# AWS Solutions Associate Architect Associate

## Notes

Introduction:

## Region

* **Cluster of Availability Zones** (usually 3, min 2, max 6) within a geographic region
* Ex: us-east-1, ap-south-1
* Consideration when selecting a region
  + Compliance (some countries require the data to be present in a data center present in that country by law)
  + Proximity to customers: reduced latency
  + Available services within a Region: new services and new features aren’t available in every Region
  + Pricing: pricing varies region to region and is transparent in the service pricing page

## Availability Zone (AZ)

* Each AZ is one or more discrete data centers with redundant power, networking, and connectivity
* Ex: us-east-1a, us-east-1b & us-east-1c
* AZs are separated from each other (isolated from disasters)
* They’re connected with high bandwidth, ultra-low latency networking

AZ name (eg. us-east-1a) is linked to an AWS account. **Same AZ name for two AWS accounts might not refer to the same physical AZ**. Use **AZ ID (unique ID for each AZ)** to coordinate AZ across accounts.

#### 

#### **Computing**

# Elastic Compute Cloud (EC2)

* Regional Service
* EC2 (Elastic Compute Cloud) is an **Infrastructure as a Service (IaaS)**
* Stopping & Starting an instance may change its public IP but not its private IP
* **AWS Compute Optimizer** recommends optimal AWS Compute resources for your workloads
* There is a vCPU-based On-Demand Instance soft limit per region

## User Data

* Some commands that run when the instance is launched for the first time (doesn't execute for subsequent runs)
* Used to automate **dynamic** boot tasks (that cannot be done using AMIs)
  + Installing updates
  + Installing software
  + Downloading common files from the internet
* Runs with the **root user privilege**

## Instance Classes

* **General Purpose**
  + Great for a diversity of workloads such as **web servers** or **code repositories**
  + Balance between compute, memory & networking
* **Compute Optimized**
  + Great for compute intensive tasks
    - Batch Processing
    - Media Transcoding
    - HPC
    - Gaming Servers
* **Memory Optimized**
  + Great for **in-memory databases** or **distributed web caches**
* **Storage Optimized**
  + Great for storage intensive tasks (accessing local databases)
    - OLTP systems
    - Distributed File System (DFS)

## Security Groups

* **Only contain Allow rules**
* External firewall for EC2 instances (if a request is blocked by SG, instance will never know)
* Security groups rules can reference a resource by IP or Security Group
* Default SG
  + inbound traffic from the same SG is allowed
  + all outbound traffic is allowed
* New SG
  + all inbound traffic is blocked
  + all outbound traffic is allowed
* A security group can be attached to multiple instances and vice versa
* Bound to a VPC (and hence to a region)
* Recommended to maintain a separate security group for SSH access
* Blocked requests will give a **Time Out** error

## IAM Roles for EC2 instances

* Never enter AWS credentials into the EC2 instance, instead attach [IAM Roles](https://notes.arkalim.org/notes/aws%20solutions%20architect%20associate/Identity%20&%20Access%20Management%20(IAM)#roles) to the instances

## Purchasing Options

#### **On-demand Instances**

* Pay per use (no upfront payment)
* Highest cost
* No long-term commitment
* Recommended for short-term, uninterrupted and **unpredictable** workloads

#### **Standard Reserved Instances**

* Reservation Period: 1 year or 3 years
* Recommended for steady-state applications (like database)
* **Sell unused instances** on the Reserved Instance Marketplace

#### **Convertible Reserved Instances**

* Can change the instance type
* Lower discount
* **Cannot sell unused instances** on the Reserved Instance Marketplace

#### **Scheduled Reserved Instances**

* reserved for a time window (ex. everyday from 9AM to 5PM)

#### **Spot Instances**

* Work on a bidding basis where you are willing to pay a specific max hourly rate for the instance. Your instance can terminate if the spot price increases.
* **Spot blocks** are designed not to be interrupted
* Good for workloads that are resilient to failure
  + Distributed jobs (resilient if some nodes go down)
  + Batch jobs

#### **Dedicated Hosts**

* Server hardware is allocated to a specific company (not shared with other companies)
* **3 year reservation** period
* Billed per host
* Useful for software that have **BYOL (Bring Your Own License)** or for companies that have strong regulatory or compliance needs

#### **Dedicated Instances**

* Dedicated hardware
* Billed per instance
* No control over instance placement

#### **On-Demand Capacity Reservations**

* Ensure you have the available capacity in an AZ to launch EC2 instances when needed
* Can reserve for a recurring schedule (ex. everyday from 9AM to 5PM)
* No need for 1 or 3-year commitment (independent of billing discounts)
* Need to specify the following to create capacity reservation: - AZ - Number of instances - Instance attributes

**Reserved Capacity & Instances Comparison**

****

## Spot Instances

#### **Spot Requests**

* **One-time**: Request once opened, spins up the spot instances and the request closes.
* **Persistent**:
  + Request will stay disabled while the spot instances are up and running.
  + It becomes active after the spot instance is interrupted.
  + If you stop the spot instance, the request will become active only after you start the spot instance.
* You can only cancel spot instance requests that are open, active, or disabled.
* Cancelling a Spot Request does not terminate instances. You must first cancel a Spot Request, and then terminate the associated Spot Instances.

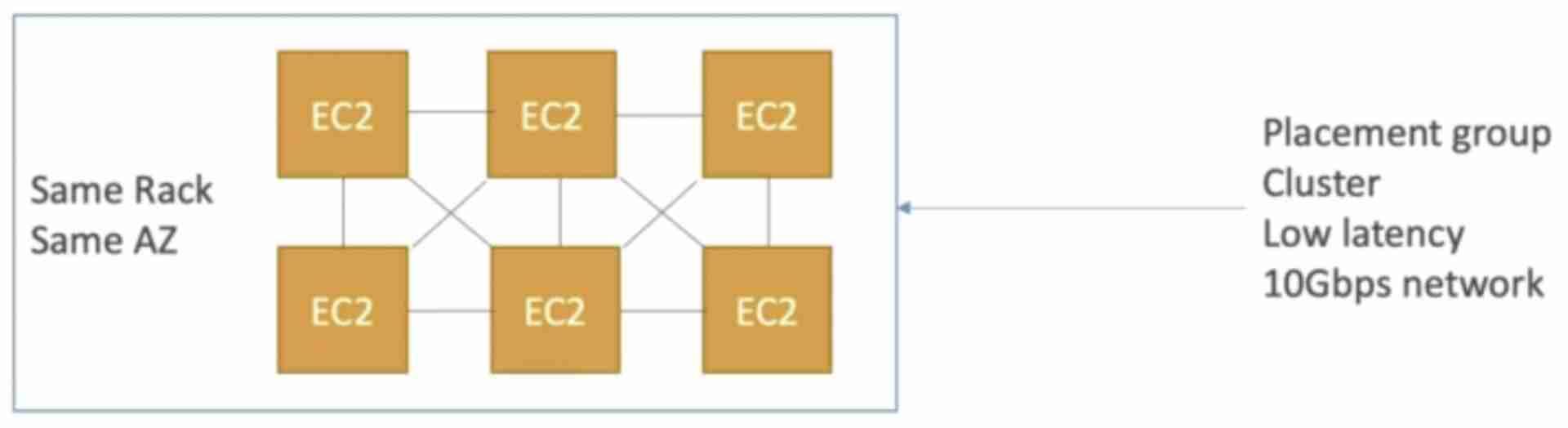
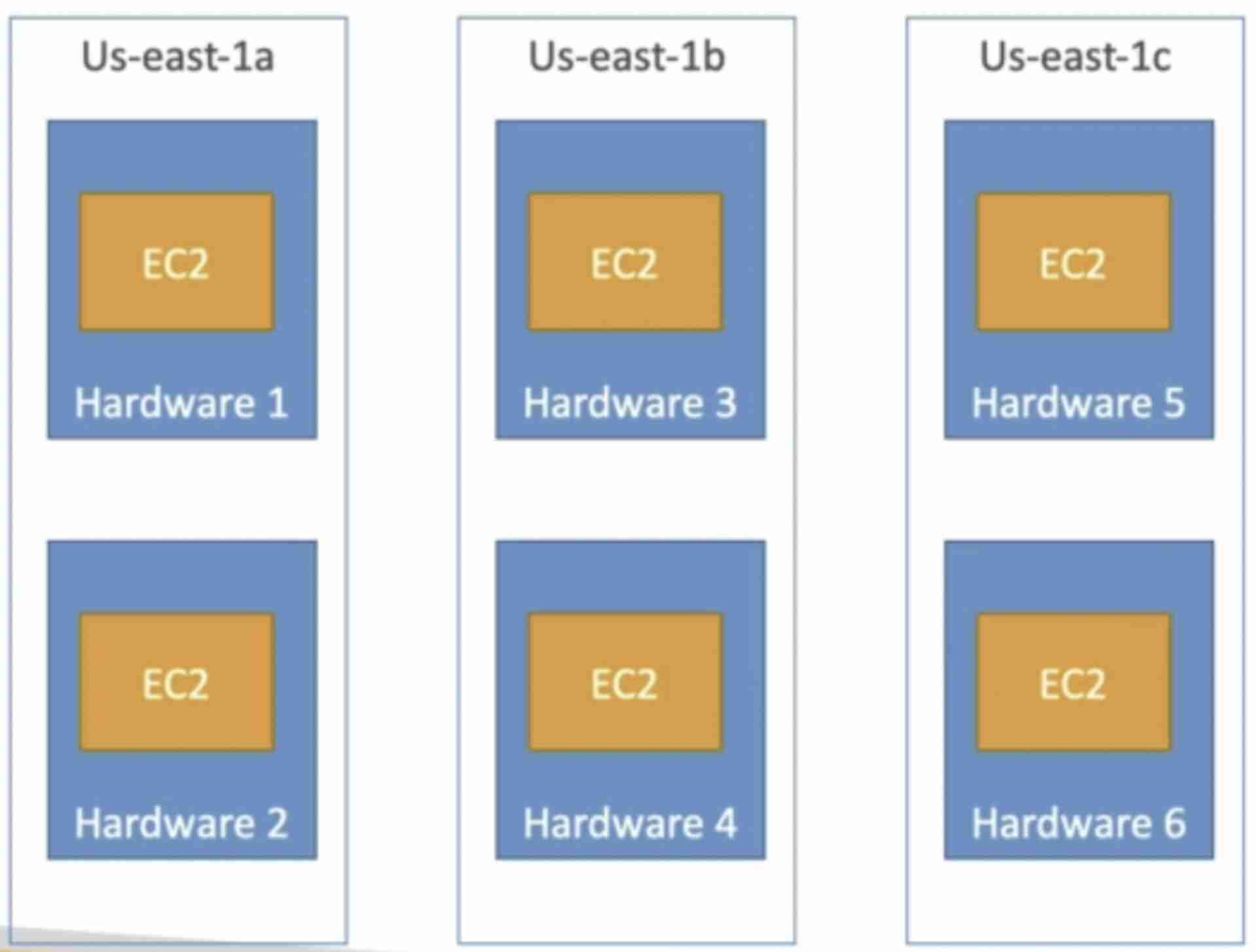
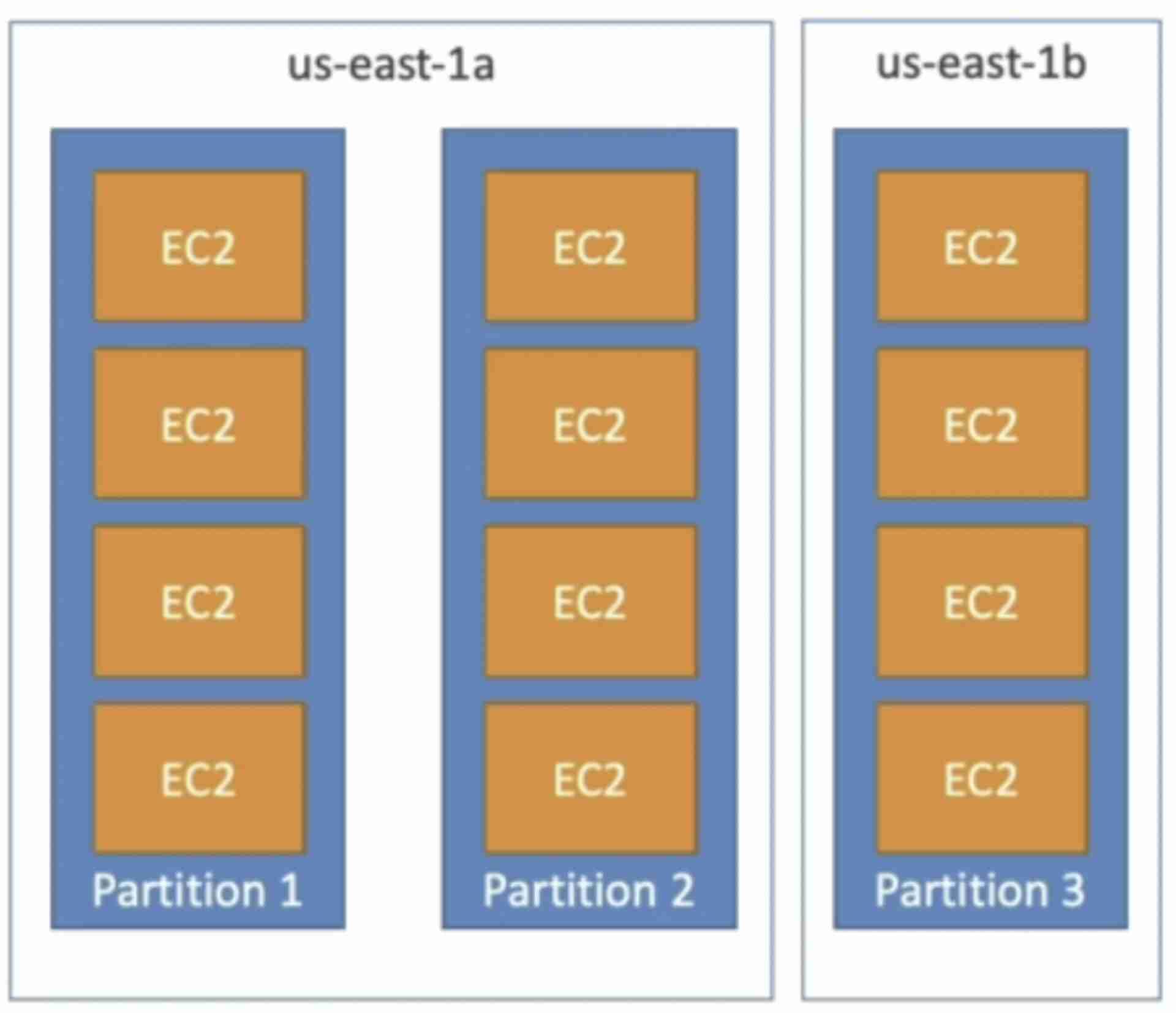
#### **Spot Fleets**

* Combination of spot and on-demand instances (optional) that tries to **optimize for cost or capacity**
* **Launch Templates must be used to have on-demand instances in the fleet**
* Can consist of instances of different classes
* Strategies to allocate Spot Instances:
  + **lowestPrice** - from the pool with the lowest price (cost optimization, short workload)
  + **diversified** - distributed across all pools (great for availability, long workloads)
  + **capacityOptimized** - pool with the optimal capacity for the number of instances

## Elastic IP

* **Static Public IP** that you own as long as you don't delete it
* Can be attached to an EC2 instance (even when it is stopped)
* Soft limit of 5 elastic IPs per account
* Doesn’t incur charges as long as the following conditions are met (EIP behaving like any other public IP randomly assigned to an EC2 instance):
  + The Elastic IP is associated with an Amazon EC2 instance
  + The instance associated with the Elastic IP is running
  + The instance has only one Elastic IP attached to it

## Placement Groups (Placement Strategies)

* **Cluster Placement Group (optimize for network)**
  + All the instances are placed on the same hardware (same rack)
  + Pros: Great network (10 Gbps bandwidth between instances)
  + Cons: If the rack fails, all instances will fail at the same time
  + Used in **HPC** (minimize inter-node latency & maximize throughput)
  + Image
    - 
* **Spread Placement Group (maximize availability)**
  + Each instance is in a separate rack (physical hardware) inside an AZ
  + Supports Multi AZ
  + Up to 7 instances per AZ per placement group (ex. for 15 instances, need 3 AZ)
  + Used for critical applications
  + Image
    - 
* **Partition Placement Group (balance of performance and availability)**
  + Instances in a partition share rack with each other
  + If the rack goes down, the entire partition goes down
  + Up to 7 partitions per AZ
  + Used in **big data** applications (Hadoop, HDFS, HBase, Cassandra, Kafka)
  + Image
    - 

If you receive a **capacity error** when launching an instance in a placement group that already has running instances, stop and start all of the instances in the placement group, and try the launch again. Restarting the instances may migrate them to hardware that has capacity for all the requested instances.

## Elastic Network Interface (ENI)

* ENI is a virtual network card that **gives a private IP to an EC2 instance**
* A primary ENI is created and attached to the instance upon creation and will be deleted automatically upon instance termination.
* We can create additional ENIs and attach them to an EC2 instance to access it via multiple private IPs.
* We can detach & attach ENIs across instances
* **ENIs are tied to the subnet** (and hence to the AZ)

## Instance States

* **Stop**
  + EBS root volume is preserved
* **Terminate**
  + EBS root volume gets destroyed
* **Hibernate**
  + Hibernation saves the contents from the instance memory (RAM) to the EBS root volume
  + EBS root volume is preserved
  + The instance boots much faster as the OS is not stopped and restarted
  + When you start your instance:
    - EBS root volume is restored to its previous state
    - RAM contents are reloaded
    - Processes that were previously running on the instance are resumed
    - Previously attached data volumes are reattached and the instance retains its instance ID
  + Should be used for applications that take a long time to start
  + **Not supported for Spot Instances**
  + Max hibernation duration = **60 days**
* **Standby**
  + Instance remains attached to the [ASG](https://notes.arkalim.org/notes/aws%20solutions%20architect%20associate/Auto%20Scaling%20Group%20(ASG)) but is temporarily put out of service (the ASG doesn't replace this instance)
  + Used to install updates or troubleshoot a running instance

## EC2 Nitro

* Newer virtualization technology for EC2 instances
* Better networking options (enhanced networking, HPC, IPv6)
* Higher Speed EBS (64,000 EBS IOPS max on Nitro instances whereas 32,000 on non-Nitro)
* Better underlying security

## vCPU & Threads

* vCPU is the total number of concurrent threads that can be run on an EC2 instance
* Usually 2 threads per CPU core (eg. 4 CPU cores ⇒ 8 vCPU)

## Storage

Instance Store

Elastic Block Storage (EBS)

Elastic File System (EFS)

## Monitoring

CloudWatch ⮕ EC2 Monitoring

## Amazon Machine Image (AMI)

* AMIs are the image of the instance after installing all the necessary OS, software and configuring everything.
* It boots much faster because the whole thing is pre-packaged and doesn’t have to be installed separately for each instance.
* Good for static configurations
* **Bound to a region** (can be copied across regions)

When the new AMI is copied from region A into region B, it automatically creates a snapshot in region B because AMIs are based on the underlying snapshots.

## Instance Metadata

* Url to fetch metadata about the instance (http://169.254.169.254/latest/meta-data)
* This URL is internal to AWS and can only be hit from the instance

## EC2 Classic & ClassicLink

* Instances run in single network shared with other customers (this is how AWS started)
* **ClassicLink** allows you to link EC2-Classic instances to a VPC in your account

## Billing

* **Reserved instances will be billed regardless of their state** (billed for a reserved period)
* **On-demand instances in stopping state when preparing to hibernate will be billed**
* If an instance is running, it will be billed
* In all the other cases, an instance will not be billed

## Run Command

* Systems Manager **Run Command** lets you remotely and securely **manage the configuration** of your **managed instances**. A *managed instance* is any EC2 instance that has been configured for **Systems Manager**.
* Run Command enables you to **automate common administrative tasks** and perform ad-hoc configuration changes at scale.
* You can use Run Command from the **AWS Console**, the AWS CLI, AWS Tools for Windows PowerShell, or the AWS SDKs. Run Command is offered at no additional cost.

## Instance Tenancy

## **Default**: Instance runs on shared hardware

* **Dedicated**: Instance runs on single-tenant hardware
* **Host**: Instance runs on dedicated host
* Tenancy of an instance can only be changed from **host to dedicated** or **dedicated to host** after the instance has been launched.
* Dedicated instance tenancy takes precedence over Default instance tenancy

## Troubleshooting

* The following are a few reasons why an instance might immediately terminate:
  + 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.

Amazon Machine Image (AMI)

AMIs are the image of the instance after installing all the necessary OS, software and configuring everything.

It boots much faster because the whole thing is pre-packaged and doesn’t have to be installed separately for each instance.

Good for static configurations

Bound to a region (can be copied across regions)

When the new AMI is copied from region A into region B, it automatically creates a snapshot in region B because AMIs are based on the underlying snapshots.

Instance Metadata

Url to fetch metadata about the instance (http://169.254.169.254/latest/meta-data)

This URL is internal to AWS and can only be hit from the instance

EC2 Classic & ClassicLink

Instances run in single network shared with other customers (this is how AWS started)

ClassicLink allows you to link EC2-Classic instances to a VPC in your account

Billing

Reserved instances will be billed regardless of their state (billed for a reserved period)

On-demand instances in stopping state when preparing to hibernate will be billed

If an instance is running, it will be billed

In all the other cases, an instance will not be billed

Run Command

Systems Manager Run Command lets you remotely and securely manage the configuration of your managed instances. A managed instance is any EC2 instance that has been configured for Systems Manager.

Run Command enables you to automate common administrative tasks and perform ad-hoc configuration changes at scale.

You can use Run Command from the AWS Console, the AWS CLI, AWS Tools for Windows PowerShell, or the AWS SDKs. Run Command is offered at no additional cost.

Instance Tenancy

Default: Instance runs on shared hardware

Dedicated: Instance runs on single-tenant hardware

Host: Instance runs on dedicated host

Tenancy of an instance can only be changed from host to dedicated or dedicated to host after the instance has been launched.

Dedicated instance tenancy takes precedence over Default instance tenancy

Troubleshooting

The following are a few reasons why an instance might immediately terminate:

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.

# Elastic Load Balancer

* Regional Service
* Supports Multi AZ
* Spread load across multiple EC2 instances
* Separate public traffic from private traffic
* Health checks allow ELB to know which instances are working properly (done on a port and a route, /health is common)
* **Does not support weighted routing**

If no targets are associated with the target groups => **503 Service Unavailable**

Using ALB & NLB, instances in peered VPCs can be used as targets using IP addresses.

## Types

#### **Classic Load Balancer (CLB) - deprecated**

* **Load Balancing to a single application**
* Supports HTTP, HTTPS (layer 7) & TCP, SSL (layer 3)
* Health checks are HTTP or TCP based
* Provides a fixed hostname (xxx.region.elb.amazonaws.com)

#### **Application Load Balancer (ALB)**

* **Load balancing to multiple applications** (target groups)
* Operates at Layer 7 (HTTP, HTTPS and WebSocket)
* Provides a fixed hostname (xxx.region.elb.amazonaws.com)
* ALB terminates the upstream connection and creates a new downstream connection to the targets (connection termination)
* **Security Groups can be attached to ALBs** to filters requests
* Great for micro services & container-based applications (Docker & ECS)
* Client info is passed in the request headers
  + Client IP => X-Forwarded-For
  + Client Port => X-Forwarded-Port
  + Protocol => X-Forwarded-Proto
* Target Groups
  + Health checks are done at the target group level
  + Target Groups could be
    - EC2 instances - HTTP
    - ECS tasks - HTTP
    - Lambda functions - HTTP request is translated into a JSON event
    - Private IP Addresses
* **Listener Rules** can be configured to route traffic to different target groups based on:
  + Path (example.com/users & example.com/posts)
  + Hostname (one.example.com & other.example.com)
  + Query String (example.com/users?id=123&order=false)
  + Request Headers
  + Source IP address

#### **Network Load Balancer (NLB)**

* Operates at Layer 4 (TCP, TLS, UDP)
* Can handle millions of request per seconds (extreme performance)
* **Lower latency** ~ 100 ms (vs 400 ms for ALB)
* **1 static public IP per AZ** (vs a static hostname for CLB & ALB)
* **Elastic IP can be assigned to NLB** (helpful for whitelisting specific IP)
* Maintains the same connection from client all the way to the target
* **No security groups can be attached to NLBs.** They just forward the incoming traffic to the right target group as if those requests were directly coming from client. So, the **attached instances must allow TCP traffic on port 80 from anywhere**.
* Within a target group, NLB can send traffic to
  + **EC2 instances**
    - If you specify targets using an instance ID, traffic is routed to instances using the **primary private IP address**
  + **IP addresses**
    - Used when you want to balance load for a physical server having a static IP.
  + **Application Load Balancer (ALB)**
    - Used when you want a static IP provided by an NLB but also want to use the features provided by ALB at the application layer.

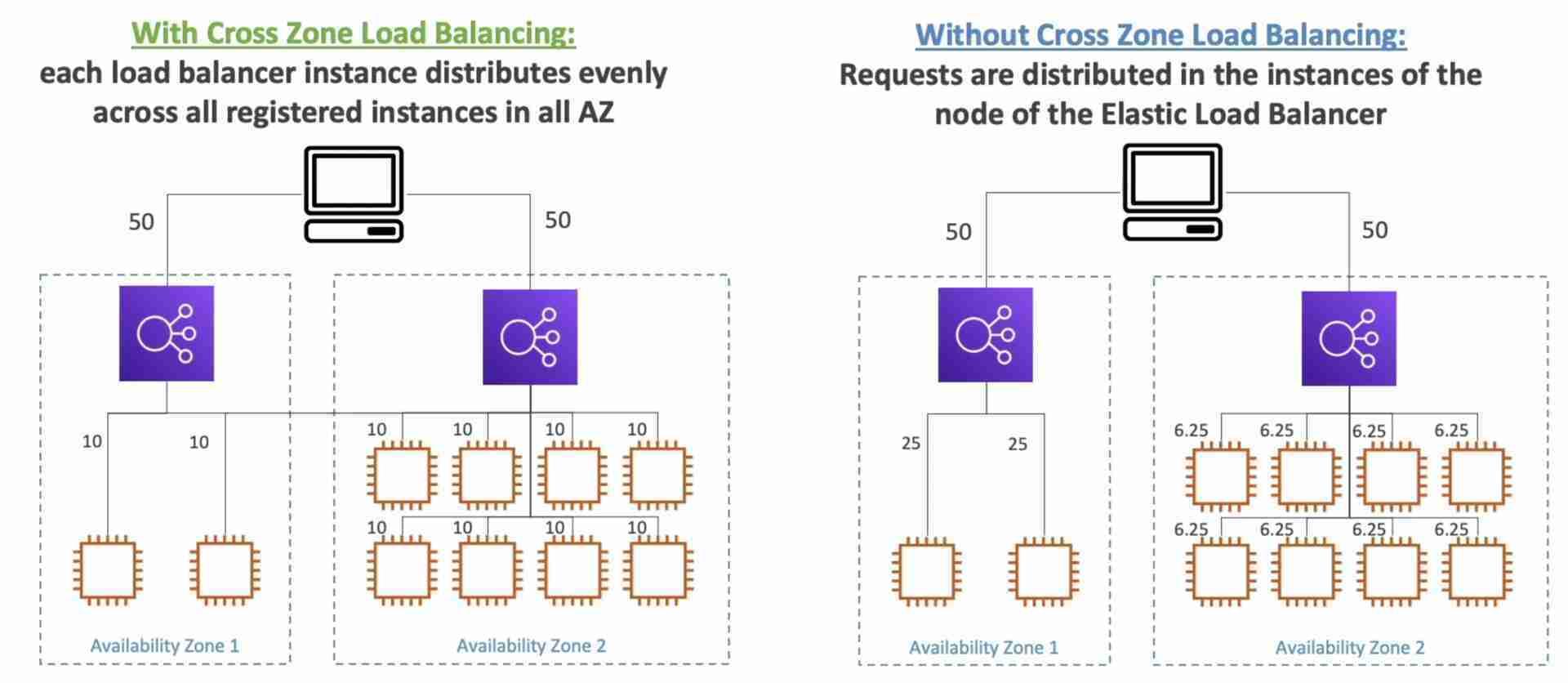
#### **Gateway Load Balancer (GWLB)**

* Operates at layer 3 (Network layer) - IP Protocol
* Used to route requests to a fleet of 3rd party virtual appliances like Firewalls, Intrusion Detection and Prevention Systems (IDPS), etc.
* Performs two functions:
  + **Transparent Network Gateway** (single entry/exit for all traffic)
  + Load Balancer (distributes traffic to virtual appliances)
* Uses GENEVE protocol
* Target groups for GWLB could be
  + EC2 instances
  + IP addresses

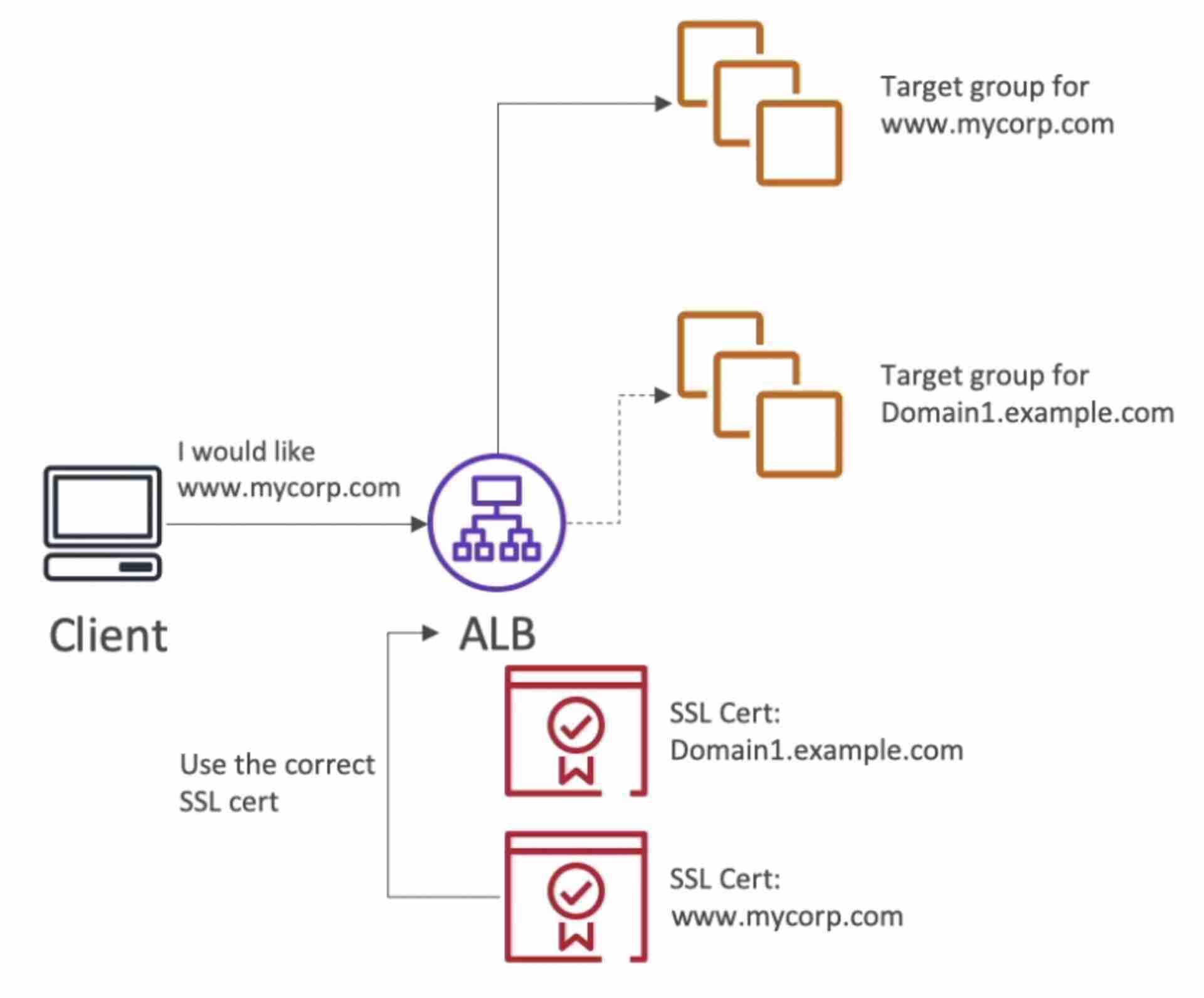
## Sticky Sessions (Session Affinity)

* Requests coming from a client is always redirected to the same instance based on a cookie. After the cookie expires, the requests coming from the same user might be redirected to another instance.
* **Only supported by CLB & ALB**
* Used to ensure the user doesn’t lose his session data, like login or cart info, while navigating between web pages.
* **Stickiness may cause load imbalance**
* Cookies could be
  + **Application-based** (TTL defined by the application)
  + **Load Balancer generated** (TTL defined by the load balancer)

## Cross-zone Load Balancing

* Allows ELBs in different AZ containing unbalanced number of instances to distribute the traffic evenly across all instances in all the AZ registered under a load balancer.
* Image
  + 
* Supported Load Balancers
  + Classic Load Balancer
    - Disabled by default
    - No charges for inter AZ data
  + Application Load Balancer
    - Always on (can’t be disabled)
    - No charges for inter AZ data
  + Network Load Balancer
    - Disabled by default
    - Charges for inter AZ data

## In-flight Encryption

* Use an NLB with a TCP listener & terminate SSL on EC2 instances
* Use an ALB with an HTTPS listener, install SSL certificates on the ALB & terminate SSL on the ALB
* Communication between ALB & EC2 instances can happen over HTTP inside the VPC
* **Server Name Indication (SNI)**
  + SNI allows us to load multiple SSL certificates on one Load Balancer to serve multiple websites securely
    - 
  + **Only works for ALB & NLB** (CLB only supports one SSL certificate)
  + Newer protocol, not every client supports it yet
  + **Supported in CloudFront** also

## Connection Draining (De-registration Delay)

* When an instance is to be de-registered from the ELB, the in-flight requests being served by that instance are given some pre-defined time to complete before the ELB de-registers it.
* ELB stops sending new requests to the EC2 instance which is de-registering
* Set manually (0 to 3600 seconds) (**default: 300 seconds**)

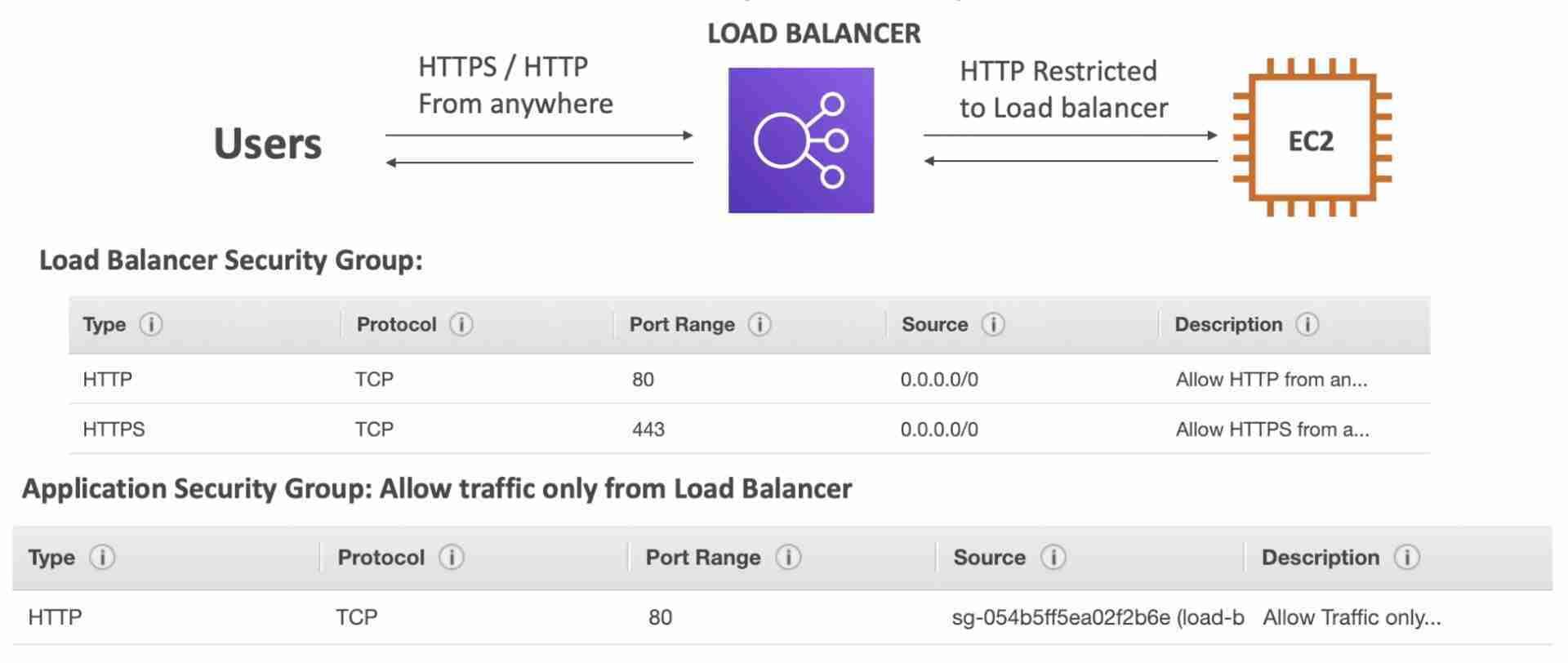
For instances behind an ELB and using ASG, increase the de-registration delay to ensure that the in-flight requests are completed before the ELB deregisters an instance which is to be terminated by the ASG.

## Access Logs

* Captures detailed information about requests sent to the load balancer
* Used to analyze traffic patterns and troubleshoot issues
* Disabled by default

## Misc

## Security Groups for a public facing ELB

* + ELB will be publicly available on the internet, so it’s security group should allow HTTP and HTTPS traffic from anywhere. EC2 should only allow traffic from the ELB, so the its security group should allow HTTP requests from ELB’s security group.
  + Image
    - 

# Auto Scaling Group (ASG)

* Regional Service
* Supports Multi AZ
* Automatically add or remove instances (scale horizontally) based on the load
* Free (pay for the underlying resources)
* IAM roles attached to an ASG will get assigned to the launched EC2 instances
* ASG can terminate instances marked as unhealthy by an ELB (and hence replace them)

Even if an ASG is deployed across 3 AZs, minimum number of instances to remain highly available is still 2

If you have an ASG with running instances and you delete the ASG, the instances will be terminated and the ASG will be deleted.

## Scaling Policies

* **Scheduled Scaling**
  + Scale based on a schedule
  + Used when the load pattern is predictable
* **Simple Scaling**
  + Scale to certain size on a CloudWatch alarm
  + Ex. when CPU > 90%, then scale to 10 instances
* **Step Scaling**
  + Scale incrementally in steps using CloudWatch alarms
  + Ex. when CPU > 70%, then add 2 units and when CPU < 30%, then remove 1 unit
  + Specify the **instance warmup time** to scale faster
* **Target Tracking Scaling**
  + ASG maintains a CloudWatch metric and scale accordingly (automatically creates CW alarms)
  + Ex. maintain CPU usage at 40%
* **Predictive Scaling**
  + Historical data is used to predict the load pattern using ML and scale automatically

## Launch Configuration & Launch Template

* Defines the following info for ASG
  + AMI (Instance Type)
  + EC2 User Data
  + EBS Volumes
  + Security Groups
  + SSH Key Pair
  + Min / Max / Desired Capacity
  + Subnets (where the instances will be created)
  + Load Balancer (specify which ELB to attach instances)
  + Scaling Policy
* **Launch Configuration** (legacy)
  + Cannot be updated (must be re-created)
  + **Does not support Spot Instances**
* **Launch Template** (newer)
  + Versioned
  + Can be updated
  + **Supports both On-Demand and Spot Instances**
  + Recommended by AWS

## Scaling Cooldown

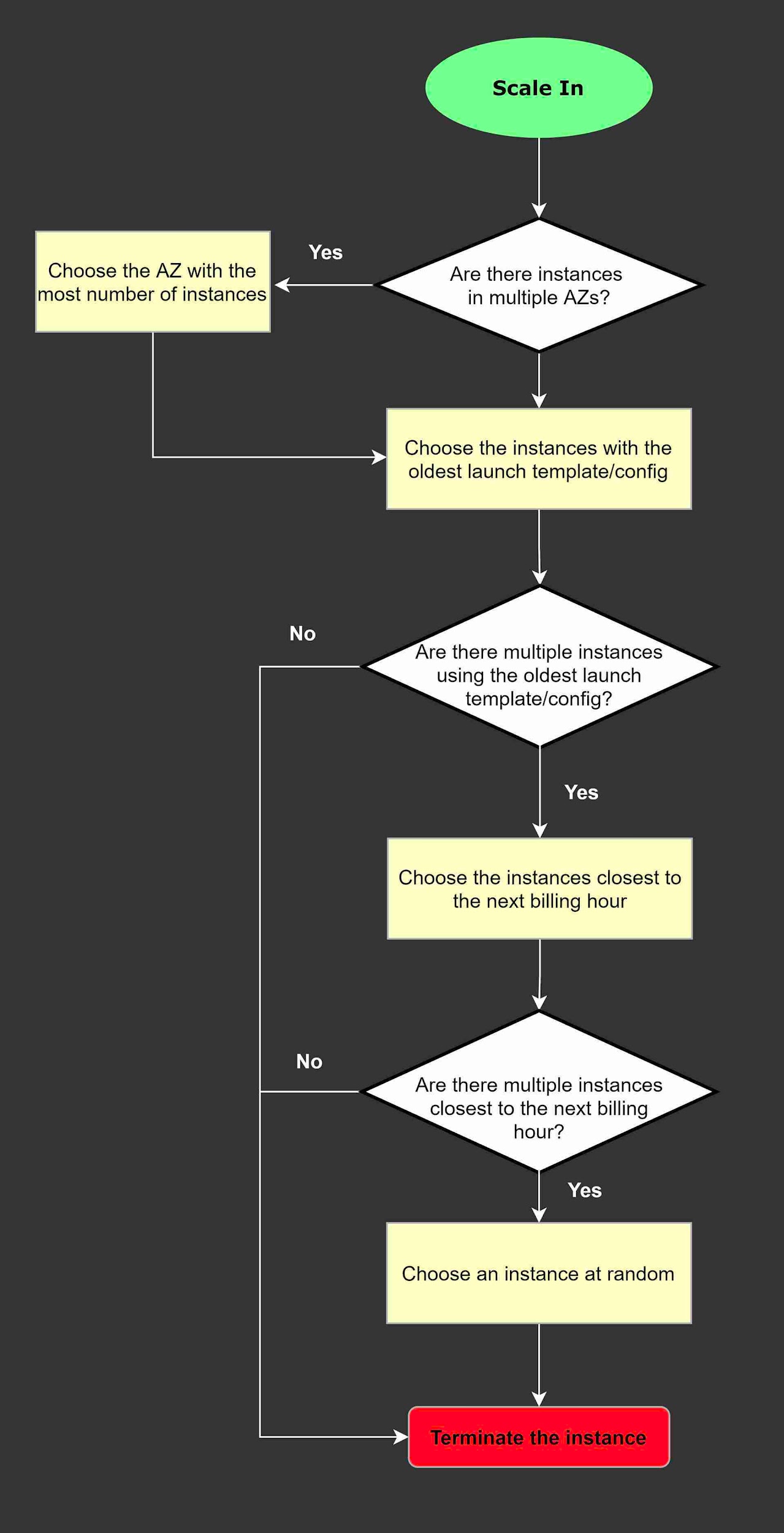
* After a scaling activity happens, the ASG goes into cooldown period (default 300 seconds) during which it does not launch or terminate additional instances (ignores scaling requests) to allow the metrics to stabilize.
* Use a ready-to-use AMI to launch instances faster to be able to reduce the cooldown period

## Health Checks

* By default, ASG uses the EC2 status check (not the ELB health check). This could explain why some instances that are labelled as unhealthy by an ELB are still not terminated by the ASG.
* To prevent ASG from replacing unhealthy instances, suspend the **ReplaceUnhealthy** process type

ASG creates a new scaling activity for terminating the unhealthy instance and then terminates it. Later, another scaling activity launches a new instance to replace the terminated instance.

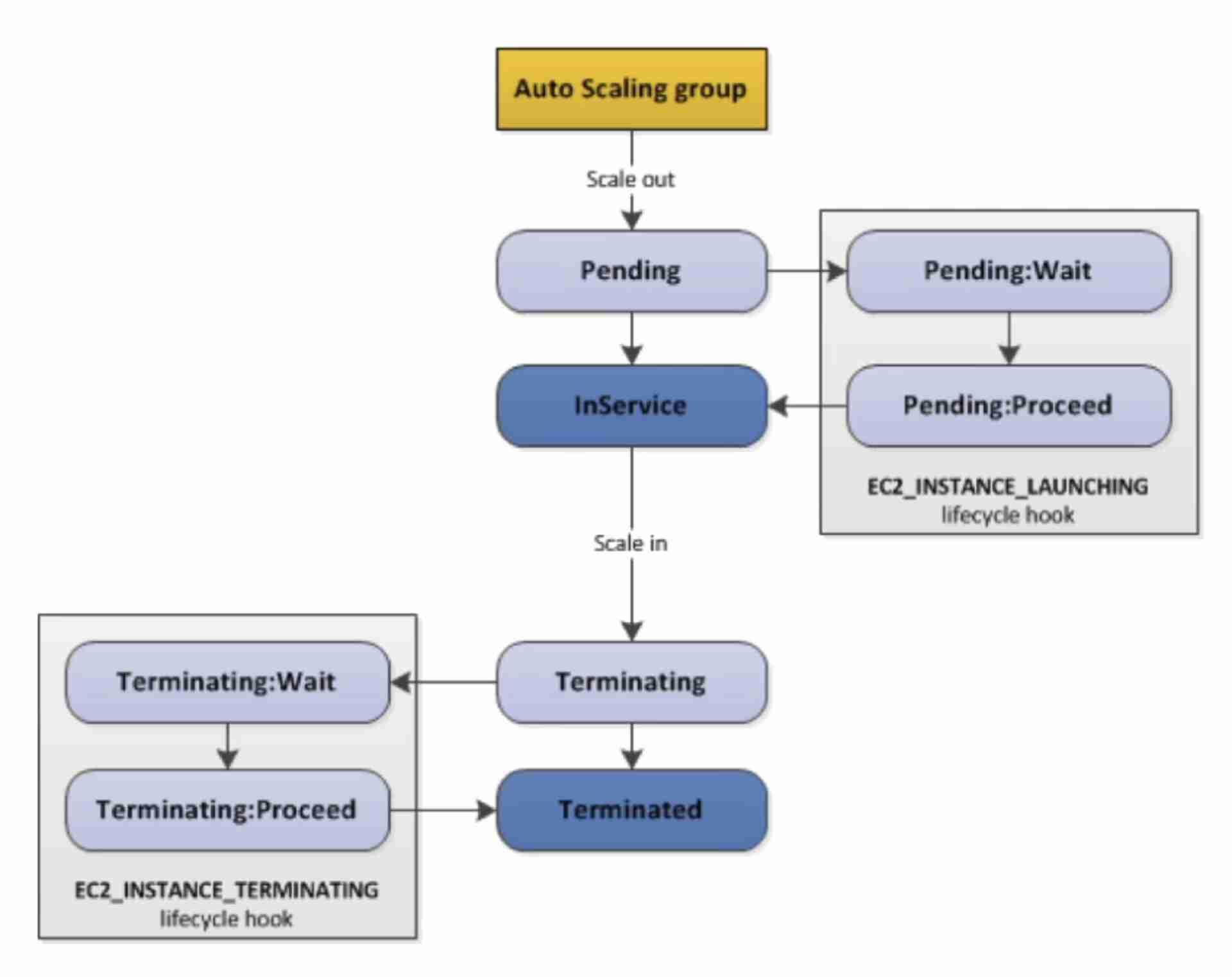
## Termination Policy

* Select the AZ with the most number of instances
* Delete the instance with the oldest launch configuration
* Delete the instance which is closest to the next billing hour
* Flow diagram
  + 
* ASG does not immediately terminate instances with an **Impaired** status, it waits a few minutes for the instance to recover.
* ASG doesn't terminate an instance that came into service based on EC2 status checks and ELB health checks until the **health check grace period** expires.

## Rebalancing AZs

* ASG ensures that the group never goes below the minimum scale. Actions such as changing the AZ for the group or explicitly terminating or detaching instances can lead to the ASG becoming unbalanced between AZs. In such cases, ASG compensates by **rebalancing** the AZs by **launching new instances before terminating the old ones**, so that rebalancing does not compromise the performance or availability of the application.

## Lifecycle Hooks

* Used to perform extra steps before creating or terminating an instance. Example:
  + Install some extra software or do some checks (during pending state) before declaring the instance as "in service"
  + Before the instance is terminated (terminating state), extract the log files
* **Without lifecycle hooks, pending and terminating states are avoided**
* Flow diagram
  + 

## Attach running instances to an existing ASG

* The running instance must meet the following criteria:
  + The AMI used to launch the instance still exists
  + The instance is not a member of another ASG
  + The instance is launched into one of the AZ defined in your ASG

# Identity & Access Management (IAM)

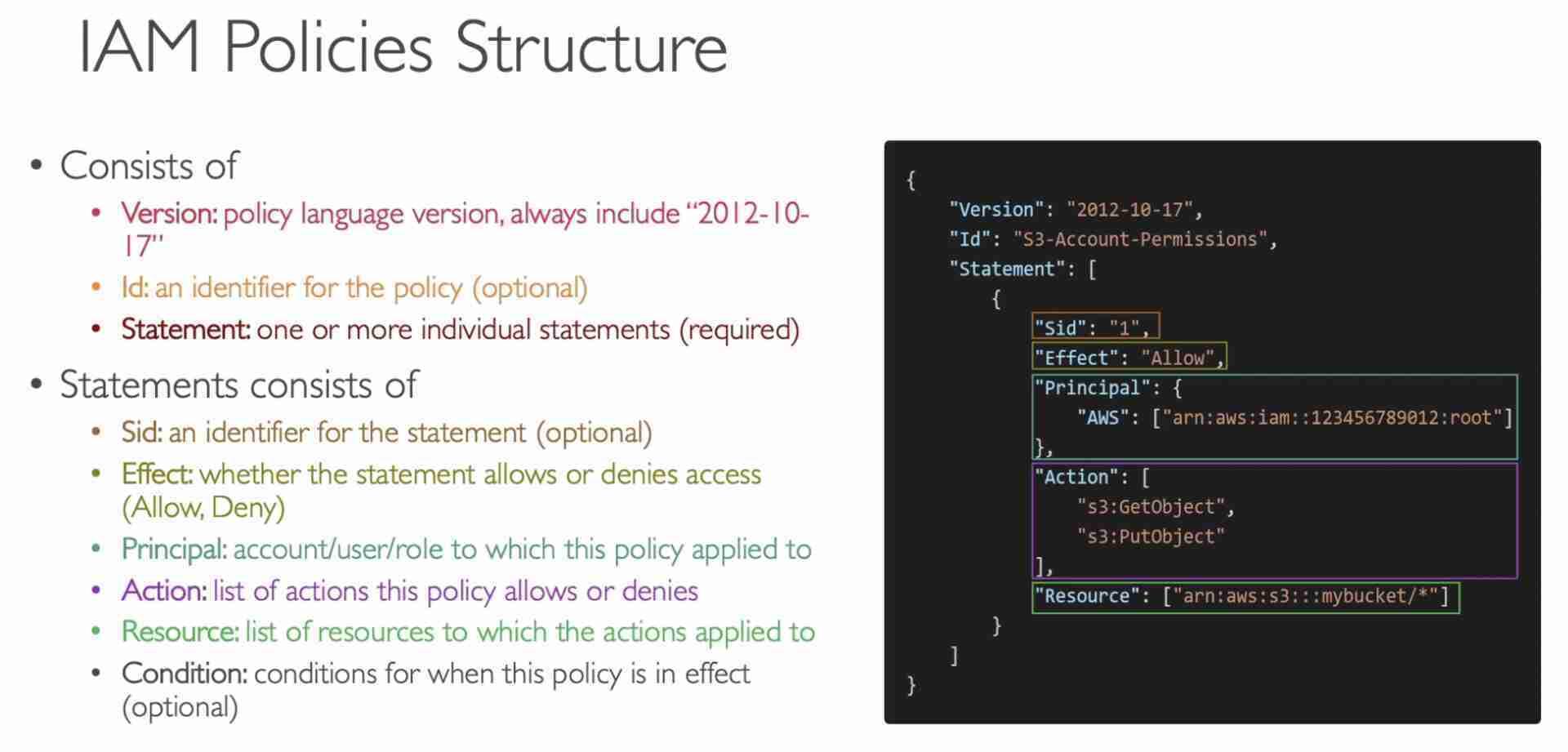
* Global Service (IAM entities like roles can be used in any region without recreation)

**IAM Query API** can be used to make direct calls to the IAM web service (using access key ID and secret access key for authentication)

## Users & Groups

* Groups are collections of users and have policies attached to them
* Groups cannot be nested
* User can belong to multiple groups
* User doesn't have to belong to a group
* **Root User** has full access to the account
* **IAM User** has limited permission to the account
* You should log in as an IAM user with **admin access** even if you have root access. This is just to be sure that nothing goes wrong by accident.
* An IAM Group is not an identity and cannot be identified as a principal in an IAM policy
* Only users and services can assume a role (not groups)
* A new IAM user created using the AWS CLI or AWS API has no AWS credentials

## Policies

* Policies are JSON documents that outline permissions for users, groups or roles
* Two types
  + **User based policies**
    - IAM policies define which API calls should be allowed for a specific user
  + **Resource based policies**
    - Control access to an AWS resource
    - Grant the specified principal permission to perform actions on the resource and define under what conditions this applies
* An IAM principal can access a resource if the user policy ALLOWS it OR the resource policy ALLOWS it AND there’s no explicit DENY.
* Policies assigned to a user are called inline policies
* Follow **least privilege principle** for IAM Policies
* Policy Structure
  + 

#### **Trust Policies**

#### Defines which principal entities (accounts, users, roles, federated users) can assume the role

* An IAM role is both an identity and a resource that supports resource-based policies.
* You must attach both a trust policy and an identity-based policy to an IAM role.
* The **IAM service supports only one type of resource-based policy** called a **role trust policy**, which is **attached to an IAM role**.

## Roles

* Collection of policies for AWS services

If you are going to use an IAM Service Role with Amazon EC2 or another AWS service that uses Amazon EC2, you must store the role in an **instance profile**. When you create an IAM service role for EC2, the role automatically has EC2 identified as a trusted entity.

## Protect IAM Accounts

* **Password Policy**
  + Used to enforce standards for password
    - password rotation
    - password reuse
  + Prevents **brute force** attack
* **Multi Factor Authentication (MFA)**
  + Both root user and IAM users should use MFA

## Reporting Tools

* **Credentials Report**
  + lists all the users and the status of their credentials (MFA, password rotation, etc.)
  + **account level** - used to audit security for all the users
* **Access Advisor**
  + shows the service permissions granted to a user and when those services were last accessed
  + **user-level**
  + used to revise policies for a specific user

## Access Keys

* Need to use access keys for AWS CLI and SDK
* Don't share access keys with anyone (every user can generate their own access keys)
* Access keys are only shown once and if you lose them you need to generate a new access key
* Access Key ID ~ username
* Secret Access Key ~ password

## Guidelines

* Use root account only for account setup
* 1 physical user = 1 IAM user
* Enforce MFA for both root and IAM users
* Never share lAM credentials & Access Keys

## Policy Simulator

* Online tool that allows us to check what API calls an IAM User, Group or Role is allowed to perform based on the permissions they have.

## Permission Boundaries

* Set the maximum permissions an IAM entity can get
* **Can be applied to users and roles (not groups)**
* Used to ensure some users can’t escalate their privileges (make themselves admin)

## Assume Role vs Resource-based Policy

* When you assume an IAM Role, you give up your original permissions and take the permissions assigned to the role
* When using a resource based policy, the principal doesn’t have to give up their permissions

## Course

[AWS Certified Solutions Architect Associate (SAA-C02) | Udemy](https://www.udemy.com/course/aws-certified-solutions-architect-associate-saa-c02/)

## Certification

* AWS Solutions Architect Associate (SAA-C02)
* Certificate Validity: 3 years
* Grading: 100-1000 (passing score = 720) - 72% correct required to pass
* Pass / Fail Exam (score doesn’t appear on the certificate)
* 65 questions - 130 minutes (2h 10min)
* Can request extra 30 mins on exam if you are a non-native English speaker