COMP5349– Cloud Computing
Week 7: Elasticity and High Availability

Dr. Ying Zhou
The University of Sydney

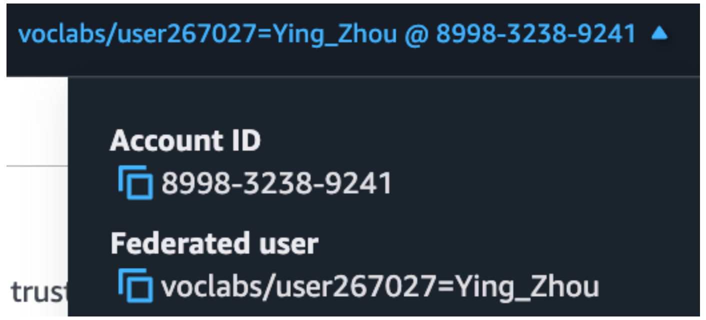
Table of Contents

Federated User in AWS Academy Lab

Session policy

All federated users in AWS academy assume the role “voclabs”

The role is likely defined under a single or a few AWS accounts



voclabs/user267027=Ying_Zhou @ 8998-3238-9241

Account ID
8998-3238-9241

Federated user
voclabs/user267027=Ying_Zhou

trus

Each lab has different sets of permissions (policies) associated with this role

At any time, different users can attempt different labs

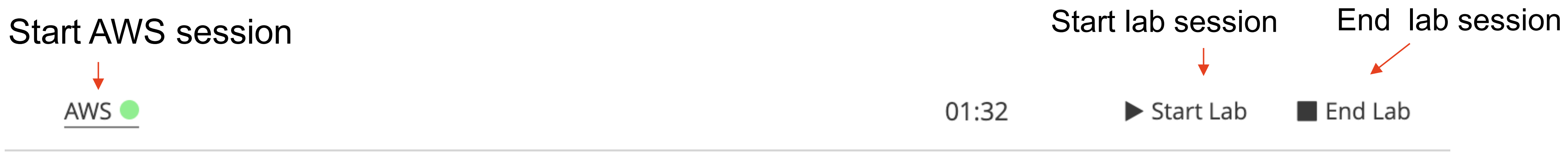
Policies are dynamically associated with this role, and multiple policies can be associated with it at the same time

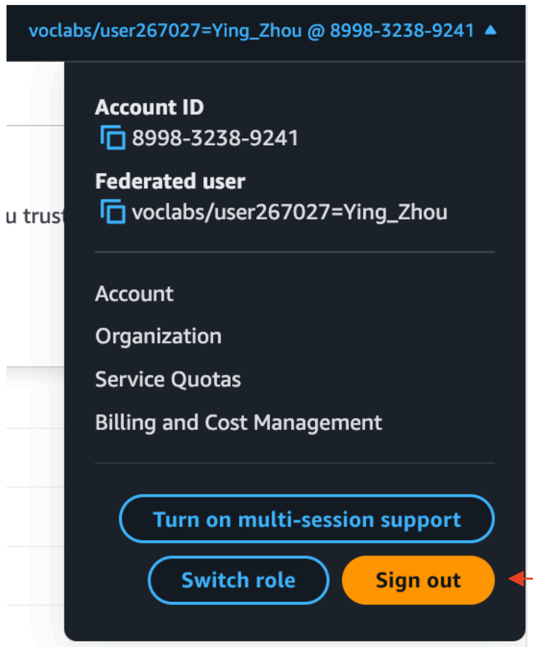
Session policy

- Session policies are advanced policies that you pass as a **parameter** when you programmatically create a temporary session for a role or federated user.
- When AWS academy backend calls AssumeRole for a federated user, it includes a Policy parameter that limit the session’s permissions

```
AssumeRole(
  RoleArn="arn:aws:sts::899832389241:role/voclabs/",
  RoleSessionName='131-5457484-8425041',
  Policy='{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": ["s3:ListBucket", "ec2:DescribeInstances"],
        "Resource": "*"
      }
    ]
  }'
)
```

Lab Session vs. AWS Session





voclabs/user267027=Ying_Zhou @ 8998-3238-9241
u trust
Account ID
8998-3238-9241
Federated user
voclabs/user267027=Ying_Zhou
Account
Organization
Service Quotas
Billing and Cost Management
Turn on multi-session support
Switch role
Sign out

End AWS session

When you end the lab session, all lab resources will be terminated, your federated user's permission will be revoked

But, your AWS session could still be alive and you can navigate the AWS console, but you can't do useful things now

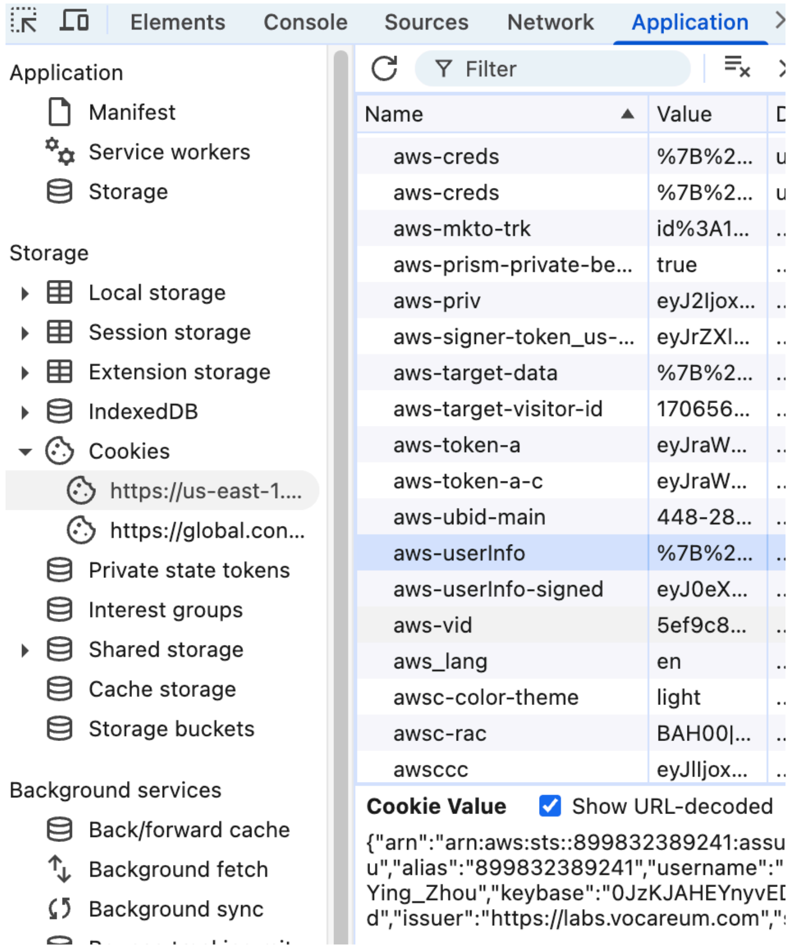
When you end your AWS session before your lab session expires, your created resources are still there and you can start AWS session again by clicking the AWS button

What does start lab do?

- We are already authenticated by AWS academy
- When we click the “start lab” button
  - It calls AWS **STS (AssumeRole)** on a preconfigured IAM role — voclabs.
    - It passes a **custom session name** like user267027=Ying_Zhou to track the authenticated user.
    - It may also attach a **session policy** to restrict what this session can do (lab-specific permissions).
  - AWS STS responds with temporary credentials
    - SessionToken, SecretAccessKey, etc
  - AWS Academy uses the information to generate a federated user sign-in URL
    - https://signin.aws.amazon.com/federation?Action=login&SigninToken=<token>&Destination=https://conso le.aws.amazon.com/
    - This is the url you will send when clicking the green AWS button
  - AWS Academy also uses a blueprint (CloudFormation template) to warm up the lab session by creating resources like a LabVPC, an EC2 instance

When we click the green AWS session

- This URL grants us **browser-based access** to the AWS Console using the temporary credentials.
- The URL is opened in a new tab to display the AWS console
- When the AWS console opens, AWS set a lot of cookies to keep track of the session
  - aws-creds –contains the STS credentials
  - aws-sessionID – tracks the console session
  - Others



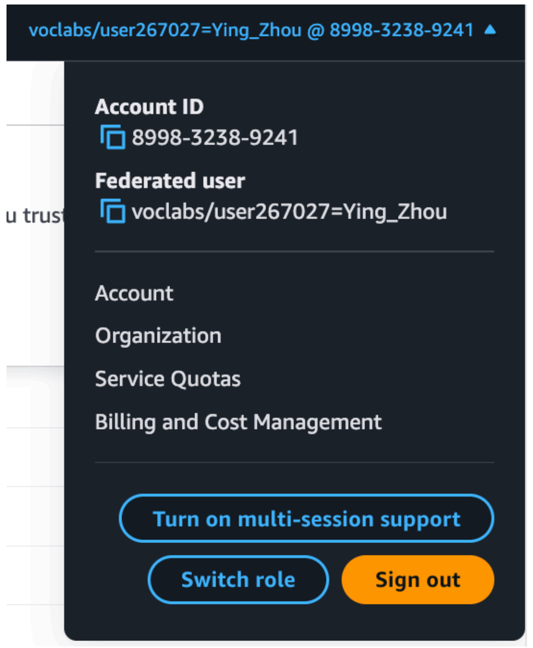
Name	Value	D
aws-creds	%7B%2...	u
aws-creds	%7B%2...	u
aws-mkto-trk	id%3A1...	..
aws-prism-private-be...	true	..
aws-priv	eyJ2Ijox...	..
aws-signer-token_us-...	eyJrZXl...	..
aws-target-data	%7B%2...	..
aws-target-visitor-id	170656...	..
aws-token-a	eyJraW...	..
aws-token-a-c	eyJraW...	..
aws-ubid-main	448-28...	..
aws-userInfo	%7B%2...	..
aws-userInfo-signed	eyJ0eX...	..
aws-vid	5ef9c8...	..
aws_lang	en	..
awsc-color-theme	light	..
awsc-rac	BAH00|...	..
awsccc	eyJlIjox...	..

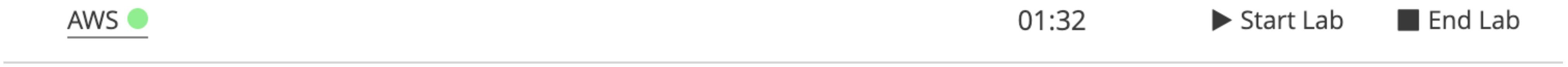
Cookie Value [✓] Show URL-decoded

```
{"arn":"arn:aws:sts::899832389241:assu
u","alias":"899832389241","username":"
Ying_Zhou","keybase":"0JzKJAHEYnyvED
d","issuer":"https://labs.vocareum.com","
```

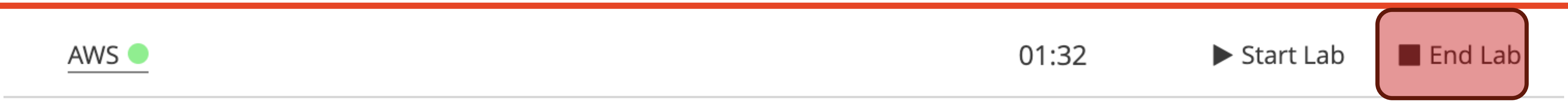
When we sign out from AWS

- All cookies will be cleared
- But, as long as the lab session is still on, we can click the green AWS button to start a new session
- The button is associated with the same federated user sign-in URL
- When we access the AWS console within the same lab session, we are recognized by AWS as the same user


voclabs/user267027=Ying_Zhou @ 8998-3238-9241
Account ID
8998-3238-9241
Federated user
voclabs/user267027=Ying_Zhou
Account
Organization
Service Quotas
Billing and Cost Management
Turn on multi-session support
Switch role
Sign out



What does end lab do



- The lab session is managed by AWS academy
- When we click end lab
  - All resources created during the lab are automatically cleared
  - The session permissions (session policy) are revoked or disabled
  - The AWS button is disabled
- If the AWS Console is still open in a browser tab, you may still **see the UI**, but most actions will fail due to permission loss.
- AWS Academy **cannot clear the AWS Console cookies** due to the browser's same-origin policy.
- You are required to explicitly logout the current AWS session to visit the console of a new lab

Session expiration

- Both AWS and lab session expire after inactivity or a set time.
- When AWS session expires
  - The effect is the same as explicitly logging out – you lose access to the resources
  - If your **lab session is still valid**, you may be able to launch the console again and resume your work.
- When the lab session expires
  - The effect is the same as clicking the end lab button
  - All resources are cleared automatically and the permissions are revoked or disabled
  - Your temporary credential becomes invalid
  - You must click **“Start Lab”** again to get new credentials and restart the environment.

What does click start lab again do?

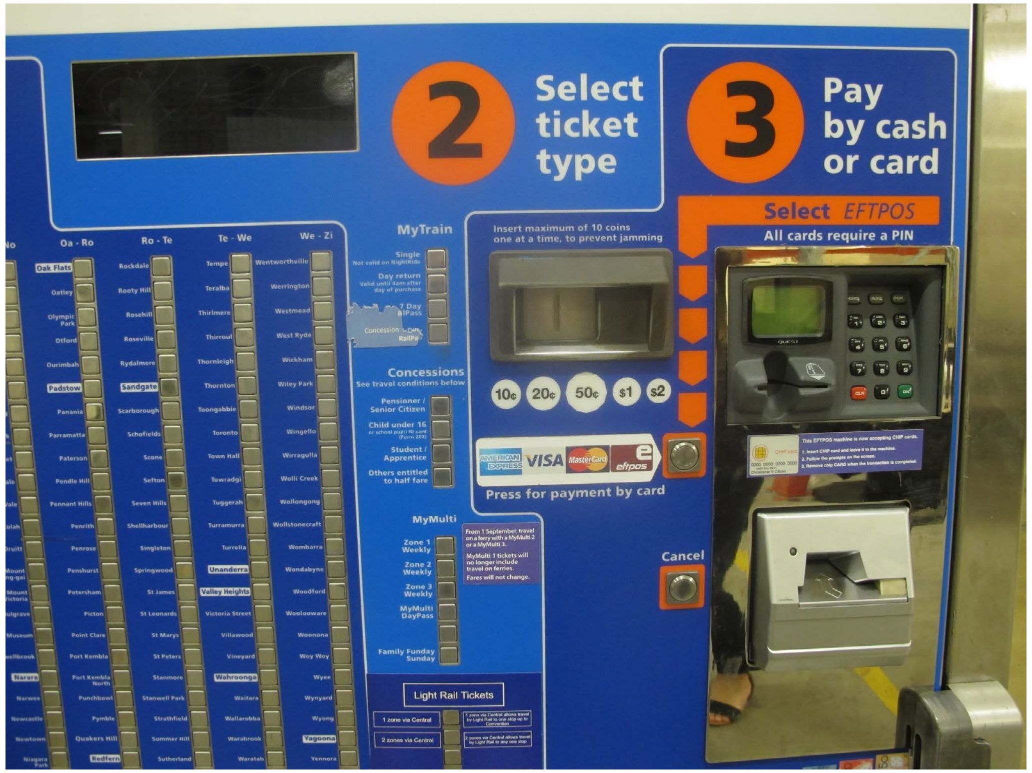
- Core action
  - This resets the timer of a current lab session, giving you additional tie to work on the same lab
  - Each lab session has a fixed duration (e.g., 2 hours), which is extended each time you click “Start Lab”
- What does not change
  - Your AWS Console credentials (temporary STS tokens) do not always refresh unless they are close to expiration.
  - Your assumed role and session information remain the same
  - Resources created during the lab session remain intact

Scalability and Elasticity

What is scalability

- **Scalability** is the property of a system to handle a *growing* amount of work.

https://en.wikipedia.org/wiki/Scalability

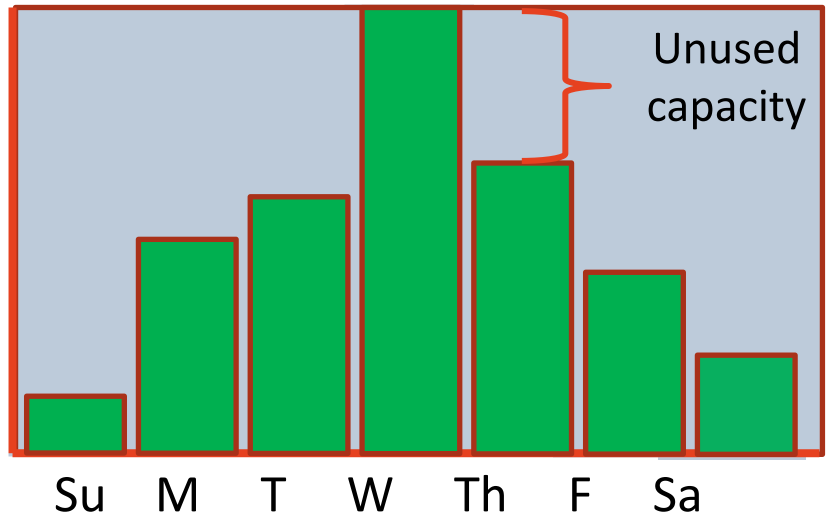


Usually judged by how easy it is to add more resources to handle such changes

An example of a system not scalable

What is elasticity?

- Supports fine grained capacity expansion and *contraction*.
  - Scalability in cloud setting
- Scalability in traditional setting is mostly about capacity expansion



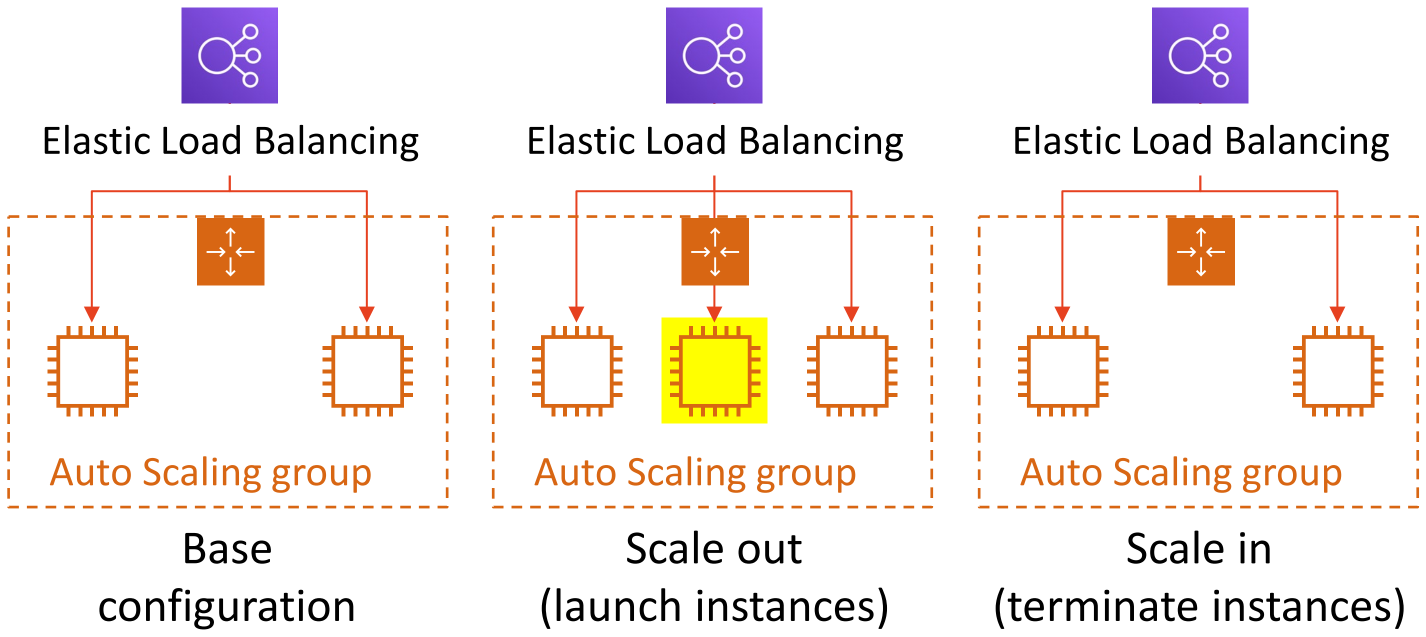
Examples:

- Increasing the number of web servers when traffic spikes
- Lowering write capacity on your database when traffic goes down
- Handling the day-to-day fluctuation of demand throughout your architecture

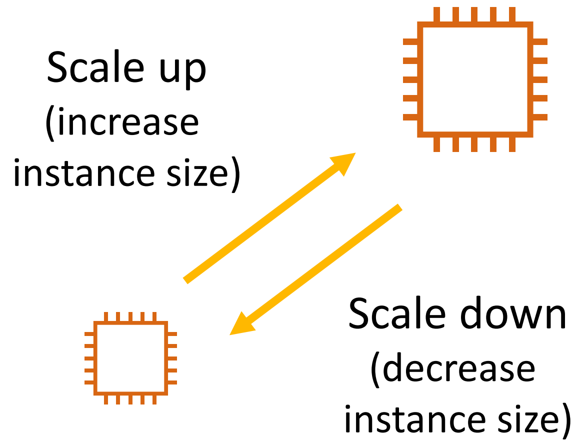
Scaling a computer system

- Is implemented by adjusting the amounts of resources based on the workload

Horizontal scaling



Vertical scaling



Scale up (increase instance size)
Scale down (decrease instance size)

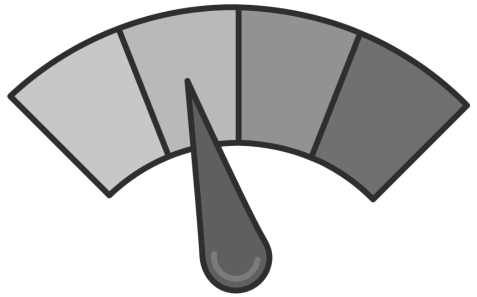
What are required to achieve elasticity in cloud

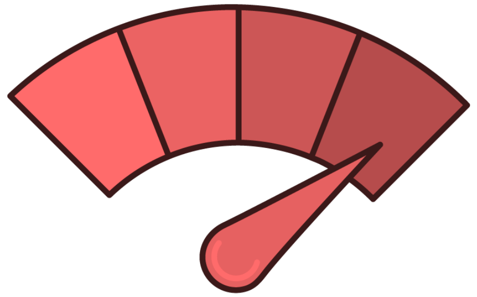
- The ability to monitor the workload and make scaling decisions accordingly
  - The cloud platform provides the mechanism, e.g. **CloudWatch**, **AWS Auto Scaling**
  - The users need to configure them properly
- The ability to automatically start/stop new resources with minimal latency
  - The cloud platform provides the mechanisms, e.g. EC2 instances can be created from AMI with very small latency
  - The users need to configure them properly
- The resource change should be recognized by the running system
  - Cloud platform provides the mechanisms, such as AWS Elastic load balancing that can direct requests to newly added resources
  - The users need to ensure that the functionalities of the application are not affected by adding or removing resources

Resource Monitoring

Amazon CloudWatch

Amazon CloudWatch

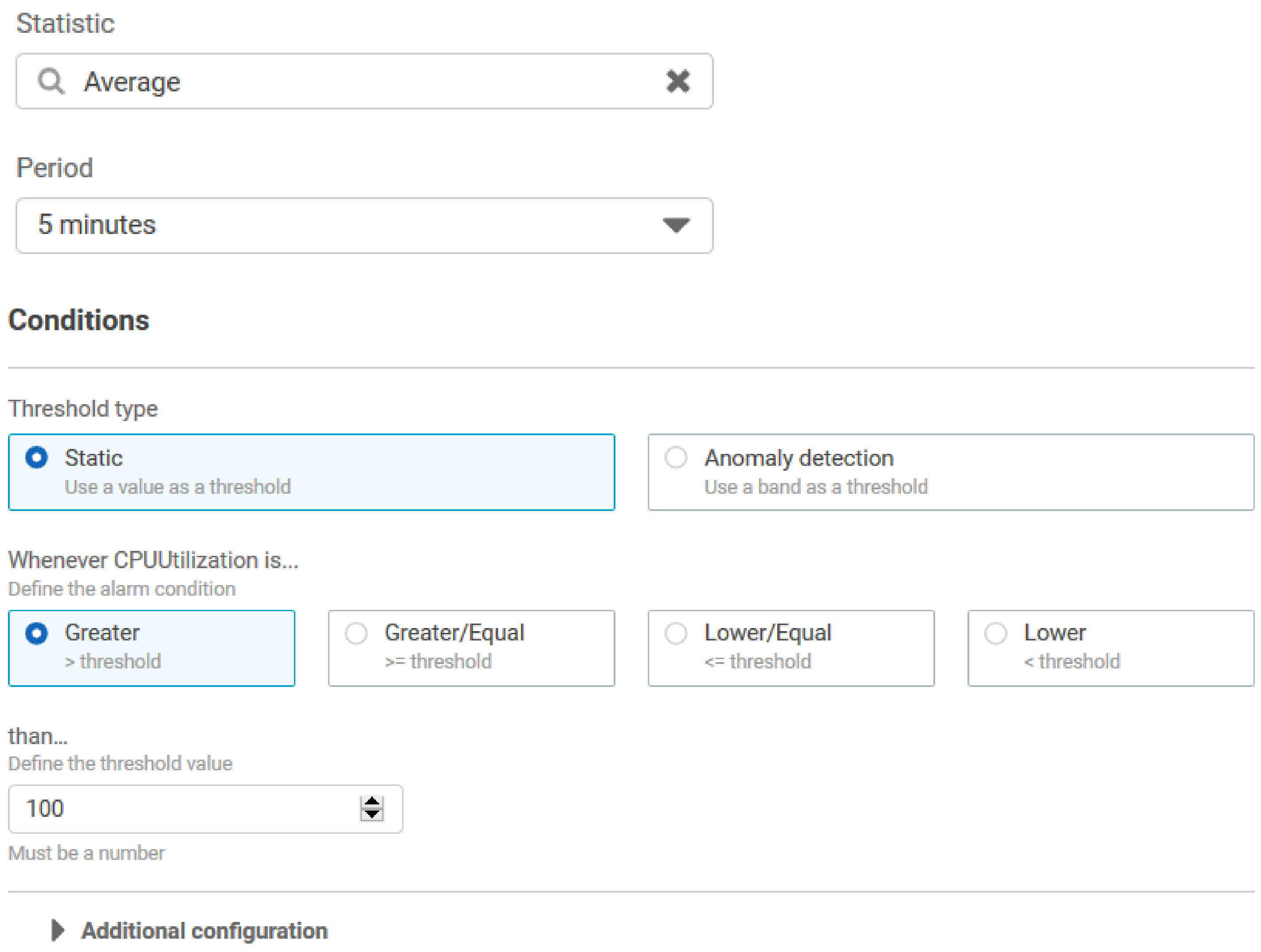




- Monitors –
  - AWS resources
  - Applications that run on AWS
- Collects and tracks –
  - Standard metrics
  - Custom metrics
- Alarms –
  - Send notifications to an Amazon SNS topic
  - Perform Amazon EC2 Auto Scaling or Amazon EC2 actions
- Events –
  - Define rules to match changes in AWS environment and route these events to one or more target functions or streams for processing

CloudWatch alarms

- Create alarms based on –
  - Static threshold
  - Anomaly detection
  - Metric math expression
- Specify –
  - Namespace
  - Metric
  - Statistic
  - Period
  - Conditions
  - Additional configuration
  - Actions


Statistic

Average

Period

5 minutes

Conditions

Threshold type

[✓ Static] Use a value as a threshold

Anomaly detection Use a band as a threshold

Whenever CPUUtilization is...

Define the alarm condition

[✓ Greater] > threshold

Greater/Equal >= threshold

Lower/Equal <= threshold

Lower < threshold

than...

Define the threshold value

100

Must be a number

Additional configuration

Scaling Compute Resource

Amazon EC2 Auto Scaling

Amazon EC2 Auto Scaling

- Manages a logical collection of Amazon Elastic Compute Cloud (Amazon EC2) instances called an Amazon EC2 Auto Scaling group across Availability Zones
- Launches or retires EC2 instances configured by launch templates
- Resizes based on events from scaling policies, load balancer health check notifications, or schedule actions
- Integrates with Elastic Load Balancing (ELB) to send new instances registrations and receive health notifications
- Balances the number of instances across Availability Zones
- Is available free of charge

Amazon EC2 Auto Scaling group components

Amazon EC2 Auto Scaling group

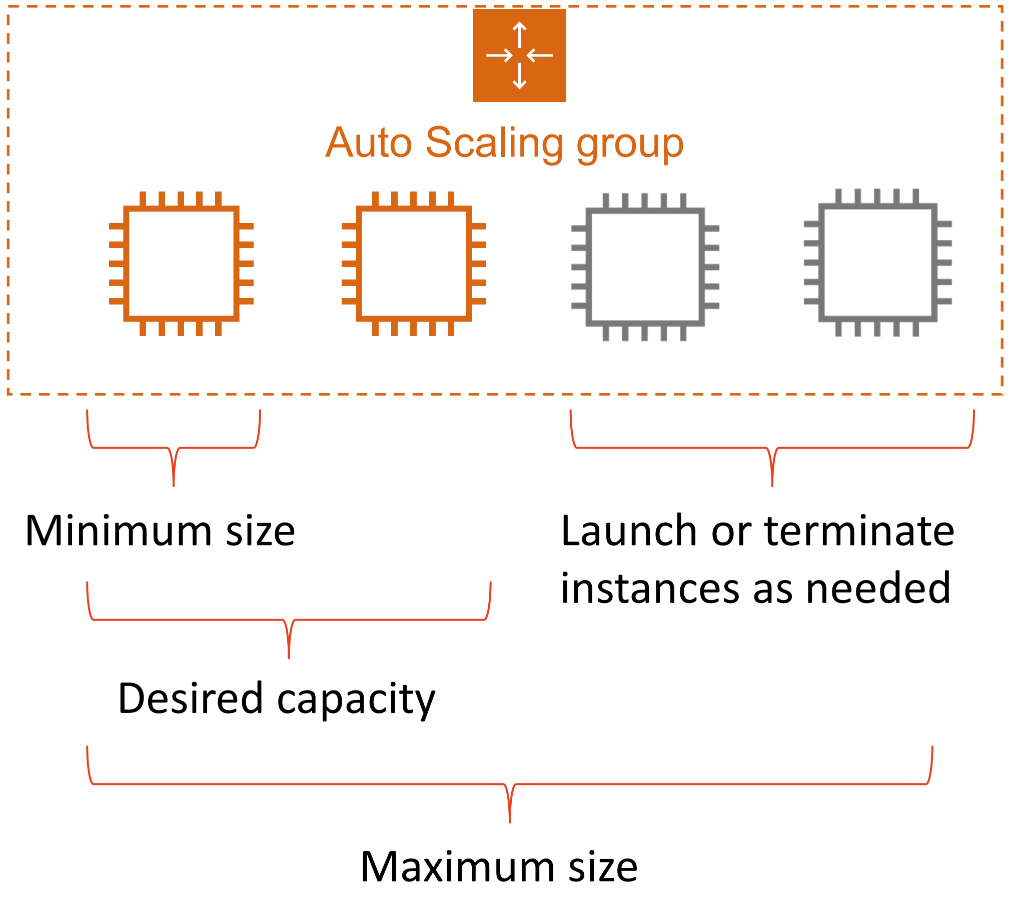
Capacity
- Minimum instances
- Maximum instances
- Desired instances

Launch template
- EC2 instance configuration
- Amazon Machine Image (AMI) ID
- Version

Load balancer
- Receives load balancer health checks
- Registers new instances with load balancer

Scaling mechanisms
- Schedule action
- Dynamic scaling policies
- Predictive scaling policy

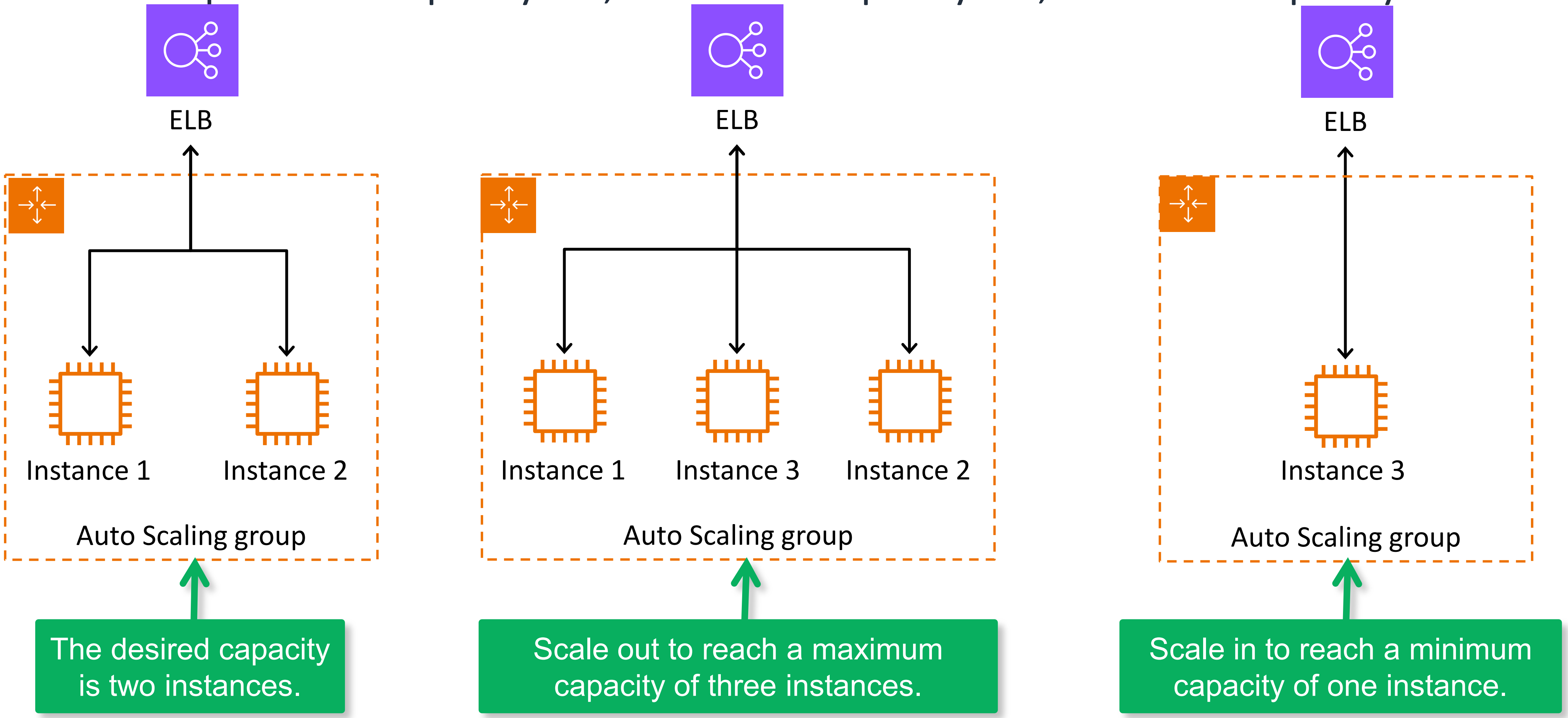
Auto Scaling Group - capacity



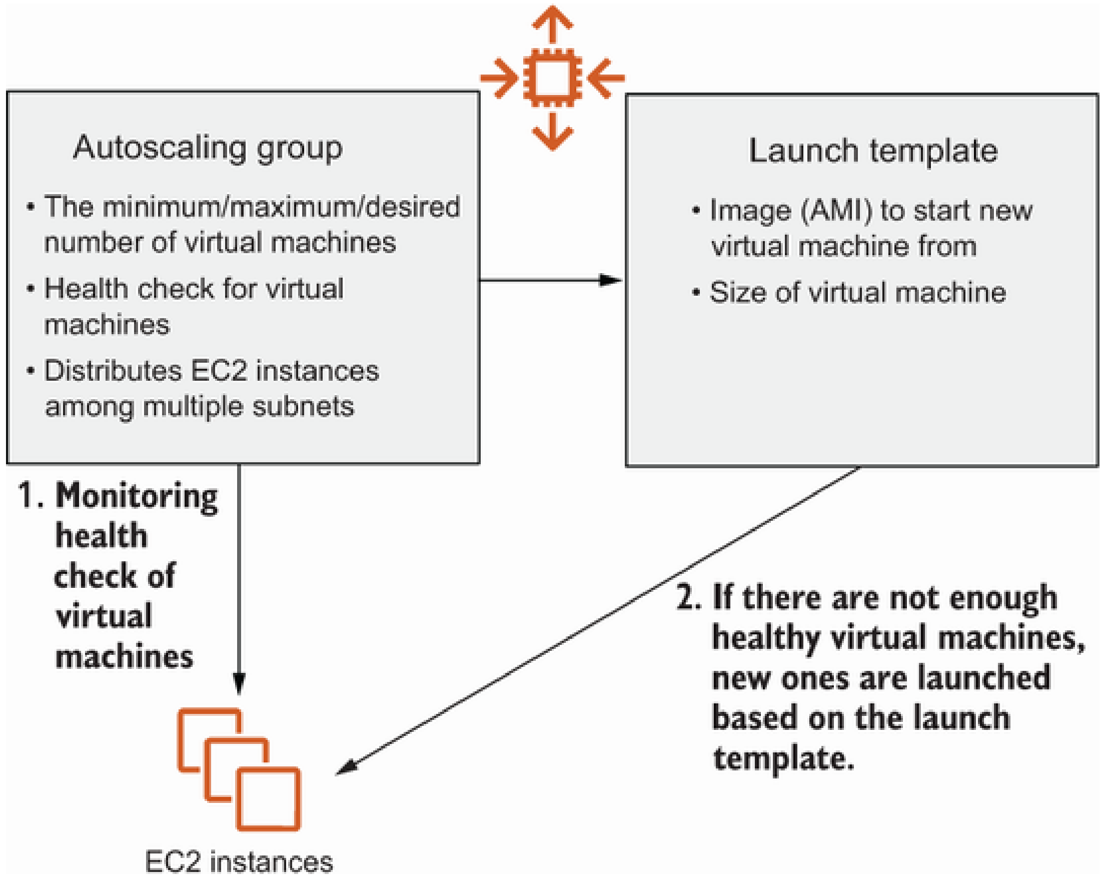
An **Auto Scaling group** is a collection of EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management.

Horizontal scaling with Amazon EC2 Auto Scaling groups

Group desired capacity = 2, maximum capacity = 3, minimum capacity = 1

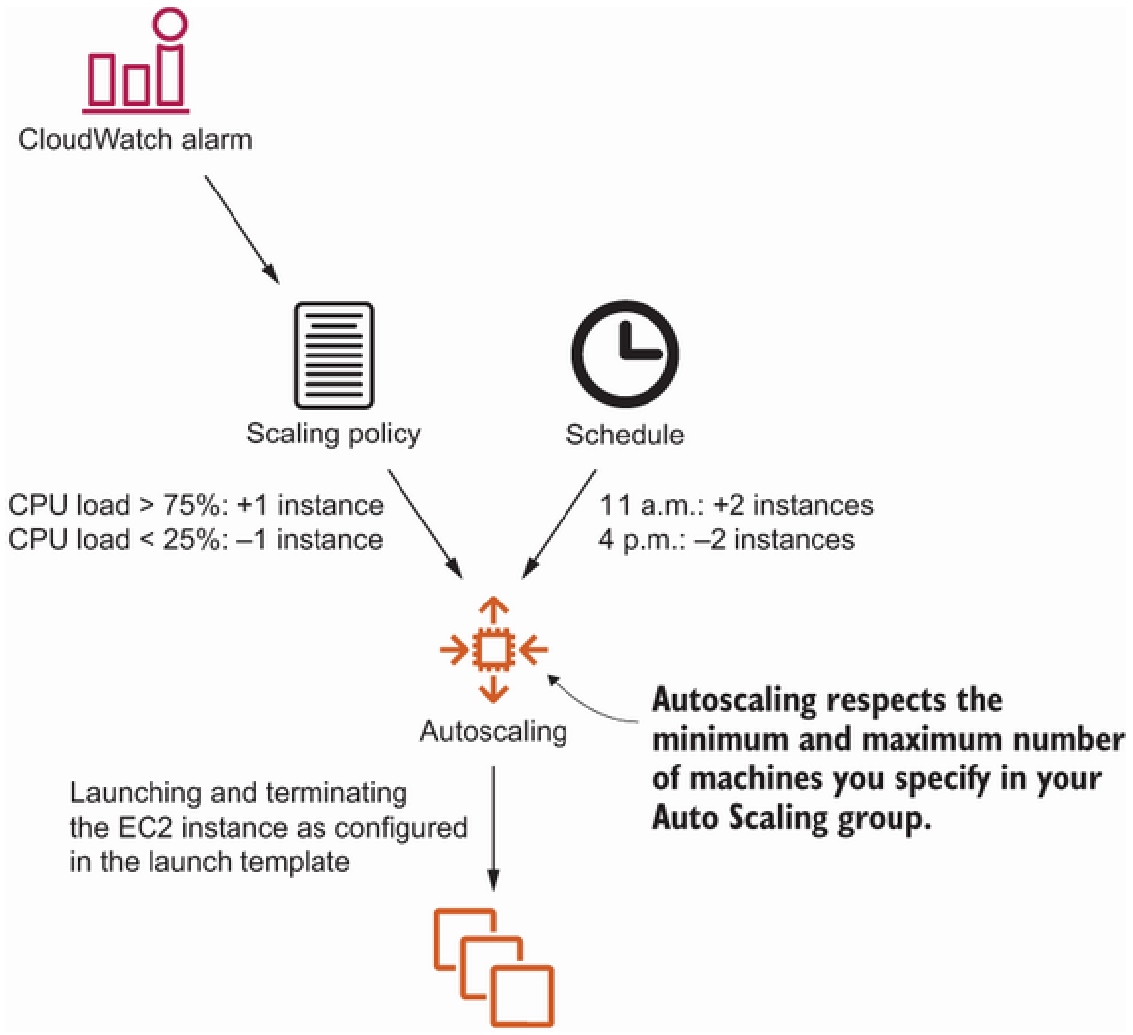

ELB
Instance 1
Instance 2
Auto Scaling group
The desired capacity is two instances.
ELB
Instance 1
Instance 3
Instance 2
Auto Scaling group
Scale out to reach a maximum capacity of three instances.
ELB
Instance 3
Auto Scaling group
Scale in to reach a minimum capacity of one instance.

Auto Scaling Group – Launch Template


Autoscaling group
• The minimum/maximum/desired number of virtual machines
• Health check for virtual machines
• Distributes EC2 instances among multiple subnets
Launch template
• Image (AMI) to start new virtual machine from
• Size of virtual machine
1. Monitoring health check of virtual machines
2. If there are not enough healthy virtual machines, new ones are launched based on the launch template.
EC2 instances

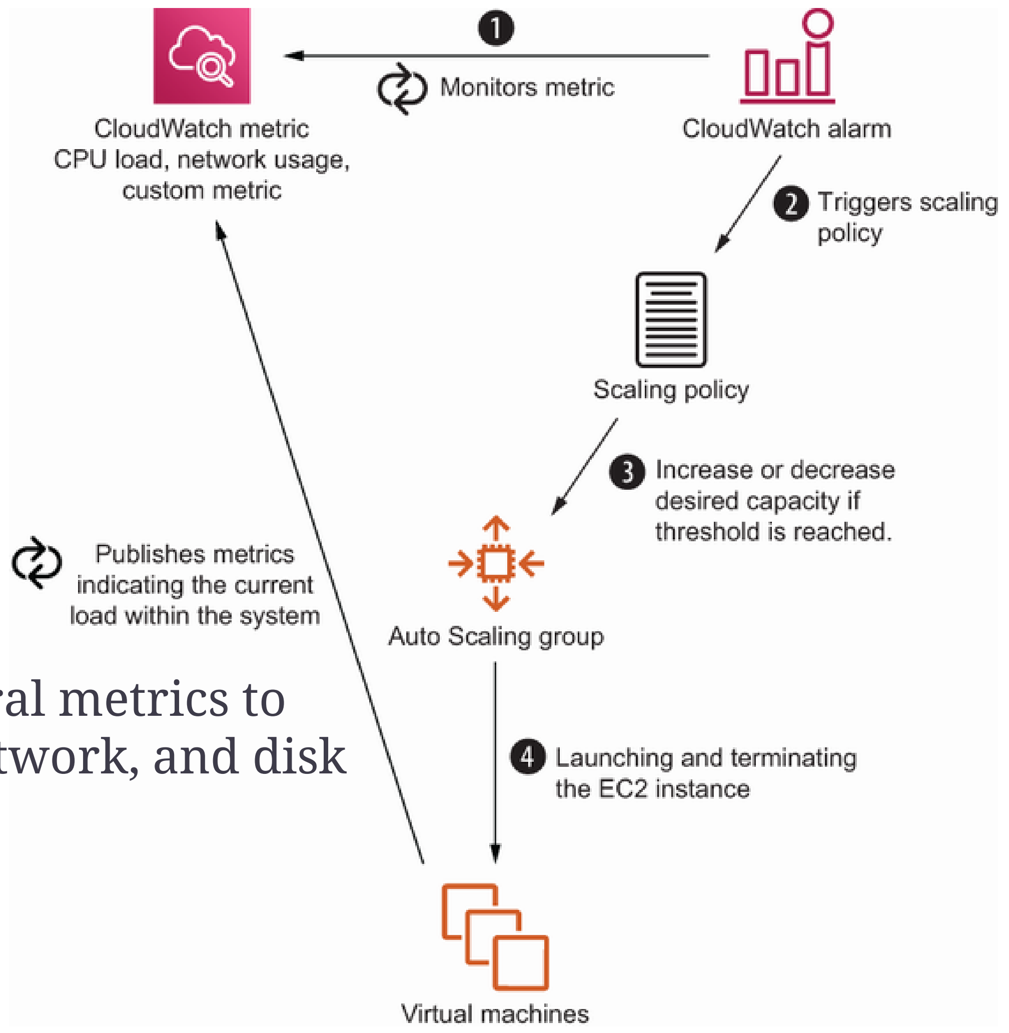
- A *launch template* contains all information needed to launch an EC2 instance
  - Instance type (size of virtual machine)
  - Image (AMI)
  - Security Group
  - User Data
- The subnet for the virtual machine isn't defined in the launch template but rather in the Auto Scaling group.

Scheduling Mechanism


CloudWatch alarm
Scaling policy
Schedule
CPU load > 75%: +1 instance
CPU load < 25%: –1 instance
11 a.m.: +2 instances
4 p.m.: –2 instances
Autoscaling
Autoscaling respects the minimum and maximum number of machines you specify in your Auto Scaling group.
Launching and terminating the EC2 instance as configured in the launch template

- Based on schedule
  - E.g. : Turning off your development and test instances at night
- Based on policy
  - Usually relying on CloudWatch alarm
  - Maintaining constant number
    - Always 2 instances
  - Target tracking
    - Maintaining an average CPU load of 60%
  - Other policies

Scheduling policy based on CloudWatch Metrics


1
Monitors metric
CloudWatch metric
CPU load, network usage, custom metric
CloudWatch alarm
2 Triggers scaling policy
Scaling policy
3 Increase or decrease desired capacity if threshold is reached.
Publishes metrics indicating the current load within the system
Auto Scaling group
4 Launching and terminating the EC2 instance
Virtual machines

An EC2 instance publishes several metrics to CloudWatch by default: CPU, network, and disk utilization and others

Elastic Load Balancing

Elastic Load Balancing

Elastic Load Balancing

A managed load balancing service that distributes incoming application traffic across multiple EC2 instances, containers, IP addresses, and Lambda functions.

- Can be external-facing or internal-facing
- Each load balancer receives a DNS name
- Recognizes and responds to unhealthy instances

Types of AWS load balancers

Application Load Balancer

- Is used for HTTP and HTTPS traffic
- Operates at OSI layer 7, the application layer
- Is used for application architectures

Network Load Balancer

- Is used for TLS offloading, UDP, and static IP addresses
- Operates at OSI layer 4, the transport layer
- Is used for millions of requests per second at ultra-low latency

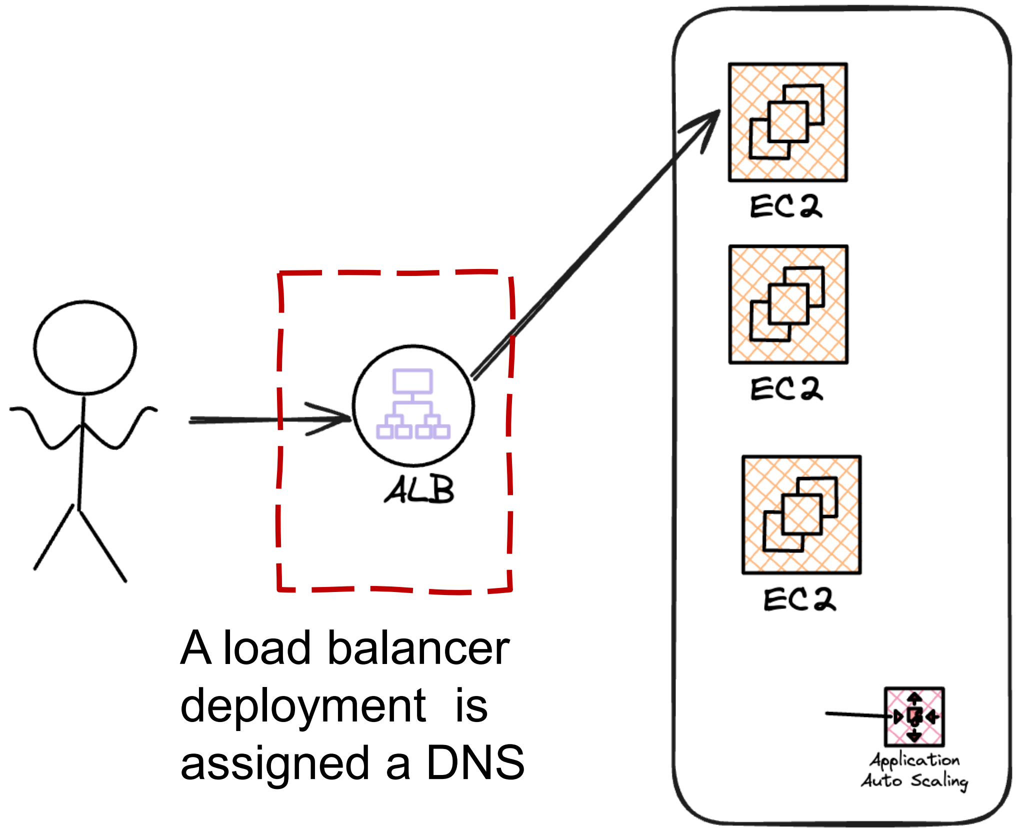
Gateway Load Balancer

- Is used for third-party virtual appliance fleet using GENEVE protocol
- Operates at OSI layer 3, the network layer
- Is used to improve security, compliance, and policy controls

Classic Load Balancer

- Is used for previous generation EC2-Classic networks
- Operates at OSI layers 3 and 7, the transport and application layers
- Is used if upgrading to other load balancers is not feasible

What does a load balancer do?



A load balancer deployment is assigned a DNS

A load balancer is used to route incoming requests to multiple EC2 instances, containers or other targets

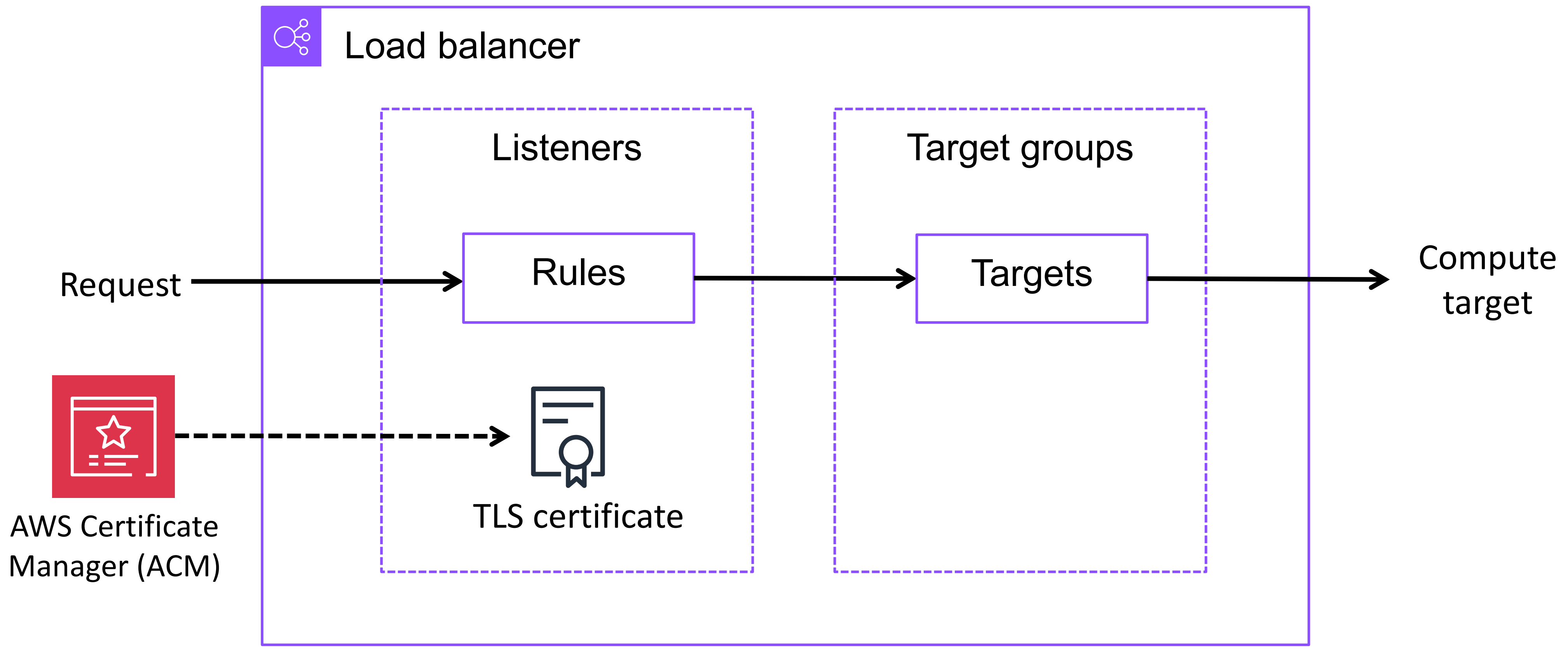
The server running on the EC2 instance must be stateless

A stateless server stores any shared data remotely in a database or storage system.

The capacity can grow or shrink depends on the workloads

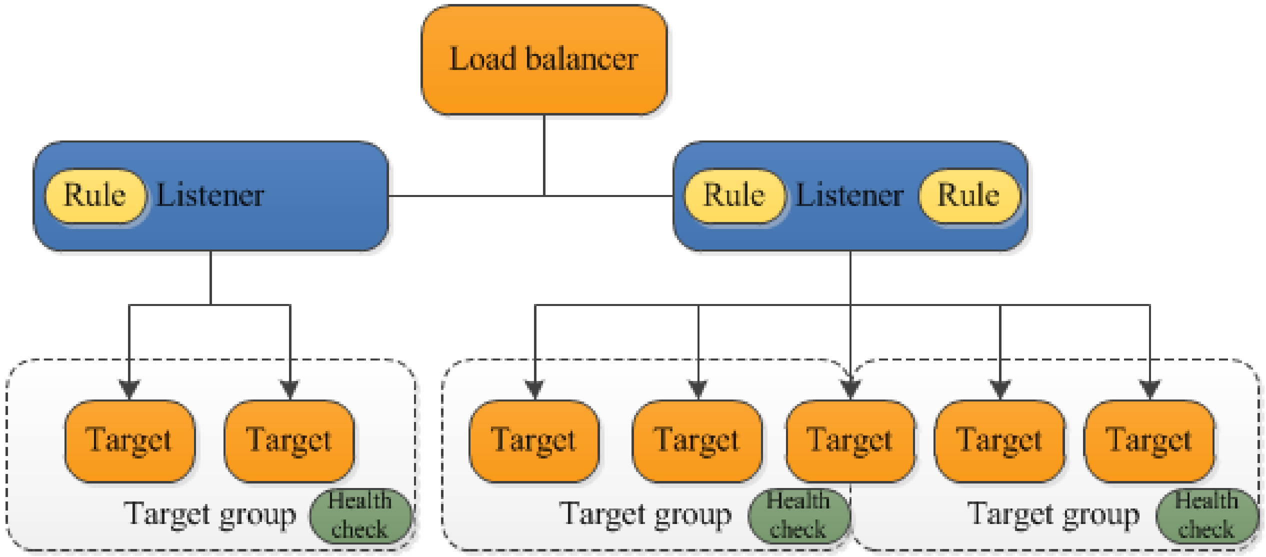
When you create an ELB and enable it in one or more Availability Zones (AZs), AWS provisions *load balancer nodes* within those AZs. These nodes are the actual components that receive incoming traffic and distribute it to your backend instances.

Load balancer components



ALB Components

- Listener
  - Defines the port and protocol which the load balancer must listen on
  - Each ALB must have at least one listener
  - Each ALB can have multiple listeners
  - Listener defines routing rules
- Target Group
  - Logical grouping of AWS resources as target for the traffic
    - E.g. An autoscaling group can be specified as the target group
  - Load balancer performs periodic health check on targes in the target groups. The request will only be routed to healthy target

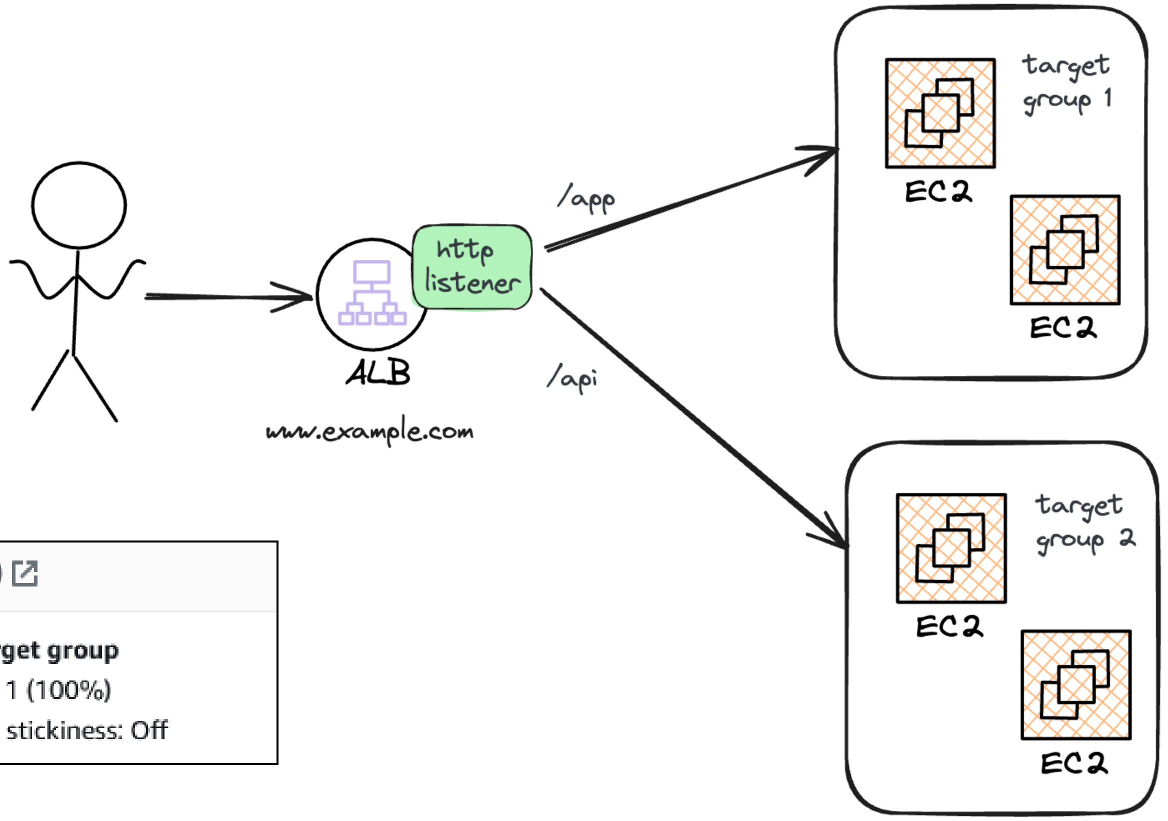


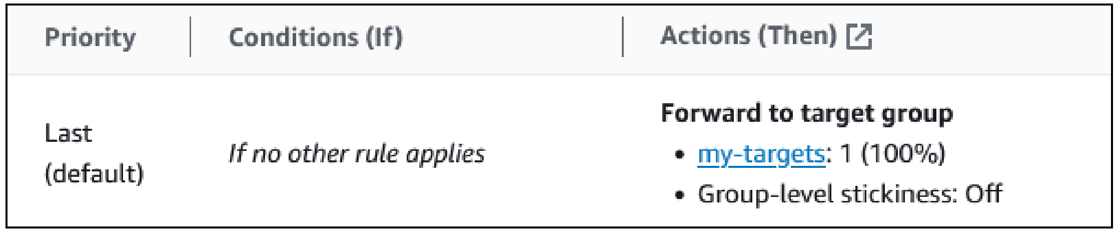
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

Listeners and rules

- Application load balancers are used to distribute HTTP and HTTPS requests
  - They support HTTP and HTTPS listeners
  - One ALB must have one listener
- Each listener defines routing rules for incoming requests
  - Rules consists of conditions and actions
  - When a request meets the condition of the rule, the action is taken
  - Typical actions
    - forwarding to a target group
    - redirect to a url
    - Responding back to the client

An example listener



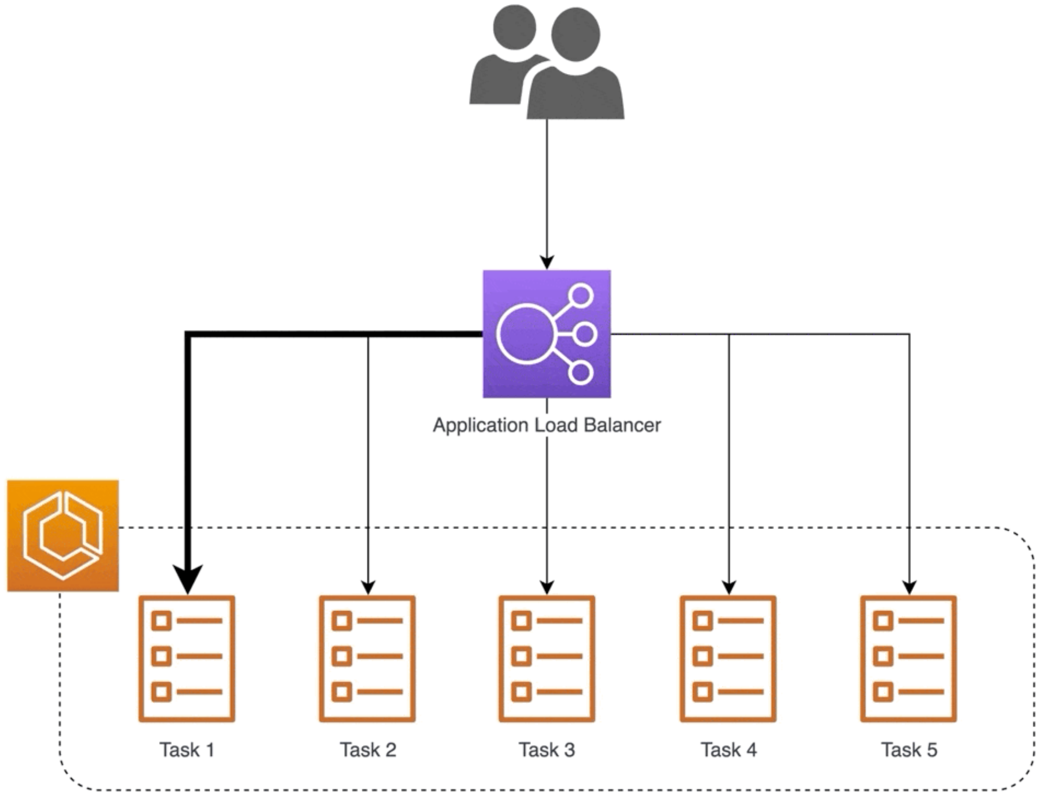


Priority	Conditions (If)	Actions (Then)
Last (default)	*If no other rule applies*	**Forward to target group** • my-targets: 1 (100%) • Group-level stickiness: Off

Default rule

Picking a target to forward the request

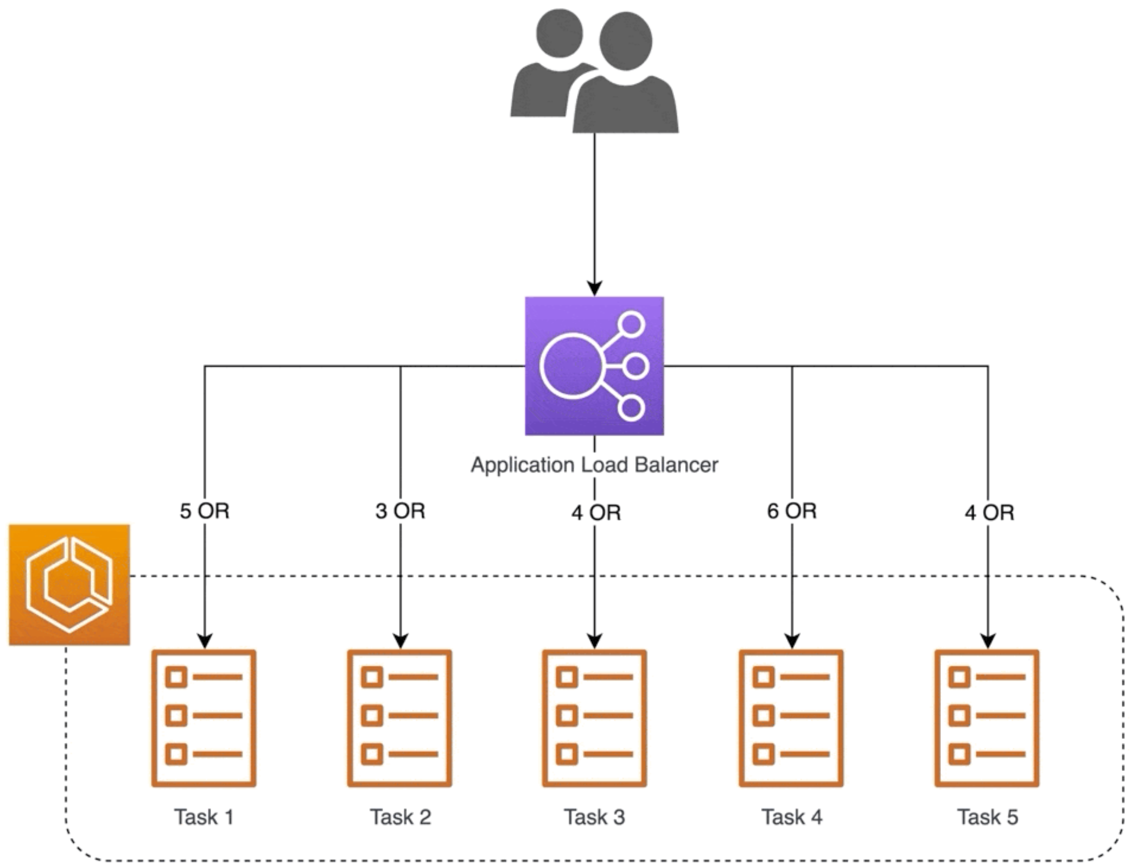
- All load balancers use some load balancing algorithms to distribute incoming requests among targets in a target group.
- ALB uses the following two algorithms
  - Default: Round-robin algorithm
    - Cycling through target
  - Configurable: Least Outstanding Request algorithm



https://medium.com/dazn-tech/aws-application-load-balancer-algorithms-765be2eca158

Least Outstanding Request algorithm

- LOR selects the target with the lowest number of requests waiting for responses.

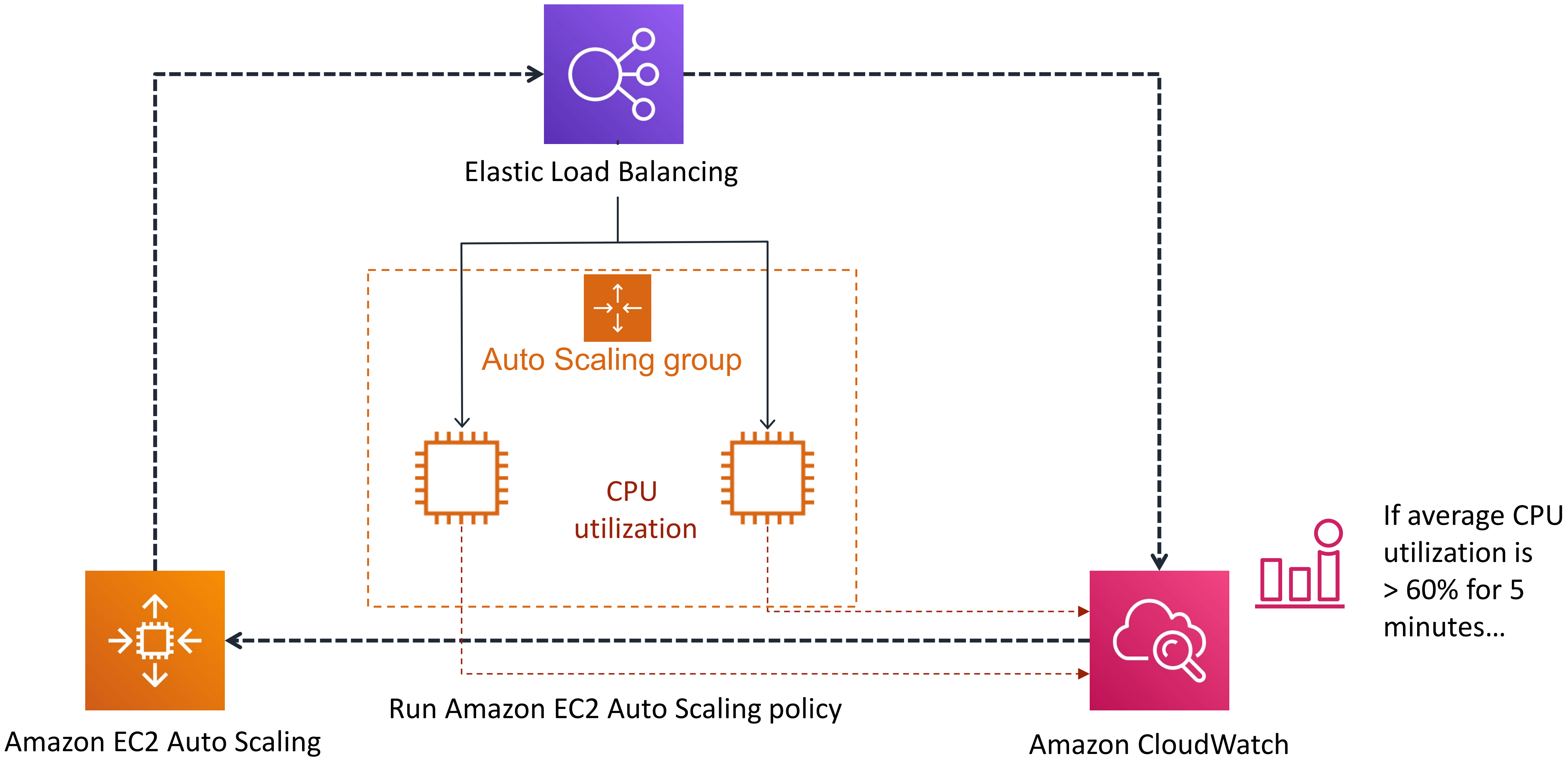


https://medium.com/dazn-tech/aws-application-load-balancer-algorithms-765be2eca158

Target group health check

- Application Load Balancer periodically sends requests to its registered targets to test their status. These tests are called *health checks*.
  - Requests are sent to healthy targets unless there is no healthy target
- The user needs to configure the health check settings
  - Protocol
  - Path
  - Timeout seconds
  - Interval
  - Etc,

Dynamic scaling and load balancing



ELB health check and ASG health check

- Auto Scaling Group always have EC2 level health check turned on
  - EC2 Auto Scaling checks that instances are running and monitors for underlying hardware or software issues that might impair an instance.
- It can be configured to use ELB heath check as well
  - If the target group marks an instance as unhealthy, the ASG will terminate it and depending on the policy may start a new instance
  - Can detect application-level issue and replace instance accordingly

Example Setting

- An **Auto Scaling Group (ASG)** managing **3 EC2 instances**.
- An **Application Load Balancer (ALB)** with a **Target Group** that receives traffic and routes it to instances in the ASG.
  - The **Target Group** does HTTP health checks every 30 seconds.
- The **ASG** is configured to use **ELB health checks**
  - it considers instances unhealthy if the Target Group marks them unhealthy

Scenario: App Crash

- The app crashes, but the EC2 instance is still “up”
- Target group health check fails
- Since the ASG is using ELB health checks, it also:
  - Marks instance as unhealthy.
  - Terminates it and launches a replacement to keep 3 instances in the group

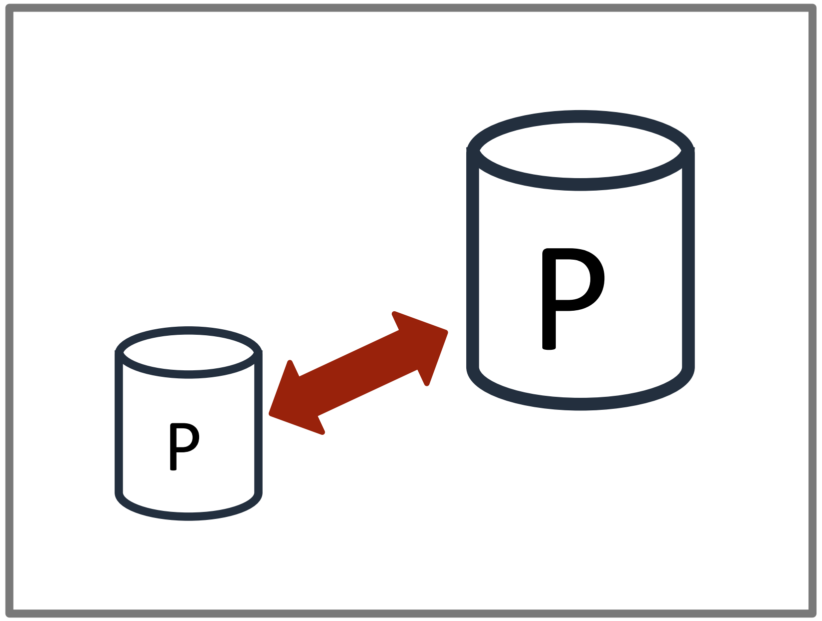
Scenario: unresponsive instance

- EC2 health check fail
  - Detected by CloudWatch and reported to ASG
- ASG will terminate the instance and start and new one
  - This could eventually be detected by the target group health check
  - The CloudWatch detection of system failure is faster and ASG can react immediately

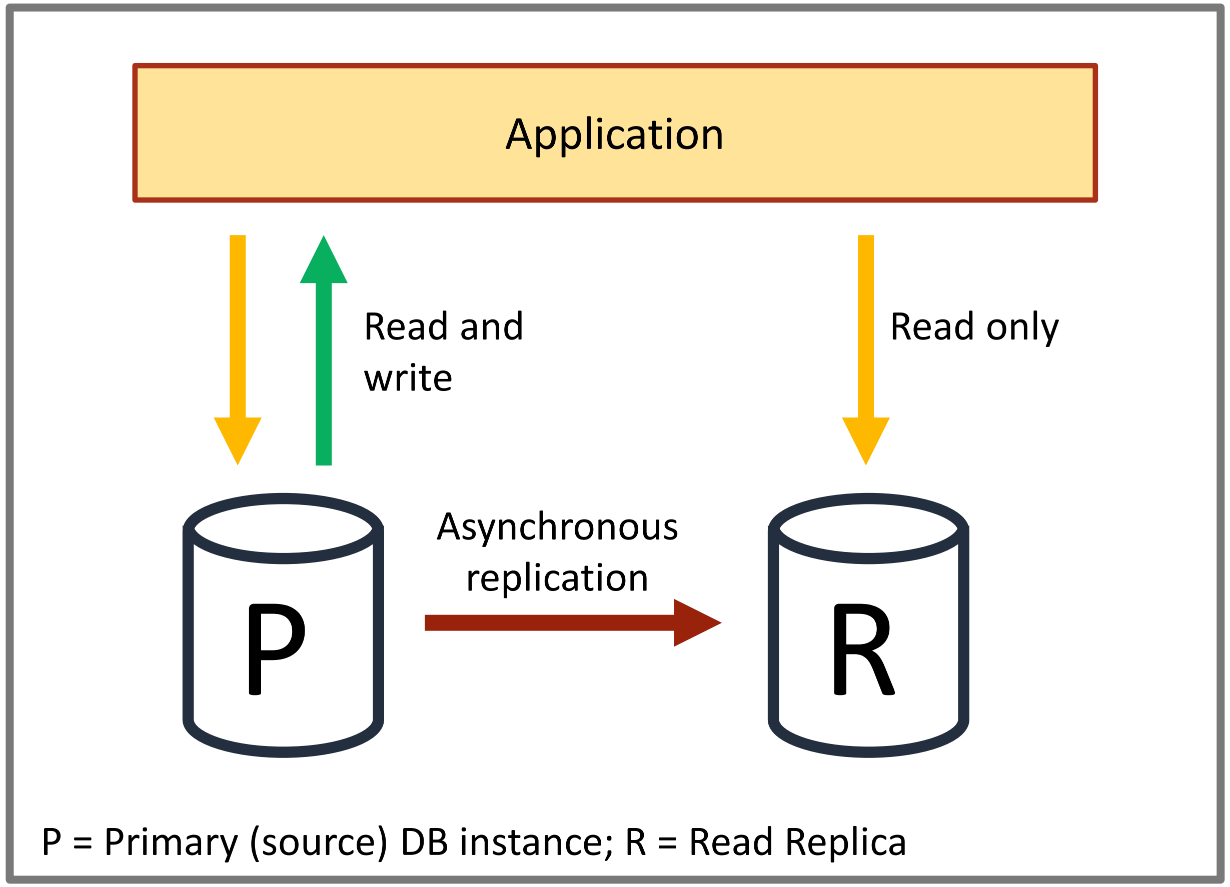
Scaling Database Resource

Vertical scaling with Amazon RDS: Push-button scaling

- Scale DB instances vertically up or down
- From micro to 24xlarge and everything in between
- Scale vertically with minimal downtime



Horizontal scaling with Amazon RDS: Read replicas

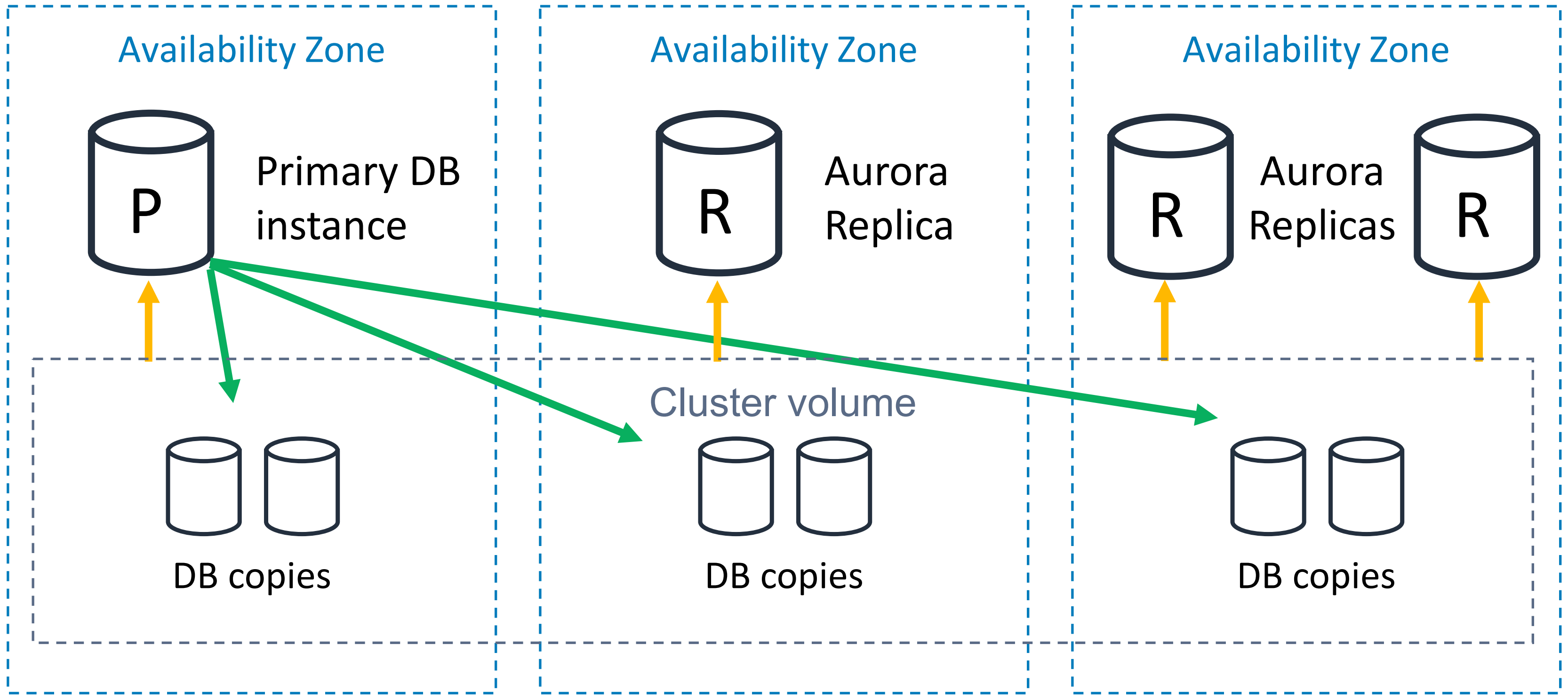


P = Primary (source) DB instance; R = Read Replica

- Horizontally scale for read-heavy workloads
- Up to five read replicas and up to 15 Aurora replicas
- Replication is asynchronous
- Available for Amazon RDS for MySQL, MariaDB, PostgreSQL, and Oracle

Scaling with Amazon Aurora

Each Aurora DB cluster can have up to 15 Aurora replicas



THE UNIVERSITY OF
SYDNEY