

# AWS Cloud Technology and Services

AWS CLOUD TECHNOLOGY AND SERVICES CONCEPTS



**Alex Kuntz**

Head of Cloud Curriculum, DataCamp

# Chapter 1: Compute Services



**AWS EC2**



**AWS Lambda**

# Chapter 2: Databases, networking, and storage



**AWS DynamoDB**



**AWS S3**

# Chapter 3: AI, machine learning, and more

## Artificial Intelligence services



Amazon Translate



Amazon Polly



Amazon Lex



Amazon Comprehend



Amazon Forecast



Amazon Rekognition




Amazon CodeGuru


# Prerequisites


- No technical experience needed
- A basic understanding of AWS could help


INTERACTIVE COURSE


## Introduction to AWS


Continue Bookmark

 Beginner

 2 hours

 12 videos

 39 exercises

 10,584 participants

2650 XP

# Course format

aws

Services

Search

[Option+S]

N. Virginia

datacamp-learner-user @ 3397-1279-7442

Console Home

Info

Reset to default layout

+ Add widgets

Recently visited

Info

S3

AWS Billing Conductor

Billing and Cost Management

IAM

Lambda

EC2

Certificate Manager

IAM Identity Center

Secrets Manager

Trusted Advisor

Security Hub

CloudWatch

View all services

Applications (0)

Info

Create application

Region: US East (N. Virginia)

us-east-1 (Current Region)

Find applications

< 1 >

Name	Description	Region	Originating account
No applications Get started by creating an application.			
Create application			

Go to myApplications

Welcome to AWS

AWS Health

Info

Cost and usage

Info

# Let's practice!

AWS CLOUD TECHNOLOGY AND SERVICES CONCEPTS

# Amazon Elastic Compute Cloud

AWS CLOUD TECHNOLOGY AND SERVICES CONCEPTS



**Alex Kuntz**

Head of Cloud Curriculum, DataCamp

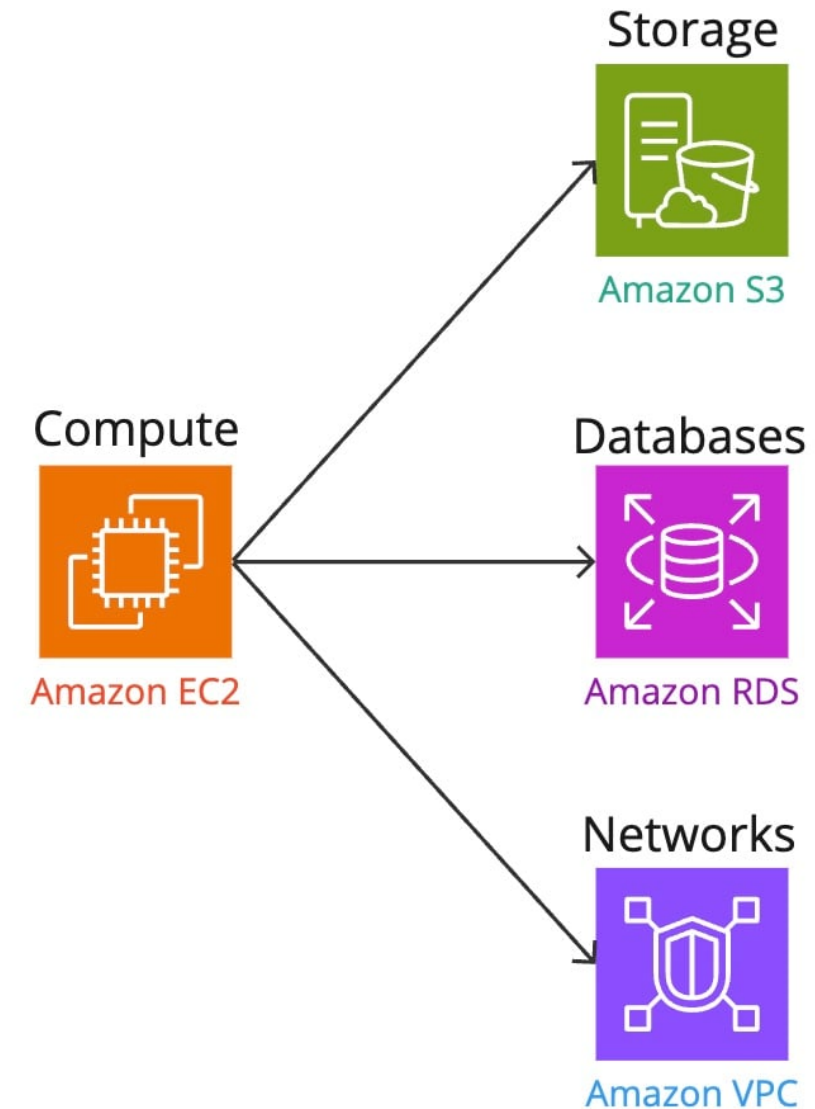


# What is Amazon Elastic Compute Cloud (EC2)?

- Provides resizable compute capacity in the cloud
- Each individual EC2 machine is referred to as an instance

## Key characteristics:

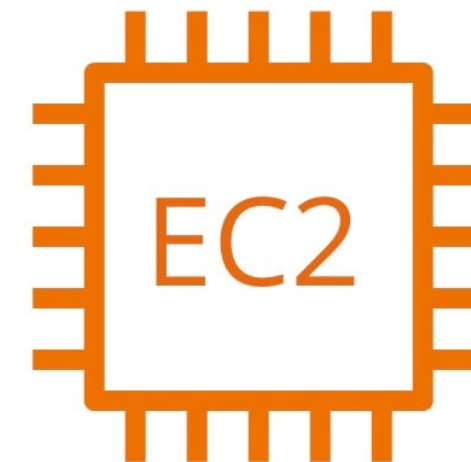
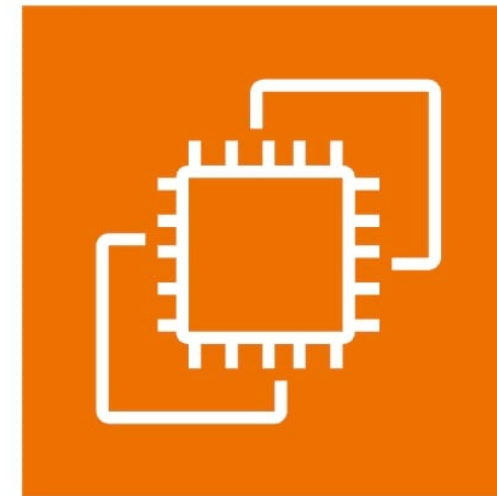
- Ability to scale up or down based on demand
- Availability of a varied range of instance types for specialized use cases



# EC2 instance types

AWS offers six categories of EC2 instances for specialized workloads

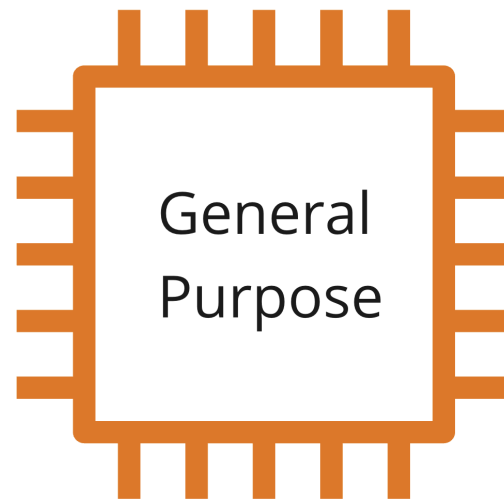
- General purpose
- Compute optimized
- Memory optimized
- Storage optimized
- Accelerated computing
- High Performance Computing (HPC) optimized



# General purpose and storage optimized instances

## General purpose instances

- Balance of compute, memory, and networking resources
- Use cases:
  - Hosting dynamic websites
  - Maintaining code repositories



## Storage optimized instances

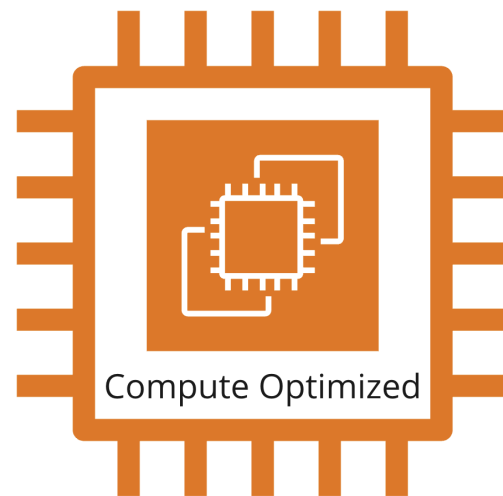
- High, sequential read and write access to large datasets
- Use cases:
  - Data warehousing
  - Refactoring large relational databases



# Compute and memory optimized instances

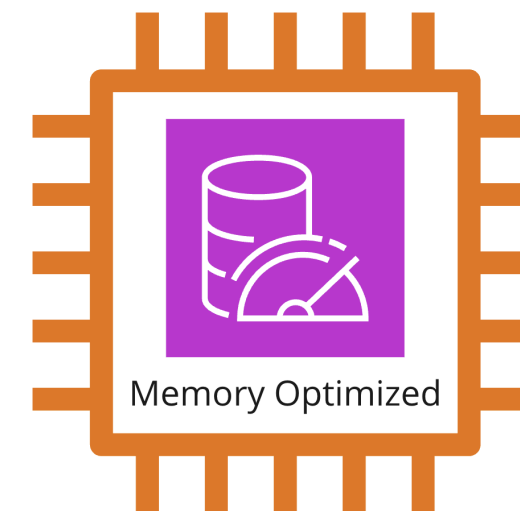
## Compute optimized instances

- Compute-intensive and high-performance workloads
- Use cases:
  - Scientific simulations
  - Financial modeling



## Memory optimized instances

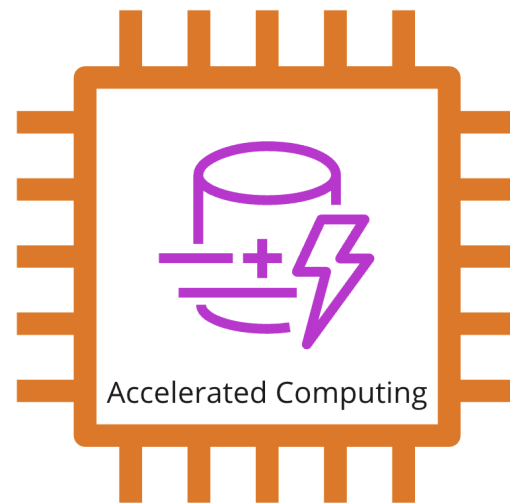
- Memory-intensive workloads not requiring high storage
- Use cases:
  - Real-time stream data analytics
  - Generating close captions



# Specialized compute instances

## Accelerated computing instances

- Contain specialized hardware accelerators, like GPUs or FPGAs
- Use cases:
  - Deep learning
  - Rendering gaming graphics

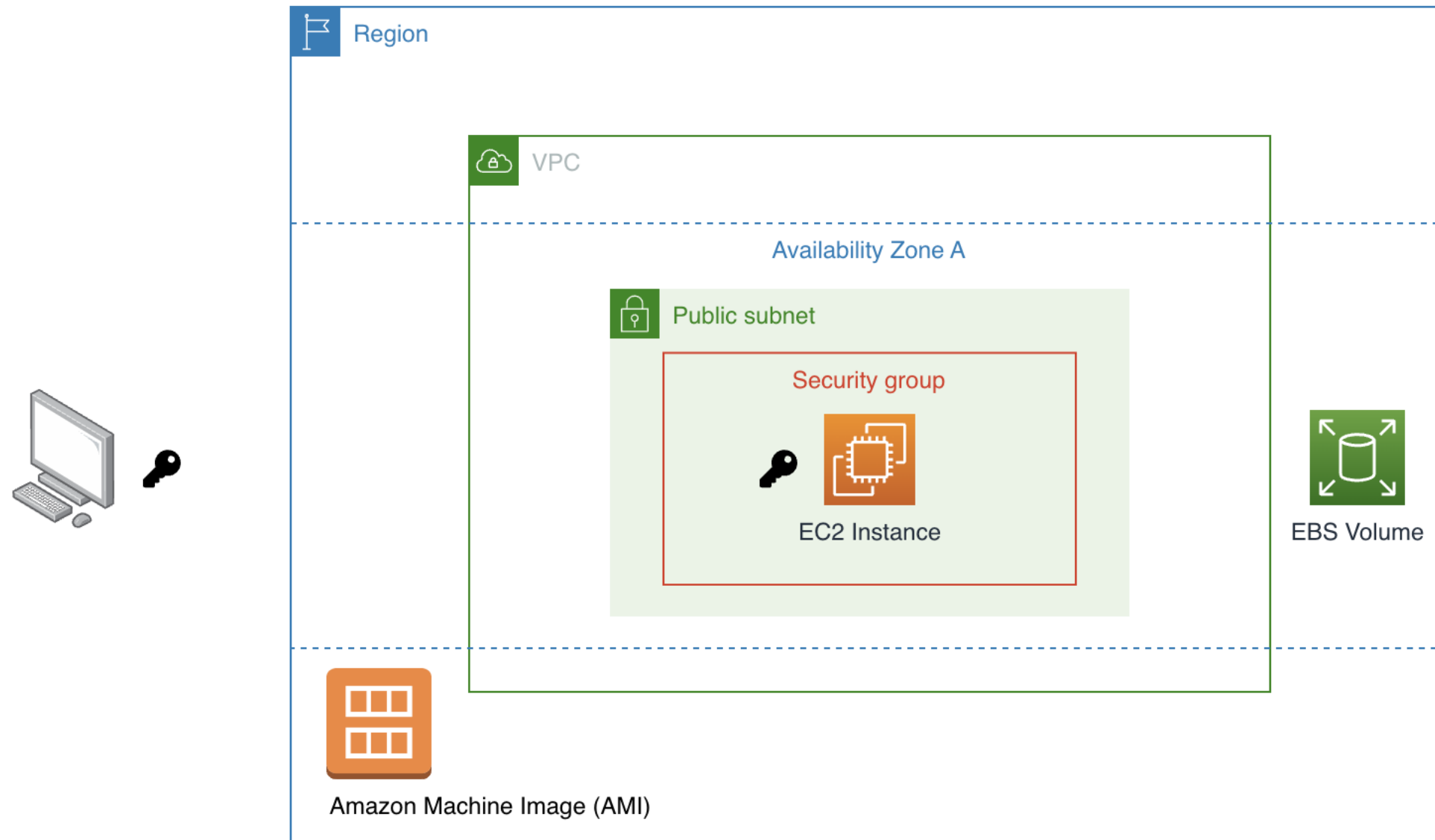


## HPC optimized instances

- Best price performance for running high performance workloads at scale
- Use cases:
  - Weather forecasting
  - Crash simulations



# Creating your EC2 instance



# Connecting to your EC2 instance: SSH Client

- SSH connects with a private key
- Those keys must be managed

## Connect to instance [Info](#)

Connect to your Instance `i-XXXXXXXXXX` (MyWebServer) using any of these options

[EC2 Instance Connect](#) | [Session Manager](#) | [SSH client](#) | [EC2 serial console](#)

Instance ID

`i-XXXXXXXXXX` (MyWebServer)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is `Test.pem`
3. Run this command, if necessary, to ensure your key is not publicly viewable.  
`chmod 400 "Test.pem"`
4. Connect to your Instance using its Public DNS:  
`ec2-XXXXXXXXXX.compute-1.amazonaws.com`

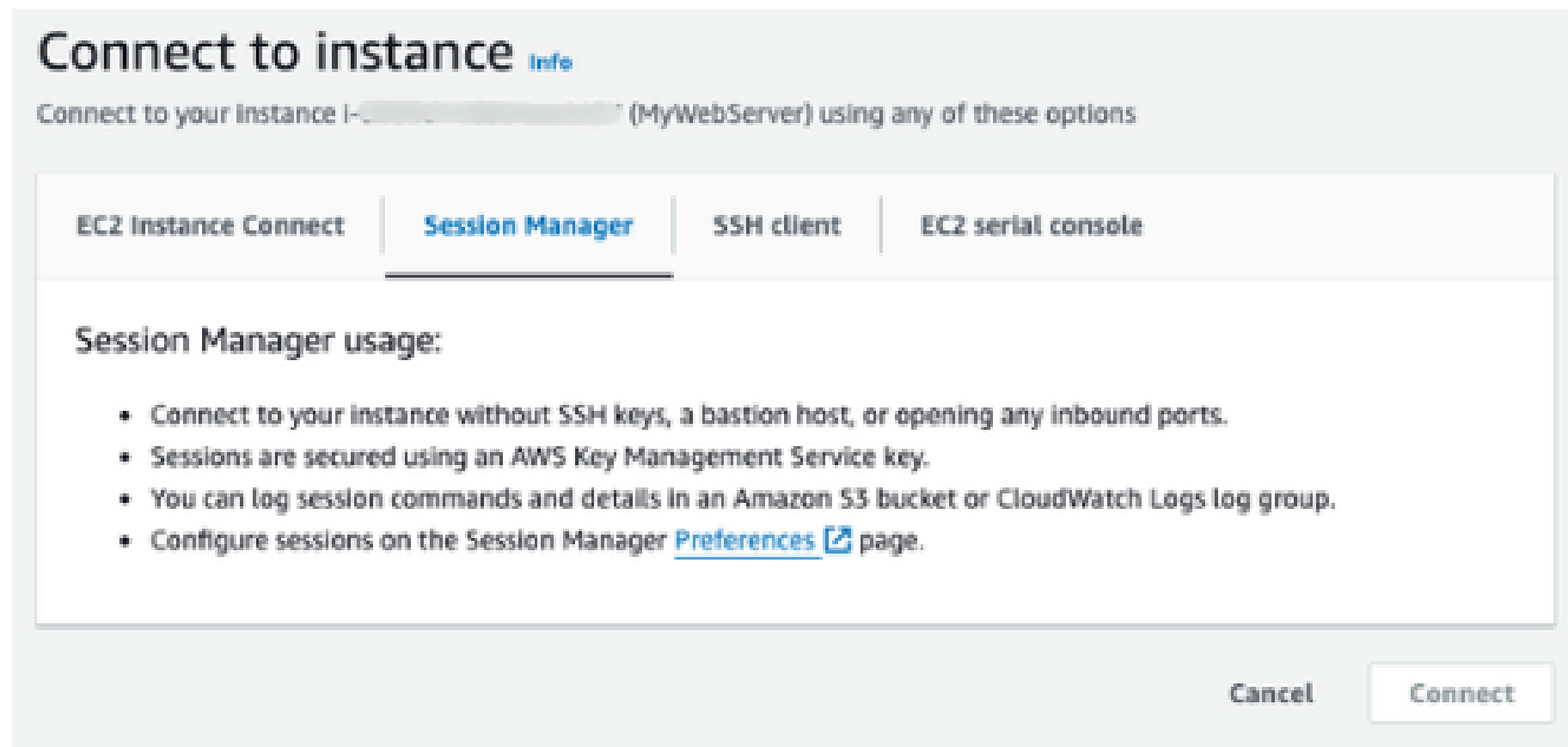
Example:

```
ssh -i "Test.pem" ec2-user@ec2-XXXXXXXXXX.compute-1.amazonaws.com
```

**Note:** In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

# Connecting to your EC2 instance: AWS Session Manager

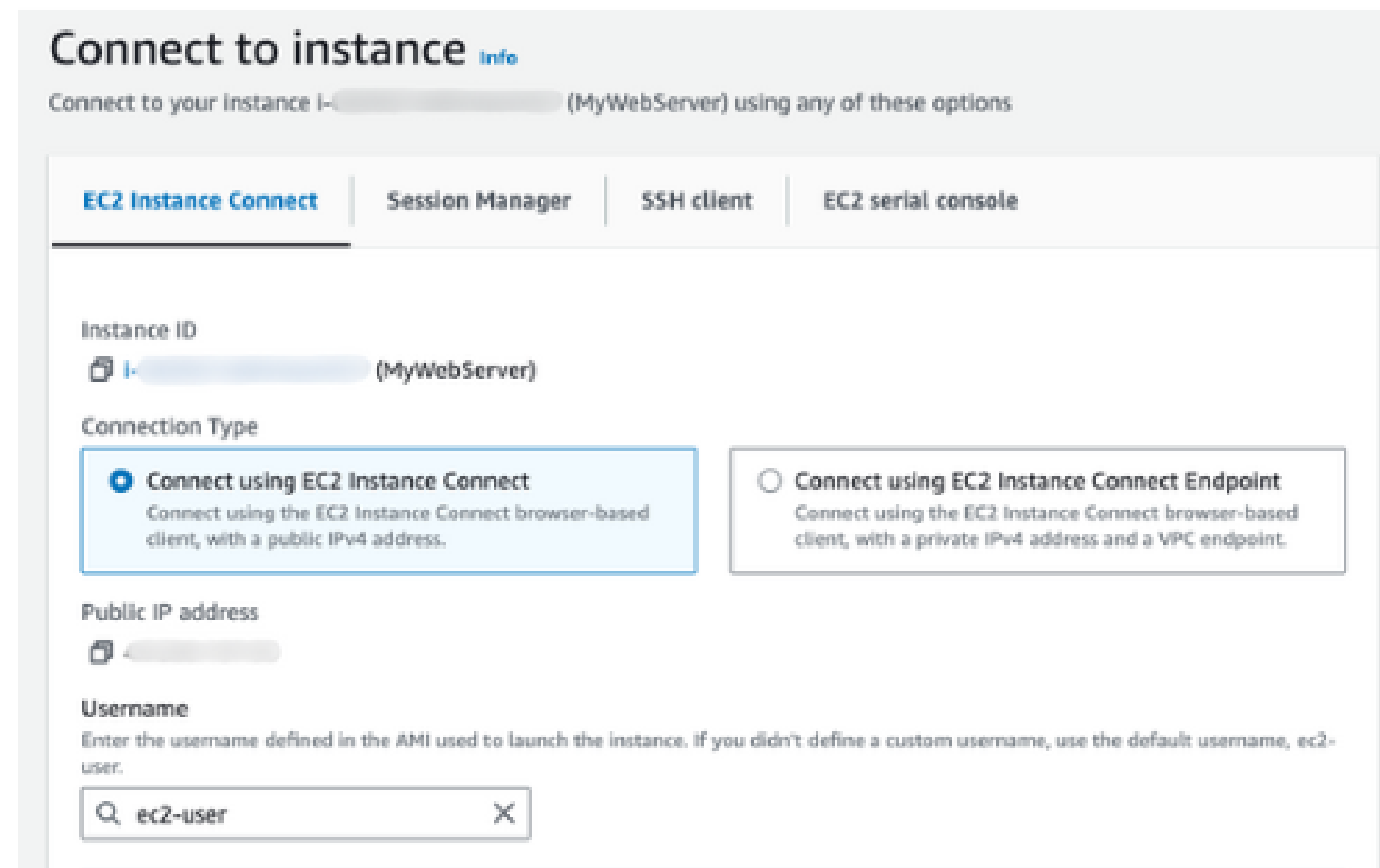
- Keyless access via the Management Console
- Integrates with Identity and Access Management (IAM)





# Connecting to your EC2 instance: EC2 Instance Connect

- Browser-based connection
- Quick and temporary access



**Connect to instance** [Info](#)

Connect to your instance **i-12345678** (MyWebServer) using any of these options

**EC2 Instance Connect** | Session Manager | SSH client | EC2 serial console

Instance ID  
**i-12345678** (MyWebServer)

Connection Type

☒ **Connect using EC2 Instance Connect**  
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

☐ **Connect using EC2 Instance Connect Endpoint**  
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address  
**44.192.164.123**

Username  
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

# Let's practice!

AWS CLOUD TECHNOLOGY AND SERVICES CONCEPTS

# Load Balancing and Auto-scaling

AWS CLOUD TECHNOLOGY AND SERVICES CONCEPTS



**Alex Kuntz**

Head of Cloud Curriculum, DataCamp

# Load balancing in AWS

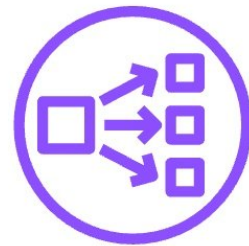
Load balancing ensures even distribution of incoming traffic among multiple EC2 instances, preventing overload on a single server

- Ensures high availability
- Provides horizontal scaling

## Types of load balancers in AWS



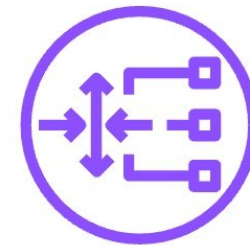
Classic Load Balancer



Network Load Balancer



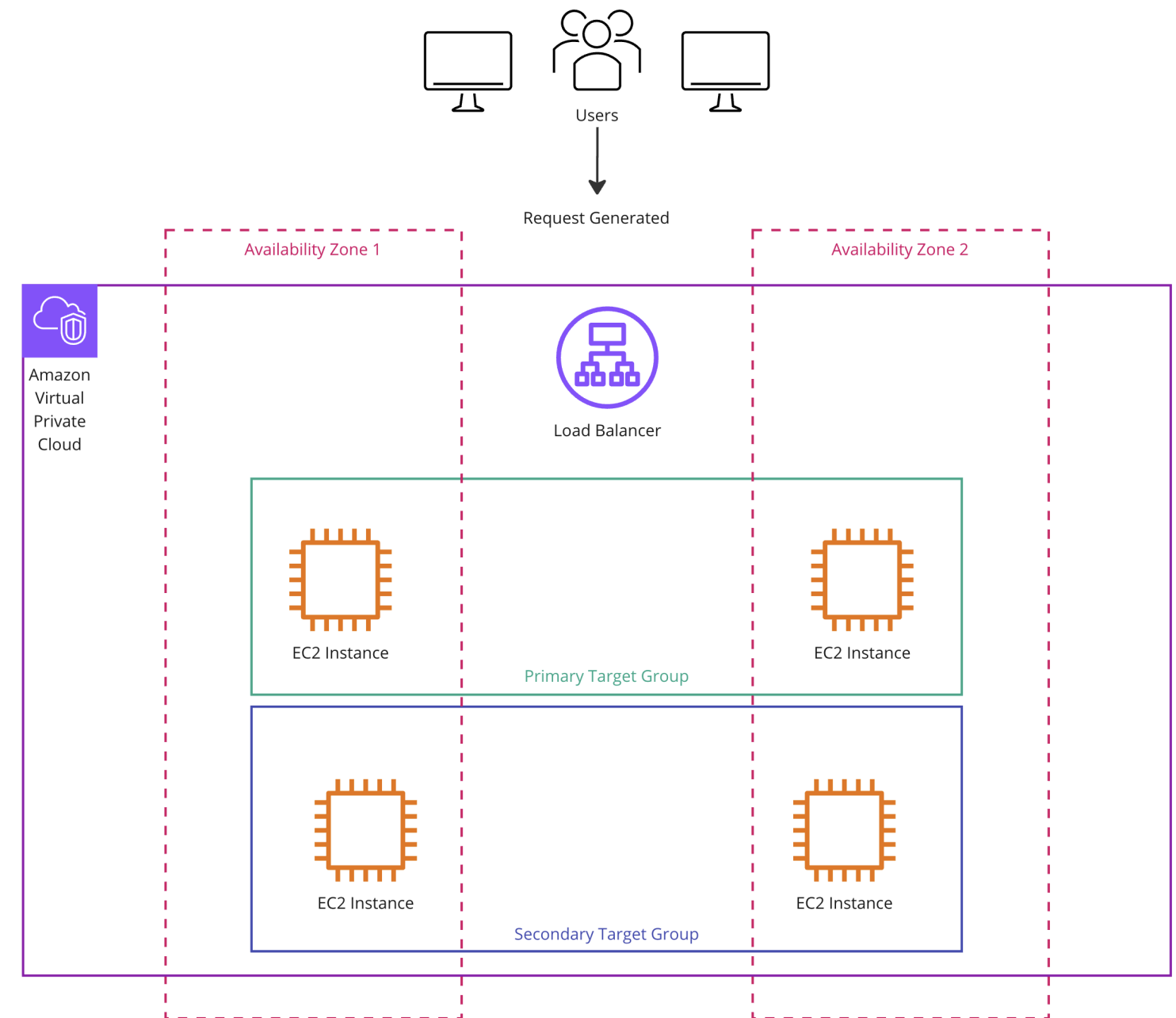
Application Load Balancer



Gateway Load Balancer

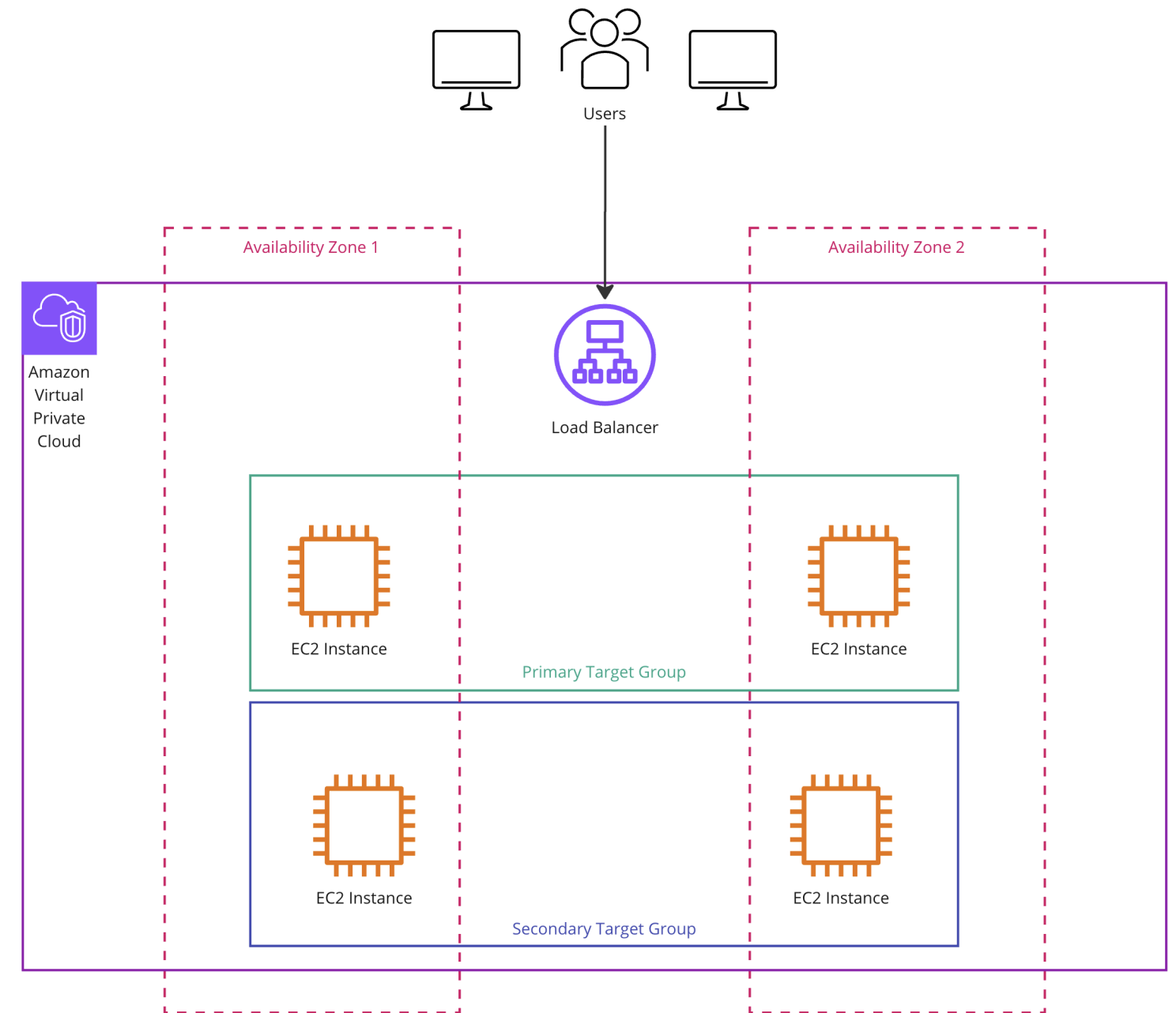
# How does load balancing work?

## 1. Users send requests



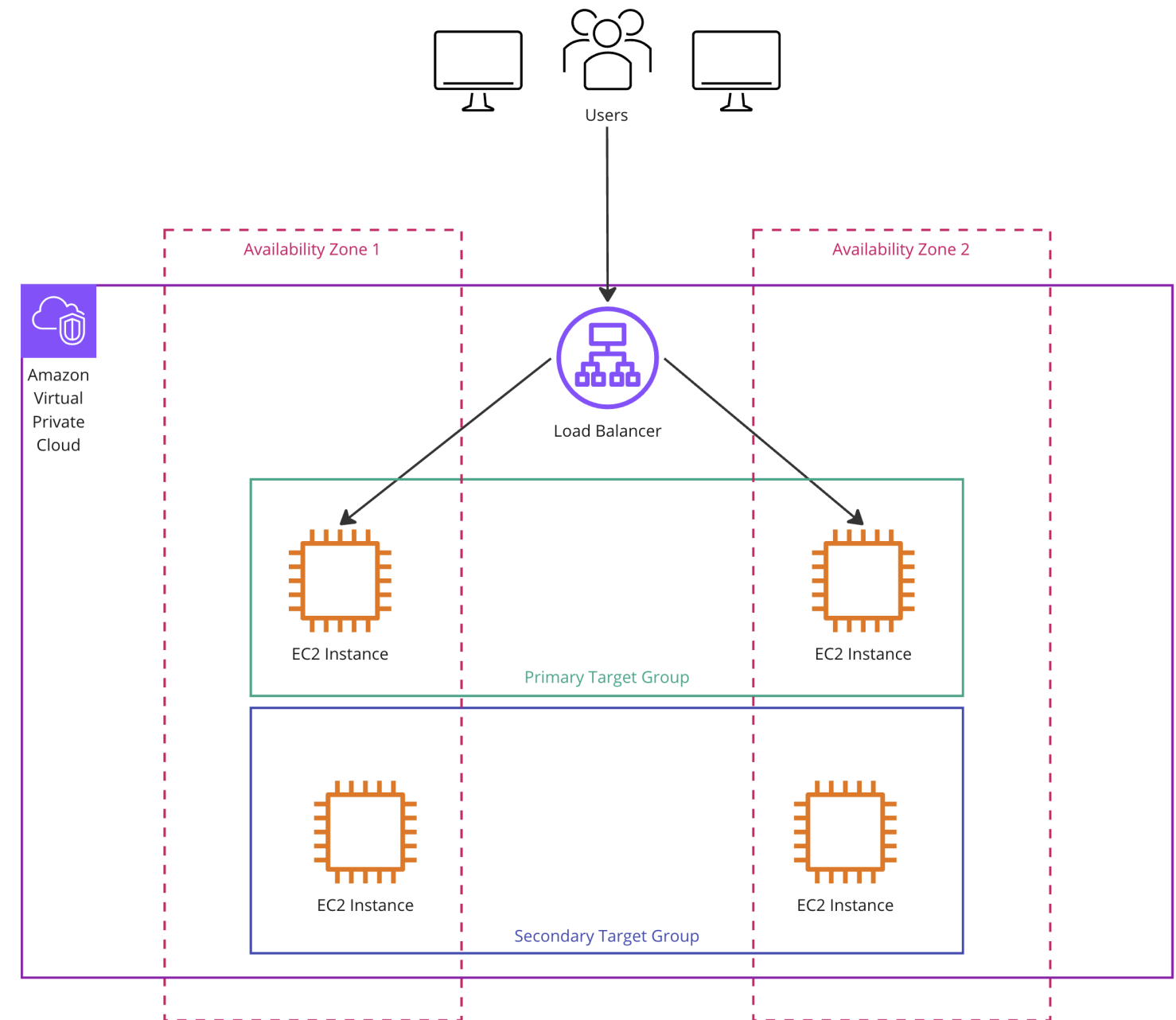
# How does load balancing work?

1. Users send requests
2. Requests hit the load balancer



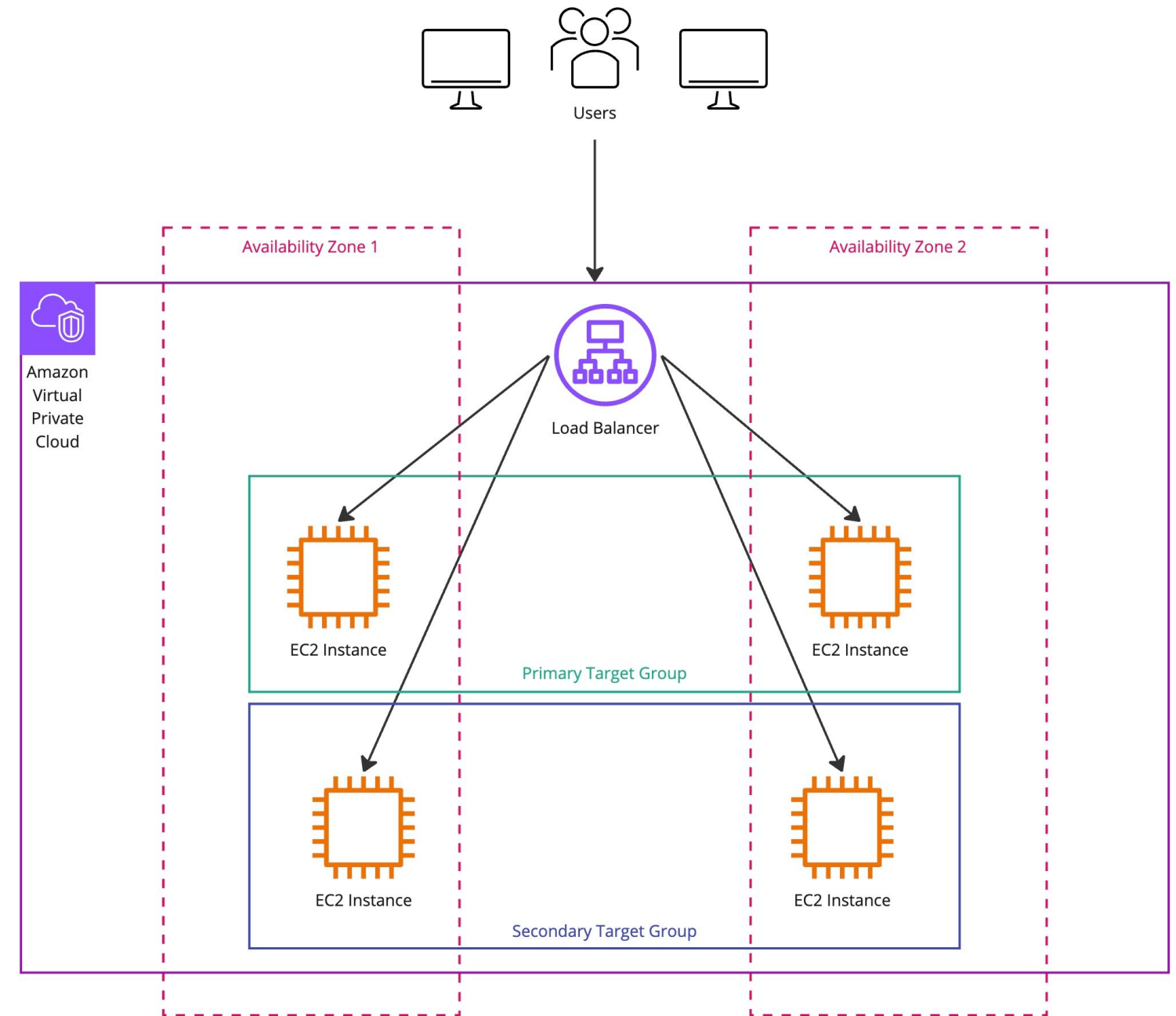
# How does load balancing work?

1. Users send requests
2. Requests hit the load balancer
3. Primary target group is instantiated by the application load balancer first



# How does load balancing work?

1. Users send requests
2. Requests hit the load balancer
3. Primary target group is instantiated by the application load balancer first
4. If demand increases, the load balancer activates the secondary target group and distributes the load across all instances





# What is compute elasticity?

Elasticity ensures your system can scale up or down based on demand, providing flexibility in resource allocation

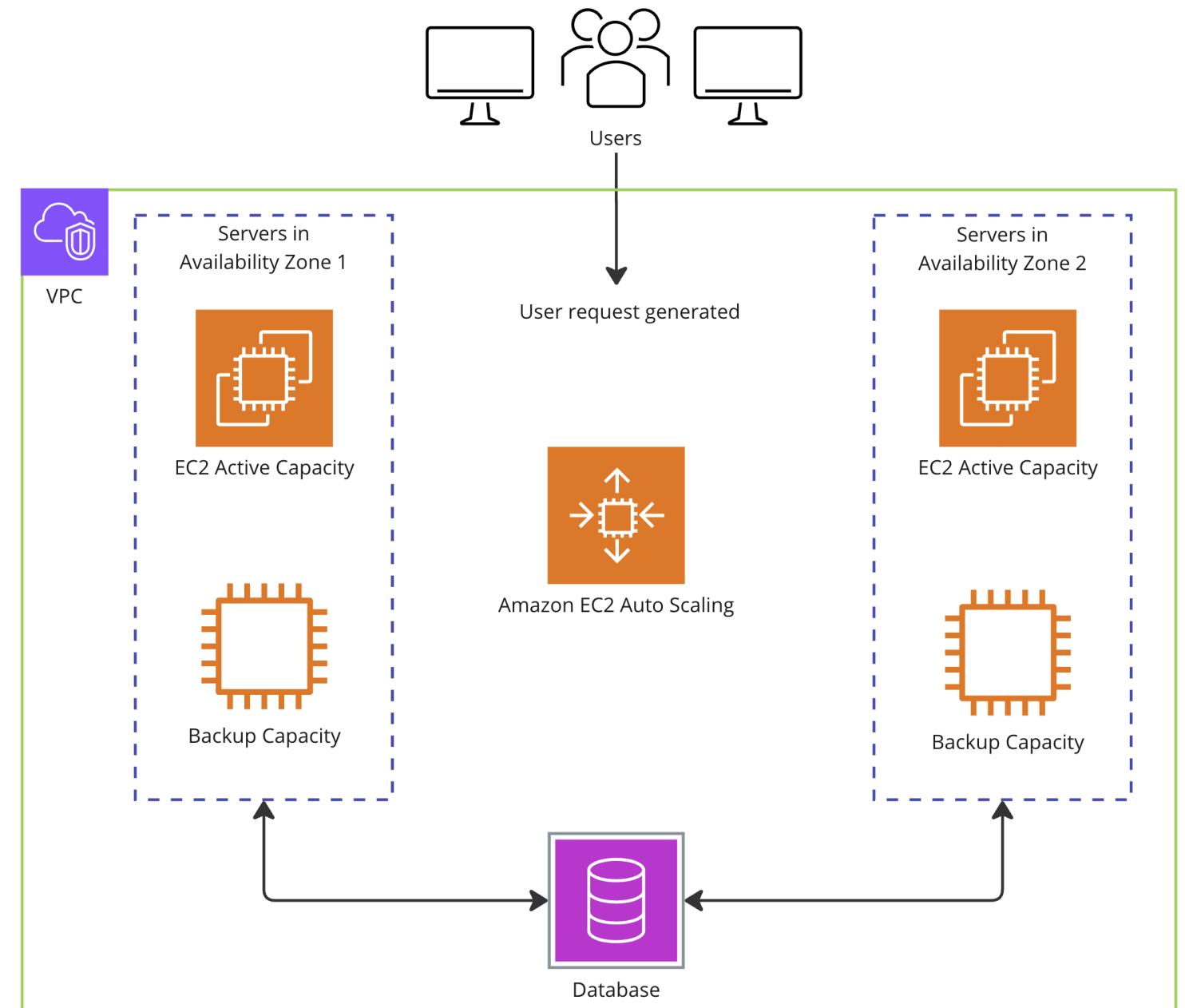
- EC2 instances achieve elasticity through EC2 Auto Scaling

## What is EC2 Auto Scaling?

- Automatically adjust the number of active instances based on usage and requirement
- Optimize costs
- Prevent over-provisioning

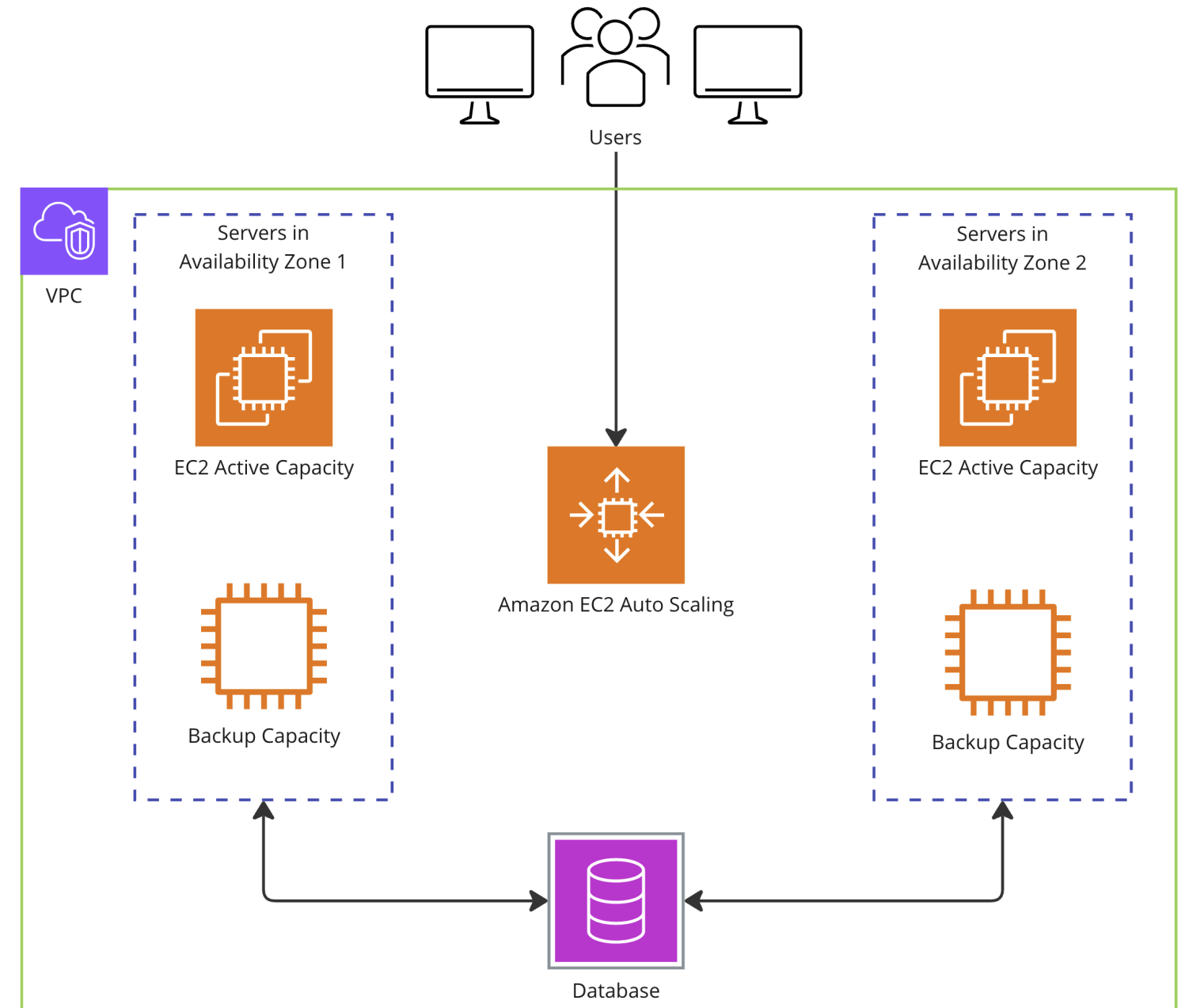
# How does auto-scaling work?

## 1. Users send requests



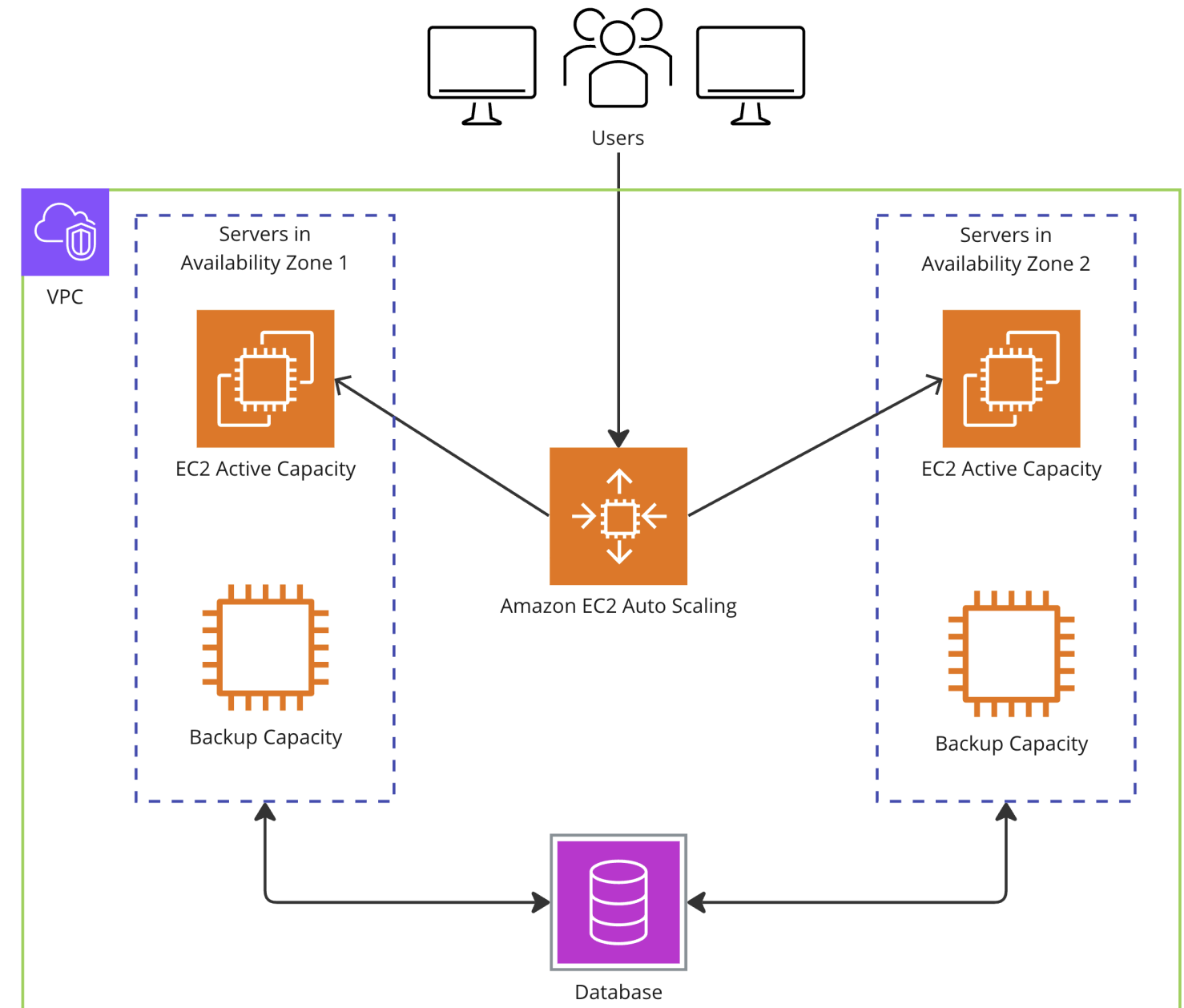
# How does auto-scaling work?

1. Users send requests
2. The requests are routed to EC2 Auto Scaling service



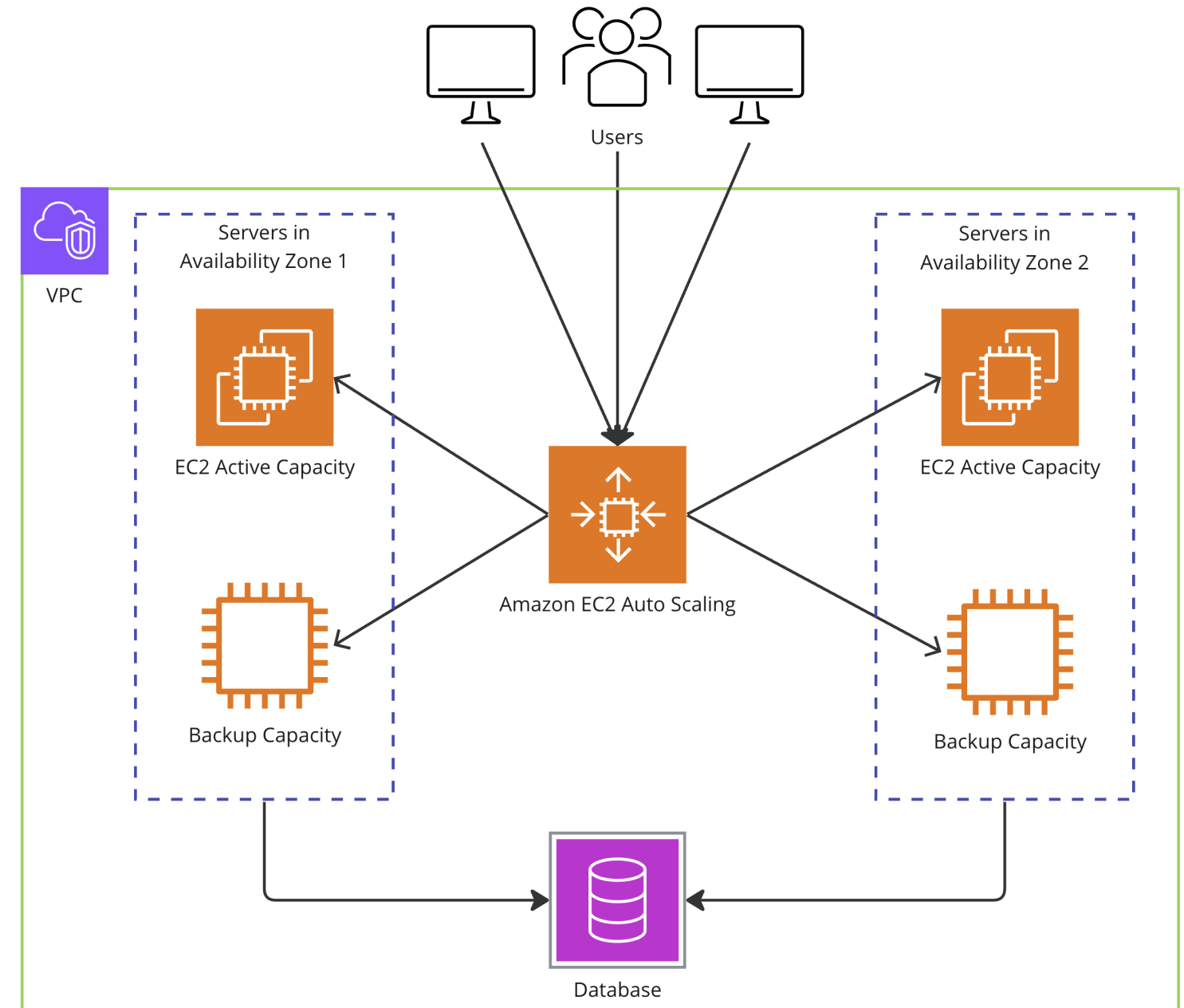
# How does auto-scaling work?

1. Users send requests
2. The requests are routed to EC2 Auto Scaling service
3. The service then routes requests to the active EC2 instances



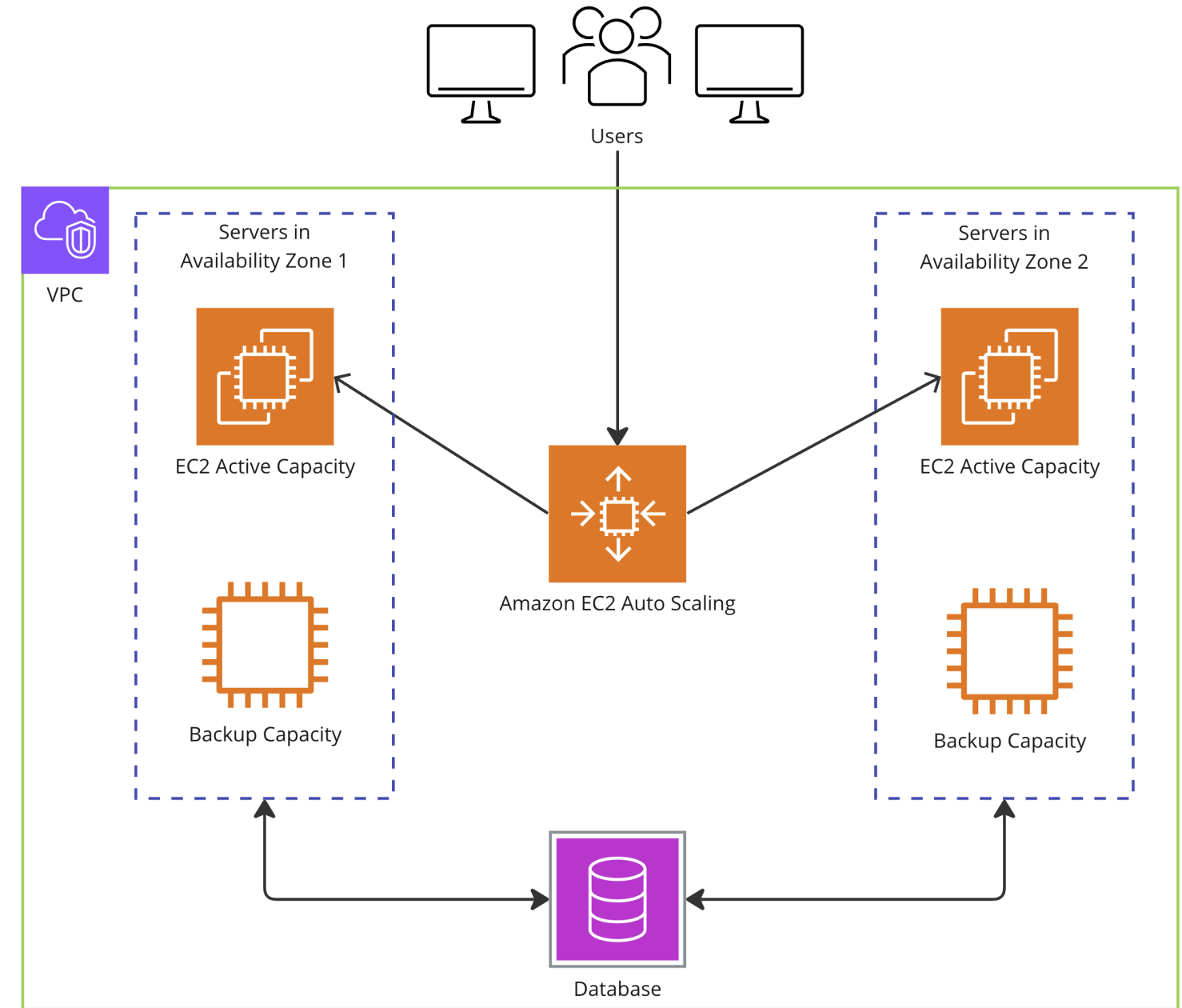
# How does auto-scaling work?

1. Users send requests
2. The requests are routed to EC2 Auto Scaling service
3. The service then routes requests to the active EC2 instances
4. If demand increases, it starts adding new EC2 instances to manage the additional load



# How does auto-scaling work?

1. Users send requests
2. The requests are routed to EC2 Auto Scaling service
3. The service then routes requests to the active EC2 instances
4. If demand increases, it starts adding new EC2 instances to manage the additional load
5. As demand goes down, the newly added EC2 instances are shut down



# Load balancing vs. auto-scaling

## Load balancing

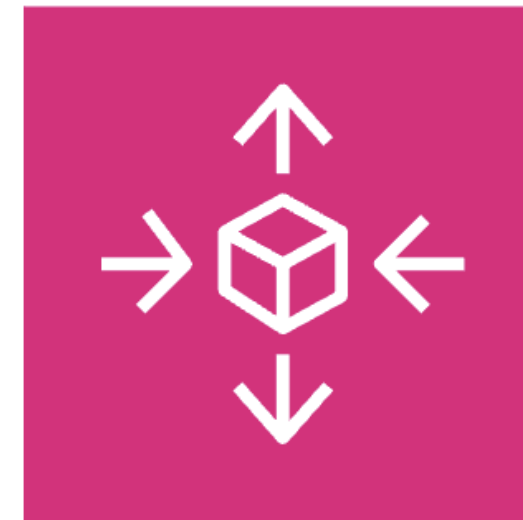
- Route traffic evenly
- Utilize existing EC2 instances



AWS Load Balancer

## Auto-scaling

- Ensure demand is always met
- Ability to add/remove EC2 instances



EC2 Auto Scaling

# Let's practice!

AWS CLOUD TECHNOLOGY AND SERVICES CONCEPTS



# Serverless Compute

AWS CLOUD TECHNOLOGY AND SERVICES CONCEPTS



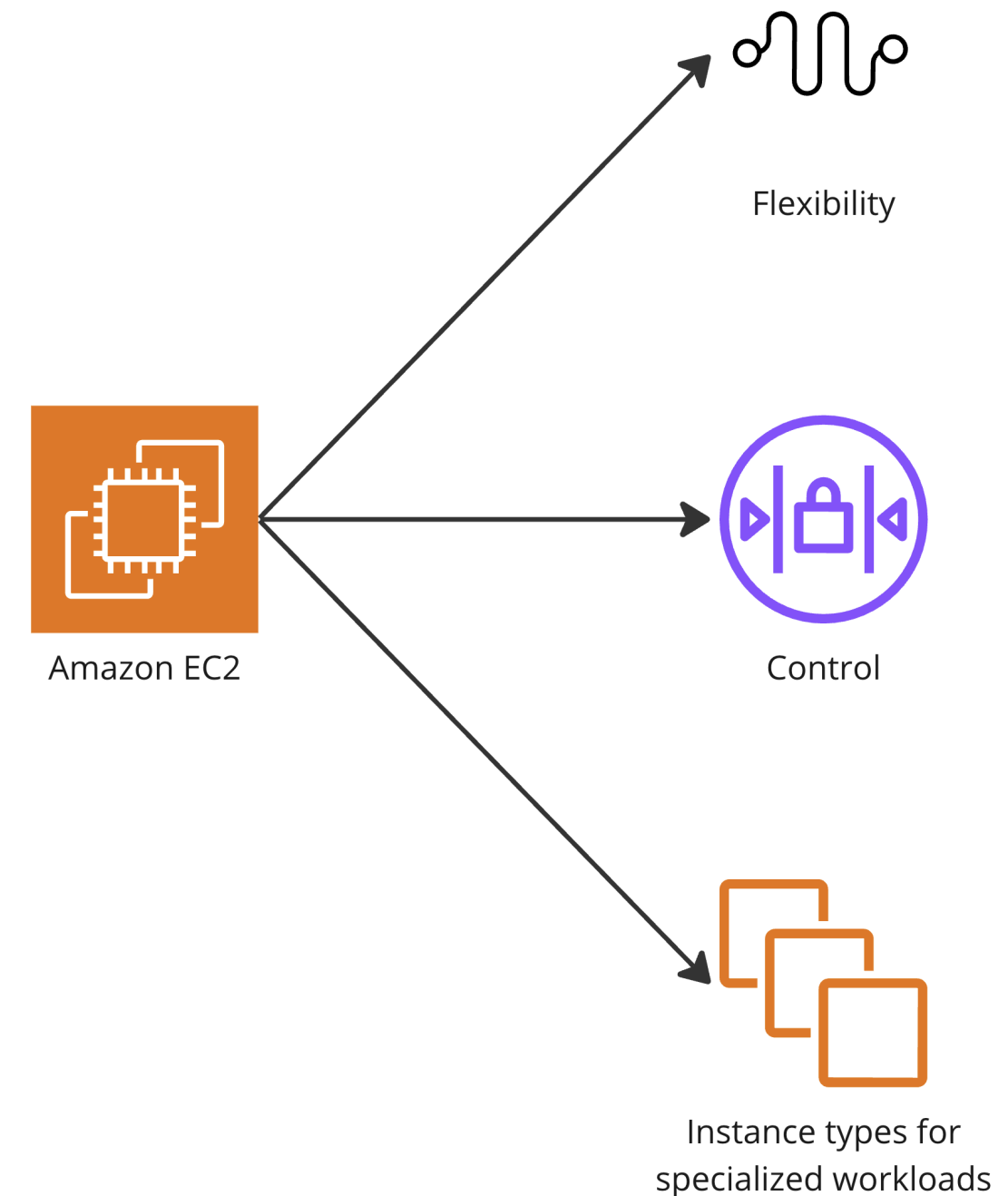
**Alex Kuntz**

Head of Cloud Curriculum, DataCamp

# Recap traditional compute

## EC2 Recap:

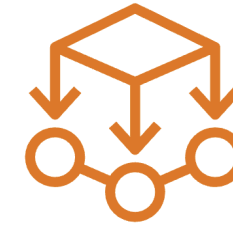
- Amazon EC2 is a service that provides compute capacity in the AWS cloud
- Using EC2 gives higher flexibility and control
- Variety of EC2 instance types optimized for different workloads



# Evolving needs: beyond traditional compute

## Today's Demands:

- Need for modular, microservices architectures

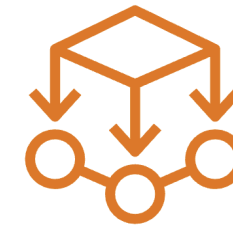


Microservice architecture

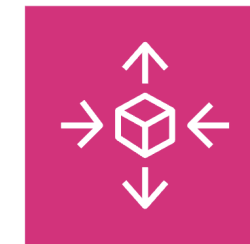
# Evolving needs: beyond traditional compute

## Today's Demands:

- Need for modular, microservices architecture
- Rapid scaling capabilities to meet fluctuating demands



Microservice architecture

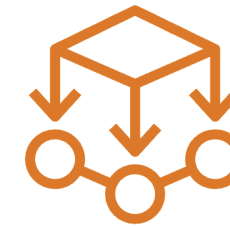


Rapid scaling

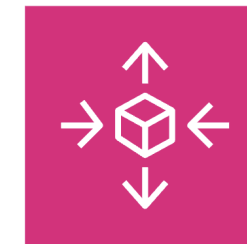
# Evolving needs: beyond traditional compute

## Today's Demands:

- Need for modular, microservices architecture
- Rapid scaling capabilities to meet fluctuating demands
- Automated infrastructure management setup without manual interventions



Microservice architecture



Rapid scaling

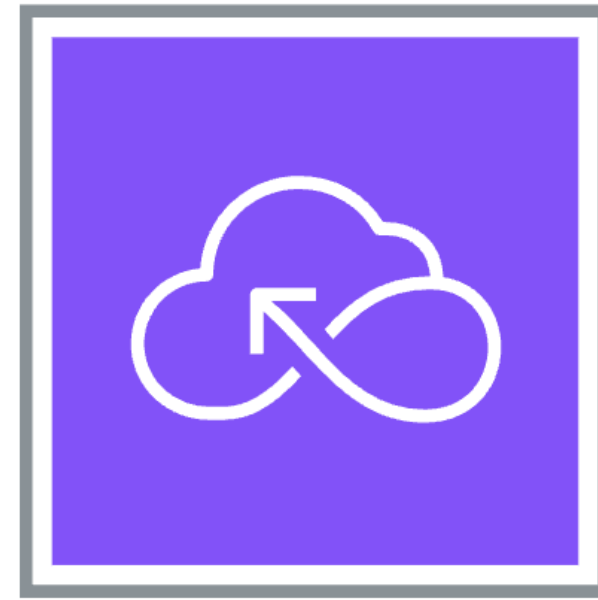


Automated infrastructure management

# Containers and serverless compute



Containers



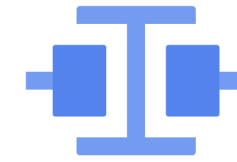
Serverless compute

# What are containers?

Containers encapsulate applications and their dependencies, in lightweight singular units

## Why containers?

- Isolate applications from underlying system dependencies



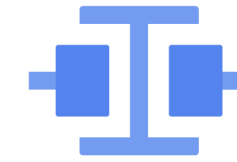
Isolation

# What are containers?

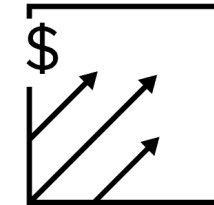
Containers encapsulate applications and their dependencies, in lightweight singular units

## Why containers?

- Isolate applications from underlying system dependencies
- Share host OS for efficient resource utilization



Isolation



Increased efficiency

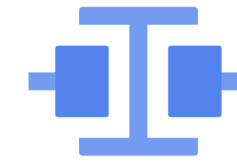


# What are containers?

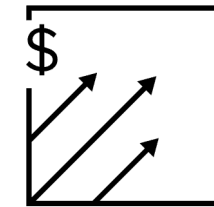
Containers encapsulate applications and their dependencies, in lightweight singular units

## Why containers?

- Isolate applications from underlying system dependencies
- Share host OS for efficient resource utilization
- Easily movable and portable across environments



Isolation



Increased efficiency



Portability

# Containers in AWS



Amazon Elastic Container  
Service



Amazon Elastic Kubernetes  
Service

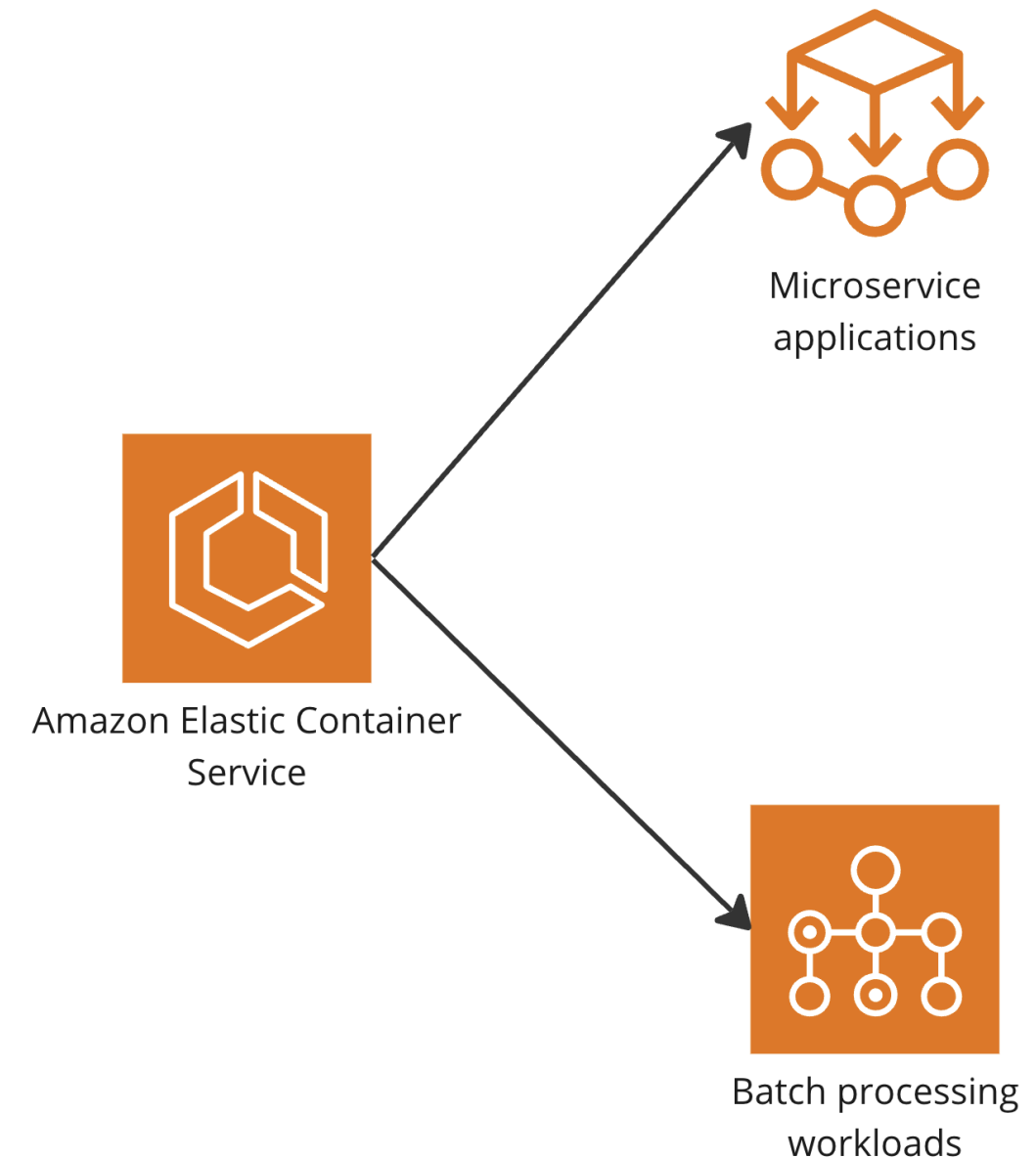
- Easily scale containerized applications up or down
- Integrate with other AWS services

# Amazon ECS

Fully managed service for efficient deployment, management, and scaling of containerized applications

## Use cases

- Deploying and managing microservices-based applications
- Plan, schedule, and run batch processing workloads across AWS services

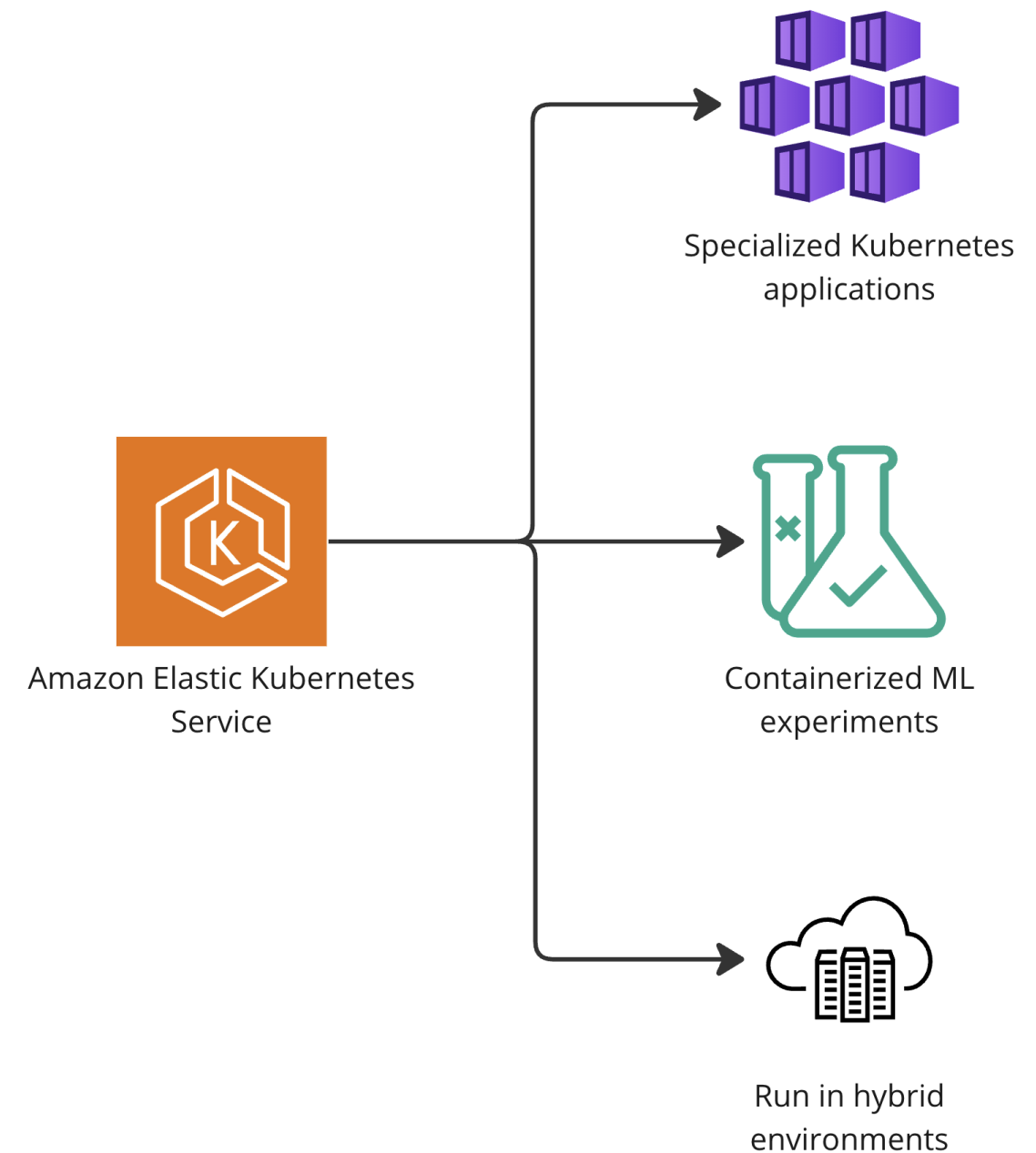


# Amazon EKS

Container orchestration service specializing in running Kubernetes-powered applications

## Use cases

- Pair with EC2 accelerated computing instances to run ML containers
- Manage clusters and applications in hybrid cloud environments



# More forms of compute?

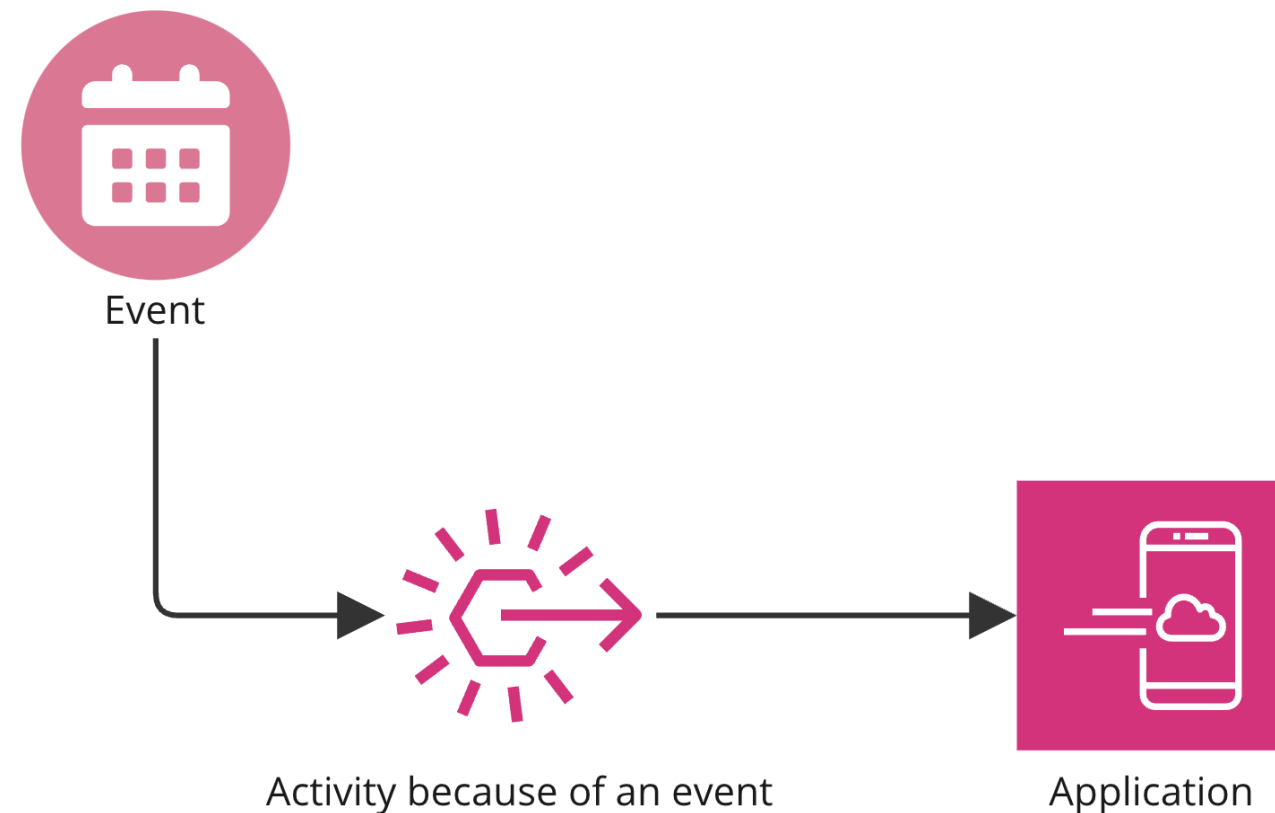


Containers

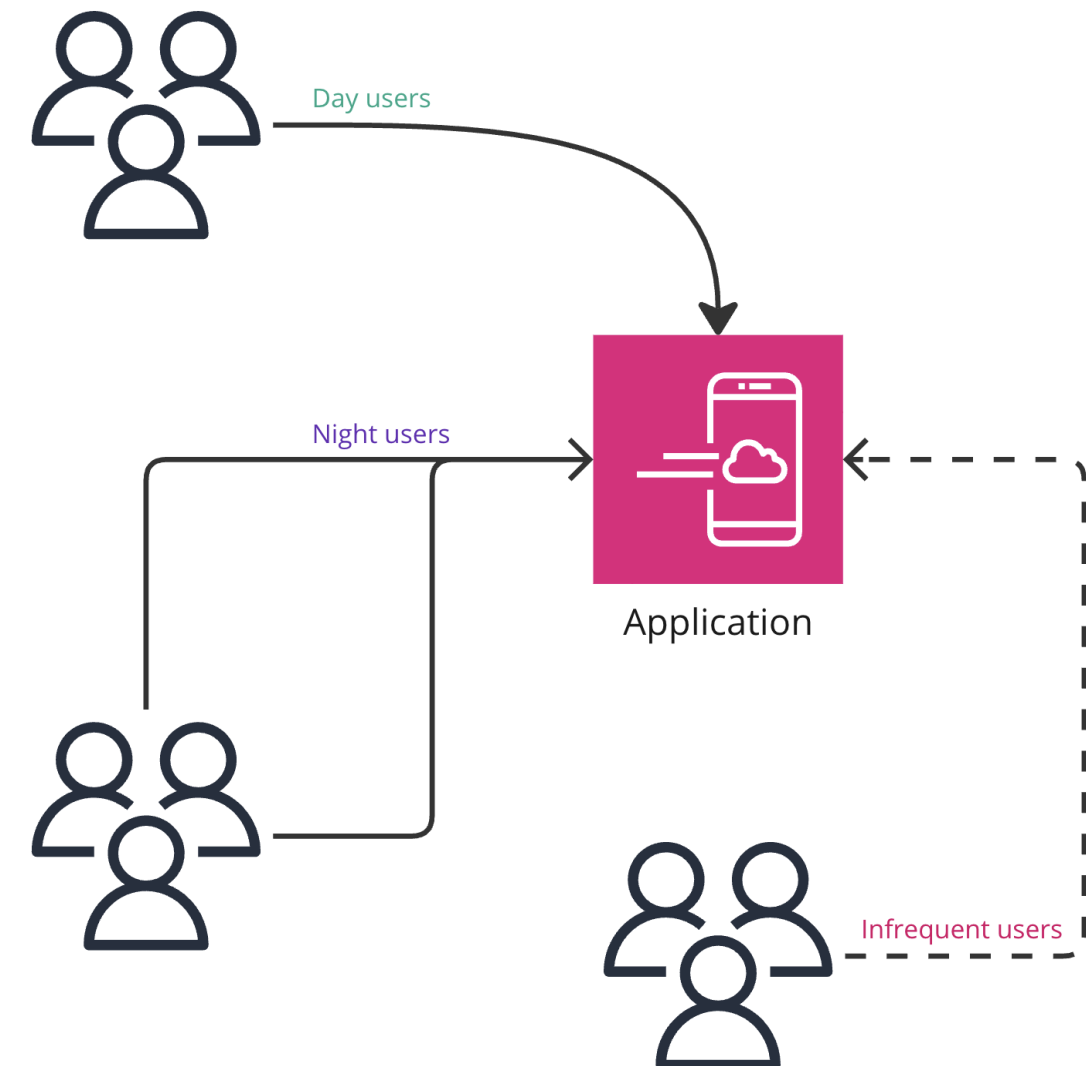
- Persistent environments
- Predictable workloads
- Resource-intensive applications

# More forms of compute?

Event-based compute changes



Balancing compute for sporadic traffic loads



# Serverless compute



Serverless compute

# What is serverless architecture?

- No server management: forget about provisioning, scaling, or maintenance



No server  
management



# What is serverless architecture?

- No server management: forget about provisioning, scaling, or maintenance
- Event-driven: functions triggered by events in real-time



No server  
management



Event-driven

# What is serverless architecture?

- No server management: forget about provisioning, scaling, or maintenance
- Event-driven: functions triggered by events in real-time
- Cost-efficient: pay only for actual usage, not pre-allocated resources



No server  
management

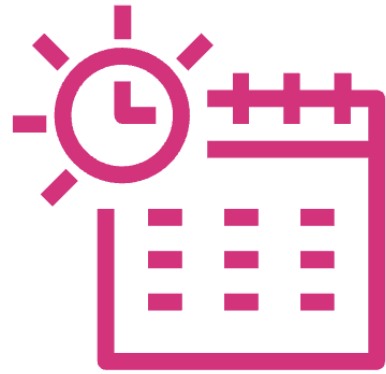


Event-driven



Cost-efficient

# When to use serverless compute?



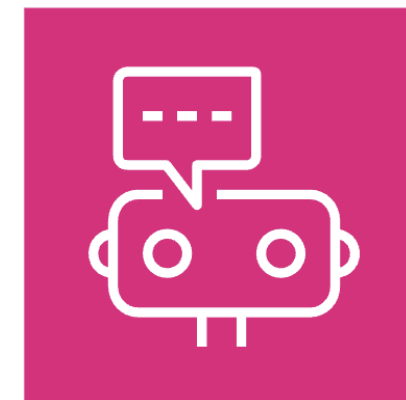
Event-driven applications



Real-time file processing



Uneven data bursts



Chatbots and voice assistants

# Serverless compute in AWS



AWS Lambda



AWS Fargate

# AWS Lambda

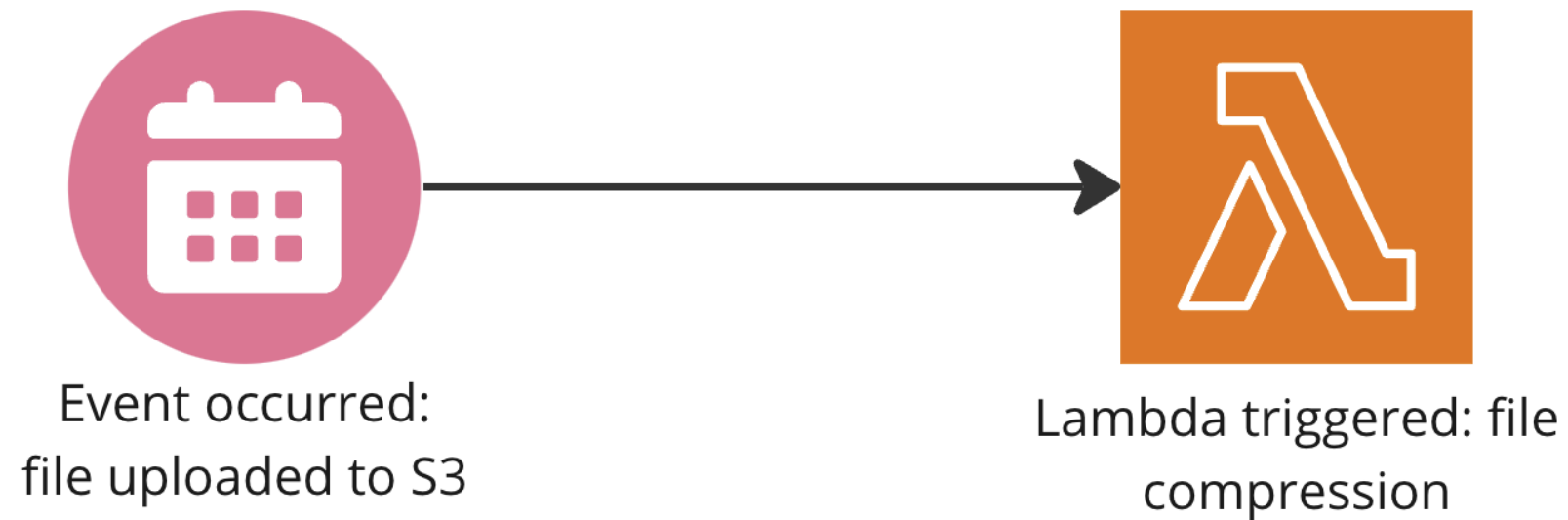
- Run code in response to events without provisioning or managing servers
- Automated compute scaling capabilities



Event occurred:  
file uploaded to S3

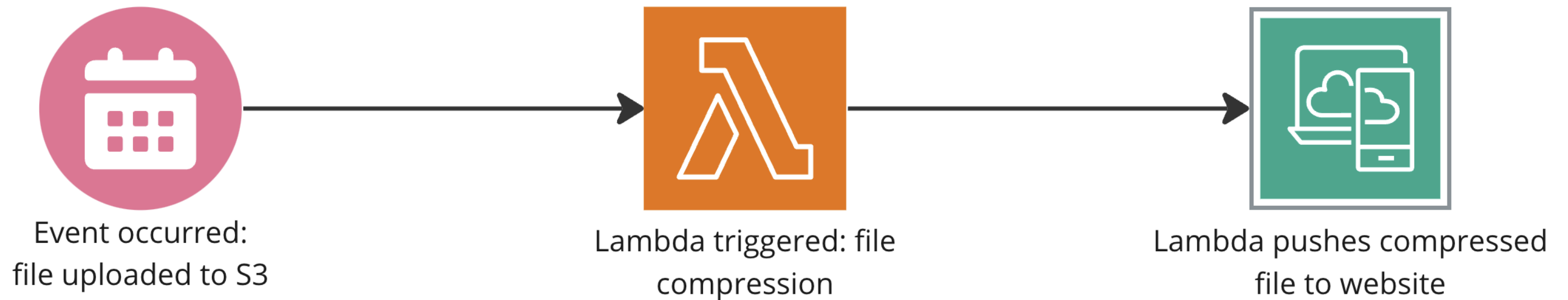
# AWS Lambda

- Run code in response to events without provisioning or managing servers
- Automated compute scaling capabilities



# AWS Lambda

- Run code in response to events without provisioning or managing servers
- Automated compute scaling capabilities

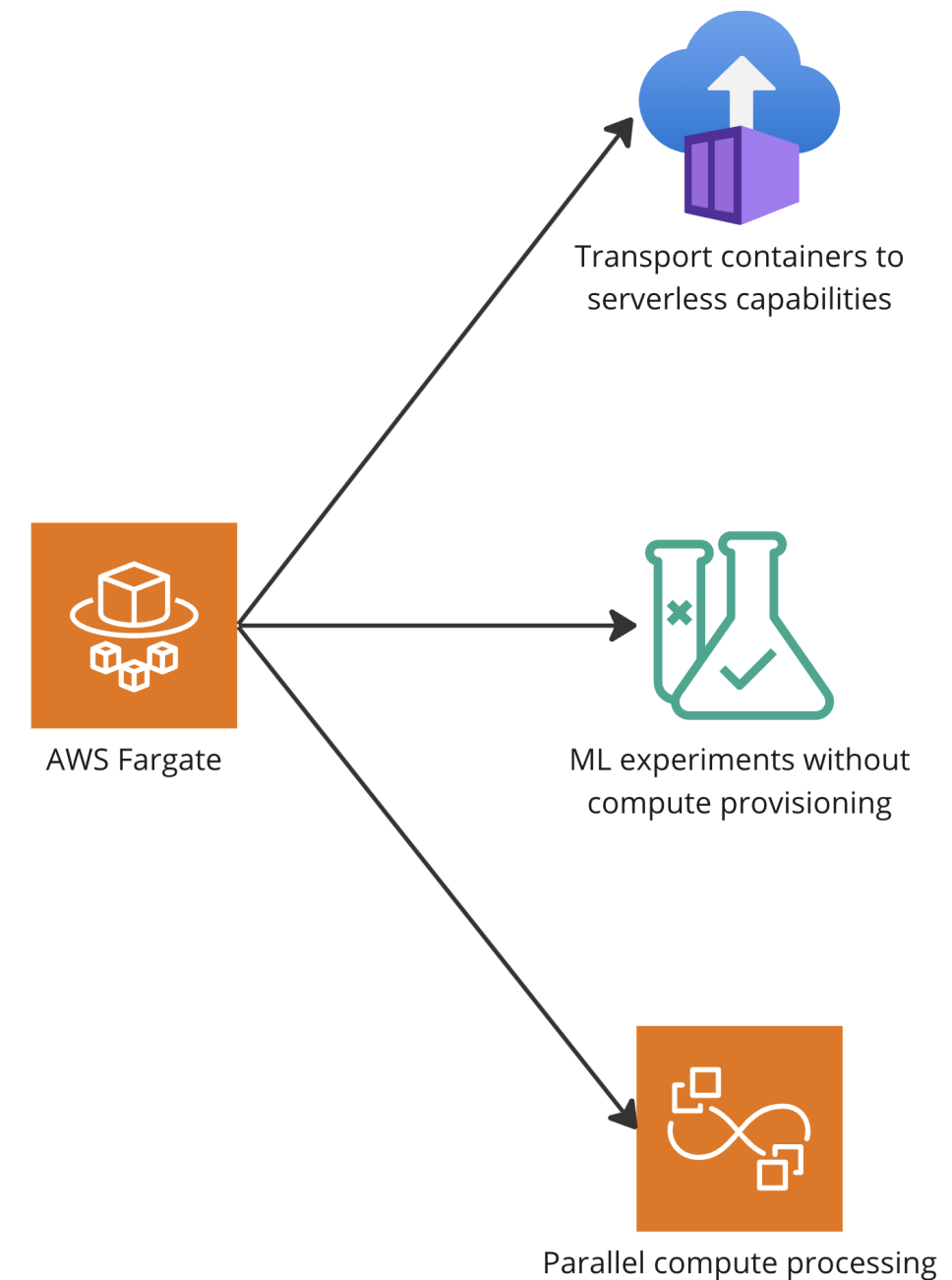


# AWS Fargate

Streamlines application development by providing serverless compute for containers

## Use cases

- Enable AI and ML applications without the need for excessive server provisioning
- Batch processing of large datasets with parallel compute capabilities





# Let's practice!

AWS CLOUD TECHNOLOGY AND SERVICES CONCEPTS