AWS Architecting and SysOps

Computing on AWS, Part 2

June-July 2019

Contents

AWS Lambda
Elastic Container Service

AWS Lambda

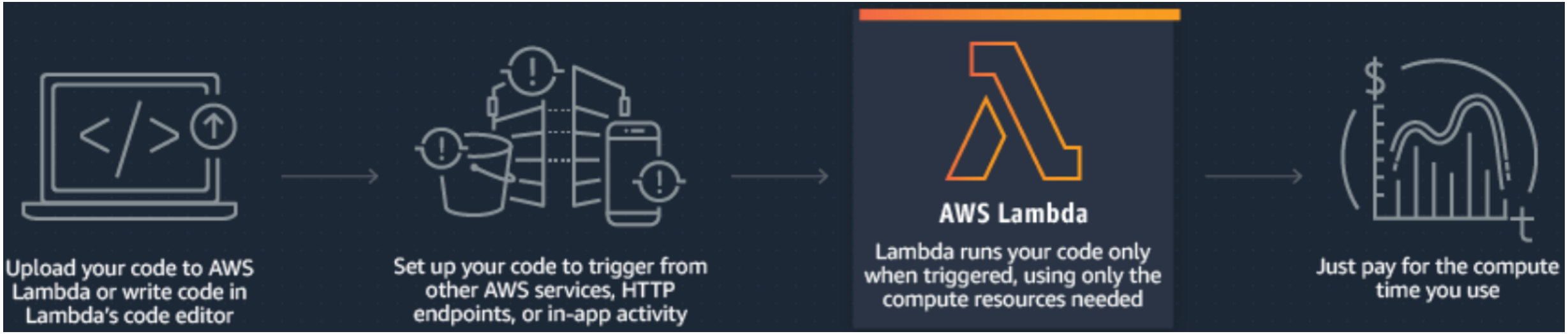
Serverless computing over AWS

Lambda

- Lambda is a serverless compute service: run code without provisioning or managing servers
- Your code runs in response to events and automatically manages the underlying compute resources for you
  - It processes each trigger individually, scaling precisely with the size of the workload
- You pay only for the compute time you consume
  - You are charged for every 100ms your code executes and the number of times your code is triggered
  - There is no charge when your code is not running
- Lambda runs your code on high-availability compute infrastructure and performs all the administration of the compute resources
  - Server and OS maintenance, capacity provisioning and automatic scaling, code monitoring and logging

Lambda: How it works

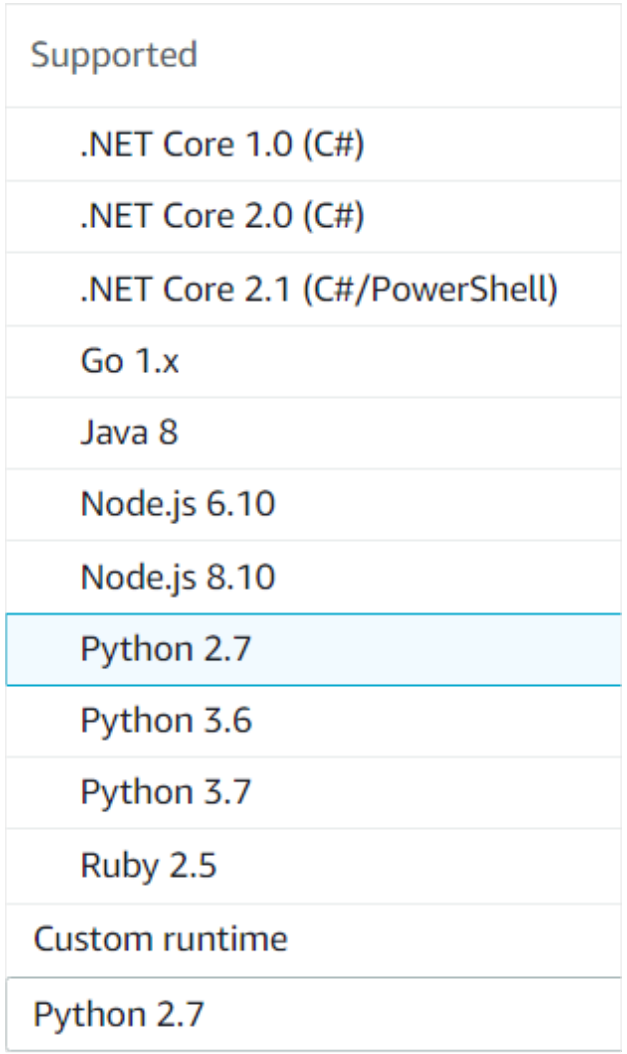
1. Upload your code to Lambda or write code in Lambda's code editor
2. Setup to trigger the code from other AWS services and so on
3. Lambda runs your code only when triggered, using the minimum resources
4. Just pay for the compute time you use

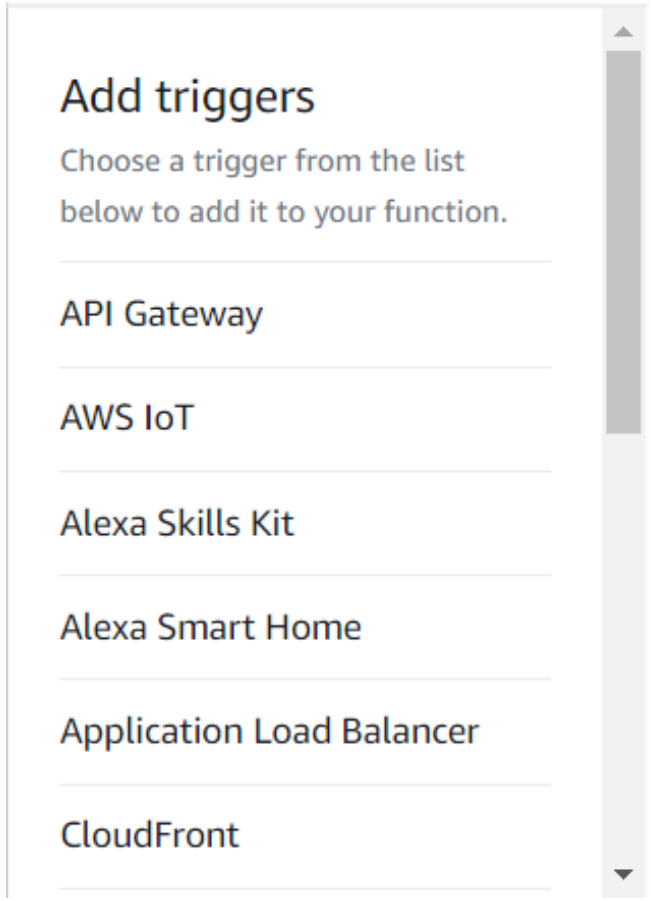


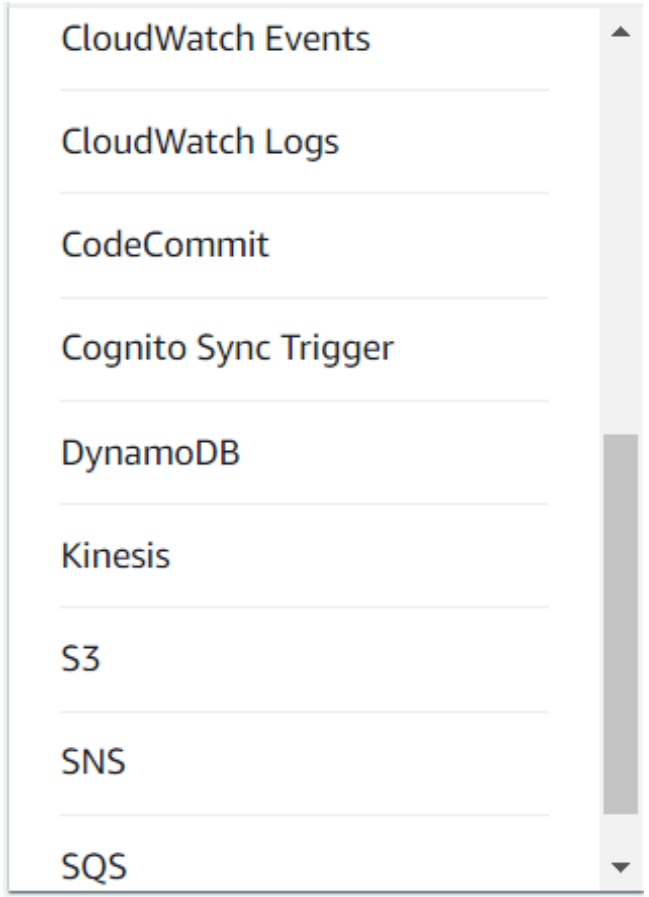
Lambda functions

- Lambda function = the code you run on AWS
- After you create your Lambda function, it is ready to run as soon as it is triggered
- Each function includes:
  - Code
  - Function name
  - Resource requirements
- Lambda functions are stateless
  - Lambda can launch as many copies as needed to scale to the rate of incoming events
- You can associate Lambda functions to AWS resources
  - In particular, S3 buckets, DynamoDB tables, Kinesis streams, or SNS notifications
- And when the resource changes, Lambda executes your function and manages the compute resources to keep up with incoming requests

Lambda supported languages and triggers





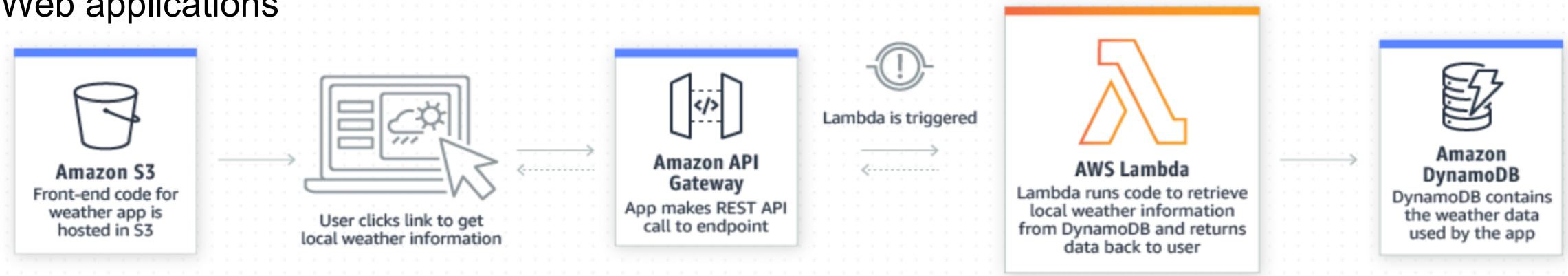


Use cases

- Lambda runs your code in response to events
- ✓ Data changes in S3 bucket or DynamoDB
- ✓ HTTP requests from API Gw

✓ Real-time file processing

✓ Web applications



Lambda pricing

- Lambda cost depends on several factors: [https://aws.amazon.com/lambda/pricing/]

✓Requests: you are charged for the total number of requests across all your functions

❖Request: each time an execution starts in response to an event notification or invoke call, including test invokes from the console

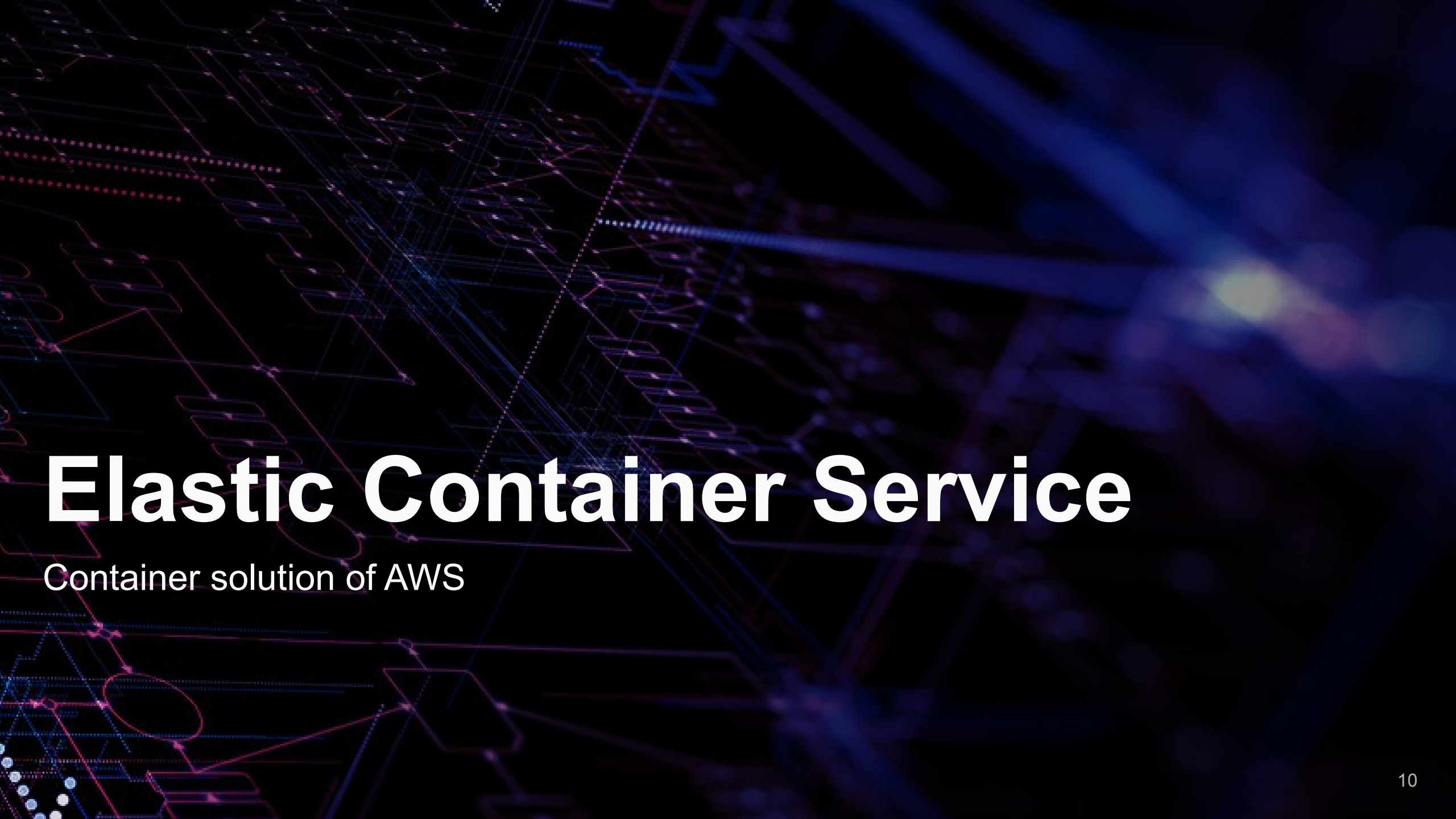
✓Duration: calculated from the time your code begins executing until it returns or otherwise terminates, rounded up to the nearest 100ms

✓Memory: price per 100ms rises depending on the amount of memory you allocate to run your function

- Usage of other AWS services or transfers are billed to the specific service you use
  - Ex/ you will be billed for read/write requests and data stored in S3
- Costs, compared to on-premises, are drastically reduced in some cases

Ex/ A web service that simply imported a data file, parsed it, and generated a report.
The service was offered with 100% availability, so it needed 2 servers running continuously

➢ The cost was reduced from $1500 / month to 5c / month using Lambda



Elastic Container Service

Container solution of AWS

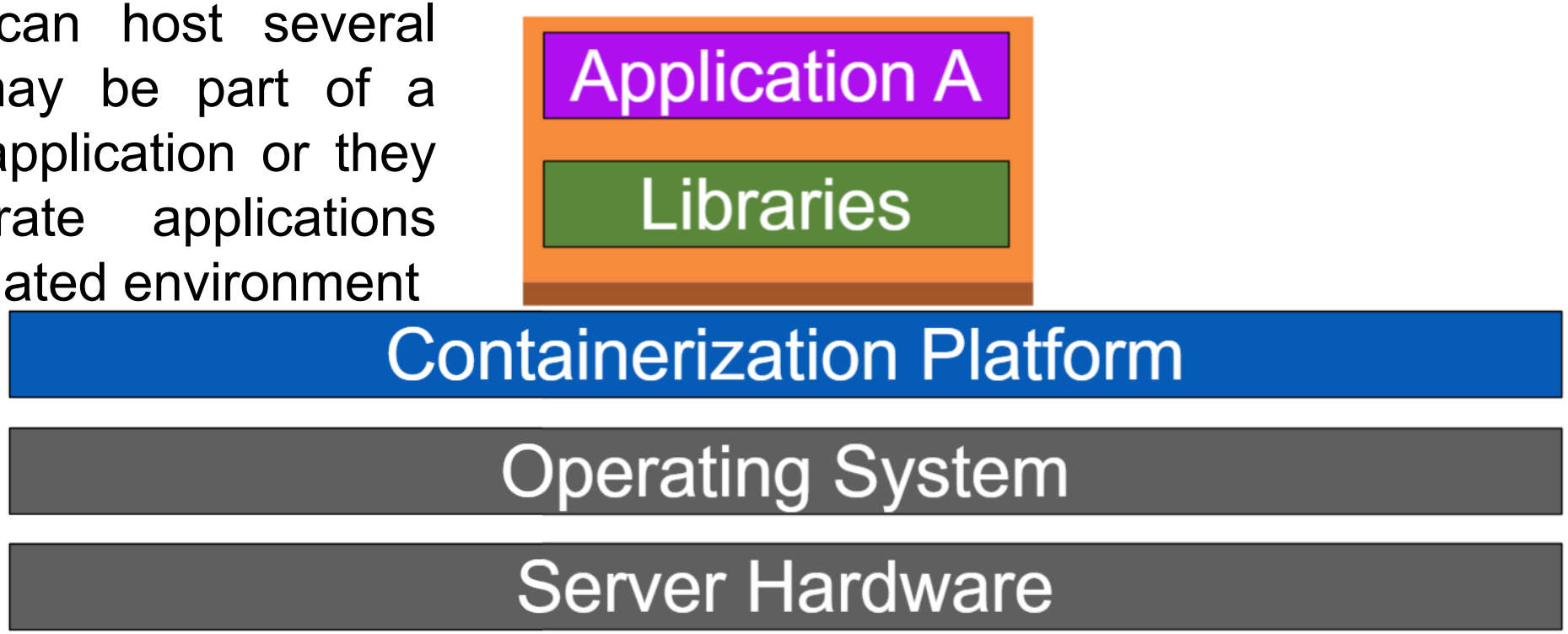
Elastic Container Service (ECS)

- ECS is a highly scalable, high-performance container orchestration service that supports Docker containers
- Allows to easily run, scale, and secure Docker container applications on AWS
- Eliminates the need to install, operate, and scale your own container orchestration and cluster management infrastructure, and allows you to focus on the resource needs and availability requirements of your containerized application
- Enables you to grow from a single container to thousands of containers across hundreds of instances without creating additional complexity in your application
- You can run applications, batch jobs, microservices…
- Integrated with features such as IAM roles, security groups, load balancers, CloudWatch Events, CloudFormation templates, and CloudTrail log

Background: What is a container?

- A container is a form of virtualization that is implemented at the OS level
- A lightweight, standalone package that include everything needed to run an application, such as code, runtime, system tools, system libraries, and settings

- A single server can host several containers that may be part of a larger enterprise application or they might be separate applications running in their isolated environment



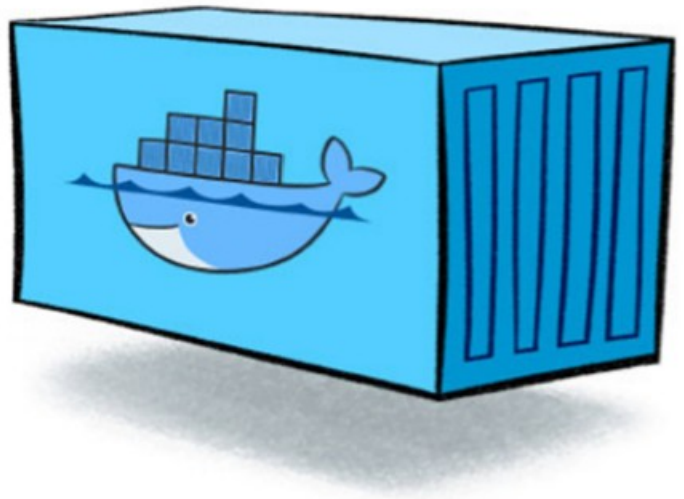
Background: Virtualization and abstraction

- Traditionally, we had the OS, associated libraries and applications running on top
  - Inefficient: costs are the same independent of utilization, applications fight for the same resources, libraries version have to be sync and compatible with all applications
- A virtualization platform over the OS increases agility
  - Isolated applications and their libraries with their own full OS into a virtual machine (VM)
  - But virtualization is heavy as you have several OS to handle, that can be just replicas
- A container runtime shares the OS kernel, enabling you to create an instance of a container image
  - Containers are lightweight, efficient, and fast
  - They can spun up and down faster than virtual machines, allowing better utilization
  - You can share libraries when needed, but you can have library isolation for your applications
  - Containers are highly portable: their code runs identically across different environments

Background: Docker

- Docker is a lightweight container virtualization platform that provides tooling for creating, storing, managing, and running containers
  - Easier to integrate with automated build, test, and deployment pipelines
- Portable application environment
  - Package application and dependencies in a single, immutable object: a Docker image
  - This image is a read-only, immutable template that is highly portable
  - A Docker container is an instance of that image

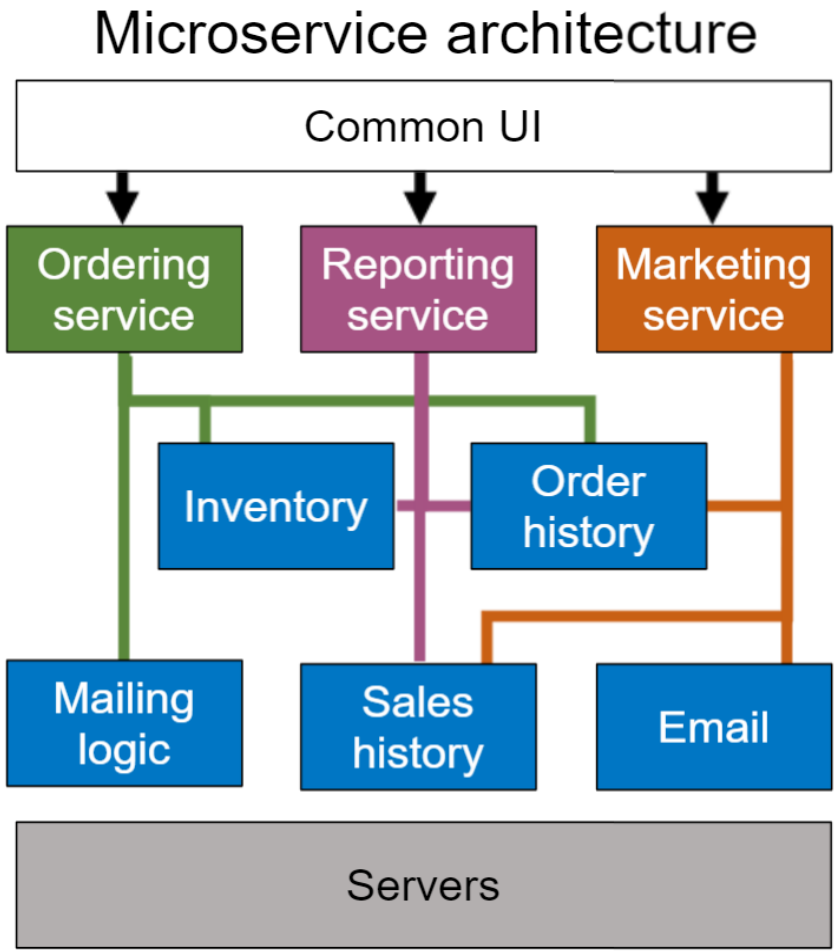
- You can run different application version with different dependencies and libraries simultaneously
- This leads to much faster development, and better resources utilization and efficiency

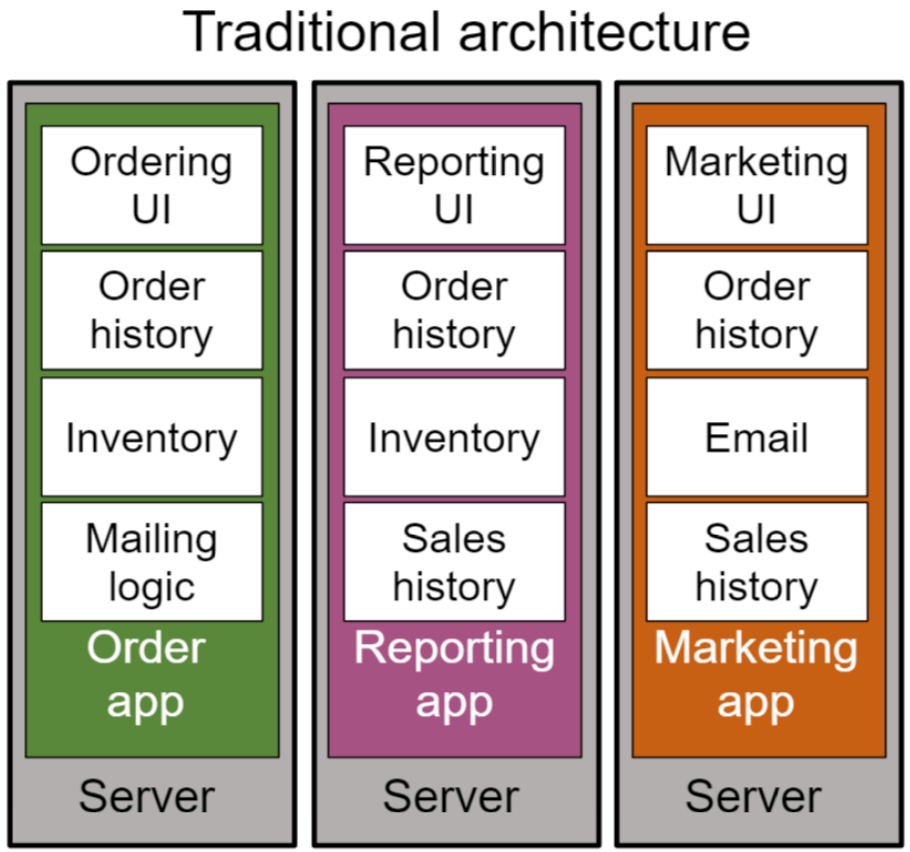


Background: Microservices architectures

- The growth of containers is due to the rise of microservices architectures

- Microservices and containers work well together
- Containers are the underlying technology that powers modern microservices
- With microservice architecture, developers can take full advantage of containers





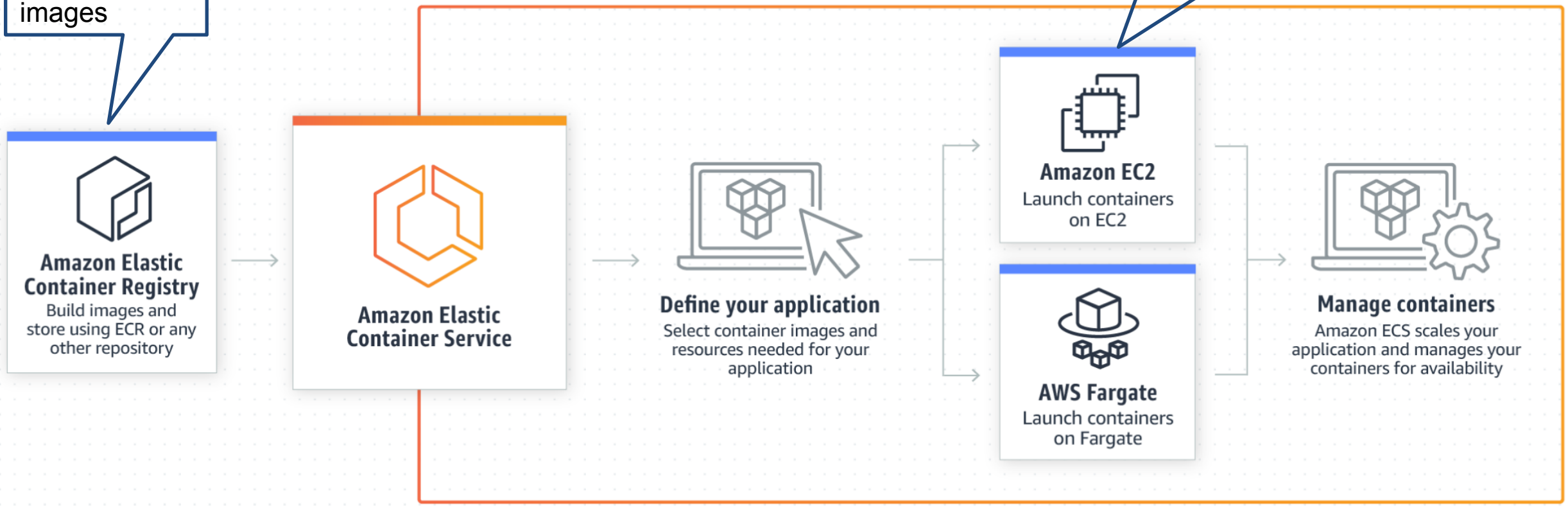
ECS Benefits

- Containers without servers
  - ECS features AWS Fargate, so you can deploy and manage containers without having to provision or manage servers
- Containers with servers
  - For developers who require more granular, server-level control over the infrastructure, ECS EC2 launch type allows you to manage a cluster of servers and schedule placement of containers on the servers
- Containerize everything
  - ECS lets you easily build all types of containerized applications
- Secure
  - ECS launches your containers in your own virtual private network (VPN), allowing you to use VPC security groups and network ACLs
- Performance at scale
  - You can launch tens or tens of thousands of Docker containers in seconds

How ECS works

ECR helps developers to store, manage and deploy Docker container images

Two possibilities to launch containers:
- Self-made: EC2
- Automatized: Fargate

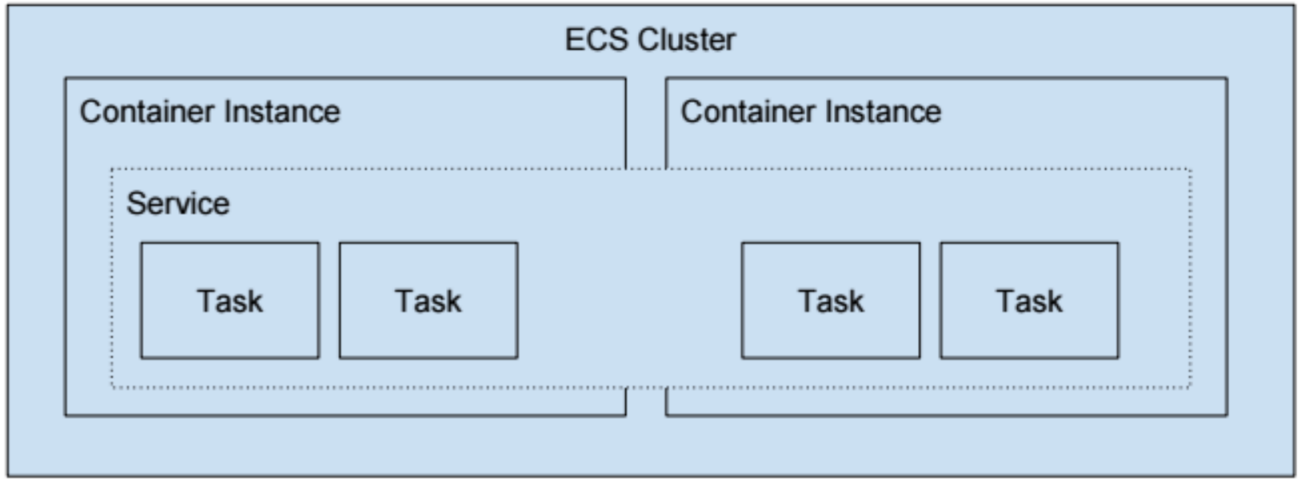


ECS Concepts

- Task Definition
  - A blueprint that describes how a Docker container should launch
  - This JSON template, contains settings: docker image, CPU shares, memory requirement, command to run and environmental variables
- Task
  - A running container with the settings defined in the Task Definition
  - An “instance” of a Task Definition
  - You can launch as many tasks as you want from a single task definition file
- Container Definition
  - Docker image, definition of the container
- Container Instance
  - An instance of a Docker image
  - An EC2 instance that is part of an ECS Cluster and has Docker and the ecs-agent running on it

ECS Concepts

- Service
  - Defines long running tasks of the same Task Definition. This can be 1 running container or multiple running containers all using the same Task Definition
- Cluster
  - A logic group of EC2 instances. When an instance launches the ecs-agent software on the server, it registers the instance to an ECS Cluster



- ✓4 running Tasks or Docker containers
- ✓They are part of an ECS Service
- ✓The Service and Tasks span 2 Container Instances
- ✓The Container Instances are part of a logical group called an ECS Cluster

ECS Management

- Programmatic control
  - You can create and delete clusters, register and deregister tasks, launch and terminate Docker containers, and provide detailed information about the state of your cluster and its instances
- Container deployments
  - Manage your containers like setting the scheduler to automatically starts new containers using the updated image and stop containers running the previous version, etc.
- Container auto-recovery
  - Automatically recover unhealthy containers to ensure that your have the desired number of containers supporting your application
- Task placements
  - Customize how tasks are place onto a cluster of EC2 instances base on built-in attributes such as instance type, AZ, or custom attributes that you define

ECS Scheduling

- Scheduling strategies that place containers across your clusters based on your resource needs (ex/ CPU or RAM) and availability requirements

- Task scheduling
  - Run processes that perform work and then stop, such as batch processing jobs
  - Manually, automatically or based on a time interval you define

- Service Scheduling
  - Run stateless services and applications. A specified number of tasks are constantly running and ECS restarts them if they fail

- Daemon Scheduling
  - Automatically runs the same task on each selected instance in your ECS cluster
  - Suitable to run tasks that provide common management functionality for a service like logging, monitoring, or backups

ECS Pricing

- ECS pricing depends on the model you choose: Fargate-based or EC2-based

- Fargate launch type model
  - Pay for the amount of vCPU and memory resources that your containerized application requests
  - vCPU and memory resources are calculated from the time your container images are pulled until the ECS Task terminates, rounded up to the nearest second
  - A minimum charge of 1 minute applies

- EC2 launch type model
  - There is no additional charge for EC2 launch type
  - You pay for AWS resources (e.g. EC2 instances or EBS volumes) you create to store and run your application
  - You only pay for what you use, as you use it; there are no minimum fees and no upfront commitments

Compute documentation

- Amazon EC2 documentation
  - https://docs.aws.amazon.com/ec2/
- Amazon EC2 Auto Scaling
  - https://aws.amazon.com/ec2/autoscaling/
- AWS Lambda
  - https://docs.aws.amazon.com/lambda/
- Amazon Elastic Container Service (ECS)
  - https://docs.aws.amazon.com/ecs/
- AWS Elastic Beanstalk
  - https://docs.aws.amazon.com/elastic-beanstalk/