amazon.com/IAM/latest/UserGuide/access\\_policies\\_examples.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_examples.html)
- S3:
  - <https://docs.aws.amazon.com/AmazonS3/latest/dev/example-bucket-policies.html>
  - <https://docs.aws.amazon.com/AmazonS3/latest/dev/example-policies-s3.html>

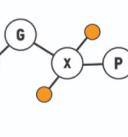
# AWS Compliance Frameworks

## Global

 <b>CSA</b> Cloud Security Alliance Controls	 <b>ISO 9001</b> International Organization for Standardization	 <b>ISO 27001</b> International Organization for Standardization	 <b>ISO 27017</b> International Organization for Standardization	 <b>ISO 27018</b> International Organization for Standardization
 <b>PCI DSS Level 1</b> Payment Card Standards	 <b>SOC 1</b> Audit Controls Report	 <b>SOC 2</b> Security, Availability, & Confidentiality Report	 <b>SOC 3</b> General Controls Report	

# AWS Compliance Frameworks

## United States

 <b>CJIS</b> Criminal Justice Information Services	 <b>DoD SRG</b> DoD Data Processing	 <b>FedRAMP</b> Government Data Standards	 <b>FERPA</b> Educational Privacy Act	 <b>FFIEC</b> Financial Institutions Regulation
 <b>FIPS</b> Government Security Standards	 <b>FISMA</b> Federal Information Security Management	 <b>GxP</b> Quality Guidelines and Regulations	 <b>HIPAA</b> Protected Health Information	 <b>ITAR</b> International Arms Regulations
 <b>MPAA</b> Protected Media Content	 <b>NIST</b> National Institute of Standards and Technology	 <b>SEC Rule 17a-4(f)</b> Financial Data Standards	 <b>VPAT / Section 508</b> Accessibility Standards	

# AWS Compliance Frameworks APAC + Europe

## Asia Pacific

**FISC [Japan]**Financial Industry  
Information Systems**IRAP [Australia]**Australian Security  
Standards**K-ISMS [Korea]**Korean Information  
Security**MTCS Tier 3****[Singapore]**  
Multi-Tier Cloud  
Security Standard**My Number Act**  
**[Japan]**Personal Information  
Protection

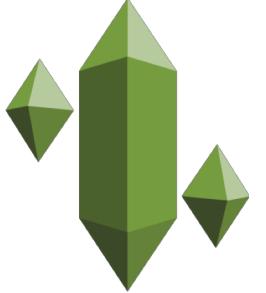
## Europe

**C5 [Germany]**Operational Security  
Attestation**Cyber Essentials**  
**Plus [UK]**Cyber Threat  
Protection**ENS High**  
**[Spain]**Spanish Government  
Standards**G-Cloud [UK]**UK Government  
Standards**IT-Grundschutz**  
**[Germany]**Baseline Protection  
Methodology

# AWS Certifications

- AWS has certification for compliance with ISO/IEC [27001:2013](https://aws.amazon.com/compliance/iso-certified/), [27017:2015](https://aws.amazon.com/compliance/iso-certified/), [27018:2014](https://aws.amazon.com/compliance/iso-certified/), and ISO/IEC [9001:2015](https://aws.amazon.com/compliance/iso-certified/).
  - List of services available here: <https://aws.amazon.com/compliance/iso-certified/>
- AWS is certified as a PCI DSS (Payment Card Industry Data Security Standard) 3.2 Level 1 Service Provider, the highest level of assessment available.
  - More details can be found at <https://aws.amazon.com/compliance/pci-dss-level-1-faqs/>
- AWS is HIPAA compliant (you can use it to store/process health data)
- AWS being compliant does not mean you are automatically compliant
- You need to get audited as well.

# AWS Artifact (not really a service)

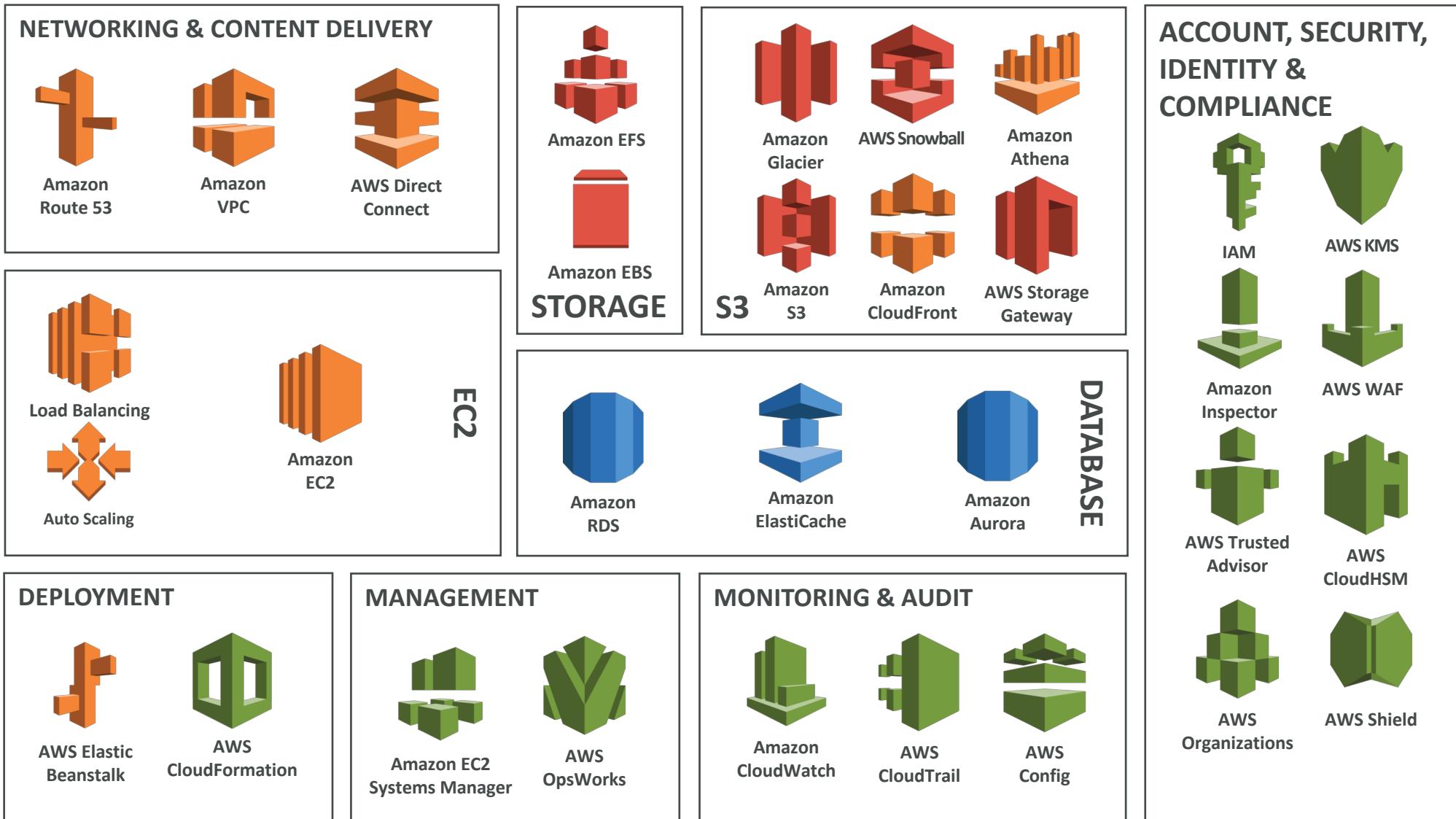


- Portal that provides customers with on-demand access to AWS compliance documentation and AWS agreements
- Artifact Reports - Allows you to download AWS security and compliance documents, like AWS ISO certifications, Payment Card Industry (PCI), and System and Organization Control (SOC) reports
- Artifact Agreements - Allows you to review, accept, and track the status of AWS agreements such as the Business Associate Addendum (BAA)
- Can be used to support internal audit or compliance

# Section Summary: Security & Compliance

- **AWS Shield:** Automatic DDoS Protection + 24/7 support for advanced
- **AWS WAF:** Firewall to filter incoming requests based on rules
- **AWS Inspector:** For EC2 only, install agent and find vulnerabilities
- **AWS GuardDuty:** Find malicious behavior with VPC, DNS & CloudTrail Logs
- **AWS Trusted Advisor:** Analyze AWS account and get recommendations
- **AWS KMS:** Encryption keys managed by AWS
- **AWS CloudHSM:** Hardware encryption, we manage keys, supports asymmetrical
- **AWS STS:** Generate Security Token
- **Identity Federation:** SAML 2.0 or Custom for Enterprise, Cognito for Apps
- **AWS Artifact:** Get access to compliance reports such as PCI, ISO, etc...
- **AWS Config:** Track config changes and compliance against rules
- **AWS CloudTrail:** Track API calls made by users within account

# Route 53



# Route 53 Section

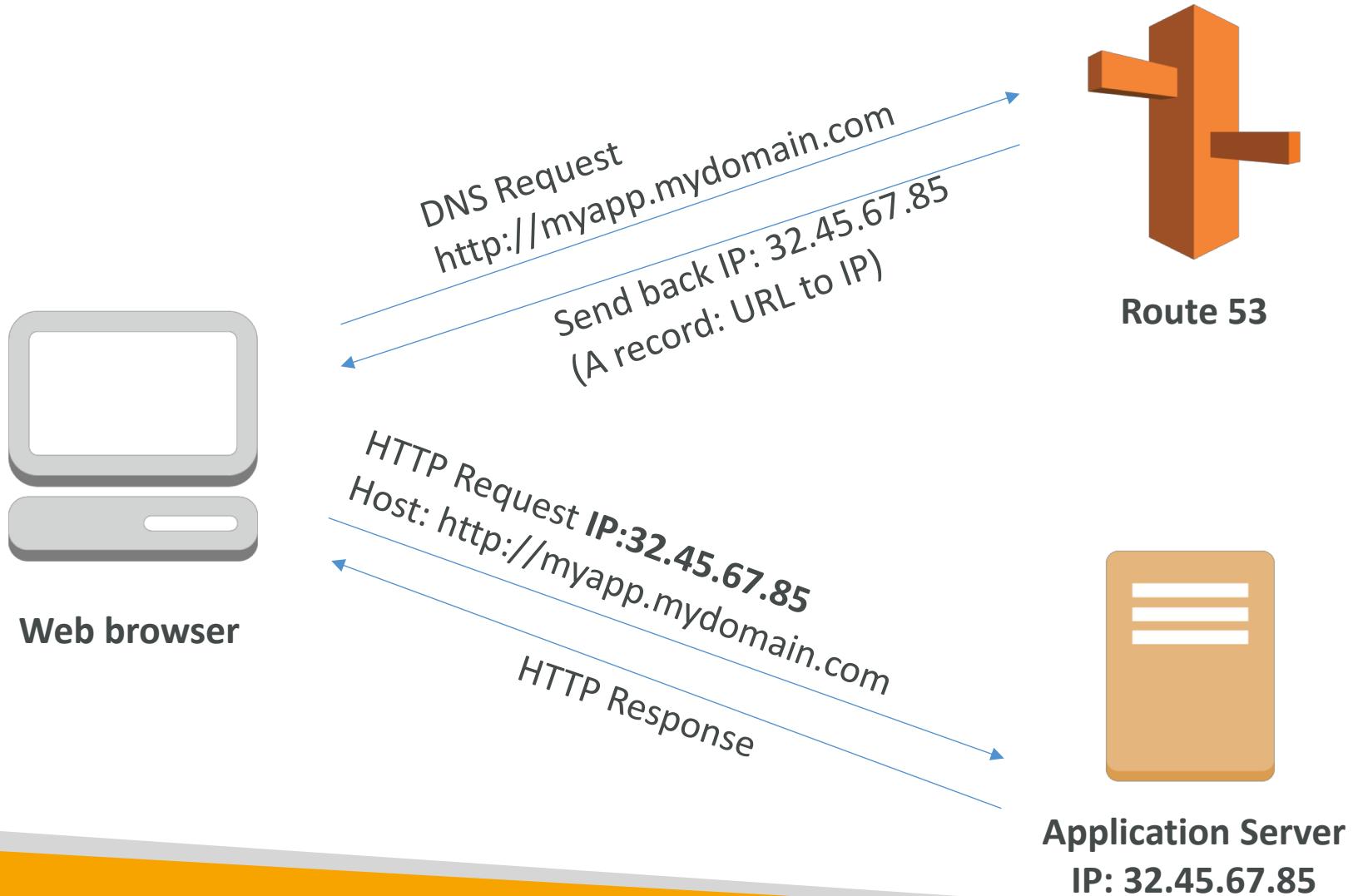
- TTL
- CNAME vs Alias
- Health Checks
- Routing Policies
  - Simple
  - Weighted
  - Latency
  - Failover
  - Geolocation
  - Multi Value
- 3<sup>rd</sup> party domains integration

# AWS Route 53 Overview



- Route53 is a Managed DNS (Domain Name System)
- DNS is a collection of rules and records which helps clients understand how to reach a server through URLs.
- In AWS, the most common records are:
  - A: URL to IPv4
  - AAAA: URL to IPv6
  - CNAME: URL to URL
  - Alias: URL to AWS resource.

# Route 53 – Diagram for A Record

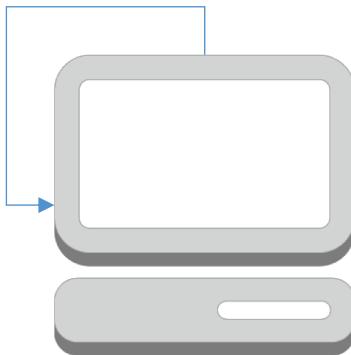


# AWS Route 53 Overview

- Route53 can use:
  - public domain names you own (or buy)  
[application1.mypublicdomain.com](http://application1.mypublicdomain.com)
  - private domain names that can be resolved by your instances in your VPCs.  
[application1.company.internal](http://application1.company.internal)
- Route53 has advanced features such as:
  - Load balancing (through DNS – also called client load balancing)
  - Health checks (although limited...)
  - Routing policy: simple, failover, geolocation, latency, weighted, multi value
- You pay \$0.50 per month per hosted zone

# DNS Records TTL (Time to Live)

DNS Cache  
For TTL duration



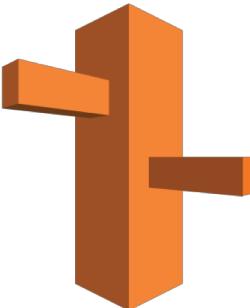
Web browser

DNS Request  
`http://myapp.mydomain.com`

Send back IP: 32.45.67.85  
(A record: URL to IP)  
+ TTL : 300 s

DNS Request  
`http://myapp.mydomain.com`

Send back IP: 195.23.45.22  
(A record: URL to IP)  
+ TTL : 300 s



Route 53

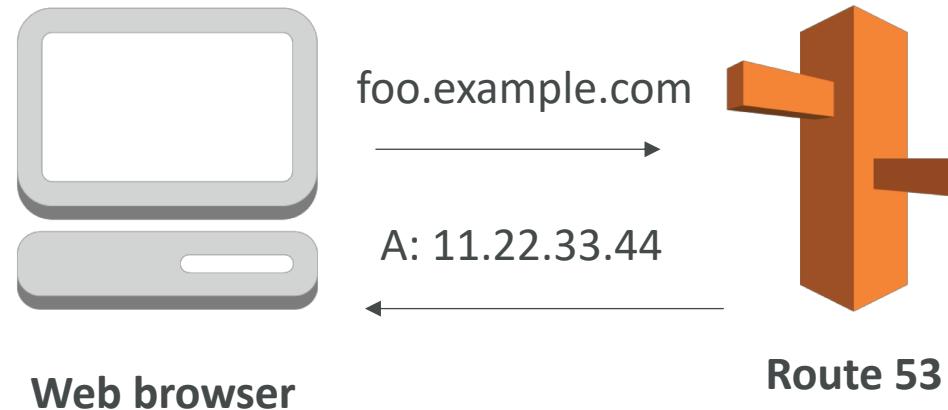
- High TTL: (e.g. 24hr)
  - Less traffic on DNS
  - Possibly outdated records
- Low TTL: (e.g 60 s)
  - More traffic on DNS
  - Records are outdated for less time
  - Easy to change records
- TTL is mandatory for each DNS record

# CNAME vs Alias

- AWS Resources (Load Balancer, CloudFront, etc...) expose an AWS URL: [lb-1234.us-east-2.elb.amazonaws.com](http://lb-1234.us-east-2.elb.amazonaws.com) and you want it to be [myapp.mydomain.com](http://myapp.mydomain.com)
- CNAME:
  - Points a URL to any other URL. (app.mydomain.com => blabla.anything.com)
  - ONLY FOR NON ROOT DOMAIN (aka. something.mydomain.com)
- Alias:
  - Points a URL to an AWS Resource (app.mydomain.com => blabla.amazonaws.com)
  - Works for ROOT DOMAIN and NON ROOT DOMAIN (aka mydomain.com)
  - Free of charge
  - Native health check

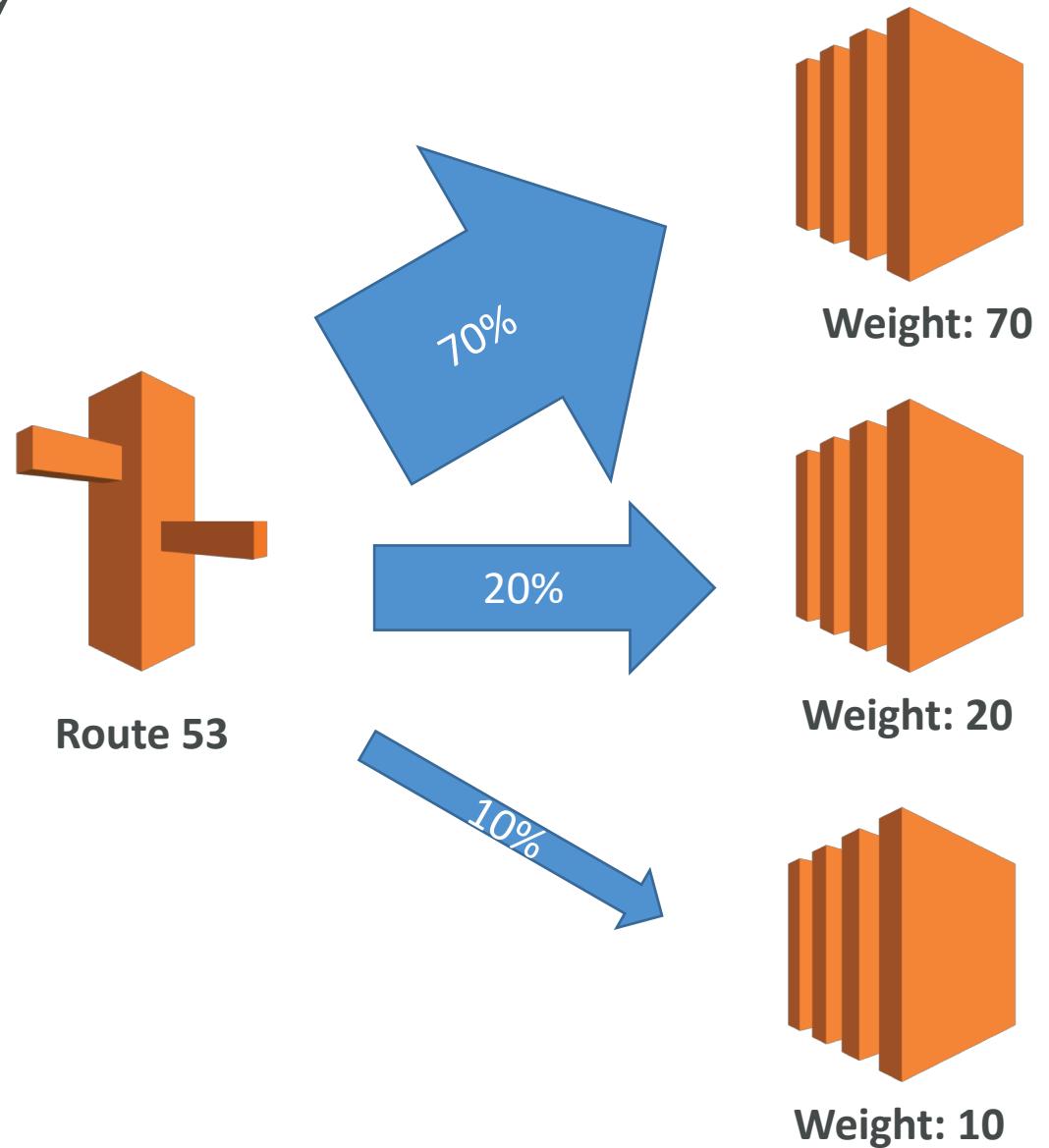
# Simple Routing Policy

- Maps a domain to one URL
- Use when you need to redirect to a single resource
- You can't attach health checks to simple routing policy
- If multiple values are returned, a random one is chosen by the client



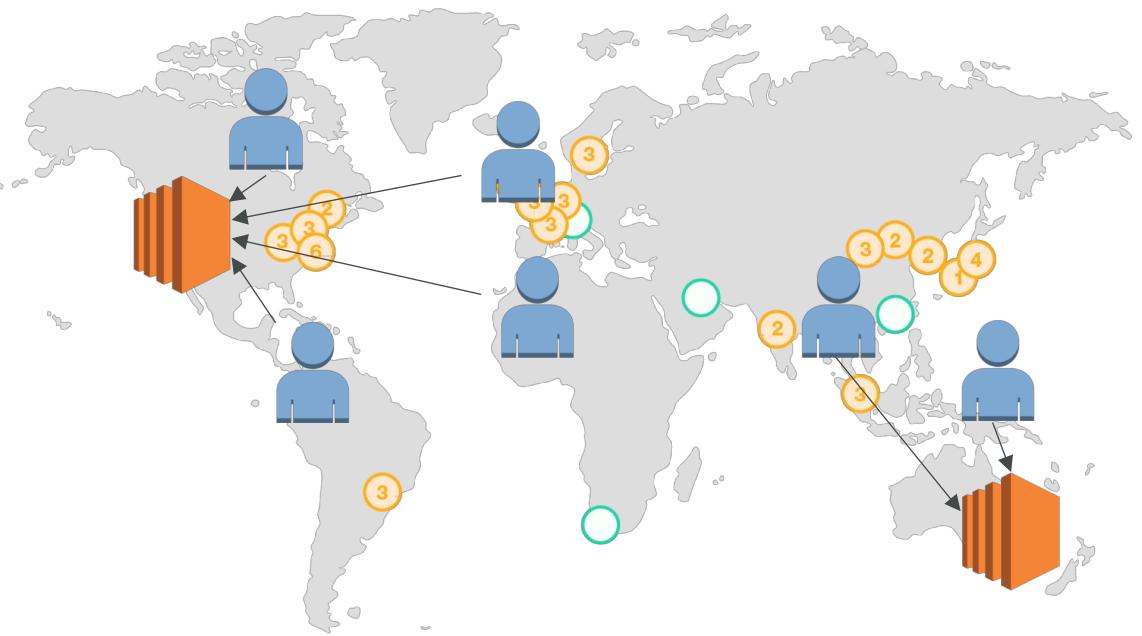
# Weighted Routing Policy

- Control the % of the requests that go to specific endpoint
- Helpful to test 1% of traffic on new app version for example
- Helpful to split traffic between two regions
- Can be associated with Health Checks



# Latency Routing Policy

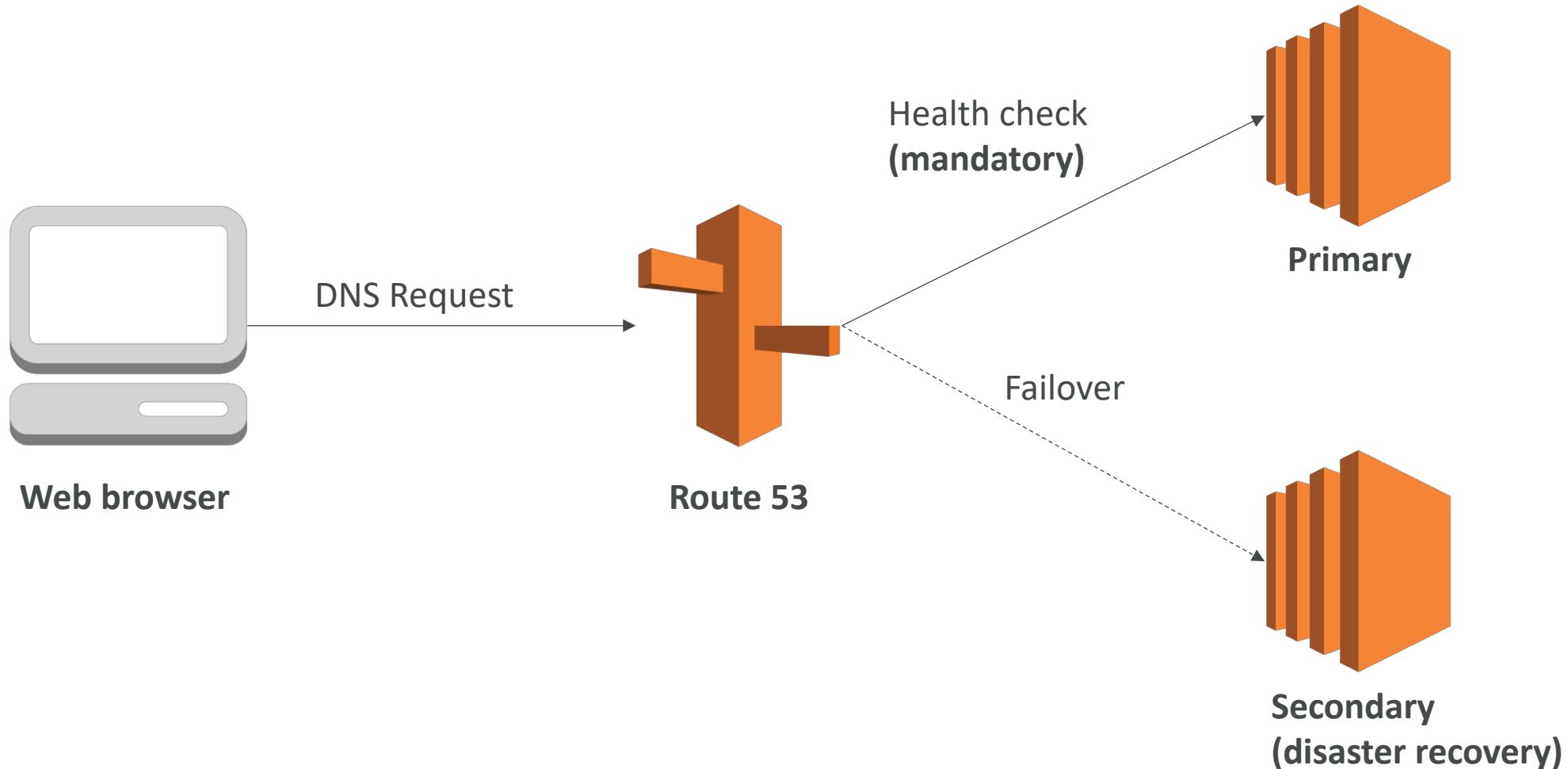
- Redirect to the server that has the least latency close to us
- Super helpful when latency of users is a priority
- Latency is evaluated in terms of user to designated AWS Region
- Germany may be directed to the US (if that's the lowest latency)



# Health Checks

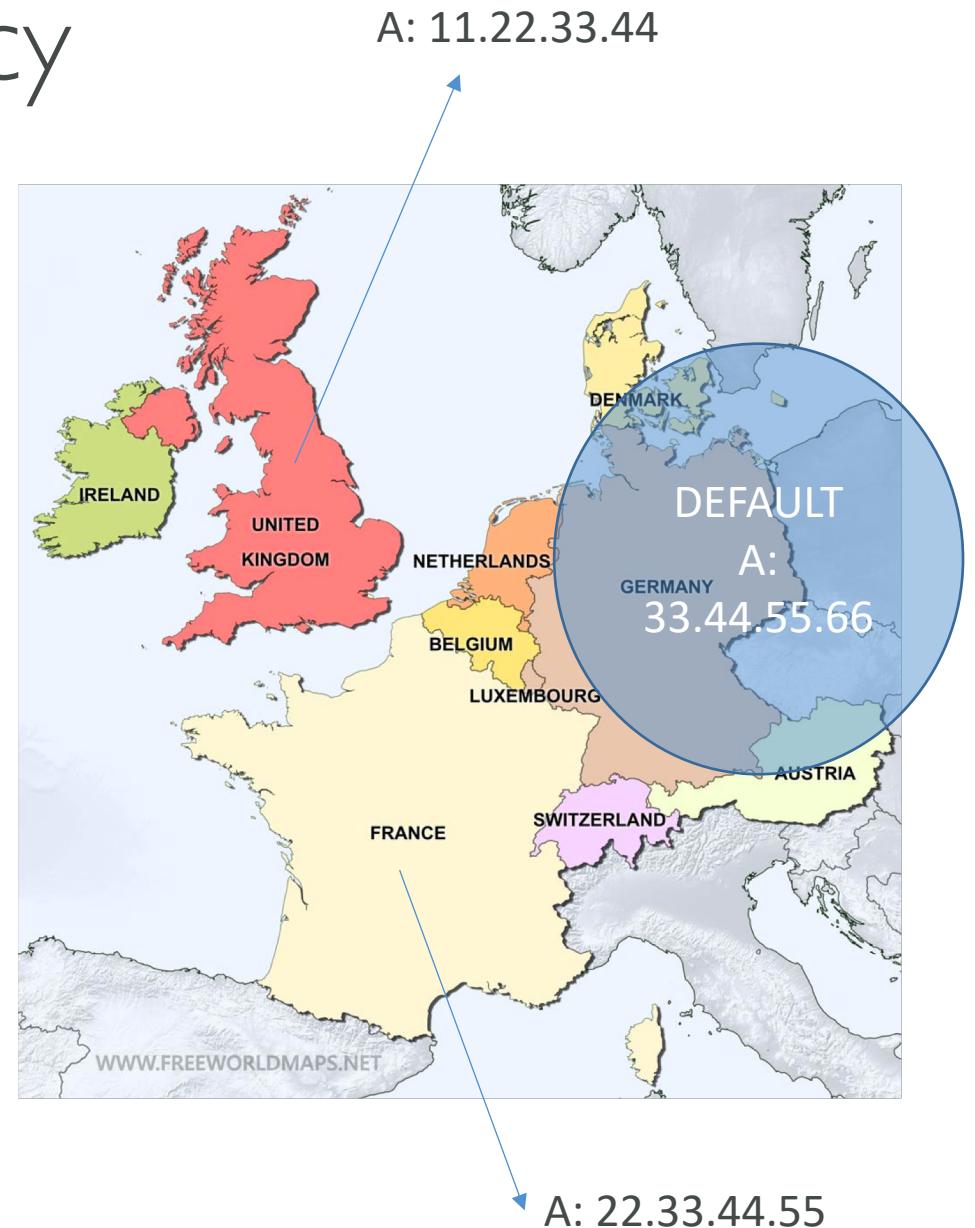
- Have X health checks failed => unhealthy (default 3)
- After X health checks passed => health (default 3)
- Default Health Check Interval: 30s (can set to 10s – higher cost)
- About 15 health checkers will check the endpoint health
- => one request every 2 seconds on average
- Can have HTTP,TCP and HTTPS health checks (no SSL verification)
- Possibility of integrating the health check with CloudWatch
- Health checks can be linked to Route53 DNS queries!

# Failover Routing Policy



# Geo Location Routing Policy

- Different from Latency based!
- This is routing based on user location
- Here we specify: traffic from the UK should go to this specific IP
- Should create a “default” policy (in case there’s no match on location)



# Multi Value Routing Policy

- Use when routing traffic to multiple resources
- Want to associate a Route 53 health checks with records
- Up to 8 healthy records are returned for each Multi Value query
- Multi Value is not a substitute for having an ELB

Name	Type	Value	TTL	Set ID	Health Check
www.example.com	A Record	192.0.2.2	60	Web1	A
www.example.com	A Record	198.51.100.2	60	Web2	B
www.example.com	A Record	203.0.113.2	60	Web3	C

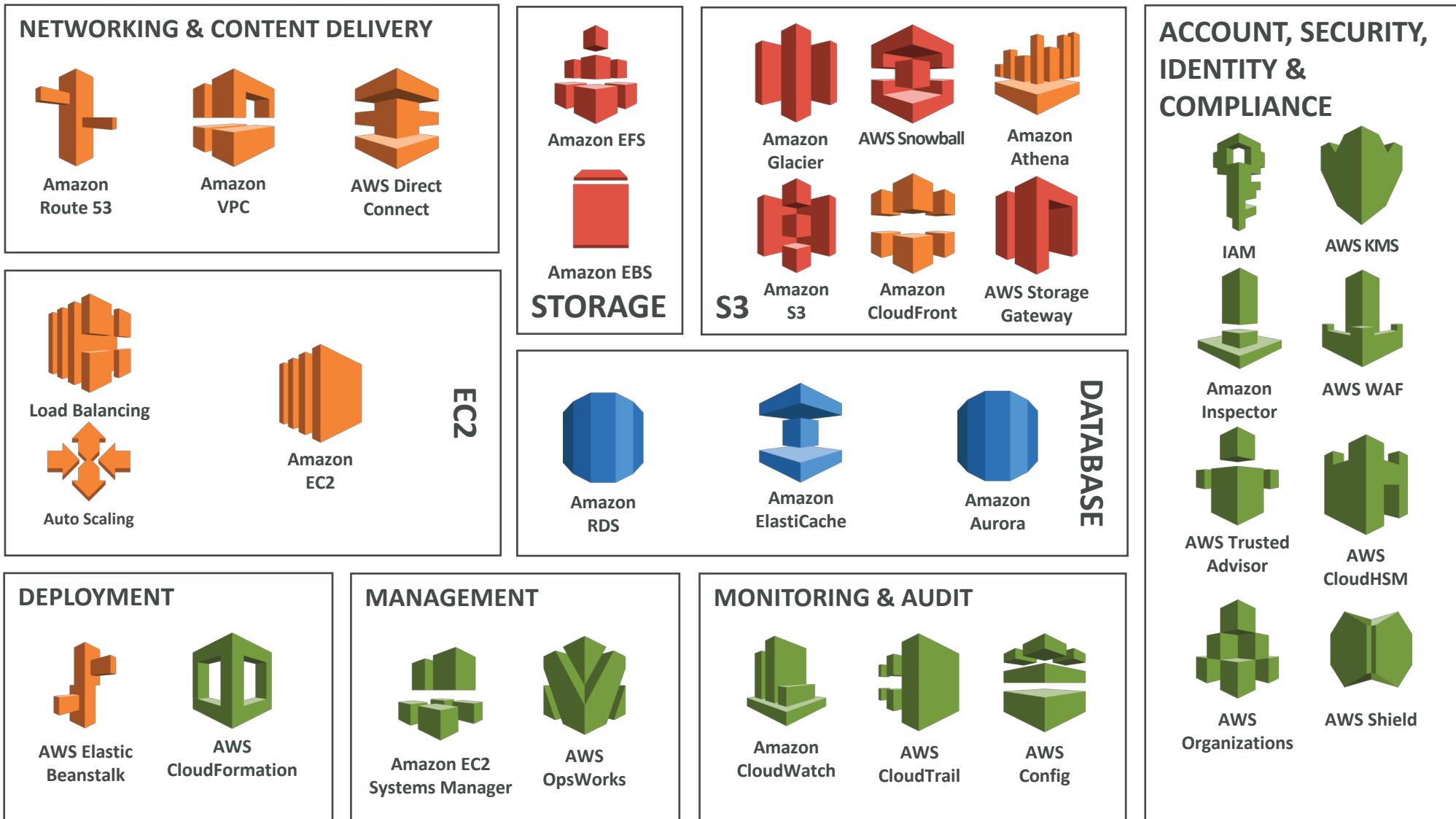
# Route53 as a Registrar

- A domain name **registrar** is an organization that manages the reservation of Internet domain names
- Famous names:
  - GoDaddy
  - Google Domains
  - Etc...
- And also... Route53 (e.g. AWS)!
- Domain Registrar != DNS

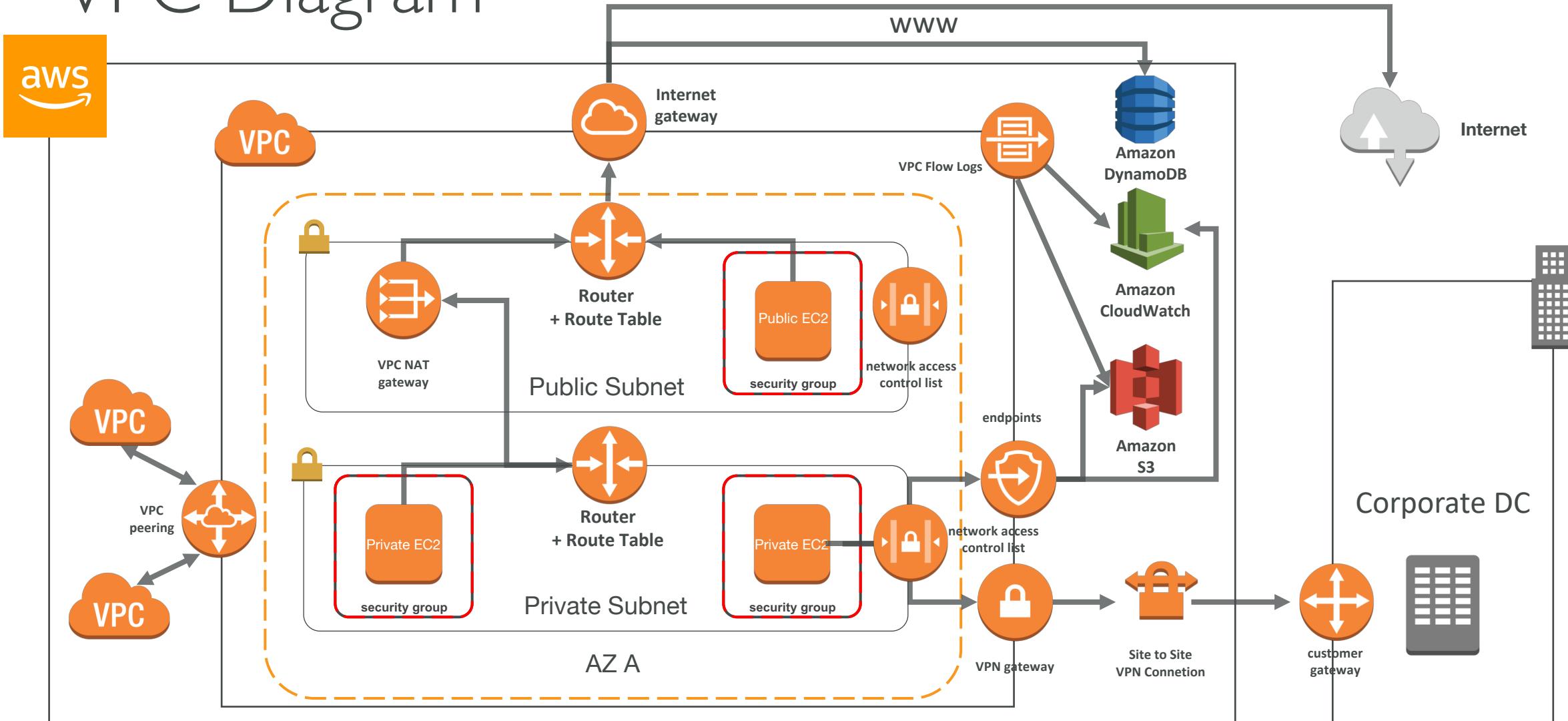
# 3<sup>rd</sup> Party Registrar with AWS Route 53

- If you buy your domain on 3<sup>rd</sup> party website, you can still use Route53.
  - 1) Create a Hosted Zone in Route 53
  - 2) Update NS Records on 3<sup>rd</sup> party website to use Route 53 name servers
- Domain Registrar != DNS
  - (But each domain registrar usually comes with some DNS features)

# Networking - VPC



# VPC Diagram



# Understanding CIDR - IPv4 (Classless Inter-Domain Routing)

- CIDR are used for Security Groups rules, or AWS networking in general

Source	
0.0.0.0/0	
122.149.196.85/32	

- They help to define an IP address range
  - We've seen WW.XX.YY.ZZ/32 == one IP
  - We've seen 0.0.0.0/0 == all IPs
  - But we can define for ex: 192.168.0.0/26: 192.168.0.0 – 192.168.0.63 (64 IP)

# Understanding CIDR

- A CIDR has two components:
  - The base IP (XX.XX.XX.XX)
  - The Subnet Mask (/26)
- The base IP represents an IP contained in the range
- The subnet mask defines how many bits can change in the IP
- The subnet mask can take two forms. Examples:
  - 255.255.255.0    ↪ less common
  - /24                 ↪ more common

# Understanding CIDRs Subnet Masks

- The subnet masks basically allows part of the underlying IP to get additional next values from the base IP
- /32 allows for 1 IP =  $2^0$
- /31 allows for 2 IP =  $2^1$
- /30 allows for 4 IP =  $2^2$
- /29 allows for 8 IP =  $2^3$
- /28 allows for 16 IP =  $2^4$
- /27 allows for 32 IP =  $2^5$
- /26 allows for 64 IP =  $2^6$
- /25 allows for 128 IP =  $2^7$
- /24 allows for 256 IP =  $2^8$
- /16 allows for 65,536 IP =  $2^{16}$
- /0 allows for all IPs =  $2^{32}$

- **Quick memo:**
- **/32 – no IP number can change**
- **/24 - last IP number can change**
- **/16 – last IP two numbers can change**
- **/8 – last IP three numbers can change**
- **/0 – all IP numbers can change**

# Understanding CIDRs

## Little exercise

- $192.168.0.0/24 = \dots ?$ 
  - $192.168.0.0 - 192.168.0.255$  (256 IP)
- $192.168.0.0/16 = \dots ?$ 
  - $192.168.0.0 - 192.168.255.255$  (65,536 IP)
- $134.56.78.123/32 = \dots ?$ 
  - Just 134.56.78.123
- $0.0.0.0/0$ 
  - All IP!
- When in doubt, use this website: <https://www.ipaddressguide.com/cidr>

# Private vs Public IP (IPv4)

## Allowed ranges

- The Internet Assigned Numbers Authority (IANA) established certain blocks of IPV4 addresses for the use of private (LAN) and public (Internet) addresses.
- Private IP can only allow certain values
  - 10.0.0 – 10.255.255.255 (10.0.0/8) <= in big networks
  - 172.16.0.0 – 172.31.255.255 (172.16.0.0/12) <= default AWS one
  - 192.168.0.0 – 192.168.255.255 (192.168.0.0/16) <= example: home networks
- All the rest of the IP on the internet are public IP

# Default VPC Walkthrough

- All new accounts have a default VPC
- New instances are launched into default VPC if no subnet is specified
- Default VPC have internet connectivity and all instances have public IP
- We also get a public and a private DNS name

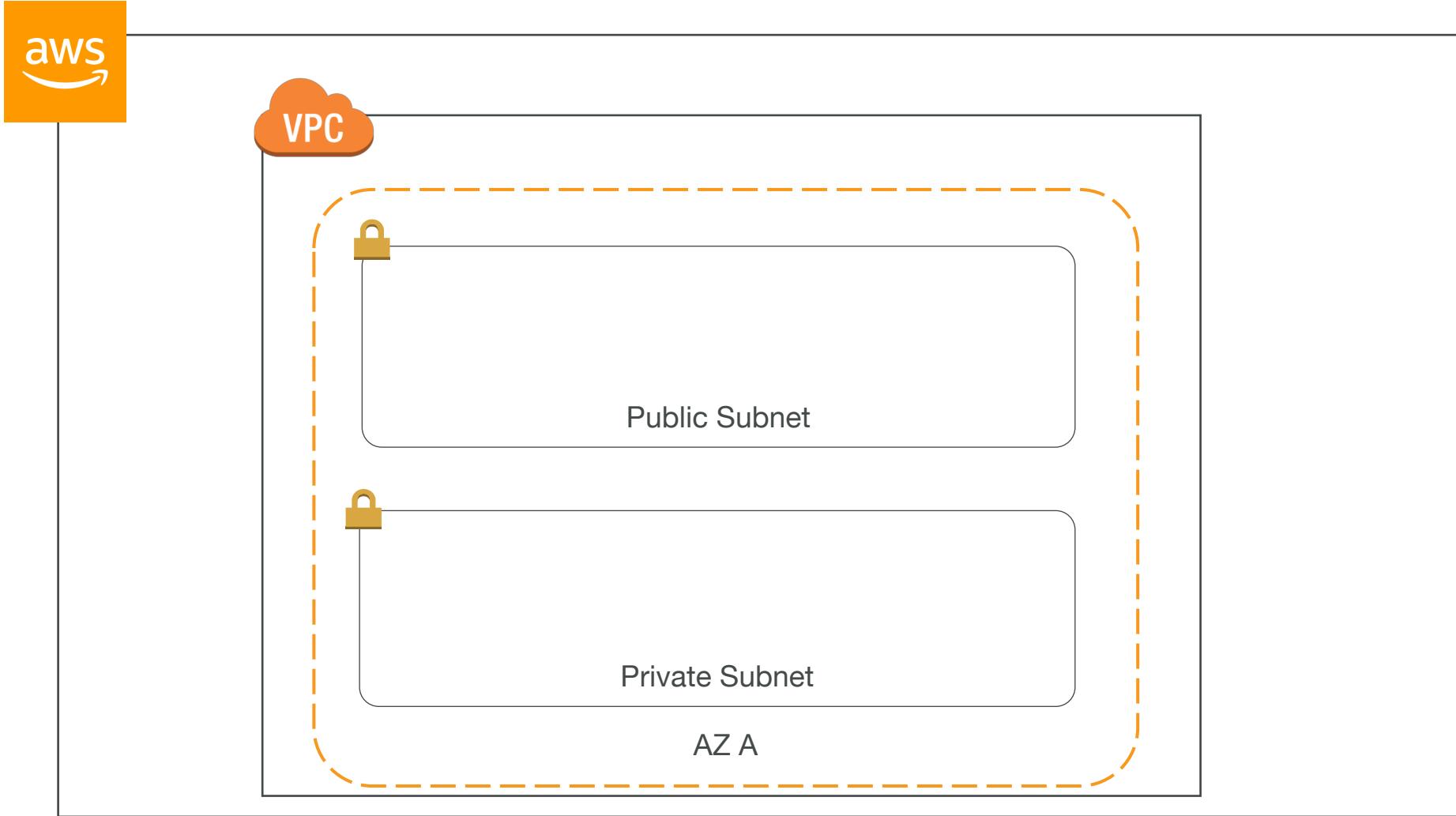
# VPC in AWS – IPv4

- VPC = Virtual Private Cloud
- You can have multiple VPCs in a region (max 5 per region – soft limit)
- Max CIDR per VPC is 5. For each CIDR:
  - Min size is /28 = 16 IP Addresses
  - Max size is /16 = 65536 IP Addresses
- Because VPC is private, only the Private IP ranges are allowed:
  - 10.0.0.0 – 10.255.255.255 (10.0.0.0/8)
  - 172.16.0.0 – 172.31.255.255 (172.16.0.0/12)
  - 192.168.0.0 – 192.168.255.255 (192.168.0.0/16)
- Your VPC CIDR should not overlap with your other networks (ex: corporate)

# State of Hands On



# Adding Subnets



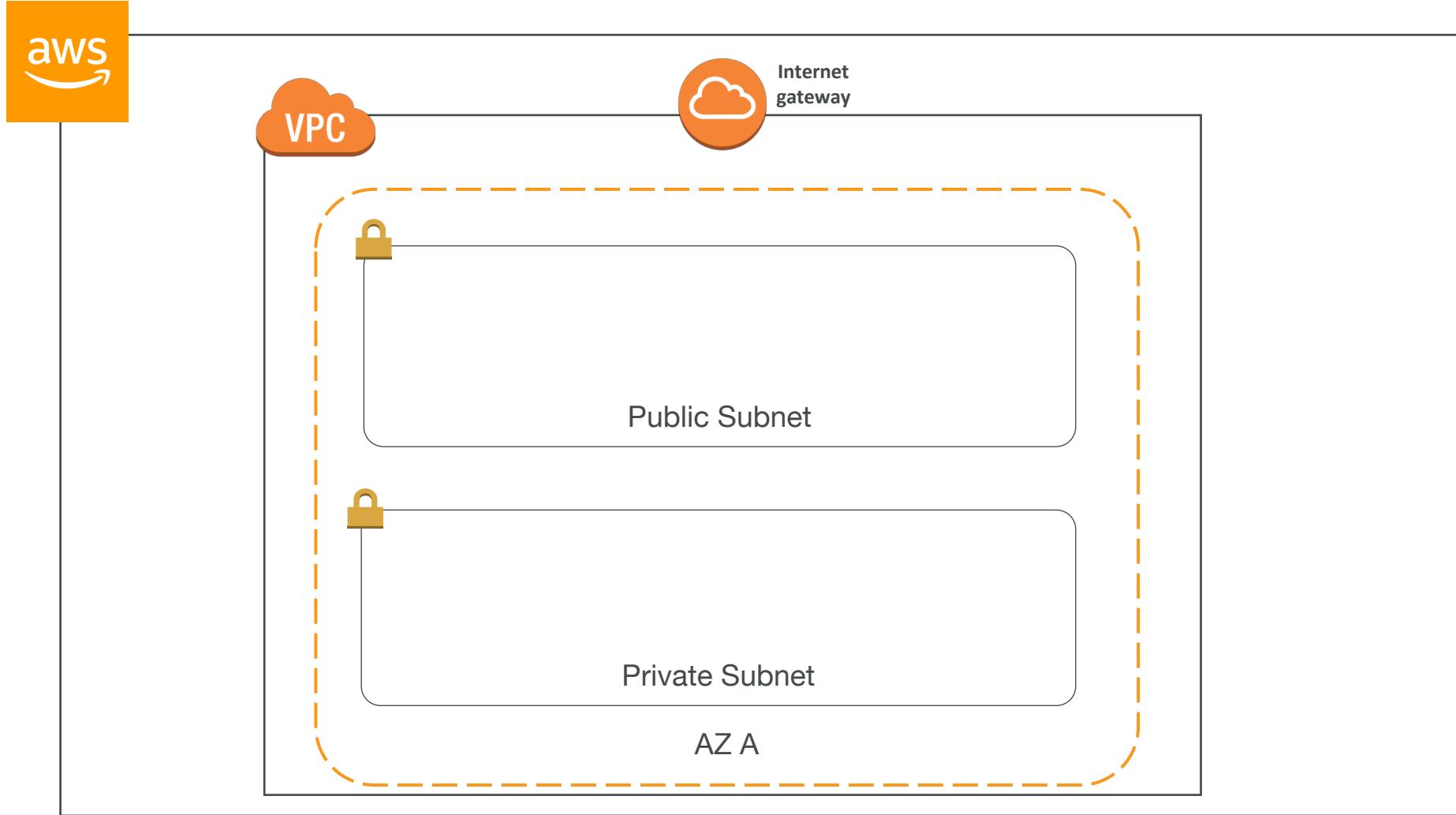
# Subnets - IPv4

- AWS reserves 5 IPs address (first 4 and last 1 IP address) in each Subnet
- These 5 IPs are not available for use and cannot be assigned to an instance
- Ex, if CIDR block 10.0.0.0/24, reserved IP are:
  - 10.0.0.0: Network address
  - 10.0.0.1: Reserved by AWS for the VPC router
  - 10.0.0.2: Reserved by AWS for mapping to Amazon-provided DNS
  - 10.0.0.3: Reserved by AWS for future use
  - 10.0.0.255: Network broadcast address. AWS does not support broadcast in a VPC, therefore the address is reserved
- Exam Tip:
  - If you need 29 IP addresses for EC2 instances, you can't choose a Subnet of size /28 (32 IP)

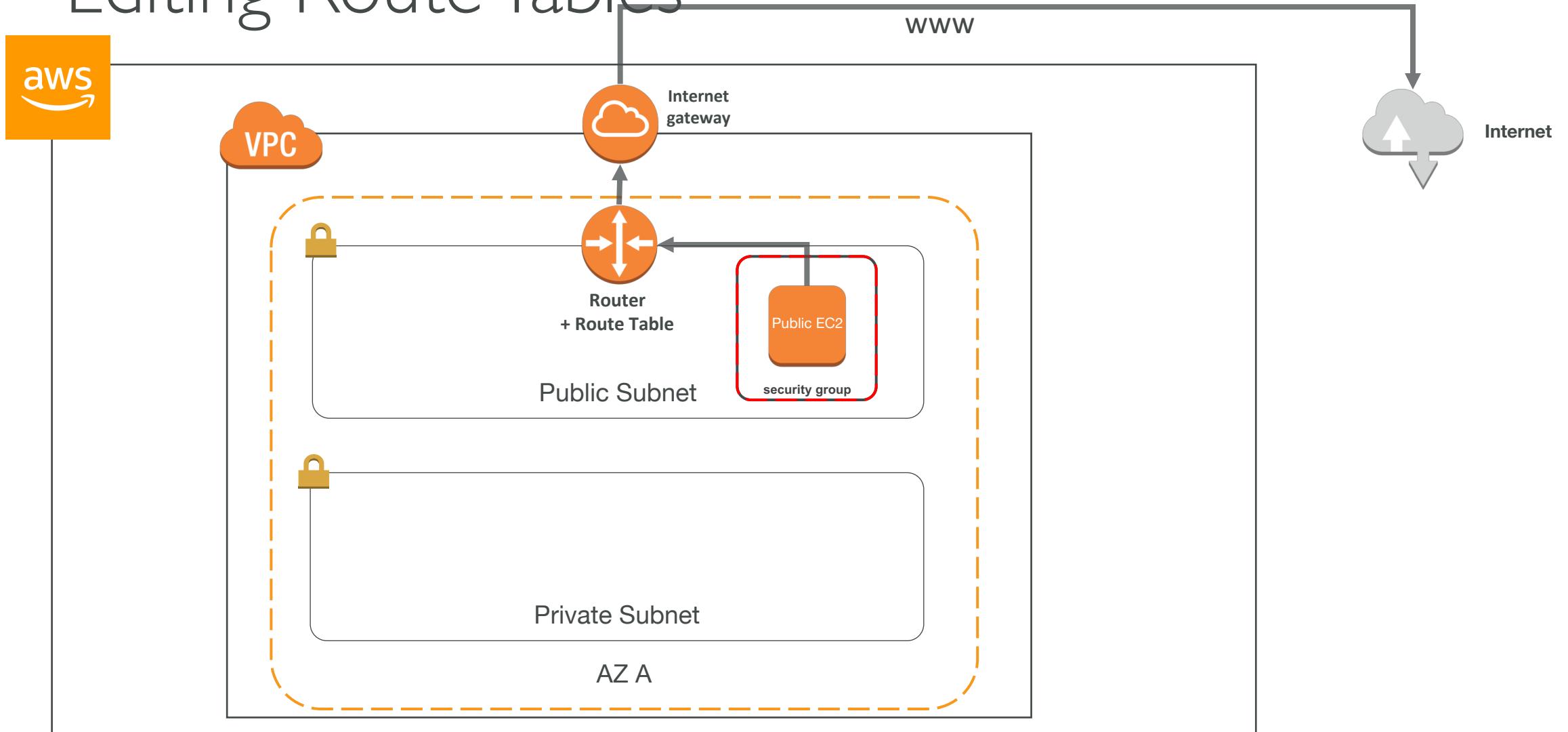
# Internet Gateways

- Internet gateways helps our VPC instances connect with the internet
- It scales horizontally and is HA and redundant
- Must be created separately from VPC
- One VPC can only be attached to one IGW and vice versa
- Internet Gateway is also a NAT for the instances that have a public IPv4
- Internet Gateways on their own do not allow internet access...
- Route tables must also be edited!

# Adding IGW



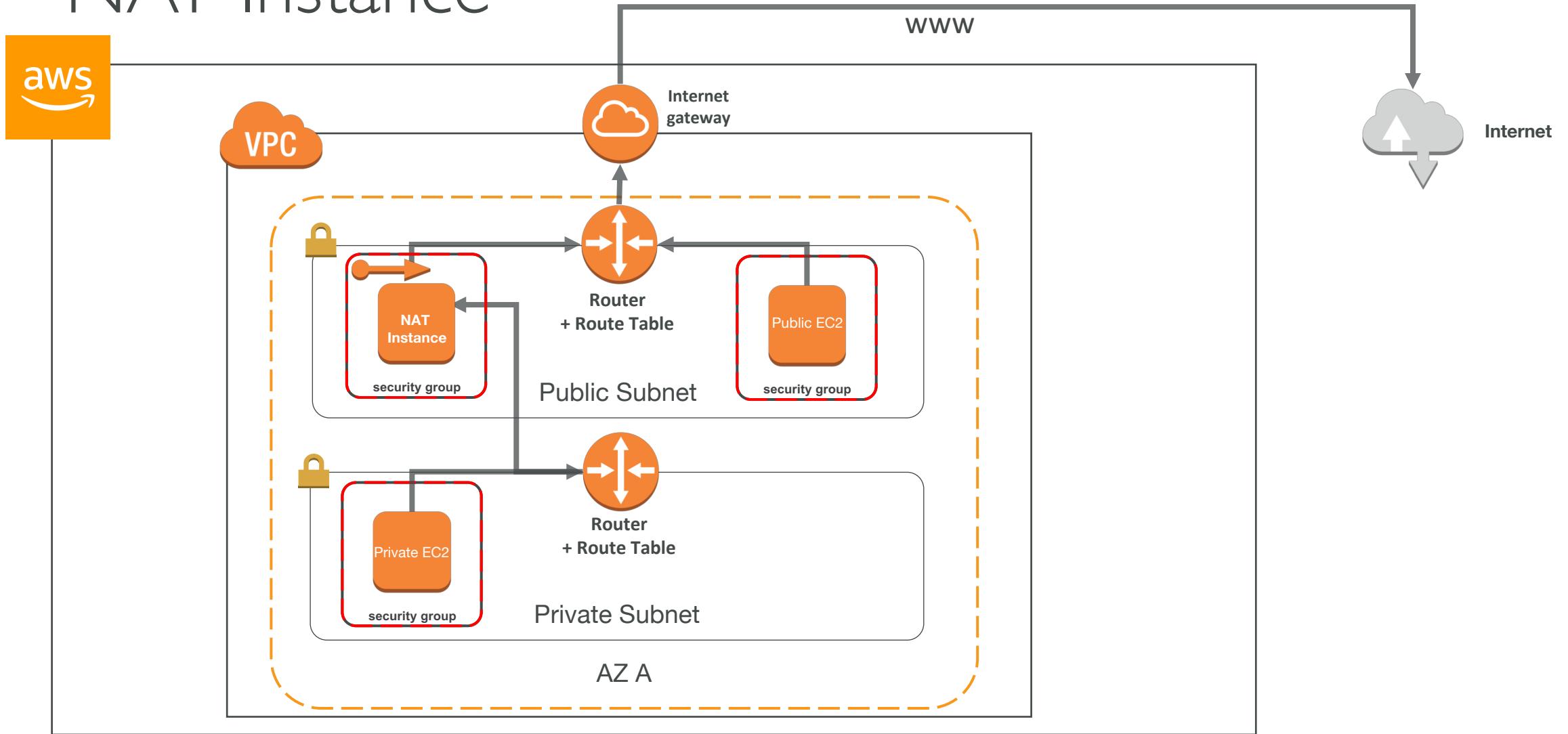
# Editing Route Tables



# NAT Instances – Network Address Translation (outdated but still at the exam)

- Allows instances in the private subnets to connect to the internet
- Must be launched in a public subnet
- Must disable EC2 flag: Source / Destination Check
- Must have Elastic IP attached to it
- Route table must be configured to route traffic from private subnets to NAT Instance

# NAT Instance



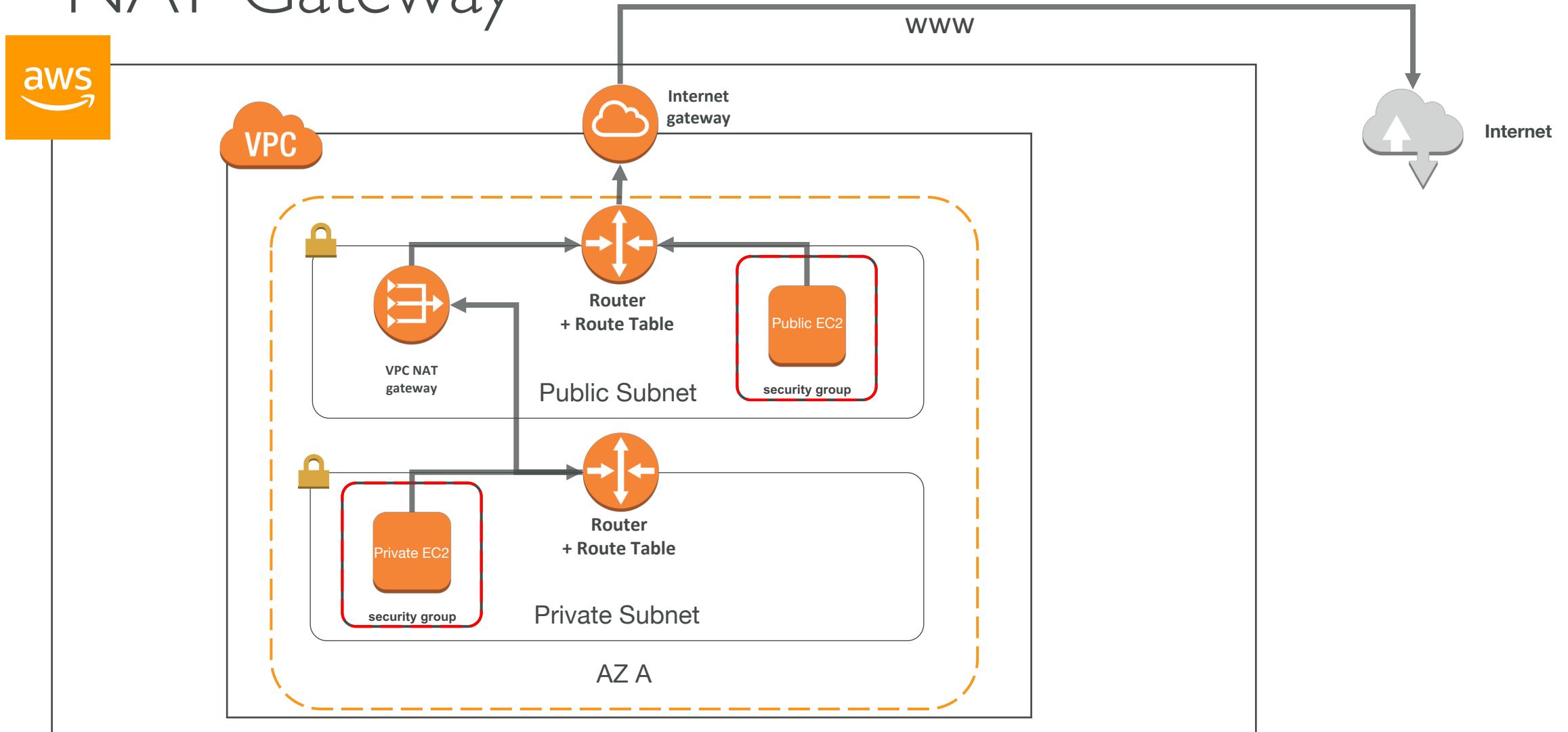
# NAT Instances – Comments

- Amazon Linux AMI pre-configured are available
- Not highly available / resilient setup out of the box
- => Would need to create ASG in multi AZ + resilient user-data script
- Internet traffic bandwidth depends on EC2 instance performance
- Must manage security groups & rules:
  - Inbound:
    - Allow HTTP / HTTPS Traffic coming from Private Subnets
    - Allow SSH from your home network (access is provided through Internet Gateway)
  - Outbound:
    - Allow HTTP / HTTPS traffic to the internet

# NAT Gateway

- AWS managed NAT, higher bandwidth, better availability, no admin
- Pay by the hour for usage and bandwidth
- NAT is created in a specific AZ, uses an EIP
- Cannot be used by an instance in that subnet (only from other subnets)
- Requires an IGW (Private Subnet => NAT => IGW)
- 5 Gbps of bandwidth with automatic scaling up to 45 Gbps
- No security group to manage / required

# NAT Gateway



# NAT Instance vs Gateway

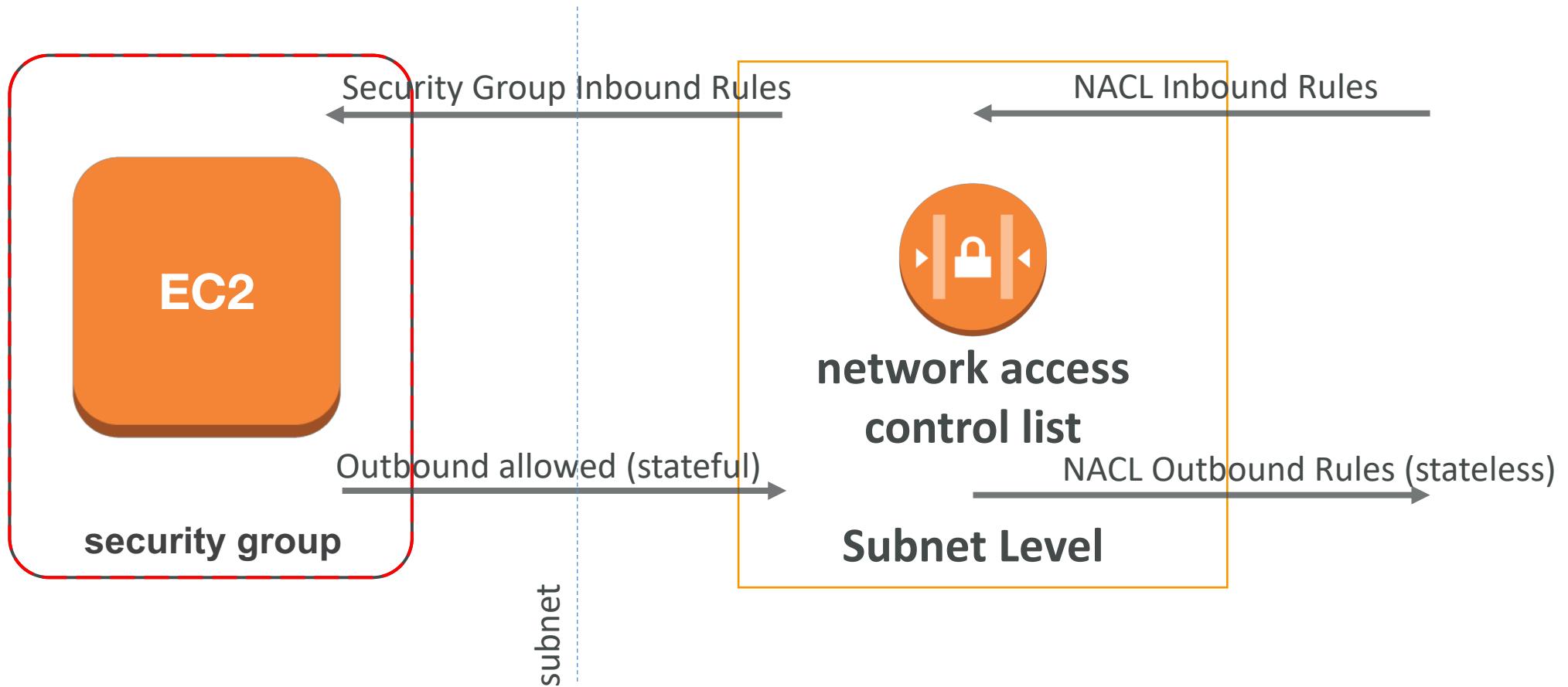
- See: <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-comparison.html>

# DNS Resolution in VPC

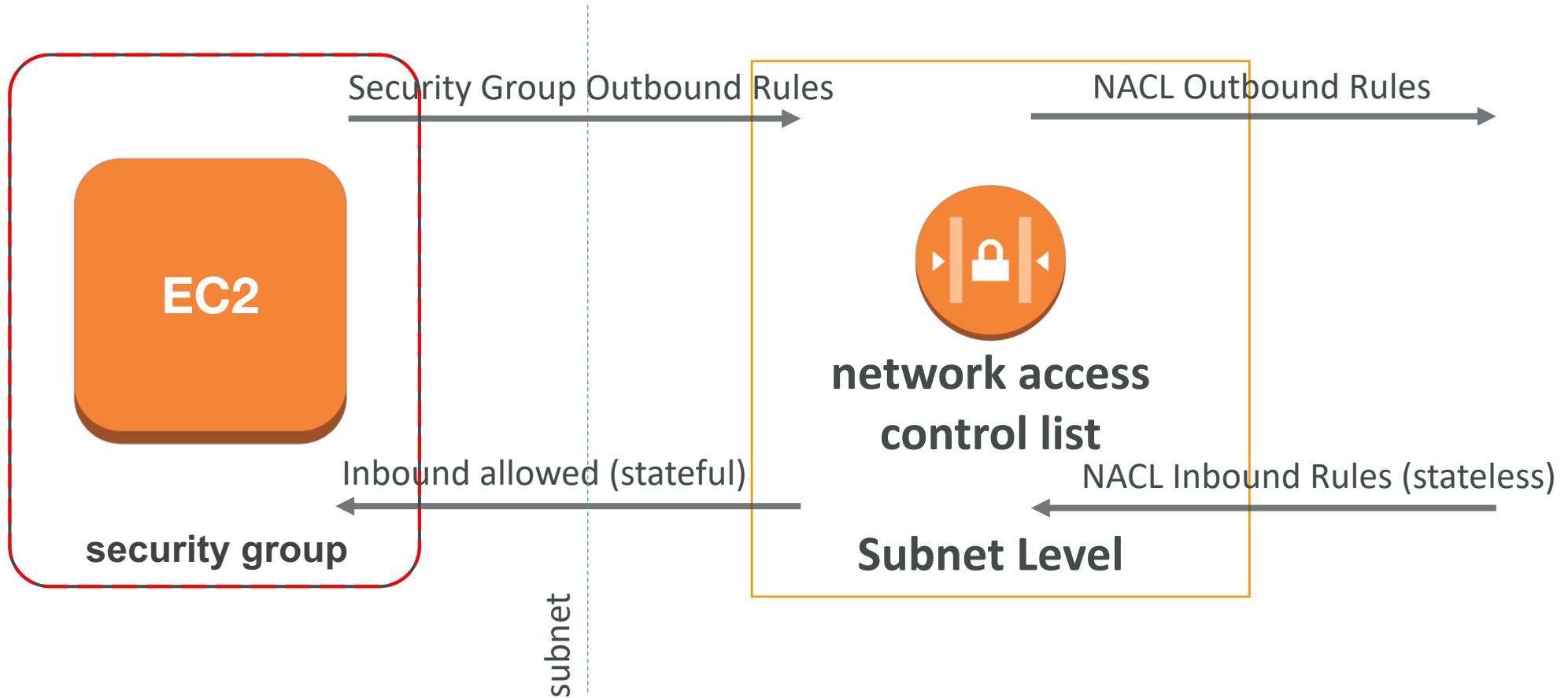
- **enableDnsSupport:** (= DNS Resolution setting)
  - Default True
  - Helps decide if DNS resolution is supported for the VPC
  - If True, queries the AWS DNS server at 169.254.169.253
- **enableDnsHostname:** (= DNS Hostname setting)
  - False by default for newly created VPC, True by default for Default VPC
  - Won't do anything unless enableDnsSupport=true
  - If True, Assign public hostname to EC2 instance if it has a public
- **If you use custom DNS domain names in a private zone in Route 53, you must set both these attributes to true**

# Network ACLs & Security Group

## Incoming Request



# Network ACLs & Security Group Outgoing Request



# Network ACLs

- NACL are like a firewall which control traffic from and to subnet
- Default NACL allows everything outbound and everything inbound
- **One NACL per Subnet, new Subnets are assigned the Default NACL**
- Define NACL rules:
  - Rules have a number (1-32766) and higher precedence with a lower number
  - E.g. If you define #100 ALLOW <IP> and #200 DENY <IP>, IP will be allowed
  - Last rule is an asterisk (\*) and denies a request in case of no rule match
  - AWS recommends adding rules by increment of 100
- Newly created NACL will deny everything
- NACL are a great way of blocking a specific IP at the subnet level

# Network ACLs vs Security Groups

Security Group	Network ACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
Is stateful: Return traffic is automatically allowed, regardless of any rules	Is stateless: Return traffic must be explicitly allowed by rules
We evaluate all rules before deciding whether to allow traffic	We process rules in number order when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets it's associated with (therefore, you don't have to rely on users to specify the security group)

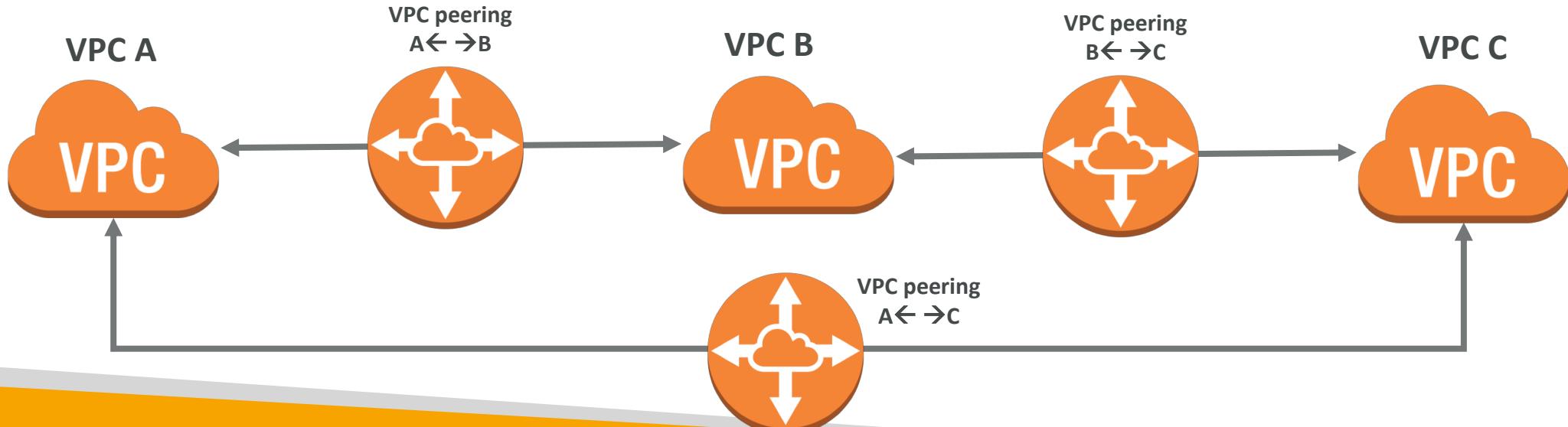
[https://docs.aws.amazon.com/vpc/latest/userguide/VPC\\_Security.html#VPC\\_Security\\_Comparison](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html#VPC_Security_Comparison)

# Example Network ACL with Ephemeral Ports

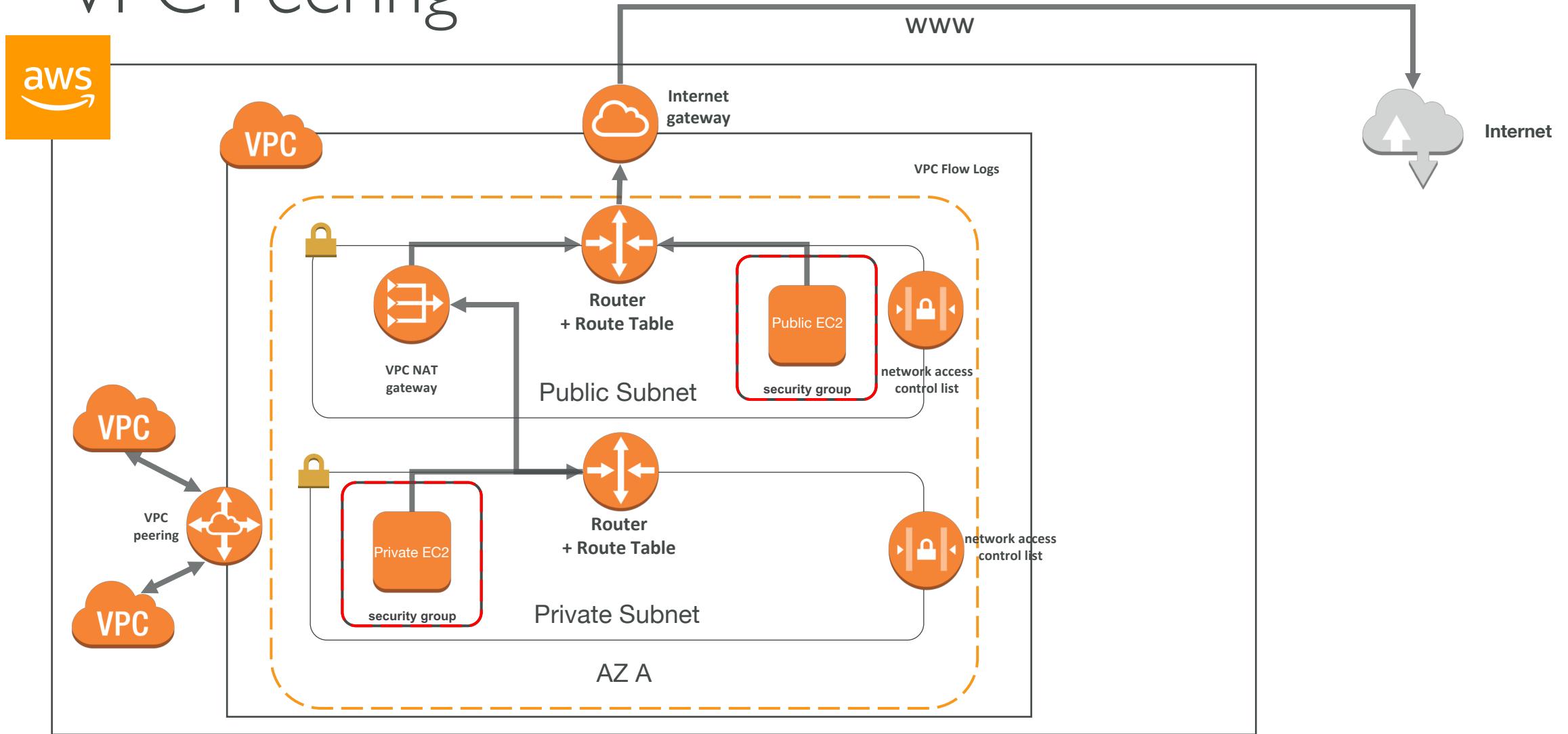
- <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html>

# VPC Peering

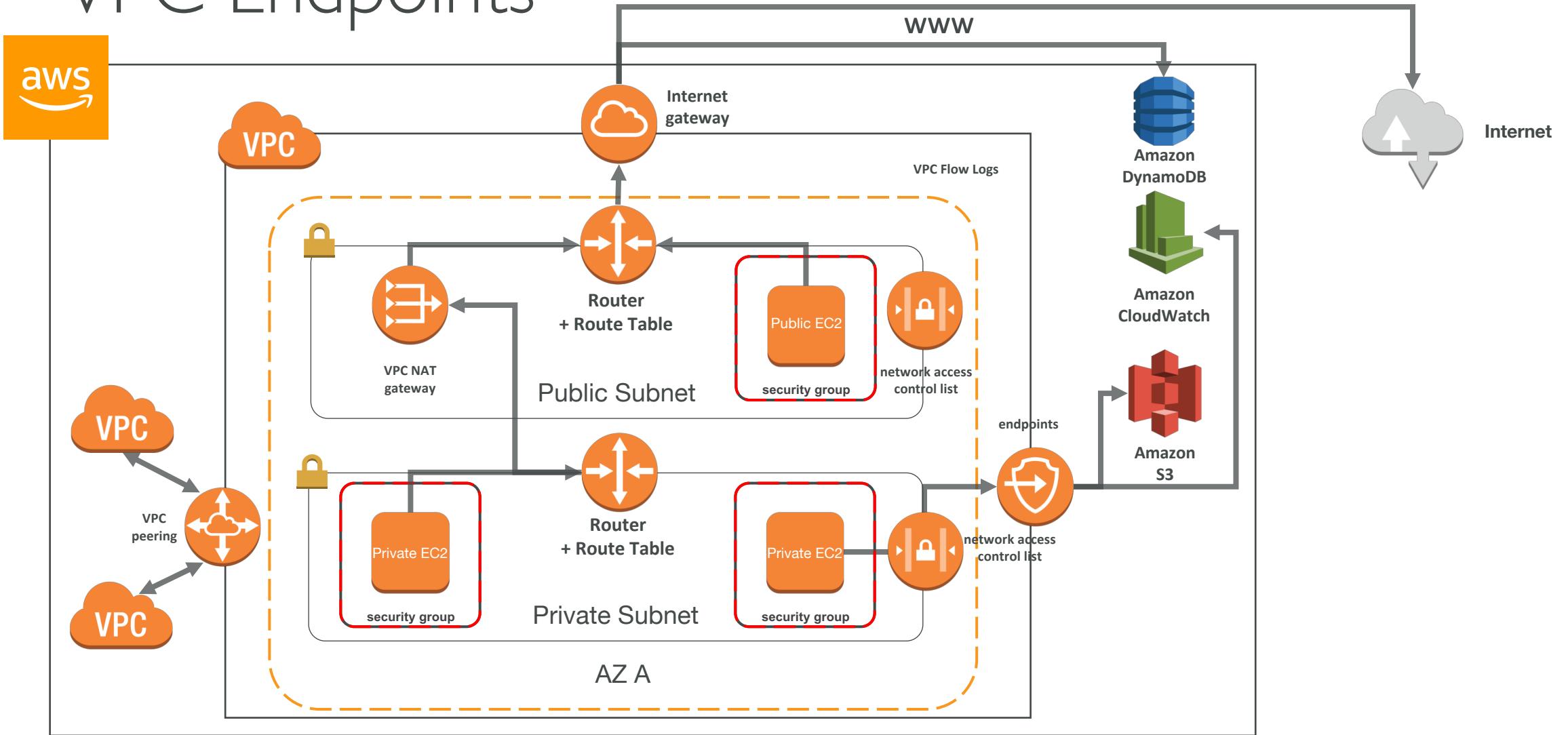
- Connect two VPC, privately using AWS' network
- Make them behave as if they were in the same network
- Must not have overlapping CIDR
- VPC Peering connection is **not transitive** (must be established for each VPC that need to communicate with one another)
- You can do VPC peering with another AWS account
- You must update route tables in each VPC's subnets to ensure instances can communicate



# VPC Peering



# VPC Endpoints



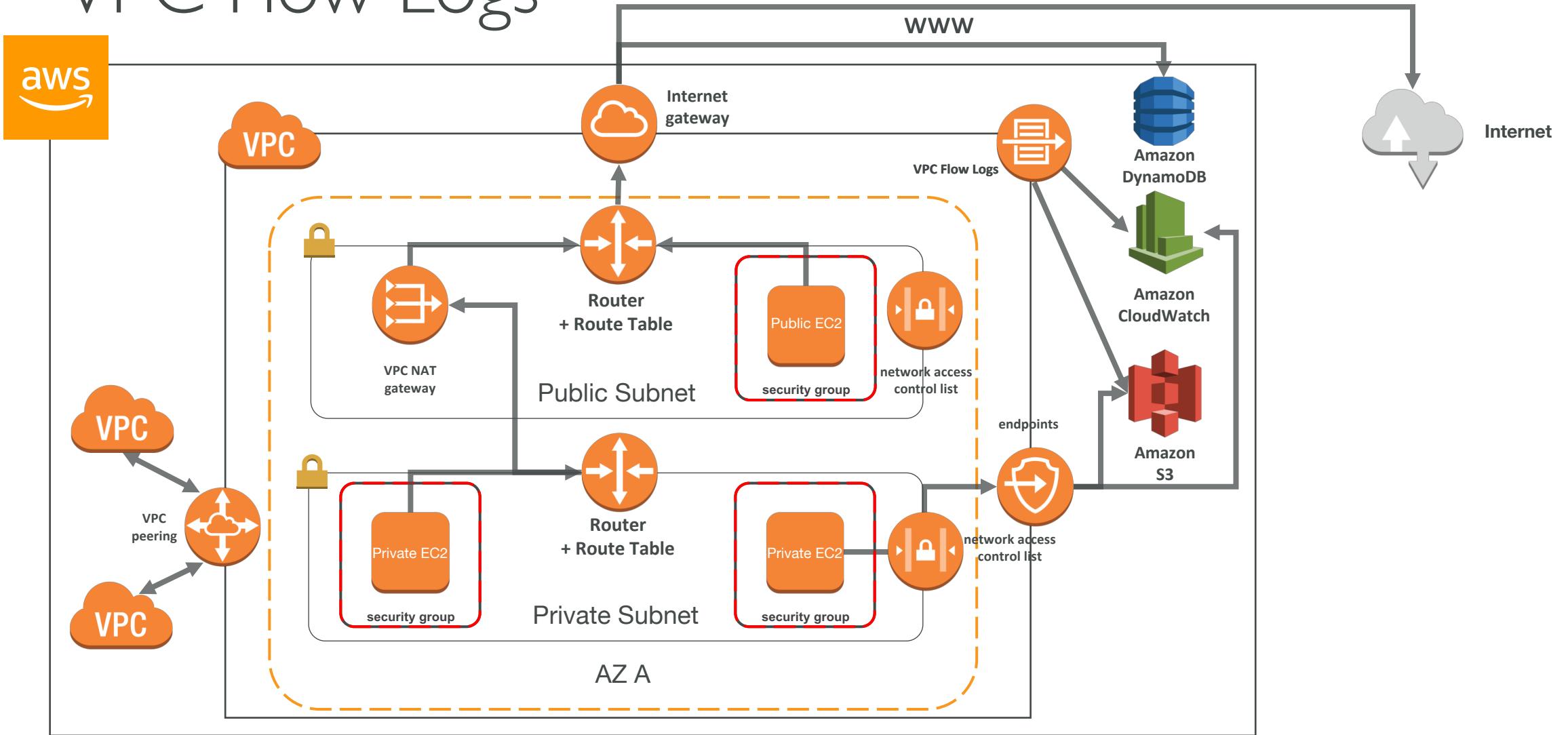
# VPC Endpoints

- Endpoints allow you to connect to AWS Services using a private network instead of the public www network
- They scale horizontally and are redundant
- They remove the need of IGW, NAT, etc... to access AWS Services
- Interface: provisions an ENI (private IP address) as an entry point (must attach security group) – most AWS services
- Gateway: provisions a target and must be used in a route table – S3 and DynamoDB
- In case of issues:
  - Check DNS Setting Resolution in your VPC
  - Check Route Tables

# Flow Logs

- Capture information about IP traffic going into your interfaces:
  - VPC Flow Logs
  - Subnet Flow Logs
  - Elastic Network Interface Flow Logs
- Helps to monitor & troubleshoot connectivity issues
- Flow logs data can go to S3 / CloudWatch Logs
- Captures network information from AWS managed interfaces too: ELB, RDS, ElastiCache, Redshift, WorkSpaces

# VPC Flow Logs



# Flow Log Syntax

- <version> <account-id> <interface-id> <srcaddr> <dstaddr> <srcport> <dstport> <protocol> <packets> <bytes> <start> <end> <action> <log-status>
- Srcaddr, dstaddr help identify problematic IP
- Srcport, dstport help identify problematic ports
- Action : success or failure of the request due to Security Group / NACL
- Can be used for analytics on usage patterns, or malicious behavior
- Flow logs example: <https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html#flow-log-records>
- Query VPC flow logs using Athena on S3 or CloudWatch Logs Insights

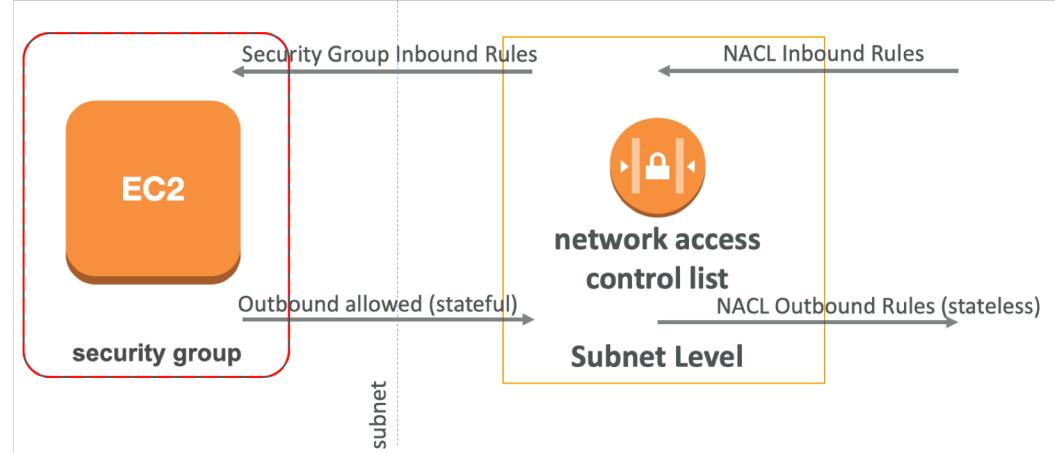
# Flow Logs: Looking at “ACTION” field

## How to troubleshoot SG vs NACL issue?

### For incoming requests

Inbound REJECT: NACL or SG

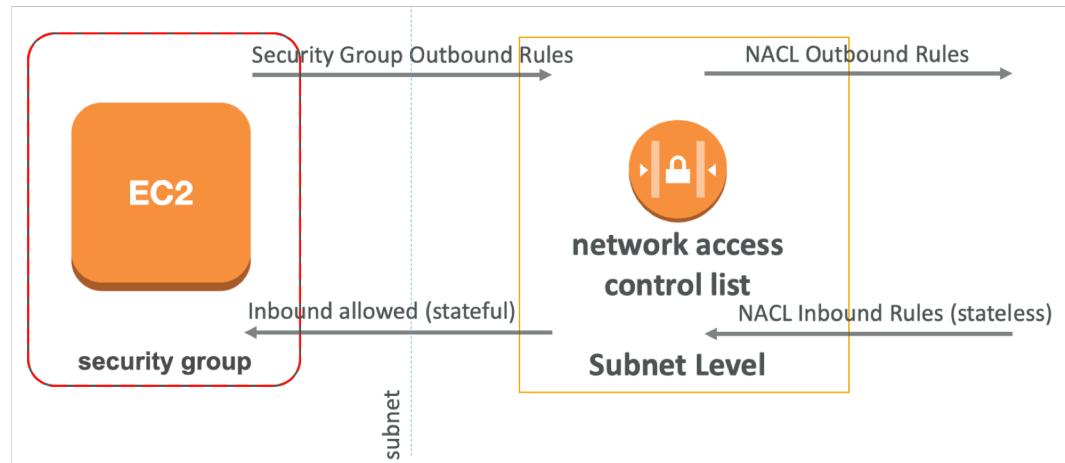
Inbound ACCEPT, outbound REJECT: NACL



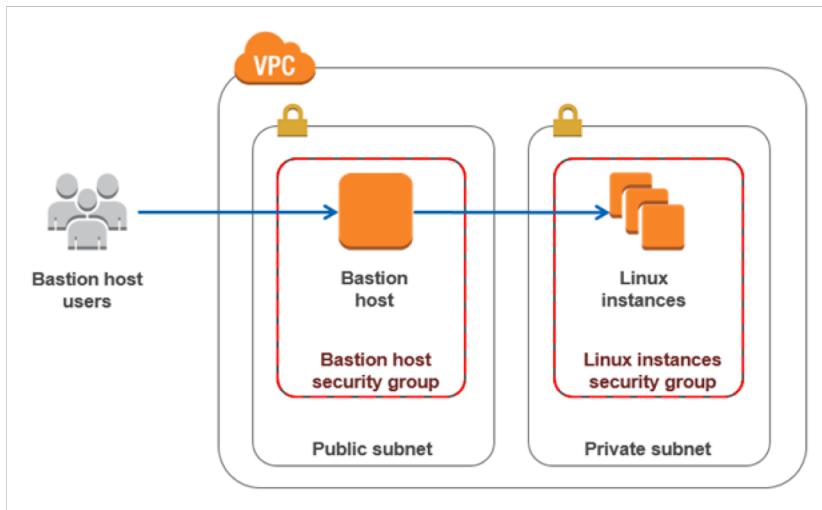
### For outgoing requests

Outbound REJECT: NACL or SG

Outbound ACCEPT, inbound REJECT: NACL

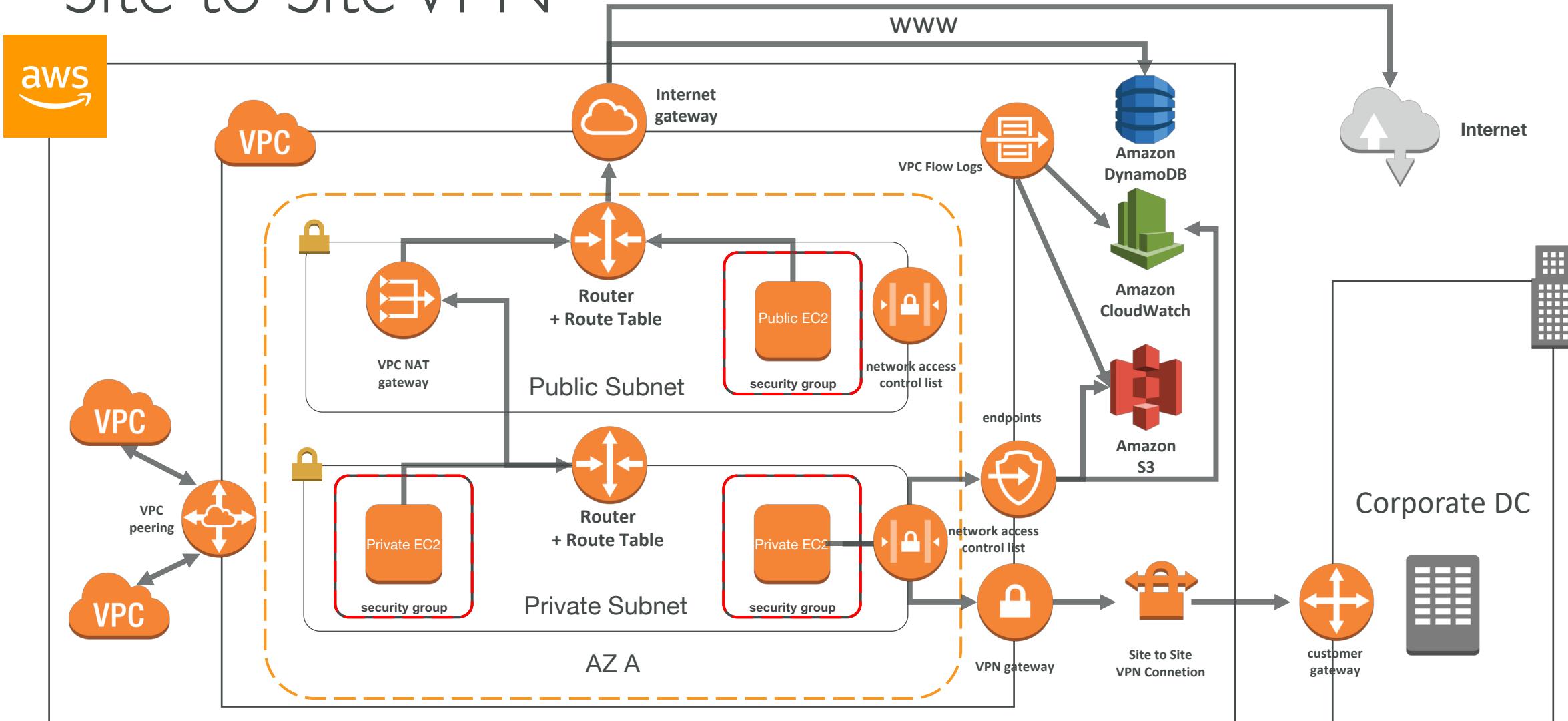


# Bastion Hosts



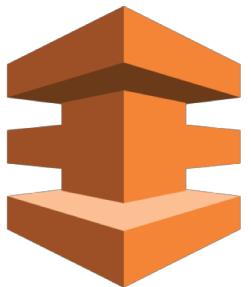
- We can use a Bastion Host to SSH into our private instances
- The bastion is in the public subnet which is then connected to all other private subnets
- **Bastion Host security group must be tightened**
- Exam Tip: Make sure the bastion host only has port 22 traffic from the IP you need, not from the security groups of your other instances

# Site to Site VPN



# Site to Site VPN

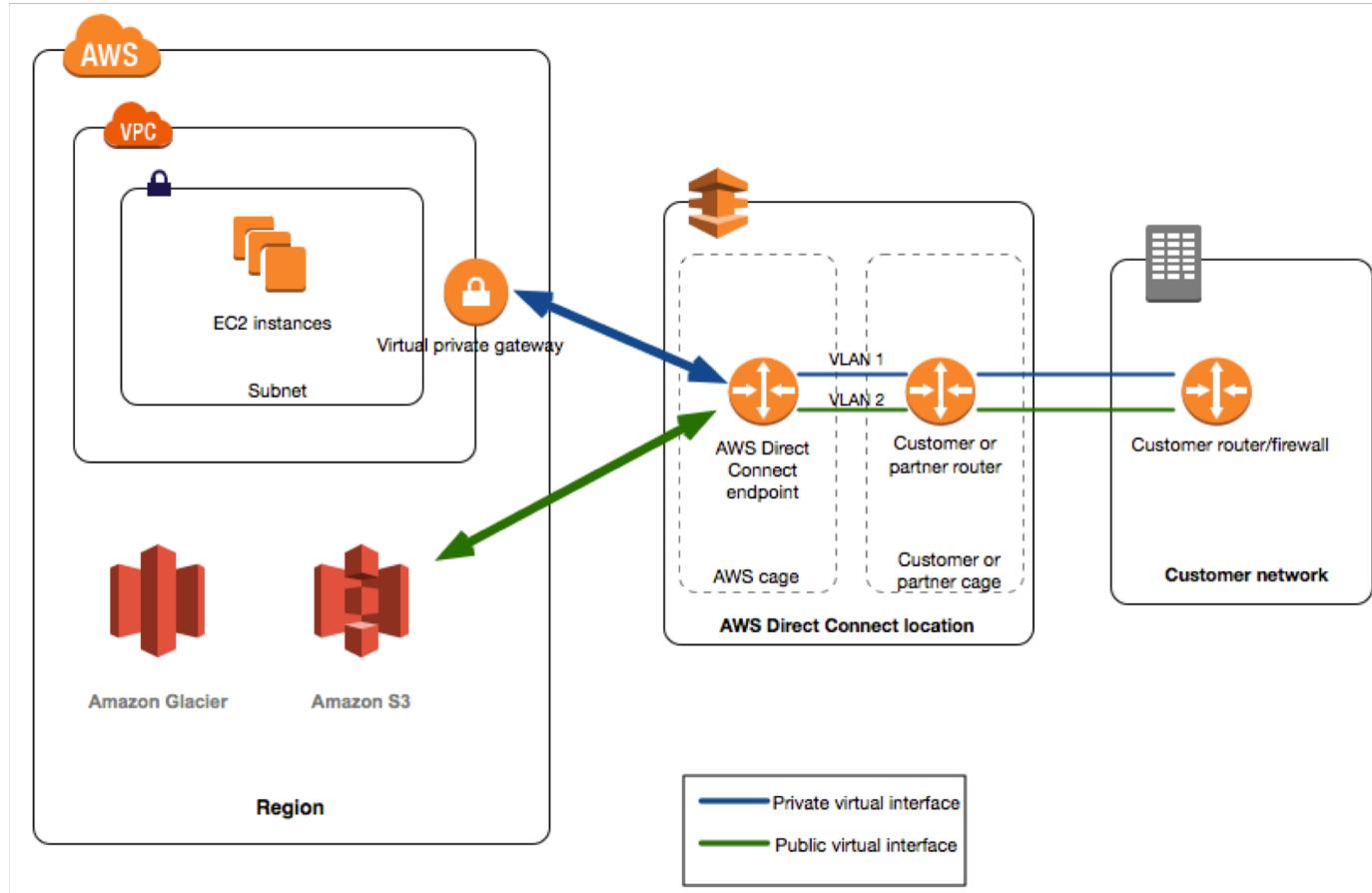
- Virtual Private Gateway:
  - VPN concentrator on the AWS side of the VPN connection
  - VGW is created and attached to the VPC from which you want to create the Site-to-Site VPN connection
  - Possibility to customize the ASN
- Customer Gateway:
  - Software application or physical device on customer side of the VPN connection
  - <https://docs.aws.amazon.com/vpc/latest/adminguide/introduction.html#devices-tested>
  - IP Address:
    - Use static, internet-routable IP address for your customer gateway device.
    - If behind a CGW behind NAT (with NAT-T), use the public IP address of the NAT



# Direct Connect

- Provides a dedicated private connection from a remote network to your VPC
- Dedicated connection must be setup between your DC and AWS Direct Connect locations
- You need to setup a Virtual Private Gateway on your VPC
- Access public resources (S3) and private (EC2) on same connection
- Use Cases:
  - Increase bandwidth throughput - working with large data sets – lower cost
  - More consistent network experience - applications using real-time data feeds
  - Hybrid Environments (on prem + cloud)
- Supports both IPv4 and IPv6

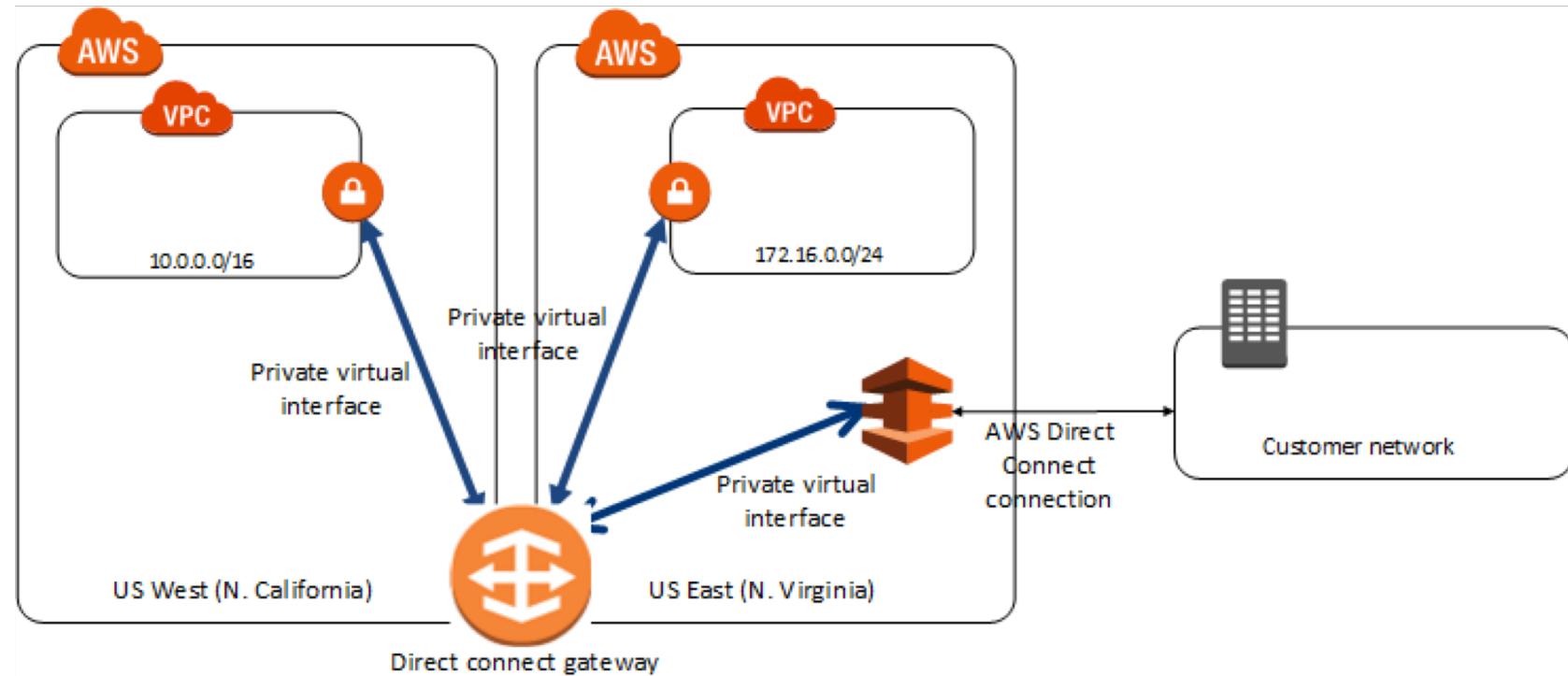
# Direct Connect Diagram



[https://docs.aws.amazon.com/directconnect/latest/UserGuide/images/direct\\_connect\\_overview.png](https://docs.aws.amazon.com/directconnect/latest/UserGuide/images/direct_connect_overview.png)

# Direct Connect Gateway

- If you want to setup a Direct Connect to one or more VPC in many different regions (same account), you must use a Direct Connect Gateway



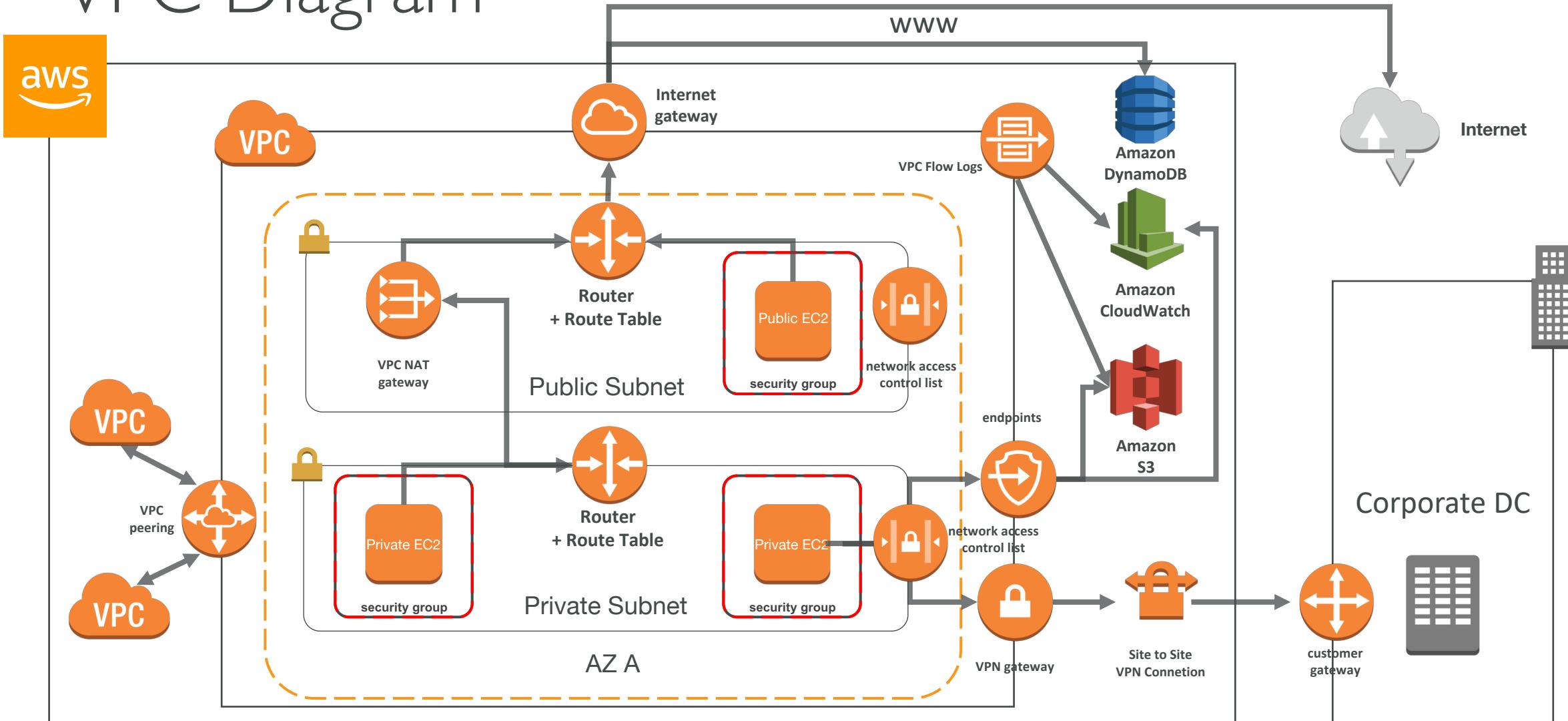
<https://docs.aws.amazon.com/directconnect/latest/UserGuide/direct-connect-gateways.html>

# Egress Only Internet Gateway



- Egress only Internet Gateway is for IPv6 only
- Similar function as a NAT, but a NAT is for IPv4
- Good to know: IPv6 are all public addresses
- Therefore all our instances with IPv6 are publicly accessible
- Egress Only Internet Gateway gives our IPv6 instances access to the internet, but they won't be directly reachable by the internet
- After creating an Egress Only Internet Gateway, edit the route tables

# VPC Diagram



# VPC Section Summary (1/2)

- **CIDR:** IP Range
- **VPC:** Virtual Private Cloud => we define a list of IPv4 & IPv6 CIDR
- **Subnets:** Tied to an AZ, we define a CIDR
- **Internet Gateway:** at the VPC level, provide IPv4 & IPv6 Internet Access
- **Route Tables:** must be edited to add routes from subnets to the IGW, VPC Peering Connections, VPC Endpoints, etc...
- **NAT Instances:** gives internet access to instances in private subnets. Old, must be setup in a public subnet, disable Source / Destination check flag
- **NAT Gateway:** managed by AWS, provides scalable internet access to private instances, IPv4 only
- **Private DNS + Route 53:** enable DNS Resolution + DNS hostnames (VPC)
- **NACL:** Stateless, subnet rules for inbound and outbound, don't forget ephemeral ports
- **Security Groups:** Stateful, operate at the EC2 instance level

# VPC Section Summary (1/2)

- **VPC Peering:** Connect two VPC with non overlapping CIDR, non transitive
- **VPC Endpoints:** Provide private access to AWS Services (S3, DynamoDB, CloudFormation, SSM) within VPC
- **VPC Flow Logs:** Can be setup at the VPC / Subnet / ENI Level, for ACCEPT and REJECT traffic, helps identifying attacks, analyze using Athena or CloudWatch Log Insights
- **Bastion Host:** Public instance to SSH into, that has SSH connectivity to instances in private subnets
- **Site to Site VPN:** setup a Customer Gateway on DC, a Virtual Private Gateway on VPC, and site-to-site VPN over public internet
- **Direct Connect:** setup a Virtual Private Gateway on VPC, and establish a direct private connection to an AWS Direct Connect Location
- **Direct Connect Gateway:** setup a Direct Connect to many VPC in different regions
- **Internet Gateway Egress:** like a NAT Gateway, but for IPv6

# Exam Review & Tips

# State of learning checkpoint

- Let's look how far we've gone on our learning journey
- <https://aws.amazon.com/certification/certified-sysops-admin-associate/>

# Practice makes perfect

- If you're new to AWS, take a bit of AWS practice thanks to this course before rushing to the exam
  - The exam recommends you to have one or more years of hands-on experience on AWS
  - Practice makes perfect!
- 
- If you feel overwhelmed by the amount of knowledge you just learned, just go through it one more time

# Proceed by elimination

- Most questions are going to be scenario based
  - For all the questions, rule out answers that you know for sure are wrong
  - For the remaining answers, understand which one makes the most sense
- 
- There are very few trick questions
  - Don't over-think it
  - If a solution seems feasible but highly complicated, it's probably wrong

# Skim the AWS Whitepapers

- You can read about some AWS White Papers here:
  - Architecting for the Cloud: AWS Best Practices
  - AWS Security Best Practices
  - Amazon Web Services: Overview of Security Processes
  - AWS Well-Architected Framework
  - Development and Test on AWS
  - Backup and Recovery Approaches Using AWS
  - Amazon Virtual Private Cloud Connectivity Options
  - How AWS Pricing Works
- Overall we've explored all the most important concepts in the course
- It's never bad to have a look at the whitepapers you think are interesting!

# Read each service's FAQ

- FAQ = Frequently asked questions
- Example: <https://aws.amazon.com/vpc/faqs/>
- FAQ cover a lot of the questions asked at the exam
- They help confirm your understanding of a service

# Get into the AWS Community

- Help out and discuss with other people in the course Q&A
  - Review questions asked by other people in the Q&A
  - Do the practice test in this section
- 
- Read forums online
  - Read online blogs
  - Attend local meetups and discuss with other AWS engineers
  - Watch re-invent videos on Youtube (AWS Conference)

# How will the exam work?

- You'll have to register online at <https://www.aws.training/>
- Fee for the exam is 150 USD
- Provide two identity documents (ID, Credit Card, details are in emails sent to you)
- No notes are allowed, no pen is allowed, no speaking
- 65 questions will be asked in 130 minutes
- At the end you can optionally review all the questions / answers
  
- You will know right away if you passed / failed the exams
- You will not know which answers were right / wrong
- You will know the overall score a few days later (email notification)
- To pass you need a score of at least 720 out of 1000
- If you fail, you can retake the exam again 14 days later

# Congratulations!

# Congratulations!

- Congrats on finishing the course!
- I hope you will pass the exam without a hitch ☺
- If you haven't done so yet, I'd love a review from you!
- If you passed, I'll be more than happy to know I've helped
  - Post it in the Q&A to help & motivate other students. Share your tips!
  - Post it on LinkedIn and tag me!
- Overall, I hope you learned how to use AWS and that you will be a tremendously good AWS SysOps