CSCI 5902 Adv. Cloud Architecting
Fall 2023
Instructor: Lu Yang

Module 11 Building Microservices and Serverless Architectures (Sections 1-5)
Nov 24, 2023

Housekeeping items and feedback

1. Start recording
2. The final is:
   9:30-11:30am, Dec 11
   CHEB room 170
3. Release more practice tests released on Brightspace
4. Do not always keep your resources running for the term project. Use IaC to keep your infrastructure.
5. CPC and SAA voucher request spreadsheet is up. The deadline to sign up is Dec 4.
6. SLEQ

AWS Academy Cloud Architecting

Module 11: Building Microservices and Serverless Architectures

Module overview

aws academy

Sections

1. Architectural need
2. Introducing microservices
3. Building microservice applications with AWS container services ECS
4. Introducing serverless architectures
5. Building serverless architectures with AWS Lambda
6. Extending serverless architectures with Amazon API Gateway
7. Orchestrating microservices with AWS Step Functions

Module objectives

At the end of this module, you should be able to:

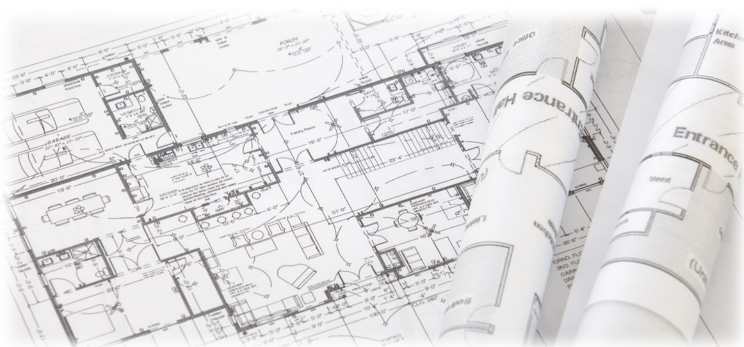
- Indicate the characteristics of microservices
- Refactor a monolithic application into microservices and use Amazon ECS to deploy the containerized microservices
- Explain serverless architecture
- Implement a serverless architecture with AWS Lambda
- Describe a common architecture for Amazon API Gateway
- Describe the types of workflows that AWS Step Functions supports

Module 13: Building Microservices and Serverless Architectures

Section 1: Architectural need

Café business requirement

The café wants to get daily reports via email about all the orders that were placed on the website. They want this information so they can anticipate demand and bake the correct number of desserts going forward (reducing waste). They also want to identify any patterns in their business (analytics).



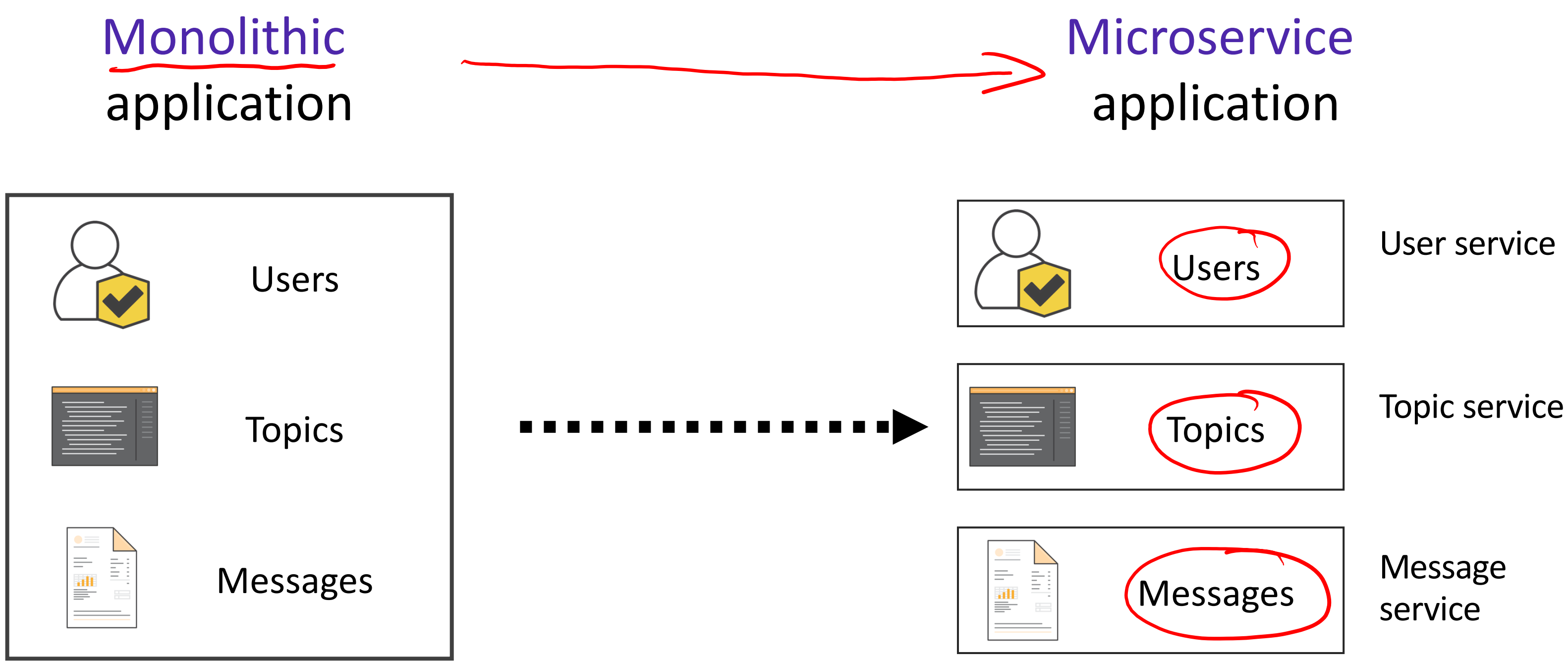
Module 13: Building Microservices and Serverless Architectures

Section 2: Introducing microservices

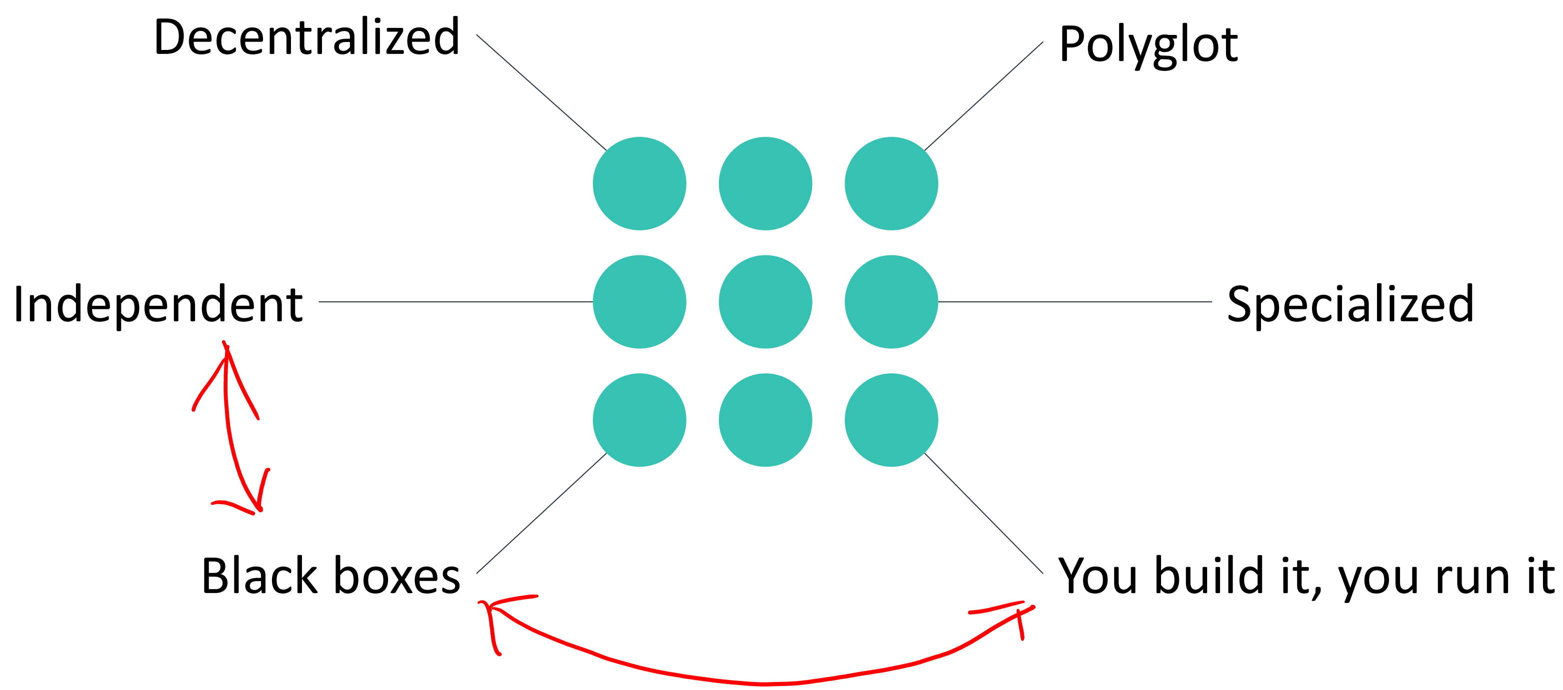
What are microservices?

Applications that are composed of independent services that communicate over well-defined APIs

Monolithic versus microservice applications



Characteristics of microservices



Section 2 key takeaways

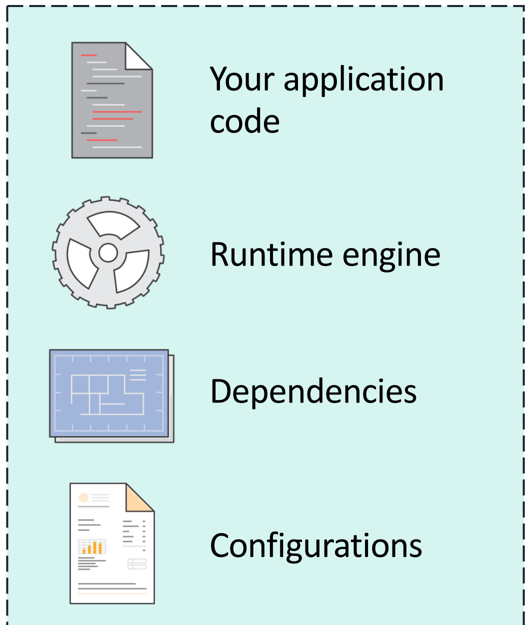
- Microservice applications are composed of independent services that communicate over well-defined APIs
- Microservices share the following characteristics –
  - Decentralized
  - Independent
  - Specialized
  - Polyglot
  - Black boxes
  - You build it, you run it

Module 13: Building Microservices and Serverless Architectures

Section 3: Building microservice applications with AWS container services

What is a container?

Your container



A problem that containers solve

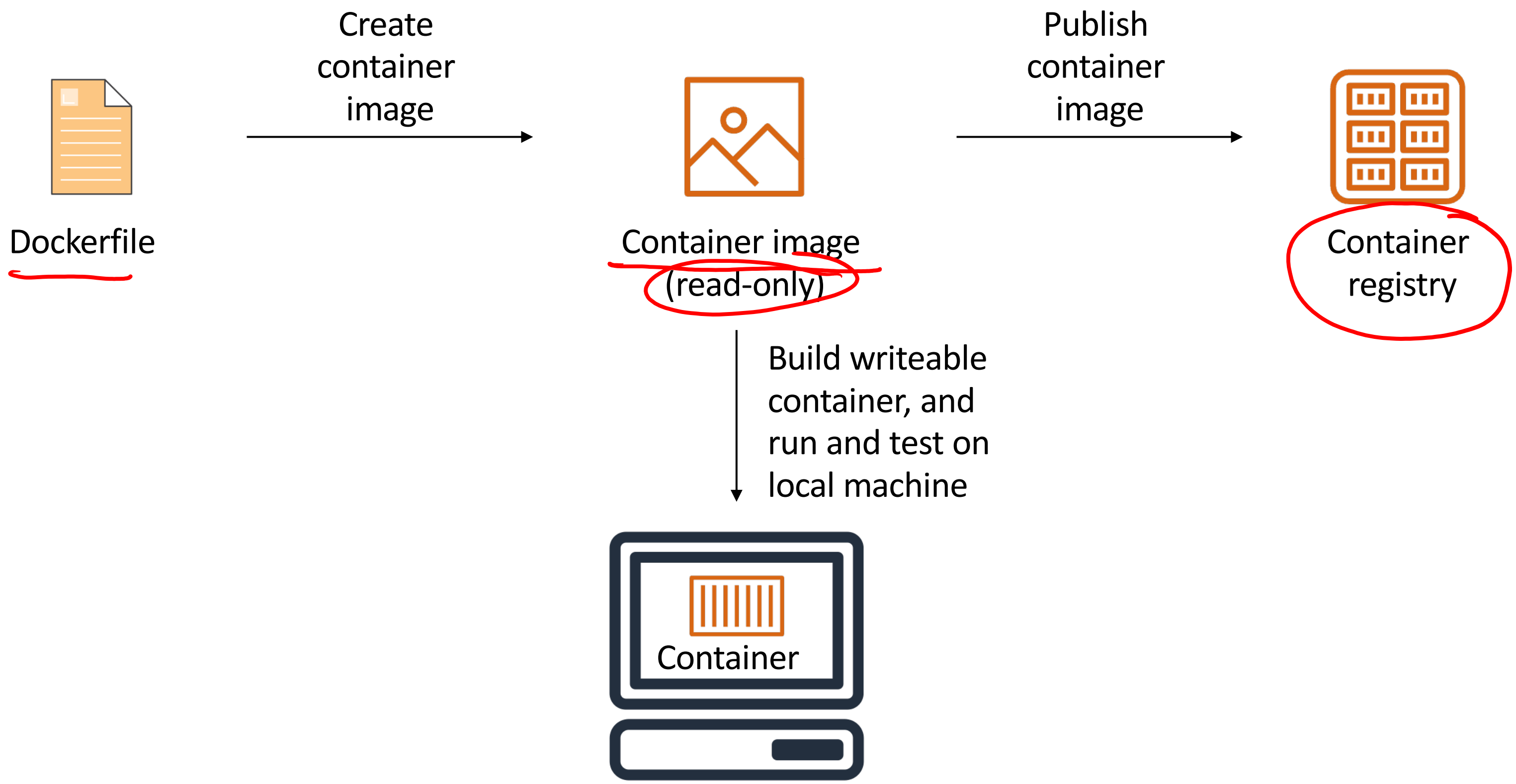
Getting software to run reliably in different work environments

Developer's workstation

Production environment

Test environment

Container terminology



Amazon ECS

aws academy

Amazon Elastic Container Service (Amazon ECS)

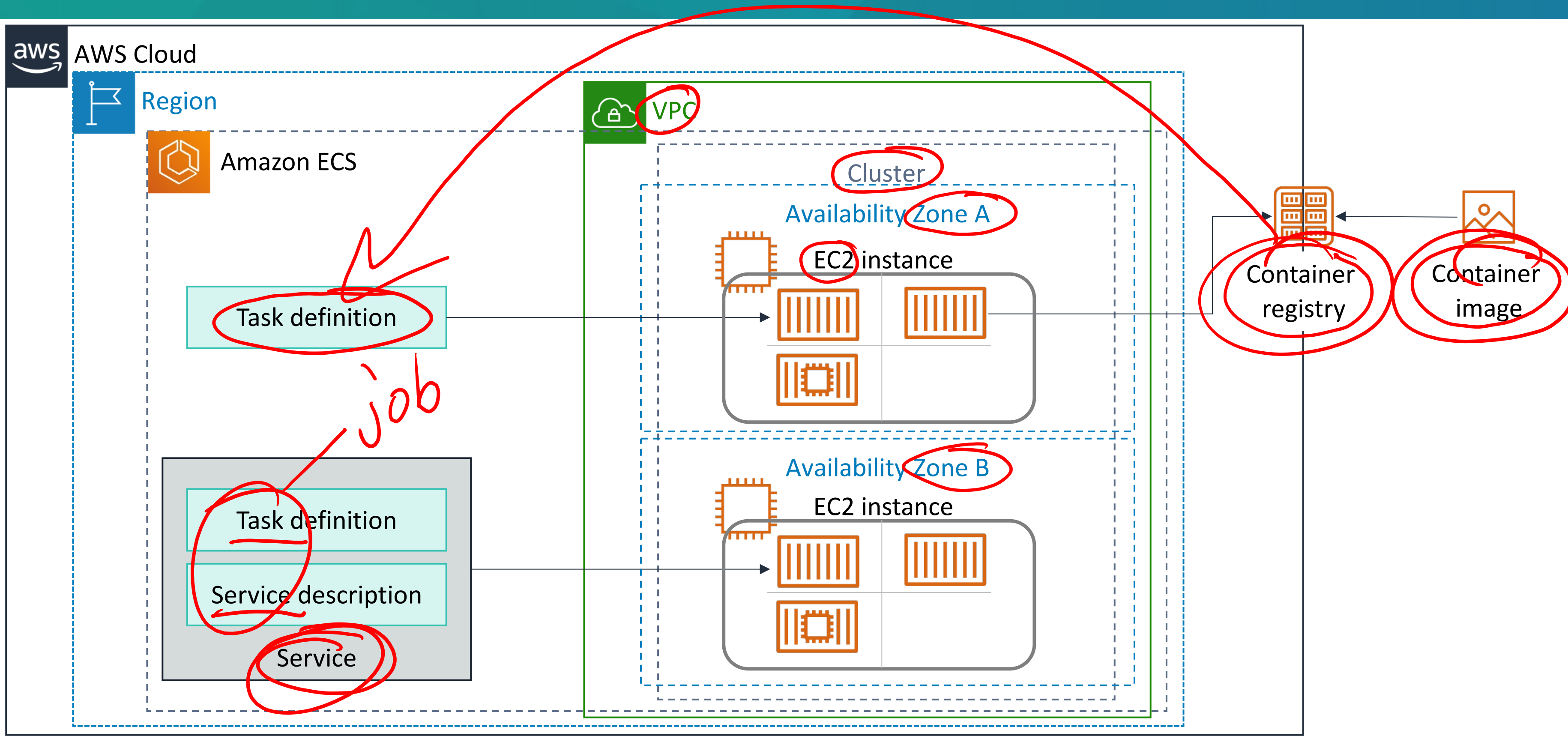
Orchestrates when containers run

Pods

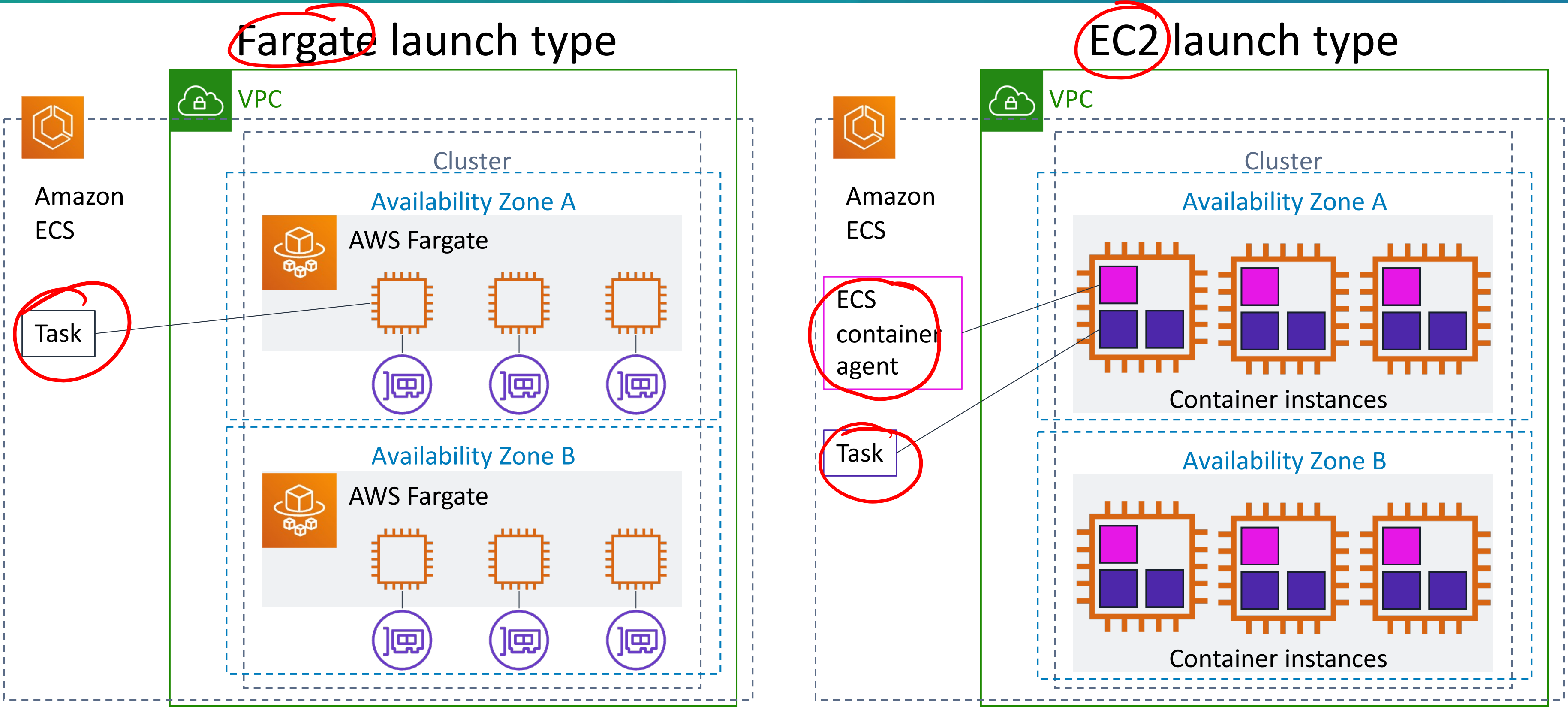
Maintains and scales the fleet of instances that run your containers

Removes the complexity of standing up the infrastructure

Amazon ECS orchestrates containers

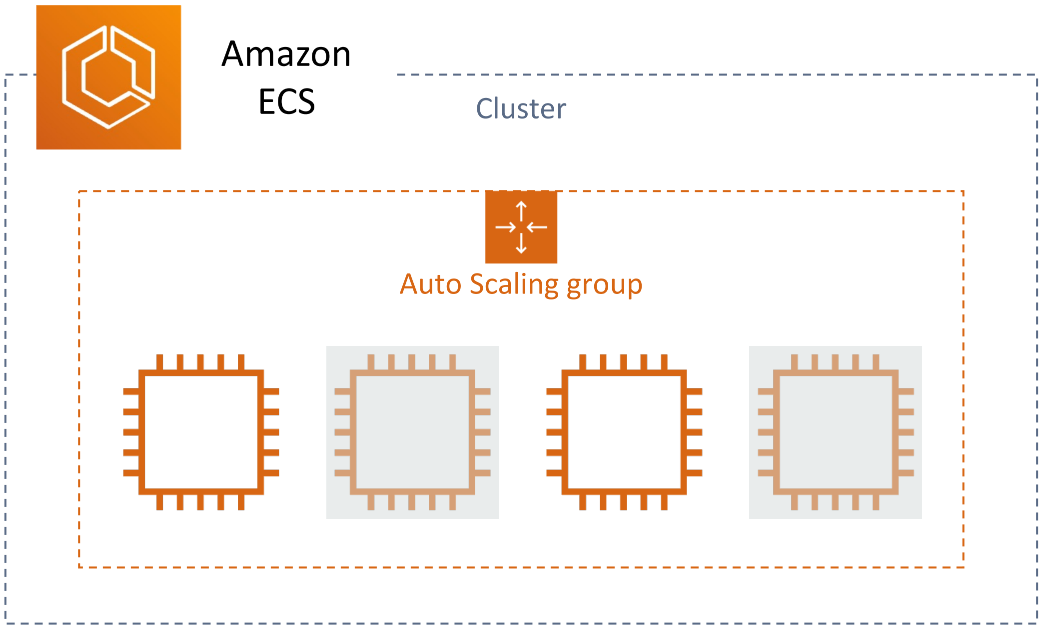


Amazon ECS launch types



Amazon ECS cluster auto scaling

aws academy

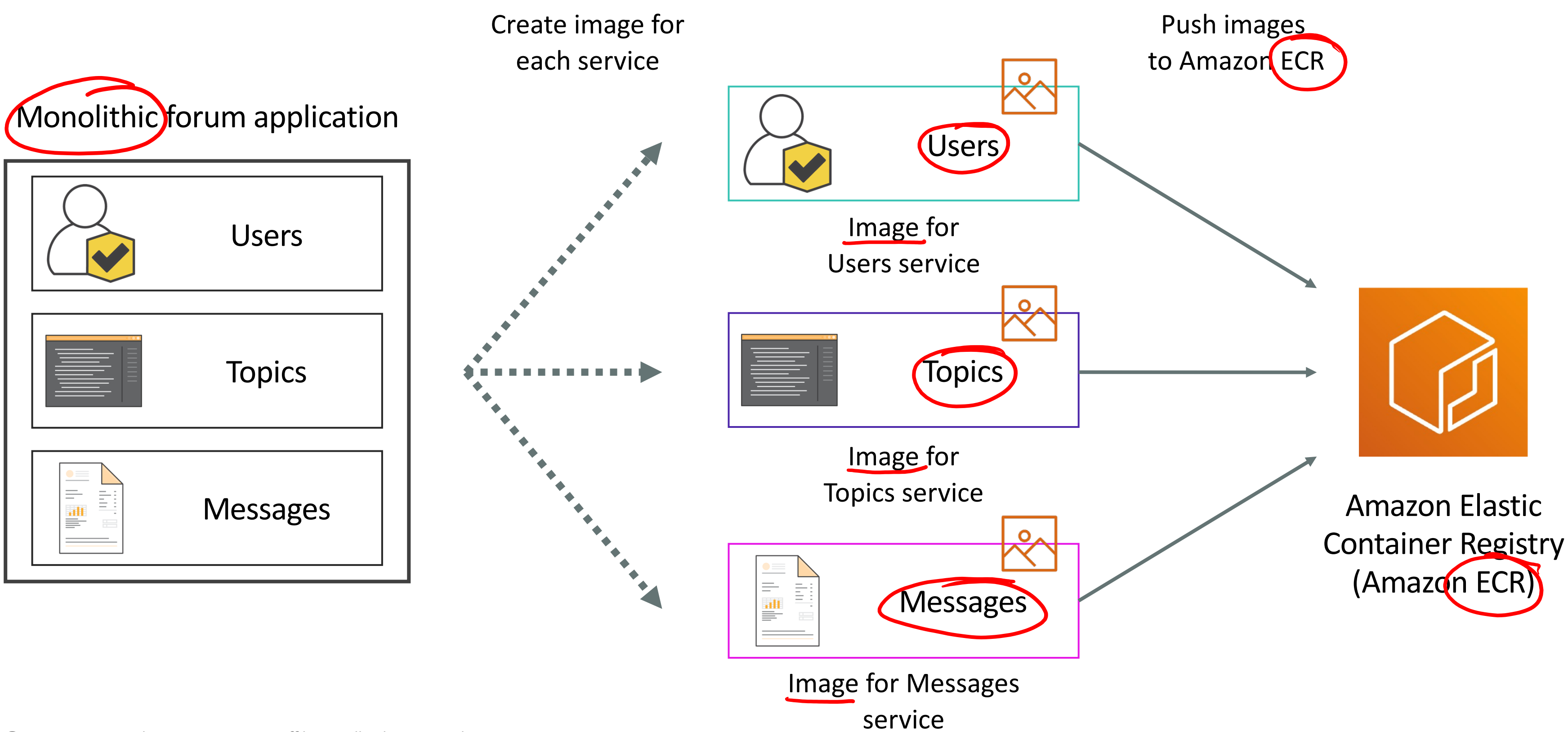


~~up/down~~

out/in

Decomposing monoliths – Step 1: Create container images

aws academy



Decomposing monoliths – Step 2: Create service task definition and target groups

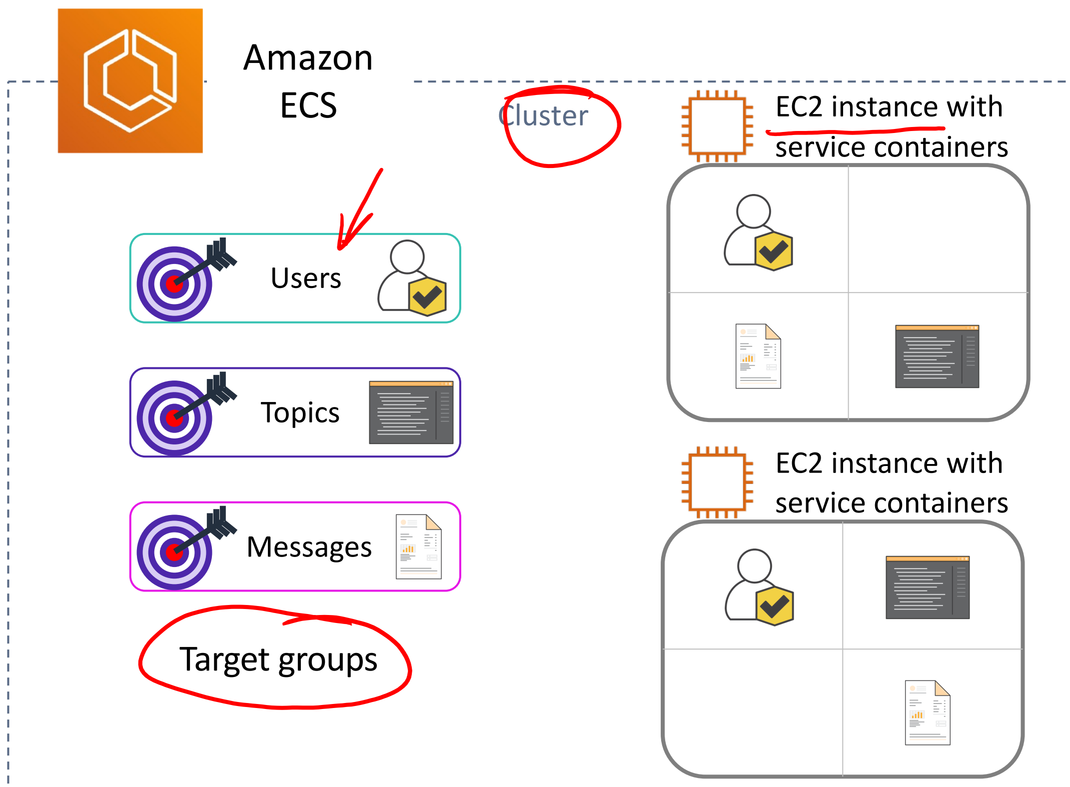
aws academy

Service Task Definition

- Launch type = [EC2 or Fargate]
- Name = [service-name]
- Image = [service ECR repo URL]:version
- CPU = [256]
- Memory = [256]
- Container port = [3000]
- Host port = [0]

Service Target Group

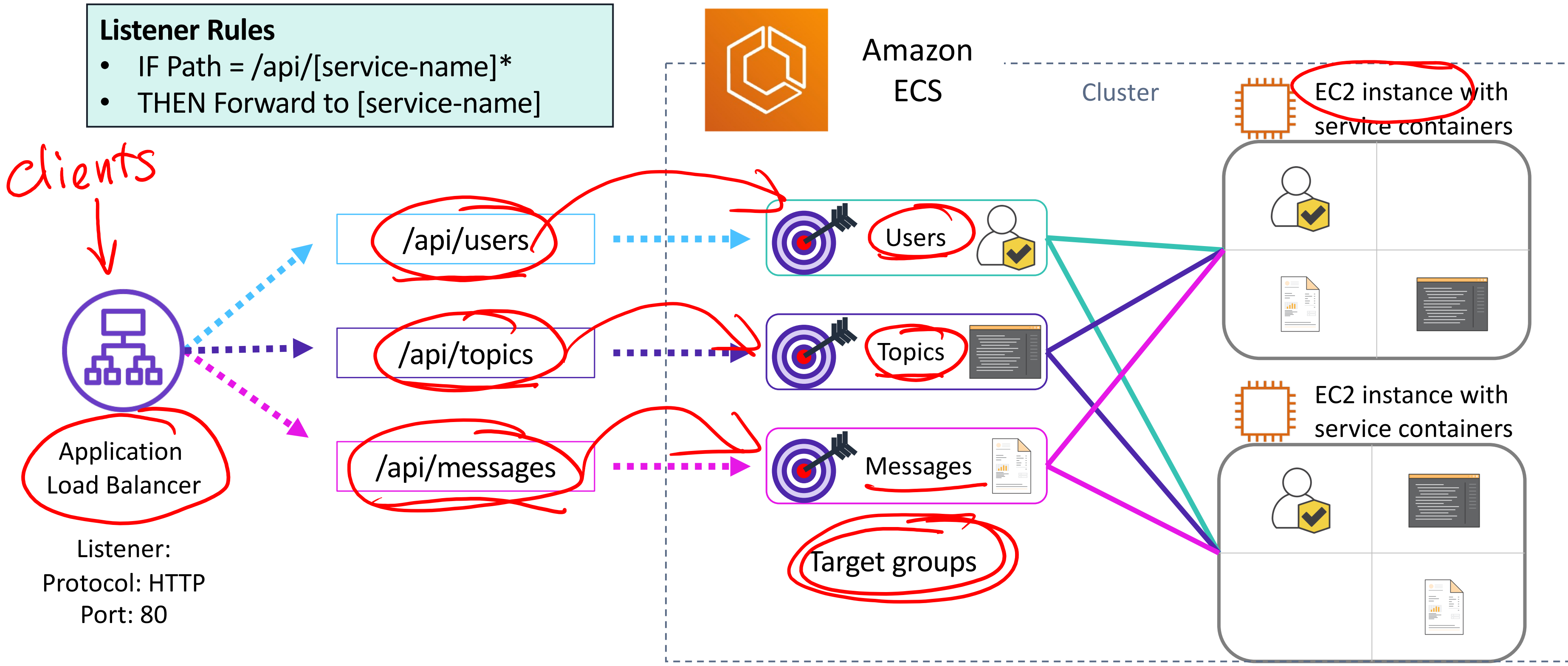
- Name = [service-name]
- Protocol = [HTTP]
- Port = [80]
- VPC = [vpc-name]



Decomposing monoliths – Step 3: Connect load balancer to services

aws academy

Listener Rules
- IF Path = /api/[service-name]*
- THEN Forward to [service-name]



Discussion: API Gateway vs. Load Balancer

aws academy

Overview	
API Gateway	**Load Balancer**
• An API gateway is a service deployed in front of an API or set of microservices, which passes requests from clients and responses or data returned by APIs. • When a client makes a request, the API Gateway splits it into multiple requests, routes them to the appropriate service, routes responses back to the client, and keeps track of everything. • API gateways have multiple benefits for microservices applications, including improved security, better request performance, centralized monitoring and control.	• Load balancers need to handle incoming requests from users for services and information. A load balancer sits between the servers handling user requests and the public Internet. • Once a load balancer receives a request, it finds an available online server and routes the request to this server. It can dynamically add servers in response to traffic spikes and drop servers when demand is low. • You can find various load balancers, including physical appliances, software instances, or a combination of the two.

Discussion: API Gateway vs. Load Balancer

aws academy

Traffic Management	
API Gateway	**Load Balancer**
API gateways manage network traffic by processing API requests from clients to determine the necessary services and destination applications to handle API calls. Clients are the software making API calls. APIs are important for integrating disparate application components and enabling them to communicate. An API gateway also manages the protocols and translations between software components.	**Application load balancers** command how traffic flows. A load balancer redirects traffic across multiple servers. This ability helps large networks handle high traffic volumes and minimizes performance issues associated with running an application on one server.

Discussion: API Gateway vs. Load Balancer

aws academy

Capabilities

API Gateway

API gateways act as translators and organizers connecting separate software components. Among their key capabilities are:

- API security including authentication and authorization.
- Rate-limiting for APIs to prevent abuse or over-utilization.
- API monitoring and logging to assist with observability.
- API transformation to enable services to communicate with each other even if they use different protocols or technology stacks.

Load Balancer

Load balancers use algorithms to direct inbound network traffic to the appropriate servers:

- Round-robin algorithms distribute traffic evenly across servers.
- Least-connection algorithms direct traffic to the least burdened server (i.e., with the fewest connections)—they ensure high availability when the servers in a given environment have varying capabilities.
- IP hash algorithms direct traffic to servers according to the origin of the requests. They are best suited for environments with servers distributed across multiple geographic regions—these route network traffic to the nearest server to minimize application latency.

Discussion: API Gateway vs. Load Balancer

aws academy

Use Cases	
API Gateway	**Load Balancer**
API gateways are best suited for designing and deploying microservices-based applications. Enterprises often build modern applications as separate services, not a monolithic architecture. These independent services use APIs to communicate, with an API gateway ensuring that all services function and collaborate properly in a unified deployment.	**Load balancers** are best suited to geographically distributed deployments that prioritize resilience and redundancy. A load balancer can redirect traffic to other instances on another server when a server fails. Enterprises usually run multiple application instances in parallel, sometimes on multiple physical servers. This approach provides redundancy to maintain high availability and ensure applications can handle all traffic.

Discussion: API Gateway vs. Load Balancer

Contribute to Functional or Non-Functional Requirements?	
API Gateway	Load Balancer
F	NF

Tools for building highly available microservice architectures

aws academy

AWS Cloud Map

- Is a fully managed discovery service for cloud resources
- Can be used to define custom names for application resources
- Maintains updated location of dynamically changing resources, which increases application availability

AWS App Mesh

- Captures metrics, logs, and traces from all your microservices
- Enables you to export this data to Amazon CloudWatch, AWS X-Ray, and compatible AWS Partner Network (APN) Partner and community tools
- Enables you to control traffic flows between microservices to help ensure that services are highly available

AWS Fargate

aws academy

AWS Fargate

- Is a fully managed container service
- Works with Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Kubernetes Service (Amazon EKS)
- Provisions, manages, and scales your container clusters
- Manages runtime environment
- Provides automatic scaling

Demonstration: ECS (https://www.youtube.com/watch?v=zs3tyVgiBQQ)

Section 3 key takeaways

- Amazon ECS is a highly scalable, high-performance container management service. It supports Docker containers and enables you to easily run applications on a managed cluster of Amazon EC2 instances.
- Cluster auto scaling gives you more control over how you scale tasks in a cluster.
- AWS Cloud Map enables you to define custom names for your application resources. It maintains the updated location of these dynamically changing resources.
- AWS App Mesh is a service mesh that provides application-level networking. It enables your services to communicate easily with each other across multiple types of compute infrastructure.
- AWS Fargate is a fully managed container service that enables you to run containers without needing to manage servers or clusters.

Module 13: Building Microservices and Serverless Architectures

Section 4: Introducing serverless architectures

What does serverless mean?

A way for you to build and run applications and services without thinking about servers

Tenets of serverless architectures

No infrastructure provisioning, no management

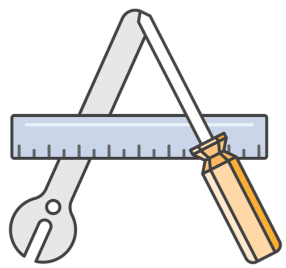
Automatic scaling

Pay for value

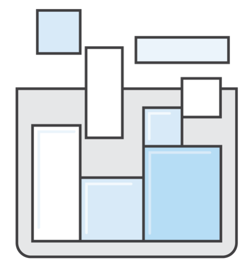
Highly available and secure

Benefits of serverless

Lower total cost of ownership



Focus on your application, not configuration



Build microservice applications

AWS serverless offerings

aws academy

Compute
- AWS Lambda and Lambda@Edge
- AWS Fargate

Storage
- Amazon S3
- Amazon EFS

Data Stores
- Amazon DynamoDB
- Amazon Aurora
- Amazon RDS Proxy

API Proxy
- Amazon API Gateway

Application integration
- Amazon SNS
- AWS AppSync
- Amazon SQS
- Amazon EventBridge

Orchestration
- AWS Step Functions

Analytics
- Amazon Kinesis
- Amazon Athena

Section 4 key takeaways

- Serverless computing enables you to build and run applications and services without provisioning or managing servers
- Serverless architectures offer the following benefits –
  - Lower total cost of ownership (TCO)
  - You can focus on your application
  - You can use them to build microservice applications

Module 13: Building Microservices and Serverless Architectures

Section 5: Building serverless architectures with AWS Lambda

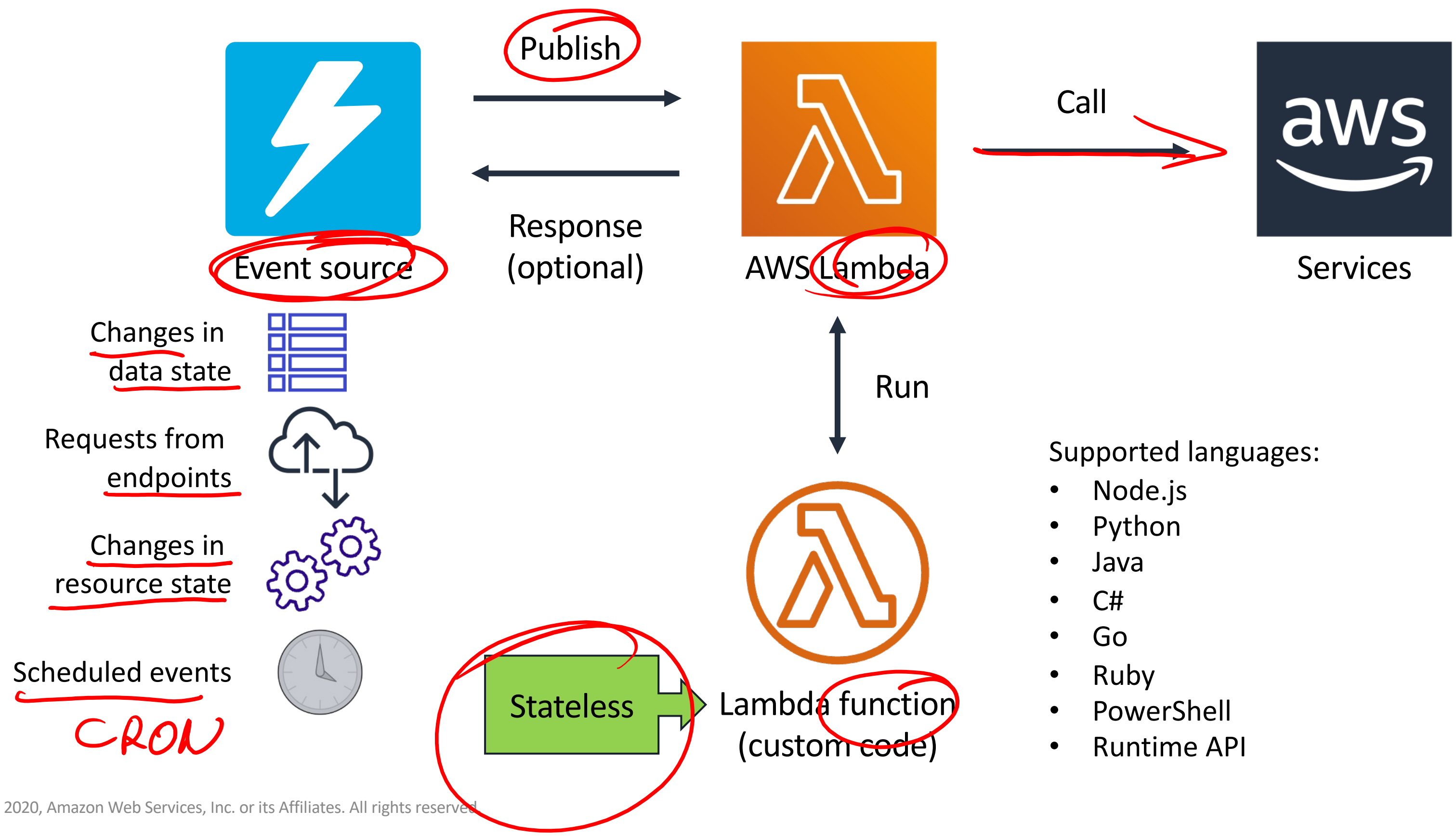
AWS Lambda

aws academy

AWS Lambda

- Is a fully managed compute service
- CRON
  Runs your code on a schedule or in response to events (for example, changes to an Amazon S3 bucket or an Amazon DynamoDB table)
- Supports Java, Go, PowerShell, Node.js, C#, Python, Ruby, and Runtime API
- Can run at edge locations closer to your users

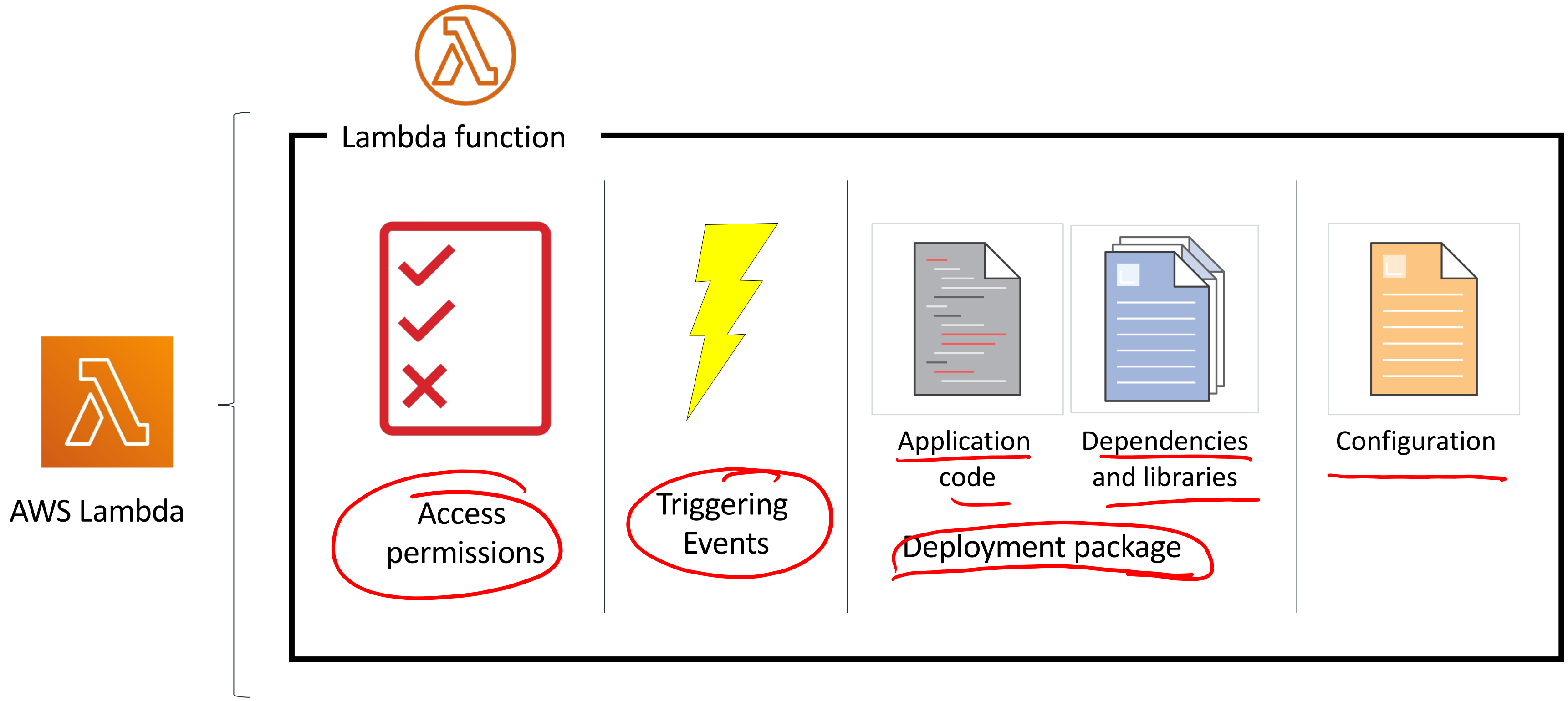
How AWS Lambda works



Supported languages:

- Node.js
- Python
- Java
- C#
- Go
- Ruby
- PowerShell
- Runtime API

Lambda functions



Anatomy of a Lambda function

aws academy

Handler()

Function to be run upon invocation

Event object

Data sent during Lambda function invocation

Context object AWS

Methods available to interact with runtime information (request ID, log group, more)

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello World')
    }
```

https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/begin

Lambda function configuration and billing

Memory – Cost per 100 ms of function duration increases as memory increases.

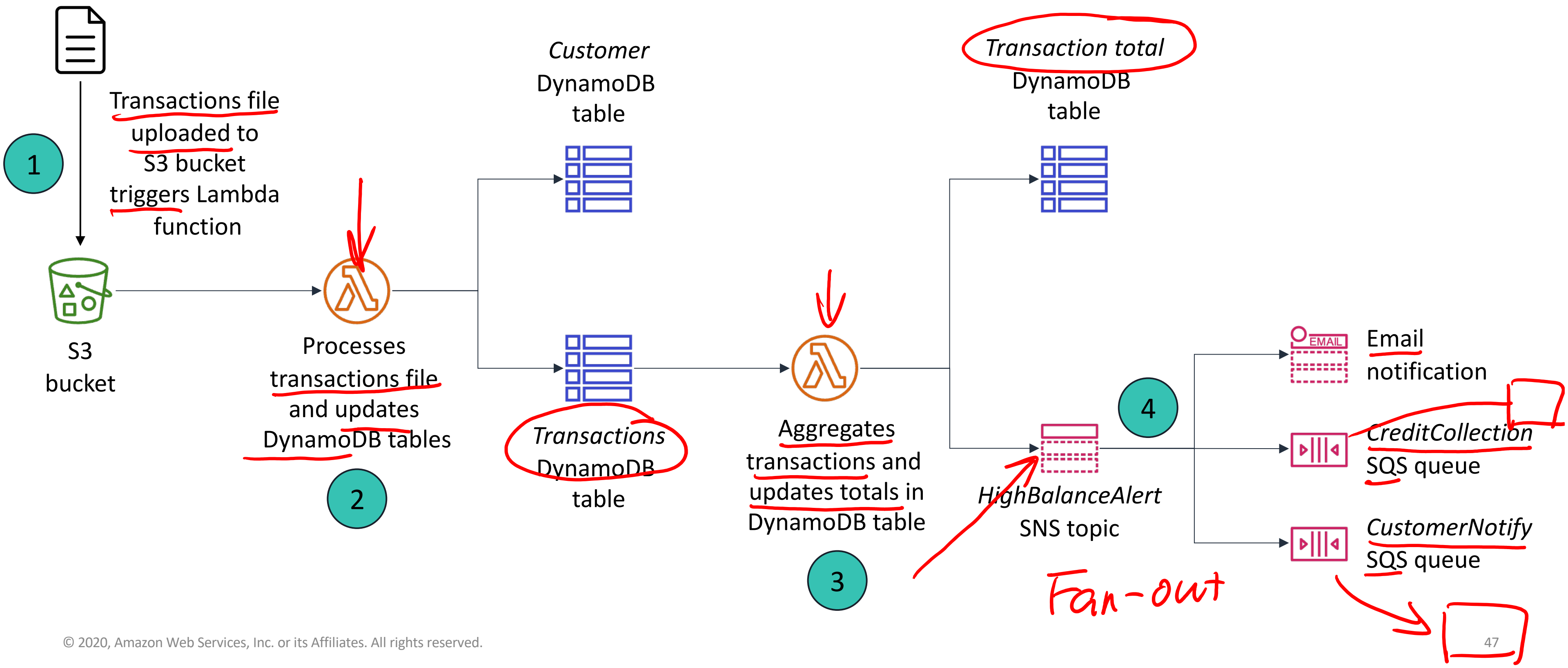
Timeout – You control the maximum duration of your function.

Pricing – You are charged based on number of requests and duration.

Demonstration: Creating an AWS Lambda function

Event-based Lambda function example: Order processing

aws academy



Lambda layers

- Enable functions to share code easily – You can upload a layer one time and reference it in any function
- Promote separation of responsibilities – Developers can iterate faster on writing business logic
- Enable you to keep your deployment packages small
- Limits – .zip
  - A function can use up to five layers a time
  - The total unzipped size of the function and all layers: less than 250 MB

5

Demonstration: Using AWS Lambda with Amazon S3

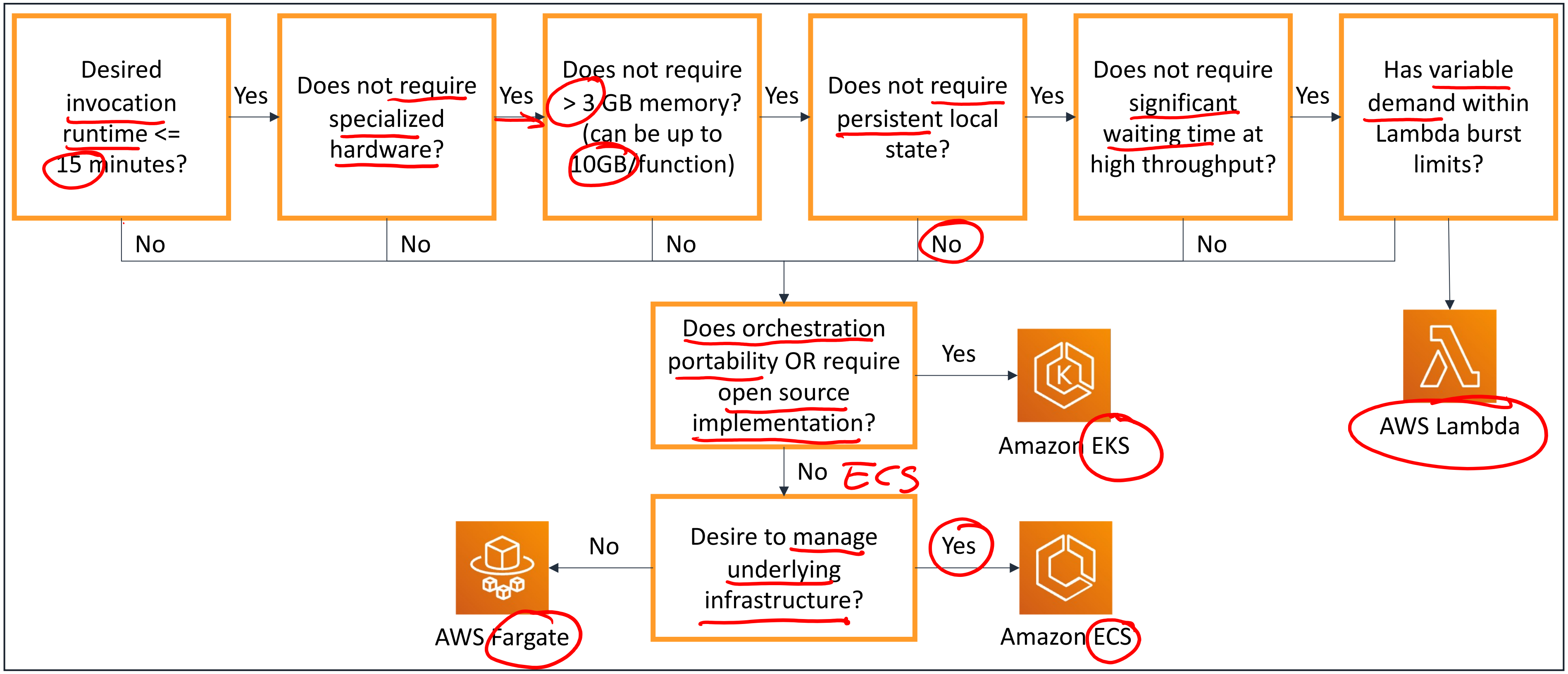
Comparison of operational responsibility for container and serverless architectures

aws academy

Less operational responsibility ↑ — More operational responsibility ↓

	AWS Manages	Customer Manages
AWS Lambda Serverless Functions	• Data source integrations • Physical hardware, software, networking, and facilities • Provisioning	• Application code
AWS Fargate Serverless Containers	• Container orchestration and provisioning • Cluster scaling • Physical hardware, host OS/kernel, networking, and facilities	• Application code • Data source integrations • Security configuration and updates, network configuration, and management tasks
Amazon ECS and Amazon EKS Container Management as a Service	• Container orchestration control plane • Physical hardware software, networking, and facilities	• Application code • Data source integrations • Work clusters • Security configuration and updates, network configuration, firewall, and management tasks

Choosing a compute platform: Containers versus AWS Lambda



Section 5 key takeaways

- Lambda is a serverless compute service that provides built-in fault tolerance and automatic scaling
- A Lambda function is custom code that you write that processes events
- A Lambda function is invoked by a handler, which takes an event object and context object as parameters
- An event source is an AWS service or developer-created application that triggers a Lambda function to run
- Lambda layers enable functions to share code and keep deployment packages small