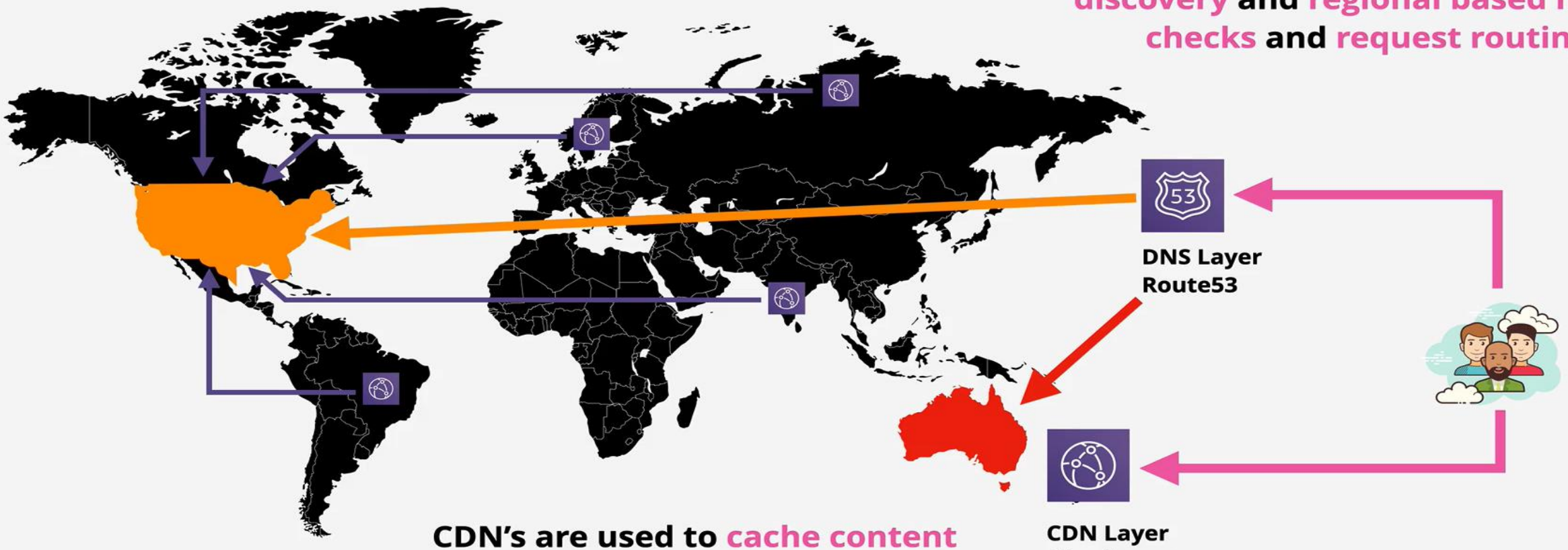
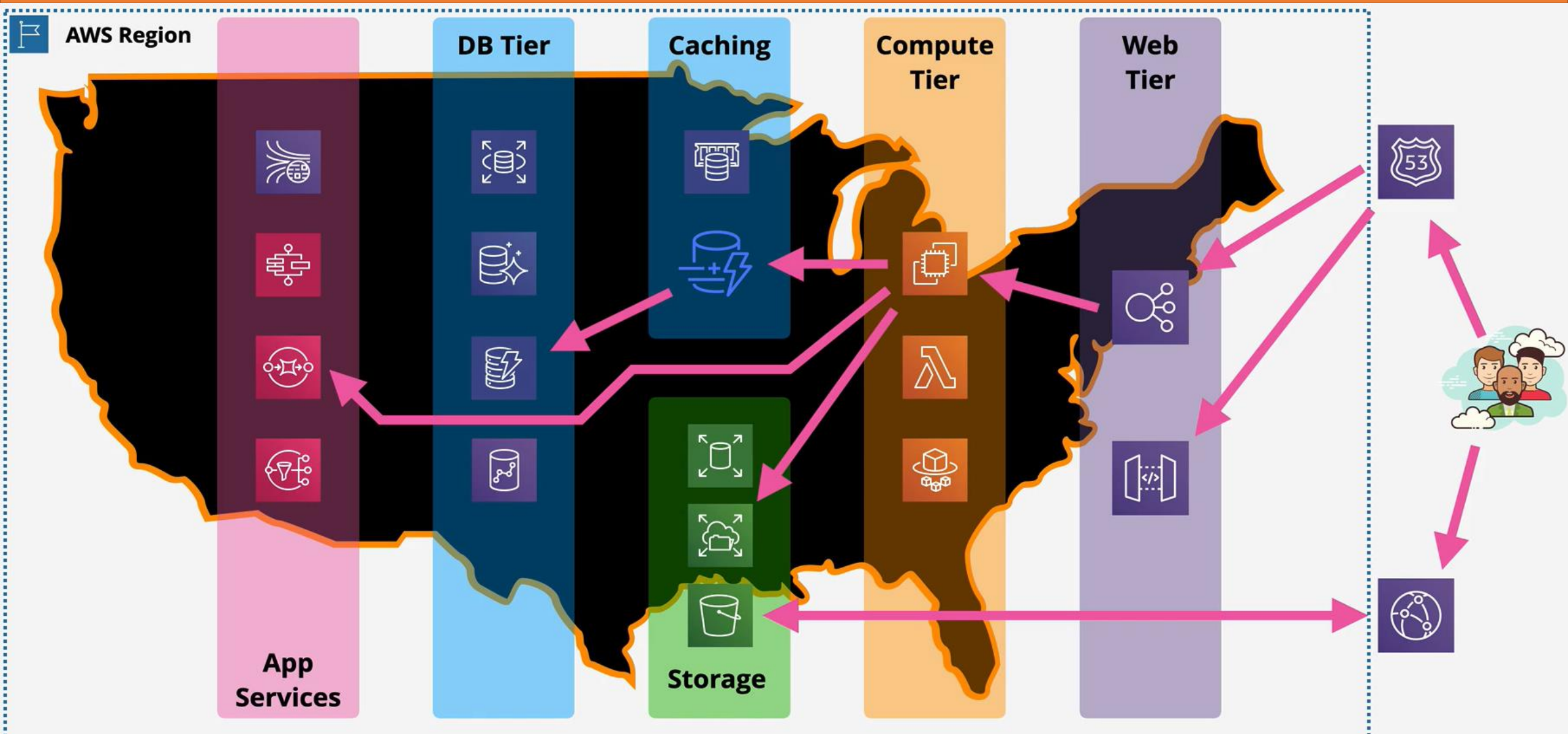
Regional and Global AWS Architecture

- Global Service Location & Discovery
- Content Delivery (CDN) and optimization
- Global health checks & Failover
- Regional entry point
- Scaling & Resilience
- Application services and components

Regional and Global AWS Architecture



Regional and Global AWS Architecture



Regional and Global AWS Architecture

- The traffic is entering one specific region of the AWS infrastructure
- Depending the architecture, this might be entering into a VPC or using public space AWS Services.
- Let’s think of our architecture in a regional sense.
- Most services runs in a region and those regions makes up AWS.
- Initially communication from your customers will generally enter at the Web Tier
- Generally, this will be an AWS service such an ALB or API Gateway, depending on the architecture that the application uses.
- The purpose of the Web Tier is to act as an entering point for your regional based applications or application components.

Regional and Global AWS Architecture

- It abstracts your customers away from the underlying infrastructure. It means that the infrastructure behind it can scale or fail or change without impacting customers.
- The functionality provided to the customers via the Web Tier is provided by the Compute Tier using services such as EC2, Lambda or Containers which use the Elastic Container Service.
- In the example provided in the architecture , the Load Balancer will use EC2 to provide compute services through to our customers.
- The Compute Tier will consume Storage services, another part of all AWS architecture.

Regional and Global AWS Architecture

- The Storage Tier will use services such as EBS(Elastic Block Store), EFS (Elastic File System) and even S3 for things like media storage.
- Many global architecture utilize CloudFront(the Global Content Delivery Network) within AWS
- CloudFront can use S3 as an origin for media.
- Netflix might store movies and TV shows on S3, and this will be cached by CloudFront.
- CloudFront can directly fetch content from S3 for delivery to global audience

Evolution of Elastic Load Balancers (ELB)

What is Elastic Load Balancing?

- Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones.
- It monitors the health of its registered targets, and routes traffic only to the healthy targets.
- Elastic Load Balancing scales your load balancer as your incoming traffic changes over time. It can automatically scale to the vast majority of workloads.

Evolution of Elastic Load Balancers (ELB)

Load balancer benefits

- A load balancer distributes workloads across multiple compute resources, such as virtual servers. Using a load balancer increases the availability and fault tolerance of your applications.
- You can add and remove compute resources from your load balancer as your needs change, without disrupting the overall flow of requests to your applications.
- You can configure health checks, which monitor the health of the compute resources, so that the load balancer sends requests only to the healthy ones.
- You can also offload the work of encryption and decryption to your load balancer so that your compute resources can focus on their main work.

Evolution of Elastic Load Balancers (ELB)

- A load balancer accepts incoming traffic from clients and routes requests to its registered targets (such as EC2 instances) in one or more Availability Zones.
- The load balancer also monitors the health of its registered targets and ensures that it routes traffic only to healthy targets. When the load balancer detects an unhealthy target, it stops routing traffic to that target.
- It then resumes routing traffic to that target when it detects that the target is healthy again.

Evolution of Elastic Load Balancers (ELB)

- You configure your load balancer to accept incoming traffic by specifying one or more *listeners*.
- A listener is a process that checks for connection requests. It is configured with a protocol and port number for connections from clients to the load balancer.
- Likewise, it is configured with a protocol and port number for connections from the load balancer to the targets.

Evolution of Elastic Load Balancers (ELB)

- 4 Types of Load balancers **(ELB)** available within AWS
- Split between **v1** (**avoid/migrate**) and **v2** (**prefer**)
- Classic Load Balancers (**CLB**) –**v1** – Introduced in 2009
- Not really layer 7, lacking features, **1 SSL per CLB**
- Application Load Balancers (**ALB**) – v2 – HTTP/S/WebSocket
- Network Load Balancers (**NLB**) –v2 – TCP, TLS &UDP
- Gateway Load Balancers
- V2 = faster, cheaper, support target groups and rules (which allows you to use a single load balancer for multiple things) or handles the load balancing different based on which customer is using it.

Evolution of Elastic Load Balancers (ELB)

- There is a key difference in how the load balancer types are configured.
- With Application Load Balancers, Network Load Balancers, and Gateway Load Balancers, you register targets in target groups, and route traffic to the target groups.
- With Classic Load Balancers, you register instances with the load balancer.

Accessing Elastic Load Balancing

You can create, access, and manage your load balancers using any of the following interfaces:

- **AWS Management Console**— Provides a web interface that you can use to access Elastic Load Balancing.
- **AWS Command Line Interface (AWS CLI)** — Provides commands for a broad set of AWS services, including Elastic Load Balancing. The AWS CLI is supported on Windows, macOS, and Linux.

ELB Related Services

Elastic Load Balancing works with the following services to improve the availability and scalability of your applications.

- **Amazon EC2** — Virtual servers that run your applications in the cloud. You can configure your load balancer to route traffic to your EC2 instances.
- **Amazon EC2 Auto Scaling** — Ensures that you are running your desired number of instances, even if an instance fails.
- Amazon EC2 Auto Scaling also enables you to automatically increase or decrease the number of instances as the demand on your instances changes.
- If you enable Auto Scaling with Elastic Load Balancing, instances that are launched by Auto Scaling are automatically registered with the load balancer. Likewise, instances that are terminated by Auto Scaling are automatically de-registered from the load balancer

ELB Related Services

- **AWS Certificate Manager** — When you create an HTTPS listener, you can specify certificates provided by ACM. The load balancer uses certificates to terminate connections and decrypt requests from clients.
- **Amazon CloudWatch** — Enables you to monitor your load balancer and to take action as needed.
- **Amazon ECS** — Enables you to run, stop, and manage Docker containers on a cluster of EC2 instances. You can configure your load balancer to route traffic to your containers.
- **AWS Global Accelerator** — Improves the availability and performance of your application. Use an accelerator to distribute traffic across multiple load balancers in one or more AWS Regions.

ELB Related Services

- **Route 53** — Provides a reliable and cost-effective way to route visitors to websites by translating domain names into the numeric IP addresses that computers use to connect to each other.
- For example, it would translate www.example.com into the numeric IP address 192.0.2.1.
- AWS assigns URLs to your resources, such as load balancers. However, you might want a URL that is easy for users to remember. For example, you can map your domain name to a load balancer.
- **AWS WAF** — You can use AWS WAF with your Application Load Balancer to allow or block requests based on the rules in a web access control list (web ACL)

Availability Zones and Load Balancer nodes

- When you enable an Availability Zone for your load balancer, Elastic Load Balancing creates a load balancer node in the Availability Zone.
- If you register targets in an Availability Zone but do not enable the Availability Zone, these registered targets do not receive traffic.
- Your load balancer is most effective when you ensure that each enabled Availability Zone has at least one registered target.

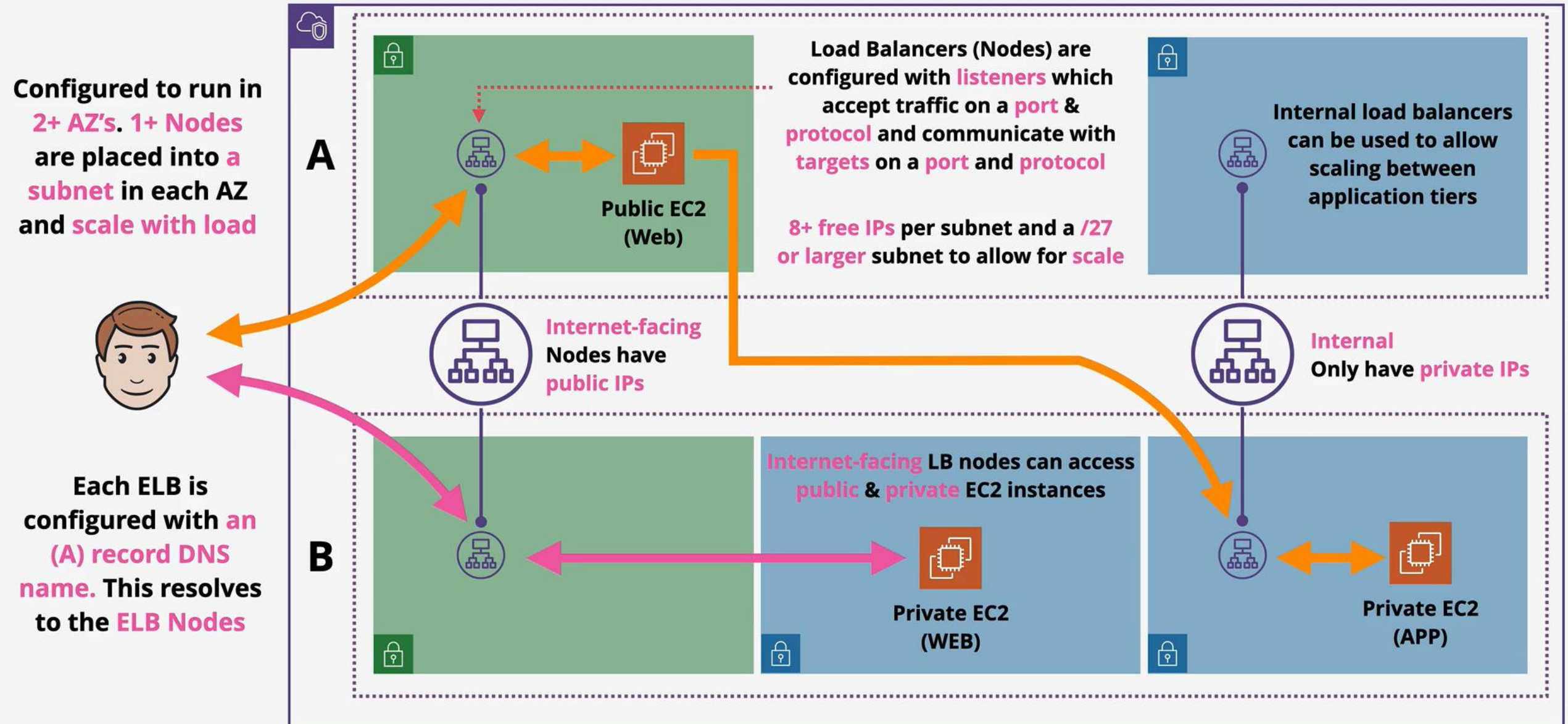
Availability Zones and Load Balancer nodes

- AWS recommend enabling multiple Availability Zones for all load balancers.
- With an Application Load Balancer however, it is a requirement that you enable at least two or more Availability Zones. This configuration helps ensure that the load balancer can continue to route traffic.
- If one Availability Zone becomes unavailable or has no healthy targets, the load balancer can route traffic to the healthy targets in another Availability Zone.
- After you disable an Availability Zone, the targets in that Availability Zone remain registered with the load balancer. However, even though they remain registered, the load balancer does not route traffic to them.

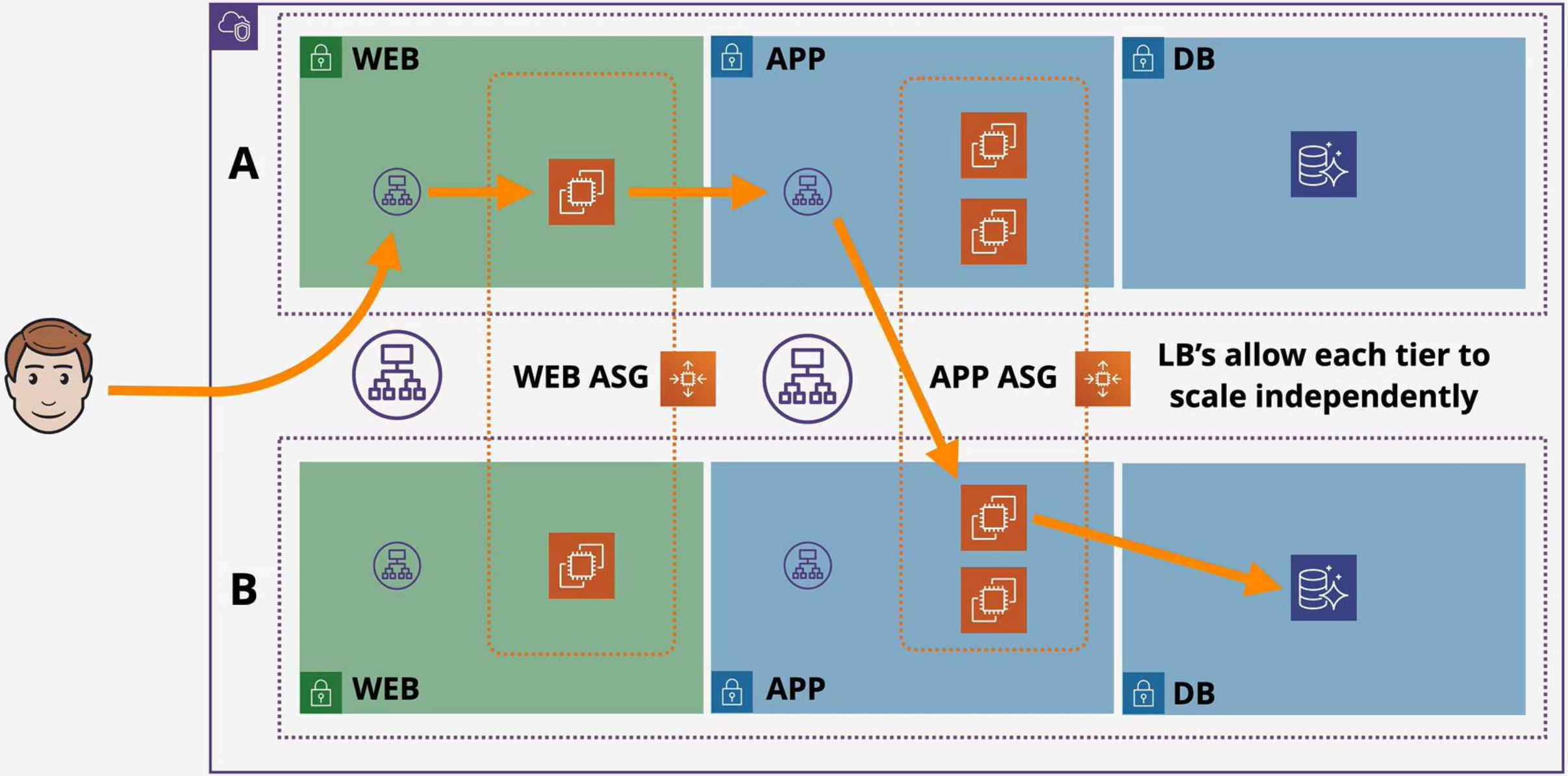
Elastic Load Balancer Architecture (ELB)

- It is the job of a Load Balancer to accept connections from customers and then to distribute those connections across any registered backend compute.
- It means the user is abstracted away from the physical infrastructure. It means the amount of infrastructure can change to increase or decrease in number without affecting customers.

ELB Architecture



ELB Architecture

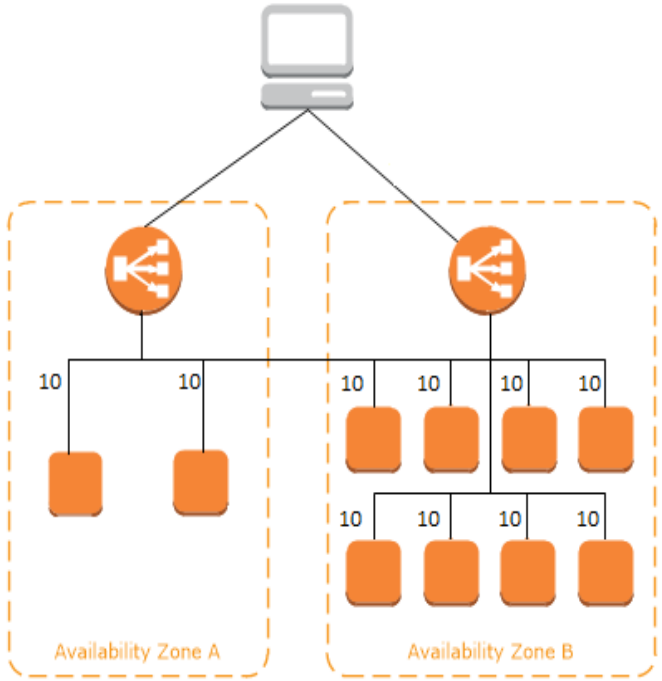


Cross-zone Load Balancing

- The nodes for your load balancer distribute requests from clients to registered targets.
- When cross-zone load balancing is enabled, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones.
- When cross-zone load balancing is disabled, each load balancer node distributes traffic only across the registered targets in its Availability Zone.

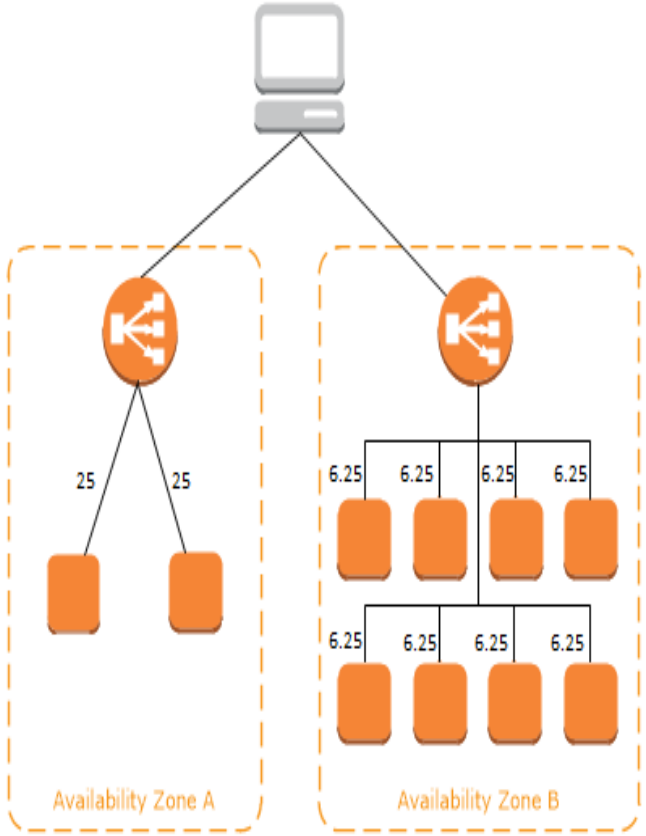
Cross-zone Load Balancing

- The following diagrams demonstrate the effect of cross-zone load balancing.
- There are two enabled Availability Zones, with two targets in Availability Zone A and eight targets in Availability Zone B. Clients send requests, and Amazon Route 53 responds to each request with the IP address of one of the load balancer nodes.
- This distributes traffic such that each load balancer node receives 50% of the traffic from the clients. Each load balancer node distributes its share of the traffic across the registered targets in its scope.
- If cross-zone load balancing is enabled, each of the 10 targets receives 10% of the traffic. This is because each load balancer node can route its 50% of the client traffic to all 10 targets.

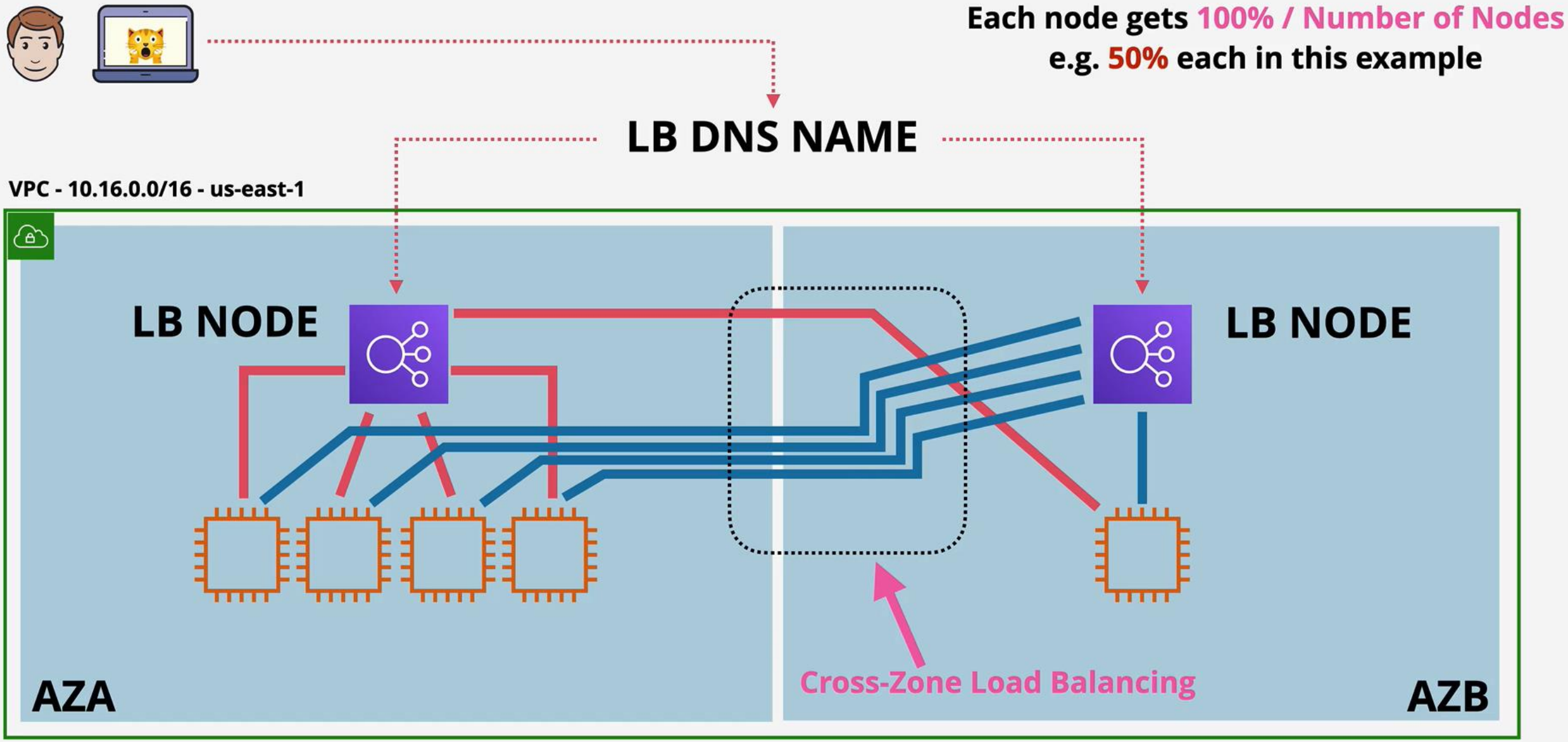


Cross-zone Load Balancing

- If cross-zone load balancing is disabled:
  - Each of the two targets in Availability Zone A receives 25% of the traffic.
  - Each of the eight targets in Availability Zone B receives 6.25% of the traffic.
- This is because each load balancer node can route its 50% the client traffic only to targets in its Availability Zone.
- With Application Load Balancers, Cross-zone load balancing is always enabled.
- With Network Load Balancers and Gateway Load Balancers cross-zone load balancing is disabled by default. After you create the load balancer, you can enable or disable cross-zone load balancing at any time.



Cross-zone Load Balancing



ELB Architecture

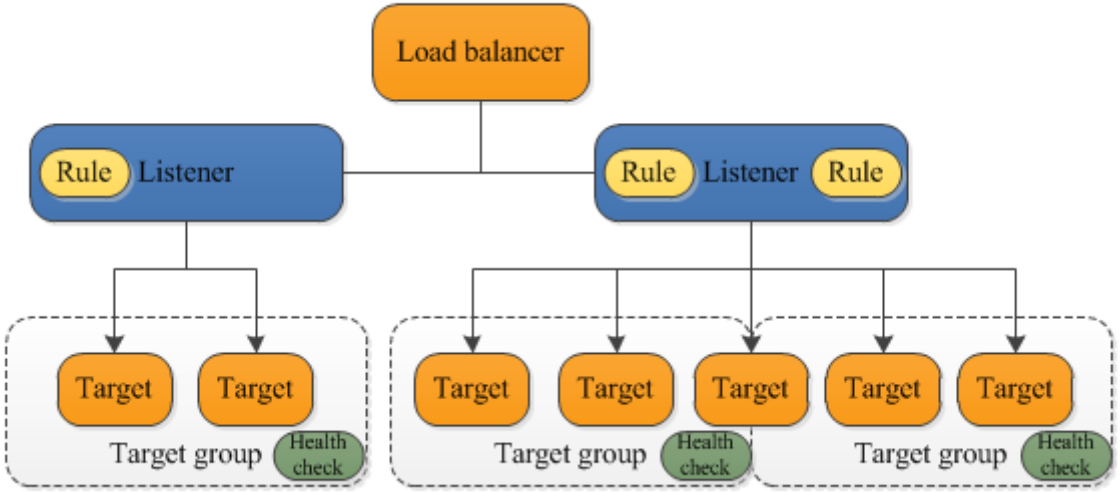
- ELB is a **DNA A** Record pointing at **1+** Nodes per AZ
- Nodes (in one subnet per AZ) can scale
- Internet-facing means nodes have public **IPv4 Ips**
- **Internal** is **private only Ips**
- EC2 **doesn’t need to be public** to work with a LB
- **Listener** Configuration controls **WHAT** the LB does
- **8+** Free Ips per subnet, and a **/27** subnet to allow scaling

Application Load Balancer components

- A ***load balancer*** serves as the single point of contact for clients. The load balancer distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones. This increases the availability of your application. You add one or more listeners to your load balancer.
- A ***listener*** checks for connection requests from clients, using the protocol and port that you configure.
- The rules that you define for a listener determine how the load balancer routes requests to its registered targets. Each rule consists of a priority, one or more actions, and one or more conditions.
- When the conditions for a rule are met, then its actions are performed. **You must define a default rule for each listener, and you can optionally define additional rules.**

Application Load Balancer components

- Each *target group* routes requests to one or more registered targets, such as EC2 instances, using the protocol and port number that you specify.
- You can register a target with multiple target groups. You can configure health checks on a per target group basis. Health checks are performed on all targets registered to a target group that is specified in a listener rule for your load balancer.



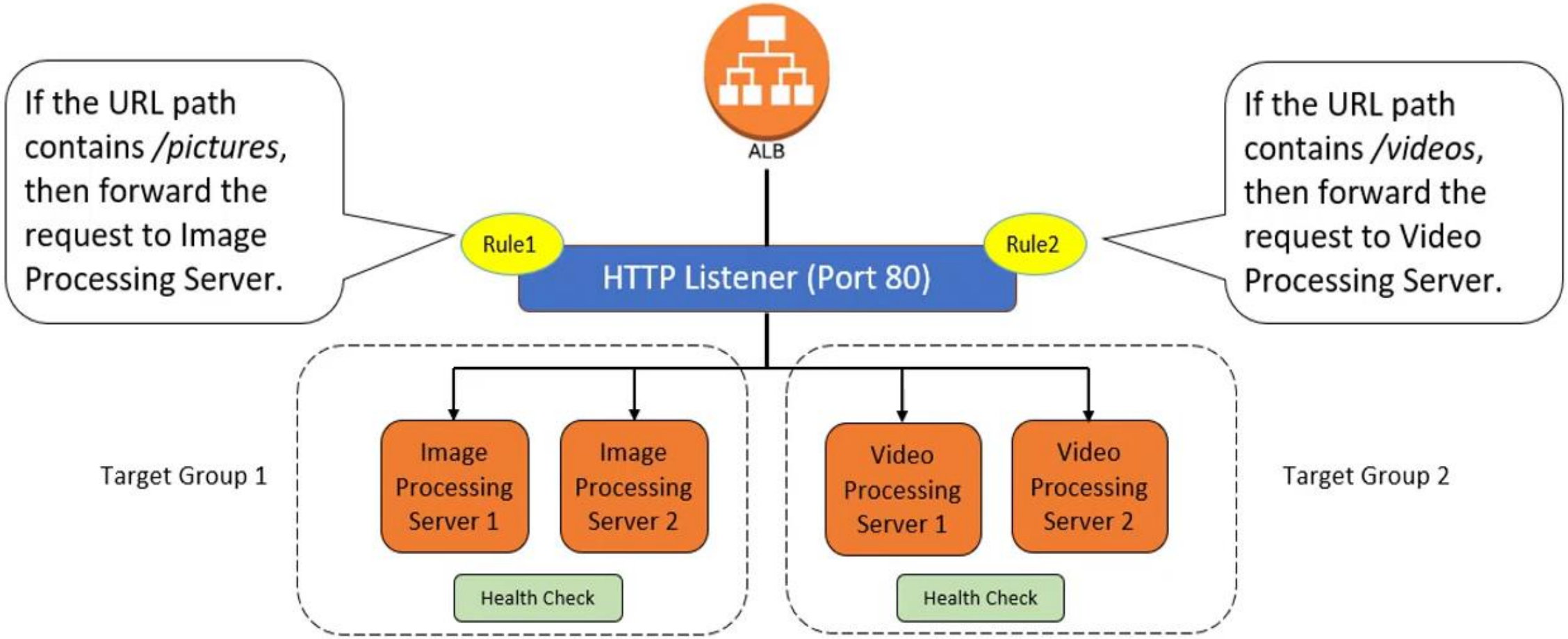
Application Load Balancer (ALB)

- An Application Load Balancer functions at the application layer, the seventh layer of the Open Systems Interconnection (OSI) model.
- After the load balancer receives a request, it evaluates the listener rules in priority order to determine which rule to apply, and then selects a target from the target group for the rule action.
- You can configure listener rules to route requests to different target groups based on the content of the application traffic.
- Routing is performed independently for each target group, even when a target is registered with multiple target groups. You can configure the routing algorithm used at the target group level. The default routing algorithm is round robin; alternatively, you can specify the least outstanding requests routing algorithm.

Application Load Balancer (ALB)

- You can add and remove targets from your load balancer as your needs change, without disrupting the overall flow of requests to your application.
- Elastic Load Balancing scales your load balancer as traffic to your application changes over time. Elastic Load Balancing can scale to the vast majority of workloads automatically.
- You can configure health checks, which are used to monitor the health of the registered targets so that the load balancer can send requests only to the healthy targets.

Application Load Balancer (ALB) - Rules



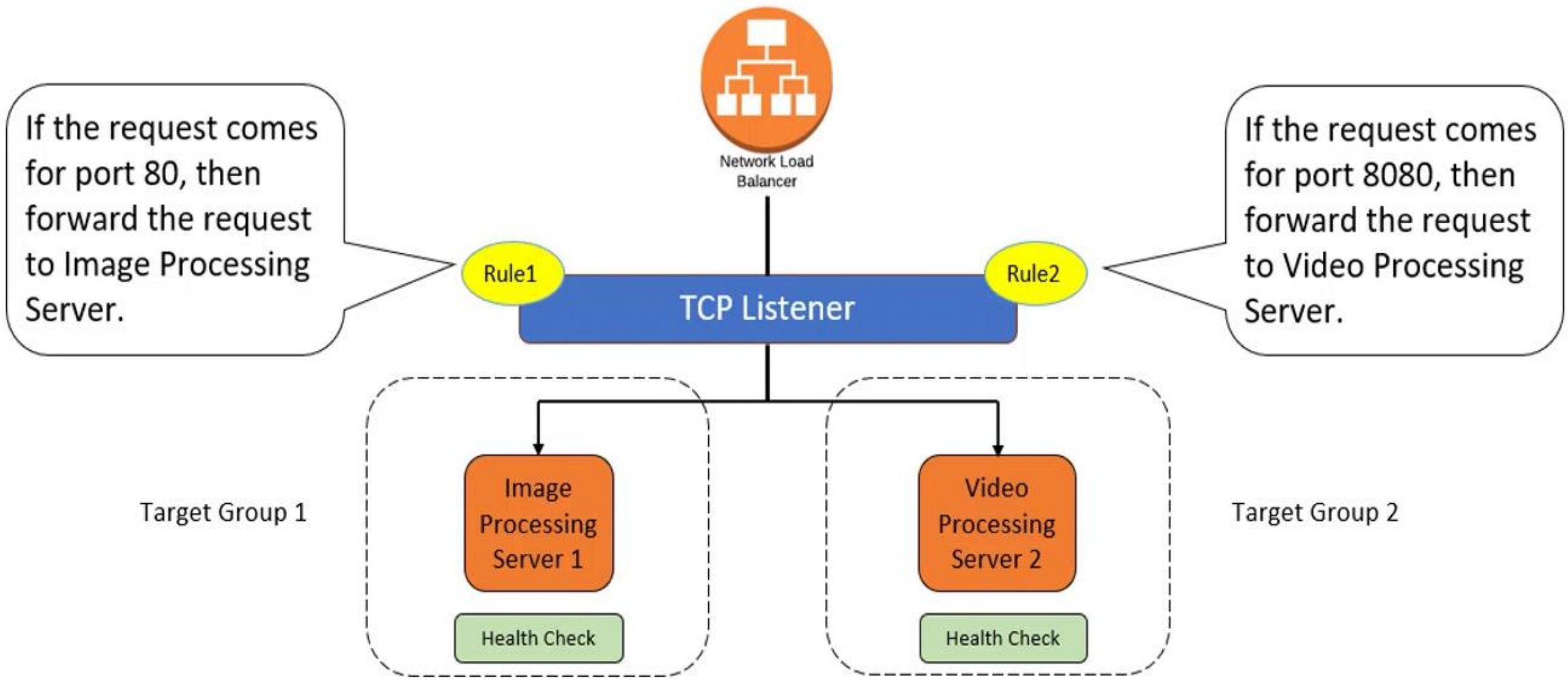
Network Load Balancer (NLB)

- A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. It can handle millions of requests per second.
- After the load balancer receives a connection request, it selects a target from the target group for the default rule.
- It attempts to open a TCP connection to the selected target on the port specified in the listener configuration.

Network Load Balancer (NLB)

- Functions at the **Transport Layer** - Layer 4 (Protocols TCP, TLS and UDP)
- For **high performance** use cases (millions of requests per second)
- Can be assigned a **Static IP/Elastic IP** – useful for whitelisting
- Routes connection based on IP protocol data (layer 4)
- Supports UDP and static IP addresses as targets
- Used with private link to provide services to other VPCs
- Routes connection based on IP protocol data (layer 4)
- Can load balance between:
  - EC2 instances
  - Containerized applications (Amazon ECS)
  - Web applications (using IP addresses)

Network Load Balancer (NLB)



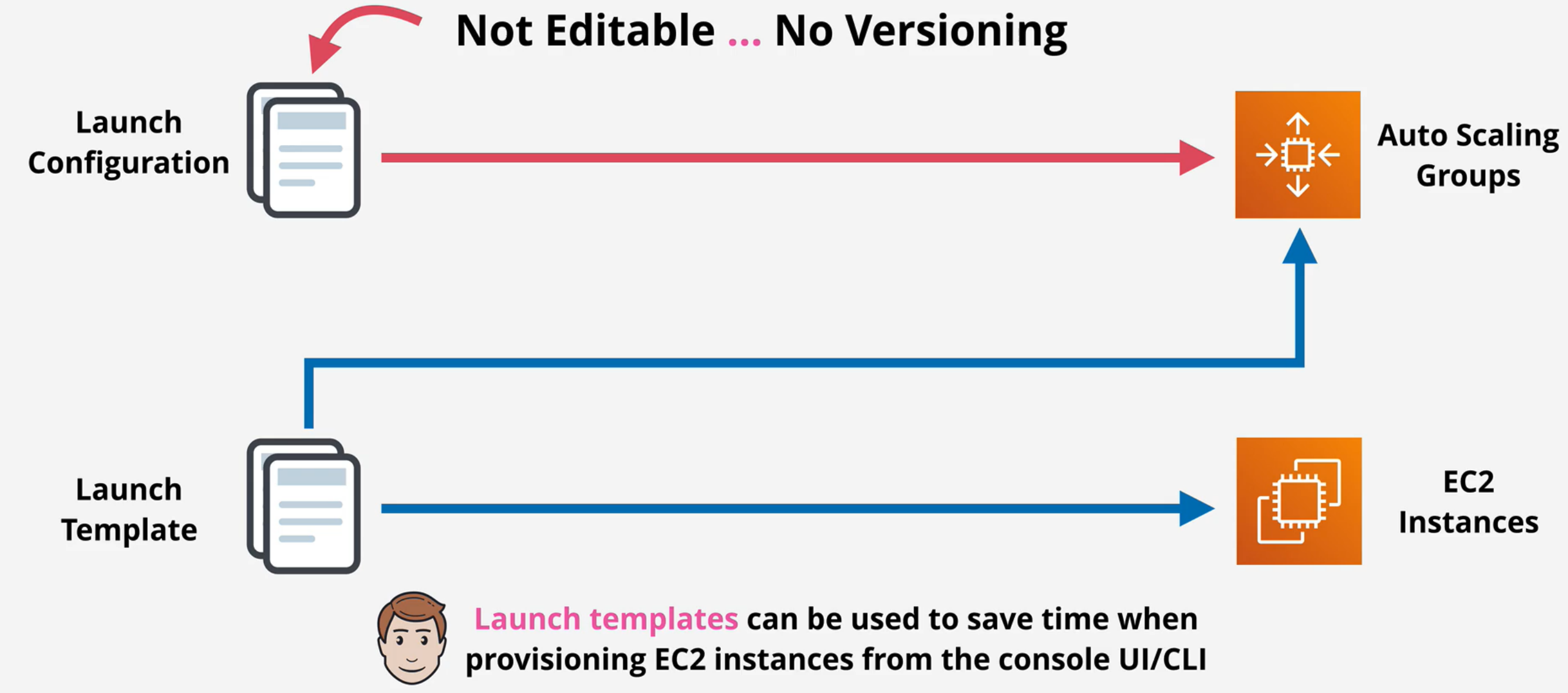
ALB vs NLB

- Unbroken encryption… NLB
- Static IP for whitelisting… NLB
- The fastest performance… NLB (millions rps)
- Protocols not HTTP or HTTPS… NLB
- Privatelink… NLB

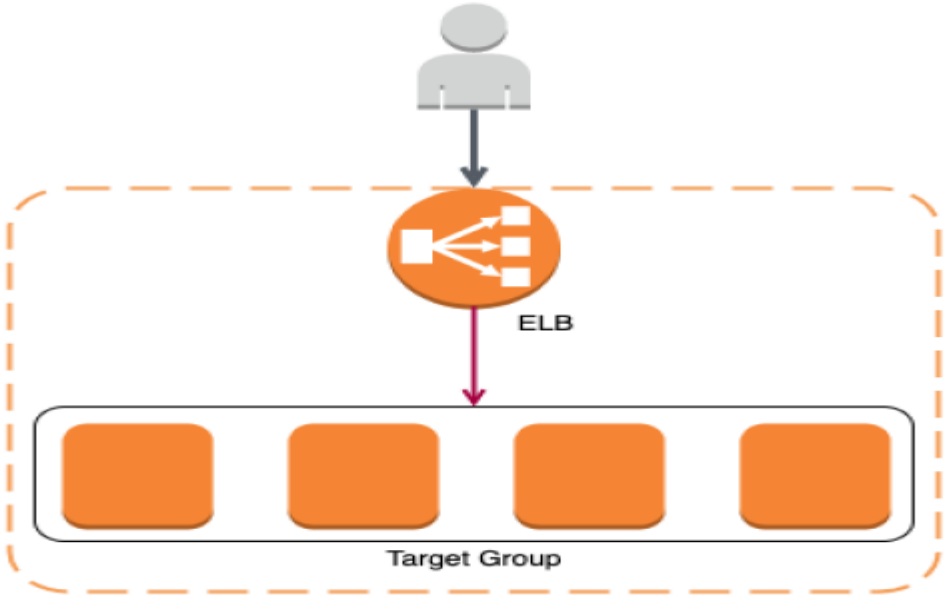
Launch Configuration and Launch Template Key Concepts

- Allow you to define the configuration of an EC2 instance in advance
- AMI, Instance Type, Storage & Key Pair
- Networking and Security Groups
- Userdata & IAM Role
- Both are NOT editable – defined once. Launch Template has versions
- Launch Template provide newer features – including T2/T3 Unlimited, Placement Groups, Capacity Reservations, Elastic Graphics.

Launch Configuration and Launch Template Key Concepts



Introduction to Auto Scaling Groups



- Target Groups are configured with a static set of instances. How do you scale out and scale in **automatically?**
  - Configure an Auto Scaling Group

Auto Scaling Groups

- An *Auto Scaling group* contains a collection of Amazon EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management.
- An Auto Scaling group also enables you to use Amazon EC2 Auto Scaling features such as health check replacements and scaling policies.
- Both maintaining the number of instances in an Auto Scaling group and automatic scaling are the core functionality of the Amazon EC2 Auto Scaling service.

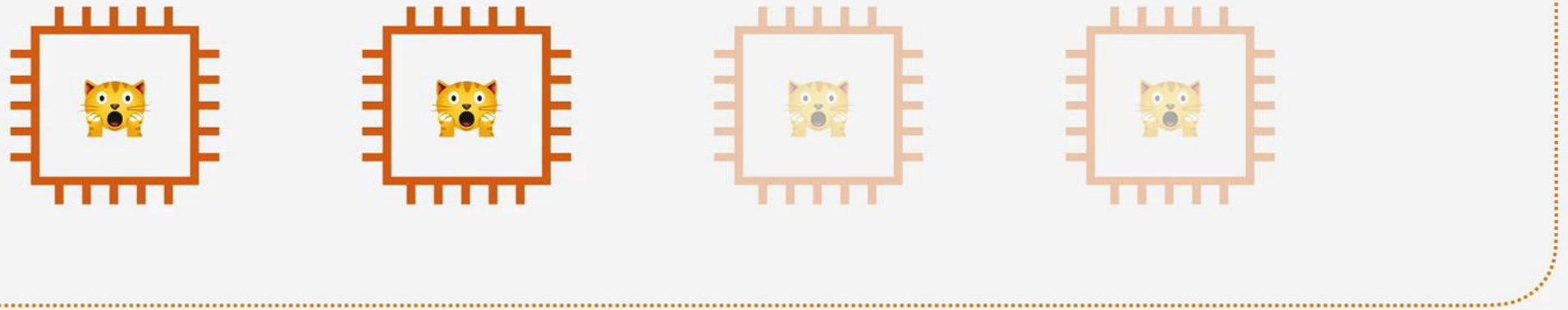
Auto Scaling Groups

- Automatic Scaling and Self-healing for EC2
- Uses Launch Templates or Configurations
- Has a Minimum, Desired and Maximum Size (e.g.; 1:2:4)
- Keep running instances at the Desired capacity by provisioning or terminating instances
- Scaling Policies automate based on Metrics

Auto Scaling Groups

Launch Template provides EC2 Config

Auto Scaling Group



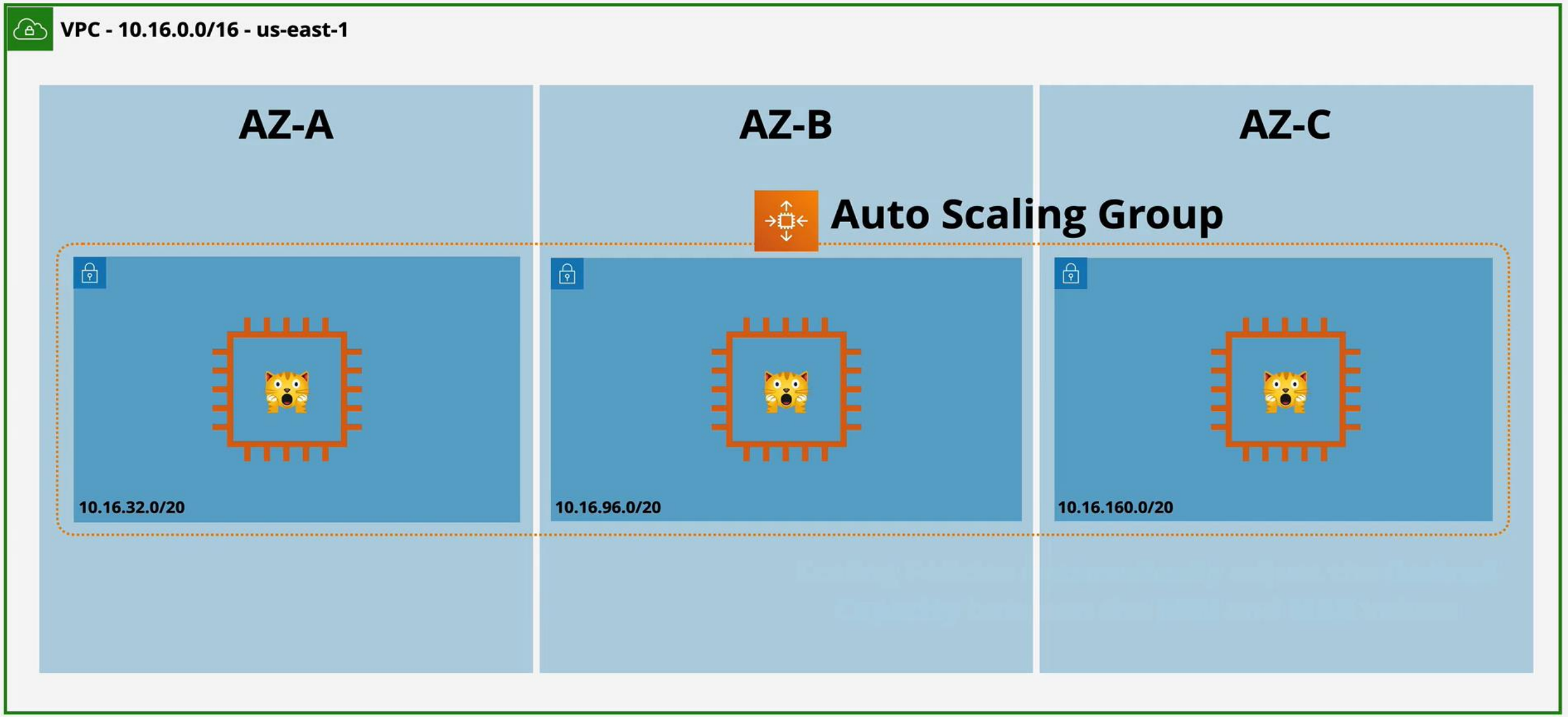
Minimum Size (1)

Desired Capacity (2)

Maximum Size (4)

Scaling Policies automatically adjust the Desired Capacity between the MIN and MAX values

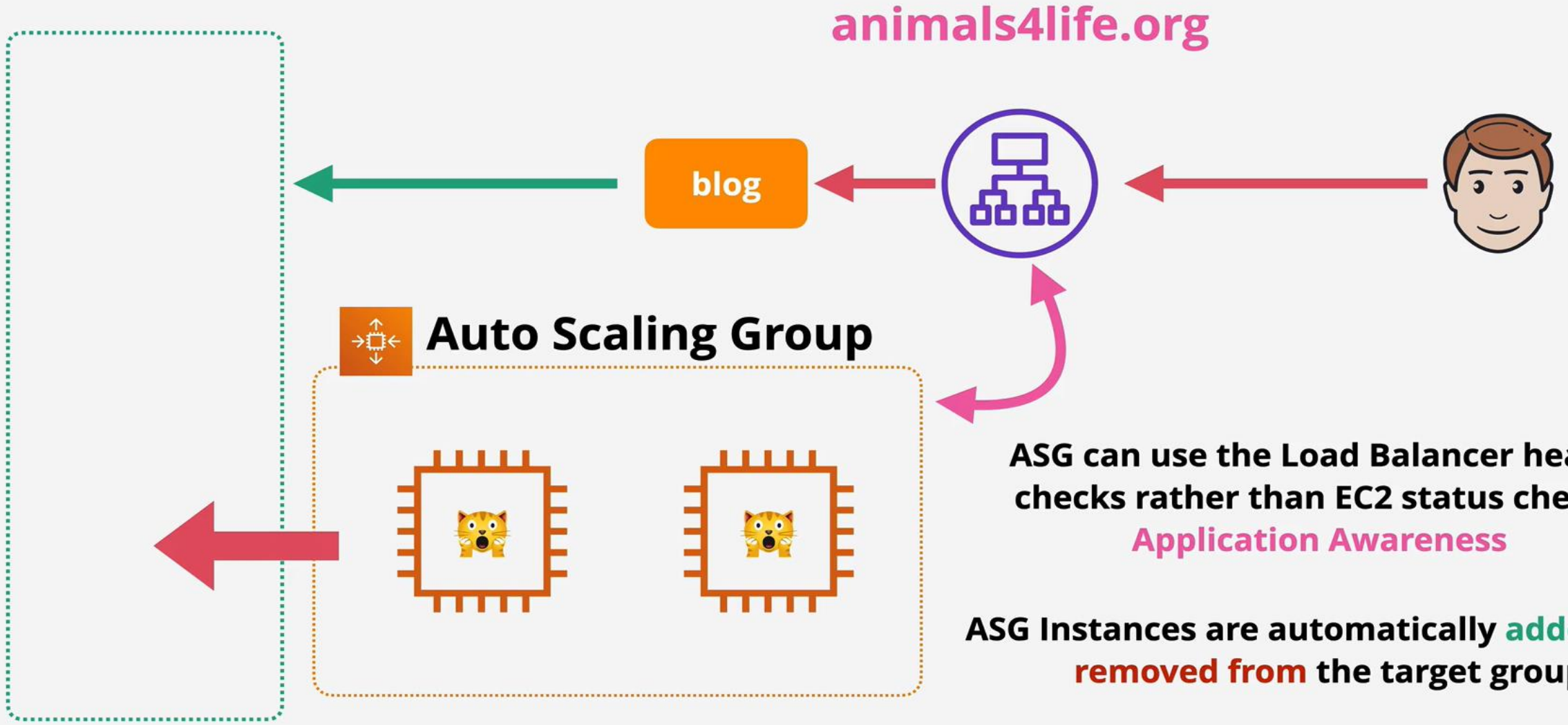
Auto Scaling Groups Architecture



Scaling Policies

- Manual Scaling – Manually adjust the desired capacity
- Scheduled Scaling – Time based adjustment – e.g., sales..
- Dynamic Scaling
  - Simple – “CPU above 50% +1”, “CPU below 50 -1”
  - Stepped Scaling – Bigger +/- based on difference
  - Target Tracking – Desired Aggregate CPU = 40%... ASG handle it
  - Cooldown Periods…

ASG + Load Balancers



Scaling Processes

- **Launch and Terminate** – SUSPEND and RESUME
- **AddToLoadBalancer –** add to LB on Launch
- **AlarmNotification** – accept notification from CW
- **AZRebalance** – Balances instances evenly across all of the Azs
- **HealthCheck** – instance health checks on/off
- **ReplaceUnhealthy** – Terminate unhealthy and replace
- **ScheduledActions** – Scheduled on/off
- **Standby** – use this for instances ‘InService vs Standby’

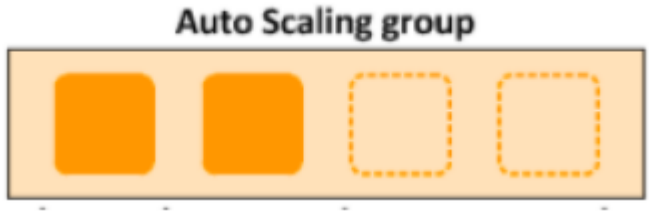
Auto Scaling Components

- Autoscaling Groups are free
- Only the resources created are billed…
- Use cool downs to avoid rapid scaling
- Think about **more, smaller** instances – **granularity**
- Use with ALB's for elasticity – **abstraction**
- ASG defines **WHEN** and **WHERE**, LT defines **WHAT**

Auto Scaling New Rules

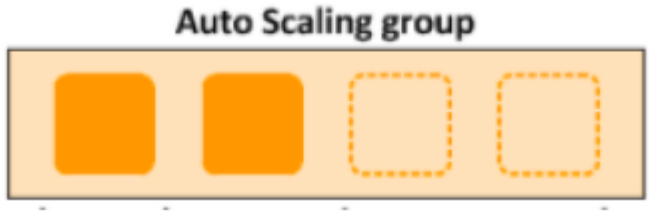
- It is now possible to define ”better” auto scaling rules that are directly managed by EC2
  - Target Average CPU Usage
  - Number of requests on the ELB per instance
  - Average Network In
  - Average Network Out
- These rules are easier to set up and can make more sense

Auto Scaling Group – Use Cases



ASG Use case	Description	More details
Maintain current instance levels at all times	min = max = desired = CONSTANT When an instance becomes unhealthy, it is replaced.	Constant load
Scale manually	Change desired capacity as needed	You need complete control over scaling
Scale based on a schedule	Schedule a date and time for scaling up and down.	Batch programs with regular schedules
Scale based on demand (Dynamic/Automatic Scaling)	Create scaling policy (what to monitor?) and scaling action (what action?)	Unpredictable load

Auto Scaling Policy Types

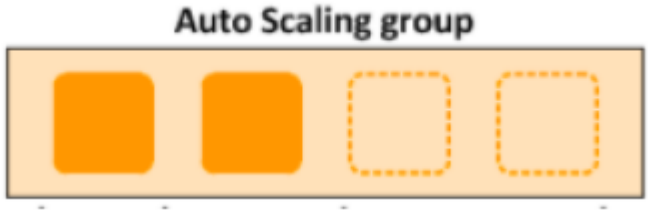


Scaling Policy	Examples	Description
Target tracking scaling	Maintain CPU Utilization at 70%.	Modify current capacity based on a target value for a specific metric.
Simple scaling	+5 if CPU utilization > 80% -3 if CPU utilization < 60%	Waits for cooldown period before triggering additional actions.
Step scaling	+1 if CPU utilization between 70% and 80% +3 if CPU utilization between 80% and 100% Similar settings for scale down	Warm up time can be configured for each instance

Scaling Policies – Background

- Two parts:
  - CloudWatch alarm (Is CPU utilization >80%? or < 60%).
  - Scaling action (+5 EC2 instances or -3 EC2 instances)

Auto Scaling - Scenarios



Scenario	Solution
Change instance type or size of ASG instances	Launch configuration or Launch template cannot be edited. Create a new version and ensure that the ASG is using the new version. Terminate instances in small groups.
Roll out a new security patch (new AMI) to ASG instances	Same as above
Perform actions before an instance is added or removed	Create a Lifecycle Hook. You can configure CloudWatch to trigger actions based on it.

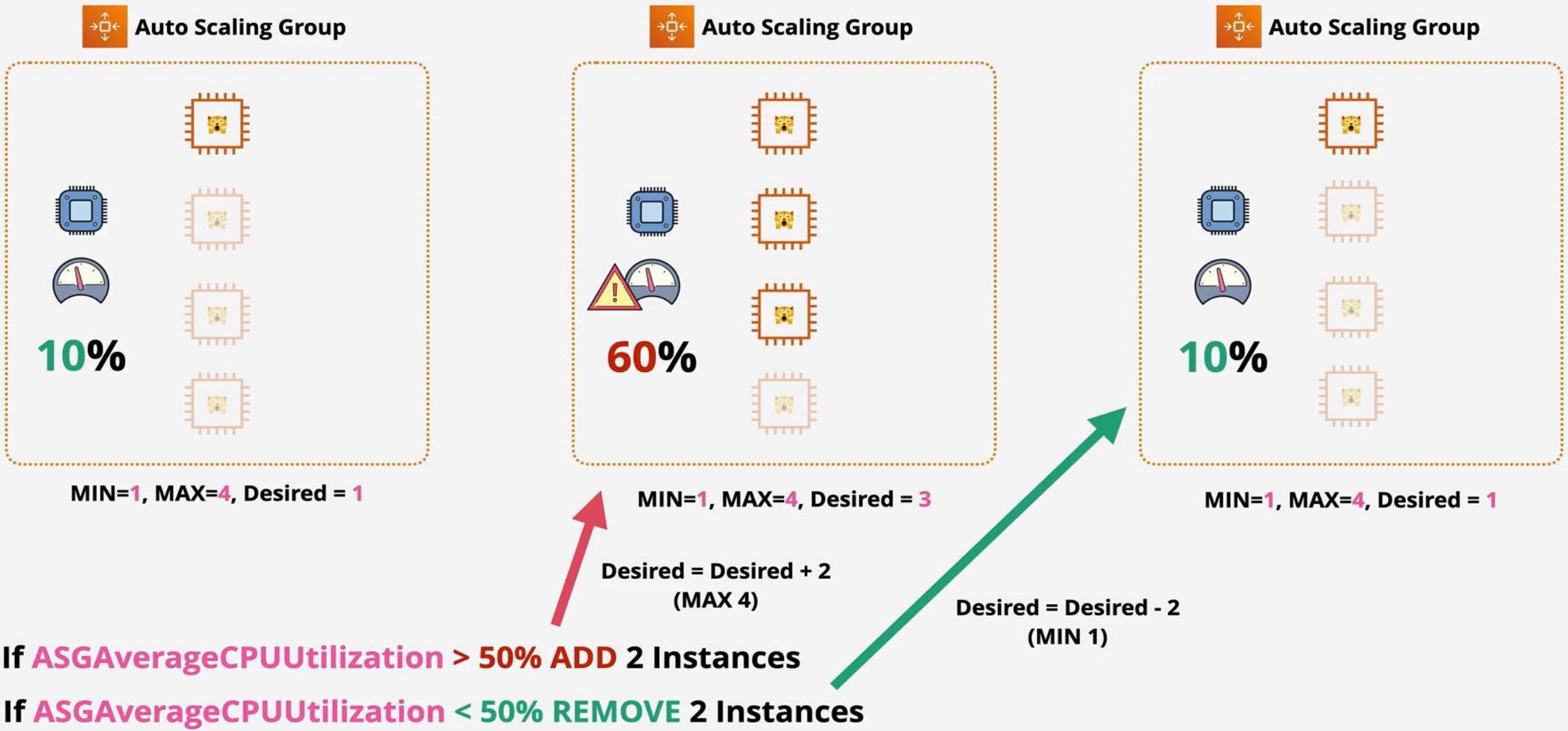
Auto Scaling - Scenarios

Scenario	Solution
Which instance in an ASG is terminated first when a scale-in happens?	**(Default Termination Policy**) Within constraints, goal is to distribute instances evenly across available AZs. Next priority is to terminate older instances.
Preventing frequent scale up and down	Adjust cooldown period to suit your need (default - 300 seconds). Align CloudWatch monitoring interval
I would want to protect newly launched instances from scale-in	Enable instance scale-in protection

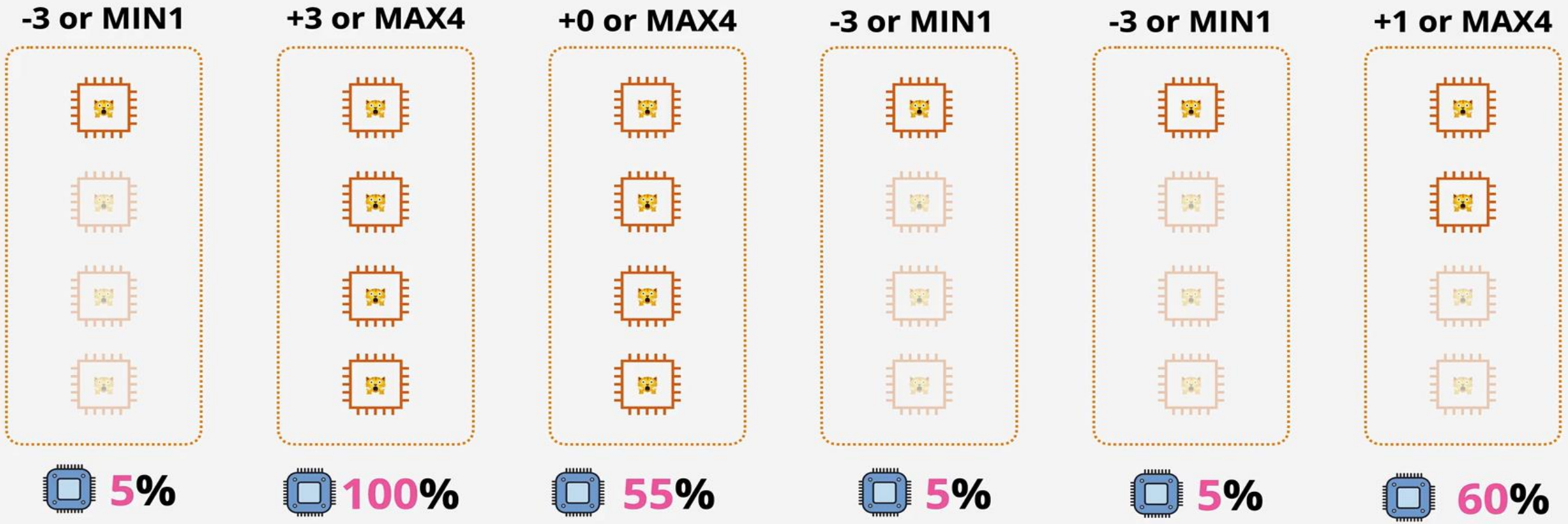
Auto Scaling Groups – Dynamic Scaling Policies

- **Target Tracking Scaling**
  - Most simple and easy to set-up
  - Example: I want the average ASG CPU to stay at around 40%
- **Simple / Step Scaling**
  - When a CloudWatch alarm is triggered (example CPU > 70%), then add 2 units
  - When a CloudWatch alarm is triggered (example CPU < 30%), then remove 1
- **Scheduled Actions**
  - Anticipate a scaling based on known usage patterns
  - Example: increase the min capacity to 10 at 5 pm on Fridays

Auto Scaling Groups – Simple Scaling



Auto Scaling Groups – Step Scaling



ASG Lifecycle Hooks

- Lifecycle hooks enable you to perform custom actions by pausing instances as an Auto Scaling group launches or terminates them.
- When an instance is paused, it remains in a wait state either until you complete the lifecycle action using the complete-lifecycle-action command or the **CompleteLifecycleAction** operation, or until the timeout period ends (one hour by default).

ASG Lifecycle Hooks

Auto Scaling Group

PENDING: Wait → **PENDING: Proceed**

PENDING ..... **InService**

e.g. Load or Index data

Scale Out

Scale In

Terminating ..... **Terminated**

e.g. Backup data or logs

State changes when timeout (3600s default) expires or CompleteLifecycleAction

Terminating: Wait → **Terminating: Proceed**

Notifications for lifecycle hooks can be sent to an SNS topic

Eventbridge can be used to initiate other processes based on Hooks

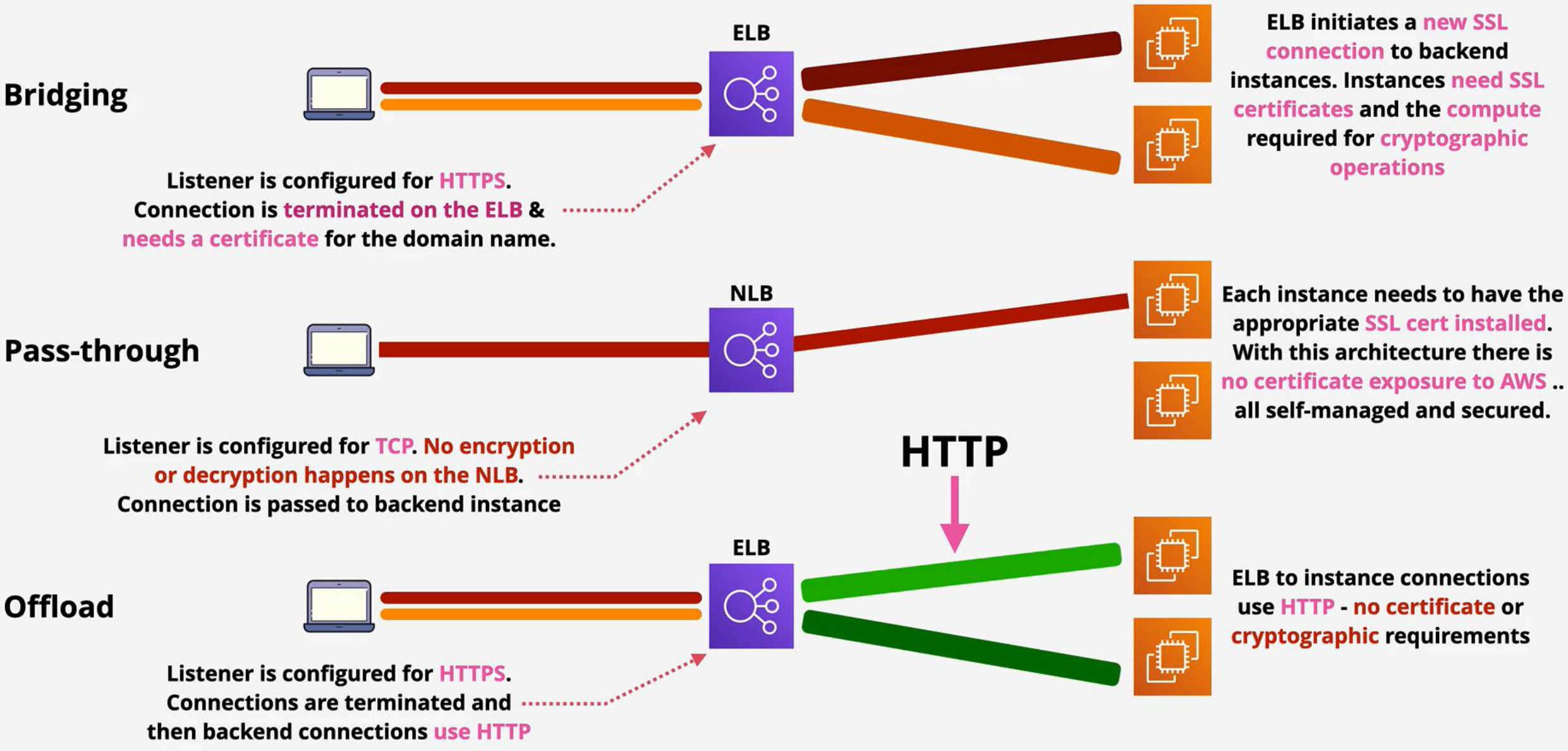
ASG Health Checks

- Amazon EC2 Auto Scaling can determine the health status of an instance using one or more of the following:
  - Status checks provided by Amazon EC2 to identify hardware and software issues that may impair an instance. The default health checks for an Auto Scaling group are EC2 status checks only.
  - Health checks provided by Elastic Load Balancing (ELB). These health checks are disabled by default but can be enabled.
  - Your custom health checks.

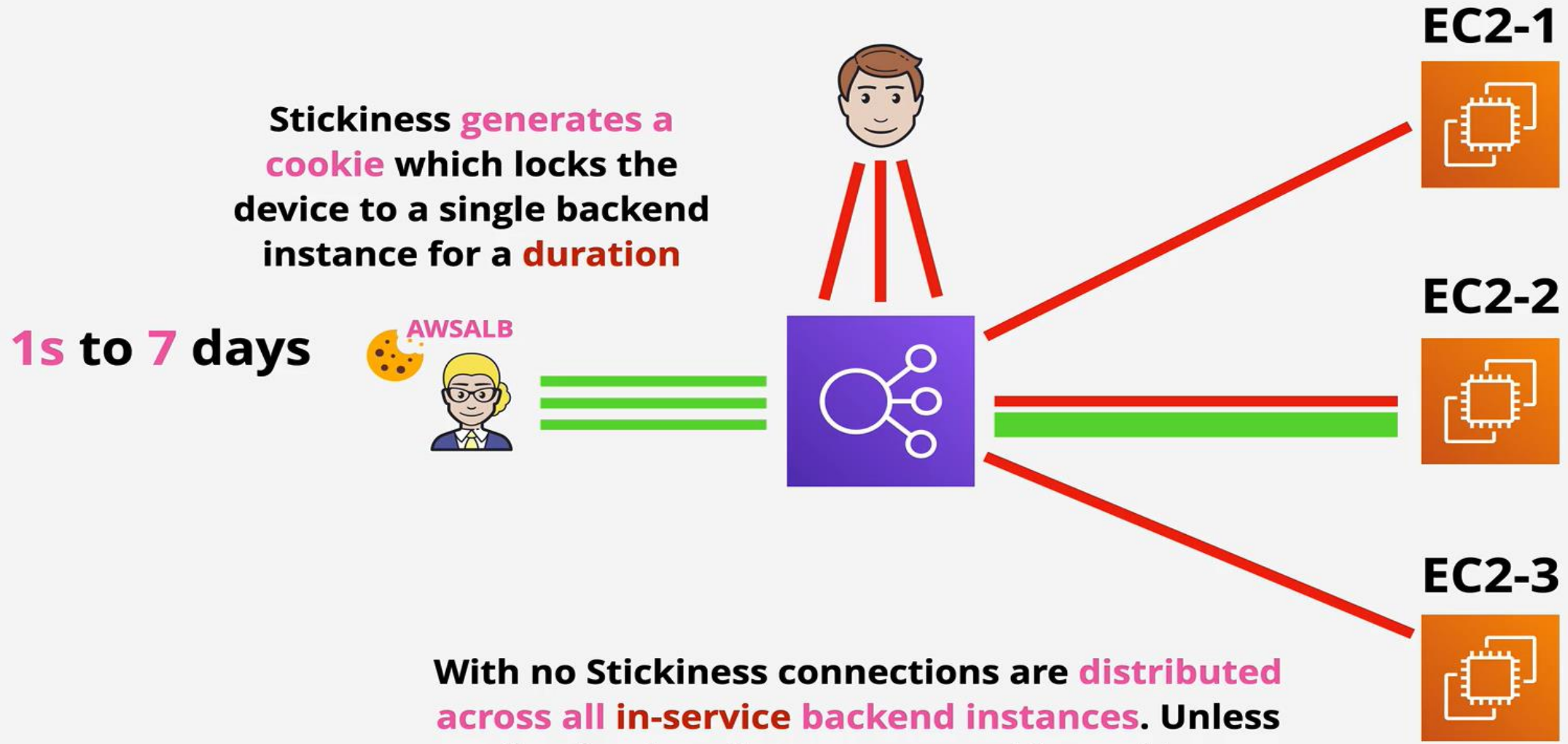
ASG Health Checks

- **EC2(Default),** ELB(can be enabled) & Custom
- **EC2**- Stopping, Stopped, Terminated, Shutting Down or Impaired (not 2/2 status) = UNHEALTHY
- **ELB** – HEALTHY = Running & passing ELB health check
- Can be more application aware (Layer 7)
- **Custom** – Instances marked healthy & unhealthy by an external system.
- Health check grace period(Default 300s)-Delay before starting checks
- …allows system launch, bootstrapping and application start

SSL Offload


Bridging
ELB
ELB initiates a new SSL connection to backend instances. Instances need SSL certificates and the compute required for cryptographic operations
Listener is configured for HTTPS.
Connection is terminated on the ELB &
needs a certificate for the domain name.
Pass-through
NLB
Each instance needs to have the appropriate SSL cert installed.
With this architecture there is
no certificate exposure to AWS ..
all self-managed and secured.
Listener is configured for TCP. No encryption
or decryption happens on the NLB.
Connection is passed to backend instance
HTTP
Offload
ELB
ELB to instance connections use HTTP - no certificate or cryptographic requirements
Listener is configured for HTTPS.
Connections are terminated and
then backend connections use HTTP

Connection Stickiness



Auto Scaling Groups – Scaling Cooldowns

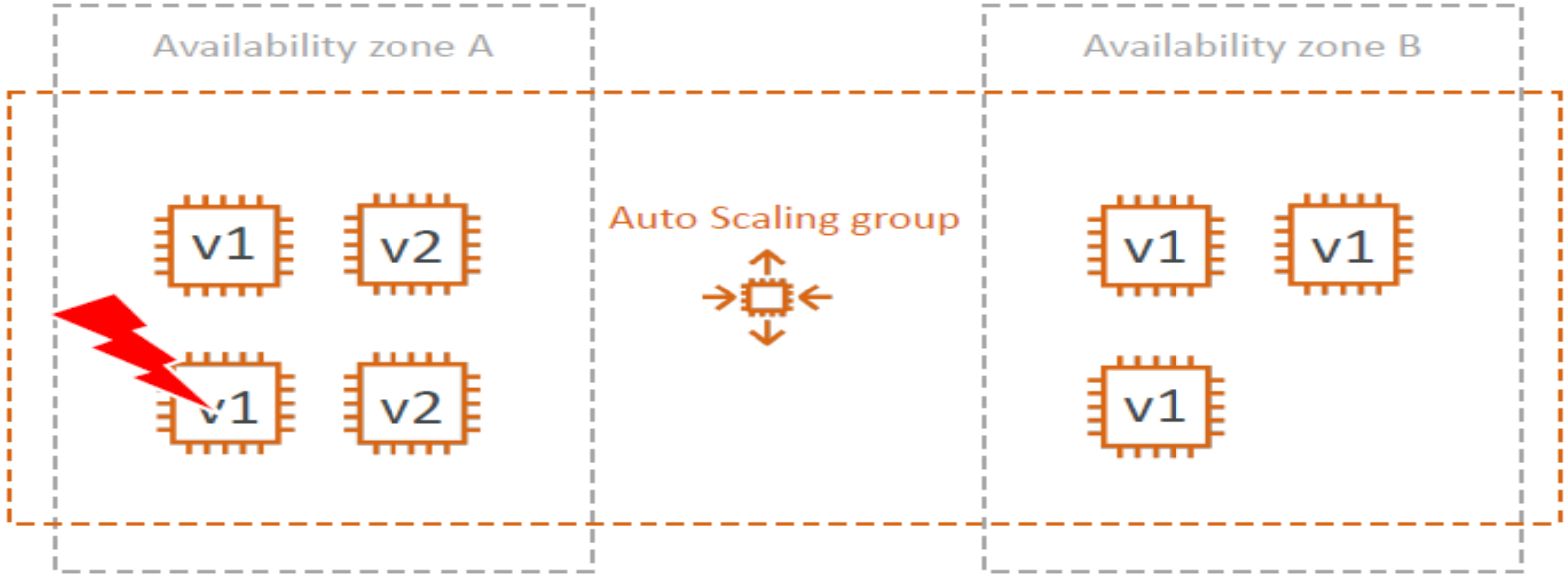
- After a scaling activity happens, you are in the **cooldown period** (**default 300 seconds)**
- During the cooldown period, the ASG will not launch or terminate additional instances (to allow for metrics to stabilize)
- Advice: Use a ready-to-use AMI to reduce configuration time in order to be serving request fasters and reduce the cool down period

ASG for Solutions Architects

ASG Default Termination Policy(simplified version):

1. Find the AZ which has the most number of instances
2. If there are multiple instances in the AZ to choose from, delete the one with the oldest launch configuration

- **ASG tries the balance the number of instances across AZ by default**



ASG for Solutions Architects Lifecycle Hooks

- By default, as soon as an instance is launched in an ASG it’s in service.
- You can perform extra steps before the instance goes in service (Pending state)
- You can perform some actions before the instance is terminated (Terminating state)

Architecture Patterns - Compute

Requirements	Solution
High availability and elastic scalability for web servers	Use Amazon EC2 Auto Scaling and an Application Load Balancers across multiple AZs
Low-latency connections over UDP to a pool of instances running a gaming application	Use Network Load Balancer with a UDP listener
Clients need to whitelist static IP addresses for a highly available load balanced application in an AWS Region	Use an NLB and create static IP addresses in each AZ
Application on EC2 in an Auto Scaling group requires disaster recovery across Regions	Create an ASG in a second Region with the capacity set to). Take snapshots and copy them across Regions (Lambda or DLM)
Application on EC2 must scale in larger increments if a big increase in traffic occurs, compared to small increases in traffic	Use Auto Scaling with Step Scaling policy and configure a larger capacity increase.
Need to scale EC2 instances behind an ALB based on the number of requests completed by each instances	Configure a target tracking policy using the ALBRequestCountPerTarget metric
Need to run a large batch computing job at the lowest cost. Must be managed. Nodes can pick up where others left off in case of interruption.	Use a managed AWS Batch job and use EC2 Spot instances.

Architecture Patterns - Compute

Requirements	Solution
A tightly coupled High performance Computing (HPC) workload requires low-latency between nodes and optimum network performance	Launch EC2 in a single AZ in a cluster placement group and use an Elastic Fabric Adapter (EFA)
Application must startup quickly when launched by ASG but requires app dependencies and code to be installed	Create an AMI that includes the application dependencies and code
Application runs on EC2 behind and ALB. Once authenticated users should not need to reauthenticate if an instance fails	Enable Sticky session for the target group or alternatively use a session store such as DynamoDB

Advanced Demo [Architecture Evolution]

Load Balancing Demo