Containers

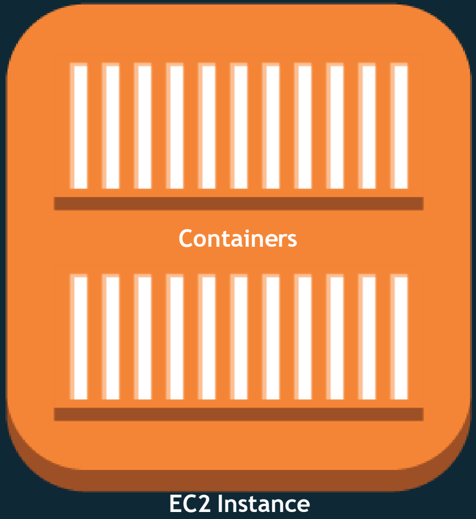
Vinícius Lima
Developer @ Infomach

Container Orchestration

At first there was

Amazon EC2

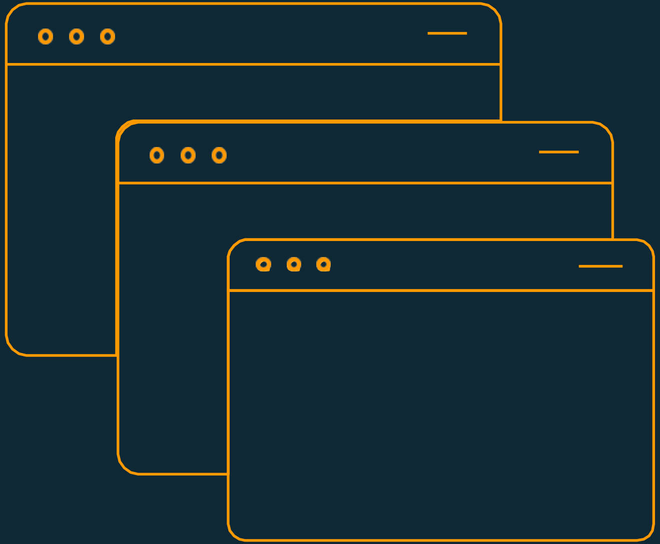
Then Docker!



Customers started containerizing applications within EC2 instances

Microservices

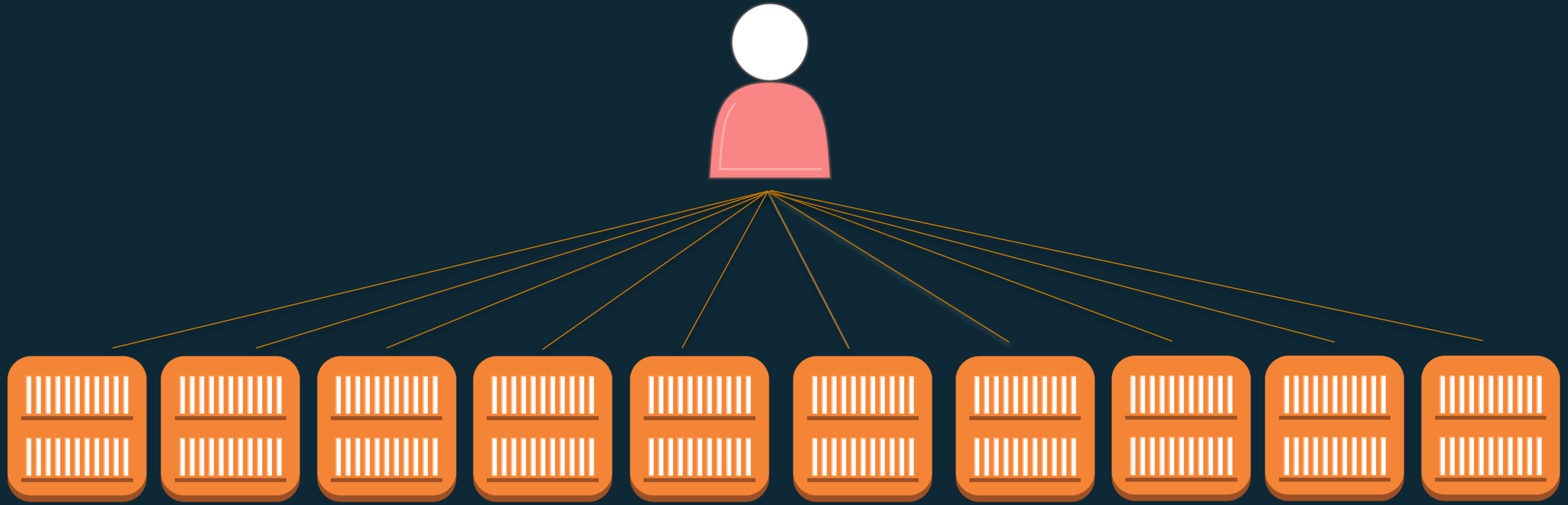
Batch Jobs



Migration to the Cloud

Containers made it easy to build and scale cloud-native applications

Customers needed an easier way to manage large clusters of instances and containers



Container services on AWS

Image Repository

Where the Docker imagesare stored

Amazon Elastic Container Registry (ECR)

Container services on AWS

Host
Where the containers are executed

Amazon EC2

AWS Fargate

Image Repository
Where the Docker images are stored

Amazon Elastic Container Registry (ECR)

Container services on AWS

Administration
Implementation, scheduling, scale and administration of the containers

Amazon Elastic Container Service (ECS)

Amazon Elastic Container Service for Kubernetes (EKS)

Host
Where the containersare executed

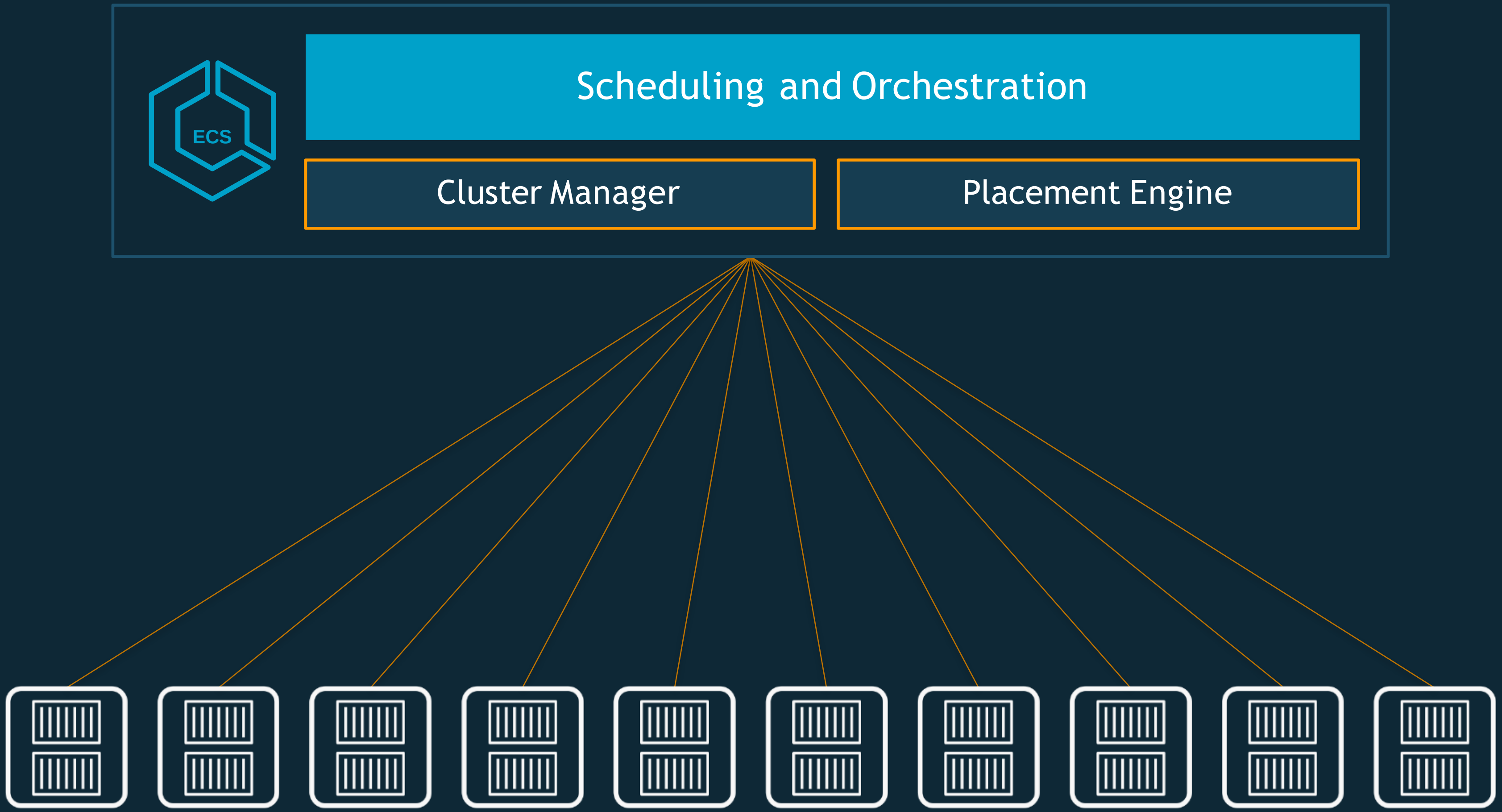
Amazon EC2

AWS Fargate

Image Repository
Where the Docker imagesare stored

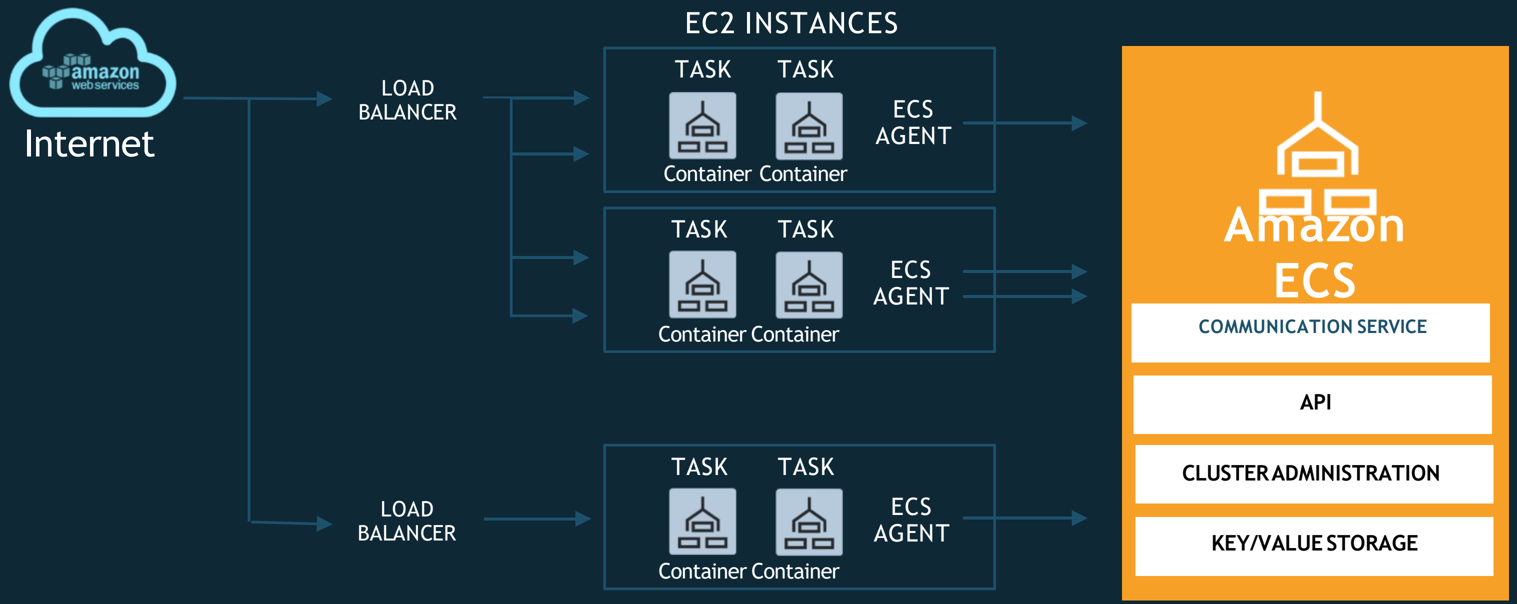
Amazon Elastic Container Registry (ECR)

Amazon Elastic Container Service (ECS)

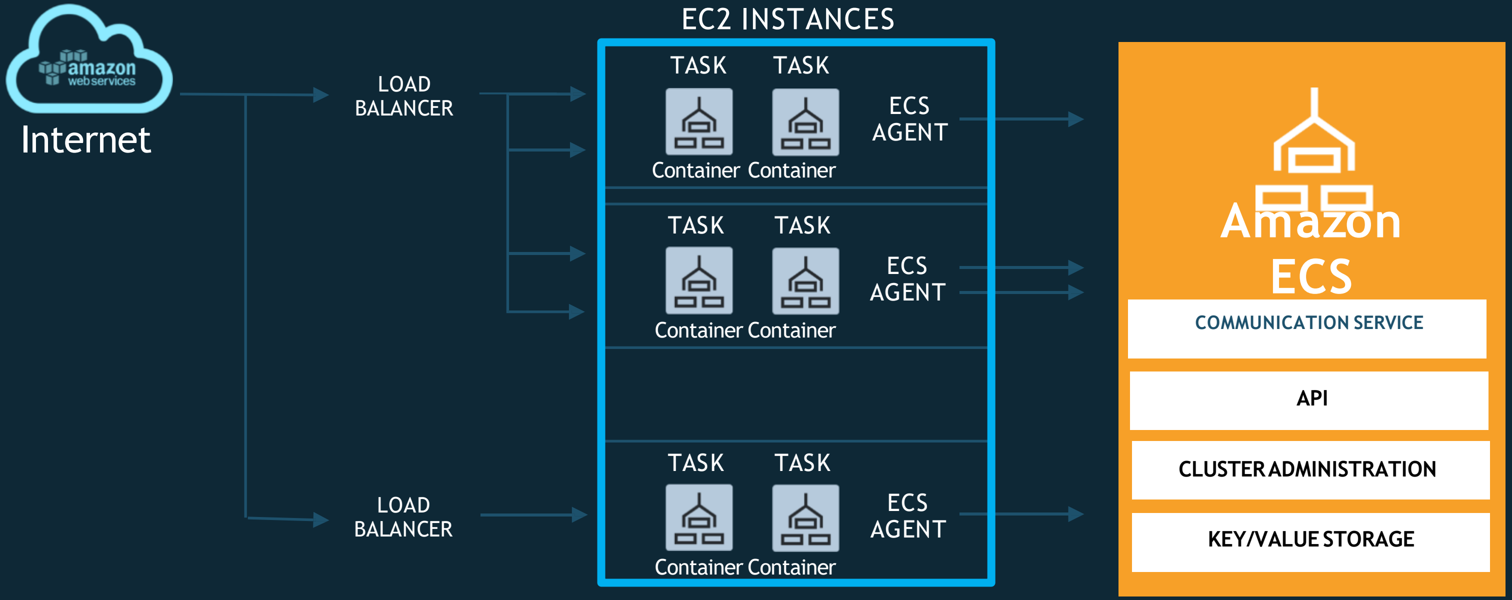

ECS
Scheduling and Orchestration
Cluster Manager
Placement Engine

ECS overview

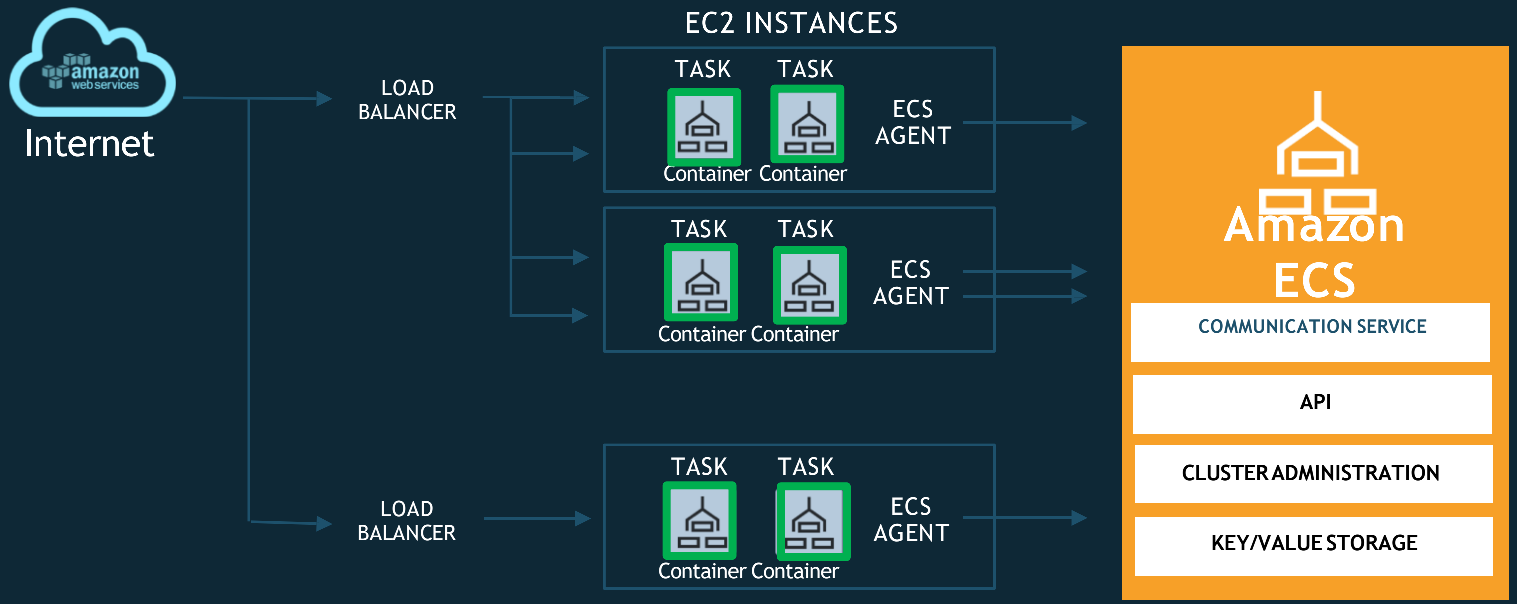
Amazon ECS


amazon web services
Internet
LOAD BALANCER
LOAD BALANCER
EC2 INSTANCES
TASK
TASK
Container
Container
ECS AGENT
TASK
TASK
Container
Container
ECS AGENT
TASK
TASK
Container
Container
ECS AGENT
Amazon ECS
COMMUNICATION SERVICE
API
CLUSTER ADMINISTRATION
KEY/VALUE STORAGE

Amazon ECS - Cluster



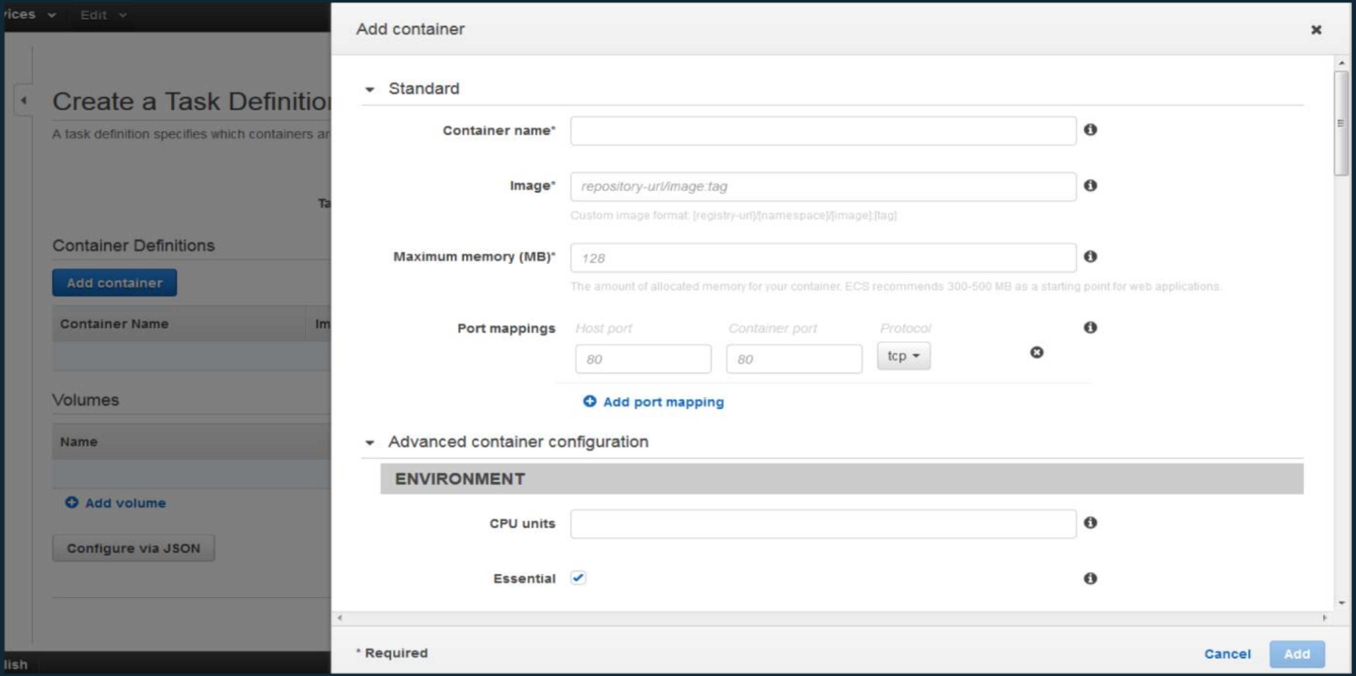
Amazon ECS - Task



Tasks

- Work Unit
- Group of containers
- Execute inside a container instance / EC2

Task Definitions



ices · Edit

Create a Task Definitio

A task definition specifies which containers ar

Ta

Container Definitions

Add container

Container Name | Im

Volumes

Name

Add volume

Configure via JSON

lish

Add container

Standard

Container name*

Image*: repository-url/image:tag

Custom image format: [registry-url]/[namespace]/[image]:[tag]

Maximum memory (MB)*: 128

The amount of allocated memory for your container. ECS recommends 300-500 MB as a starting point for web applications.

Port mappings

Host port	Container port	Protocol
80	80	tcp

Add port mapping

Advanced container configuration

ENVIRONMENT

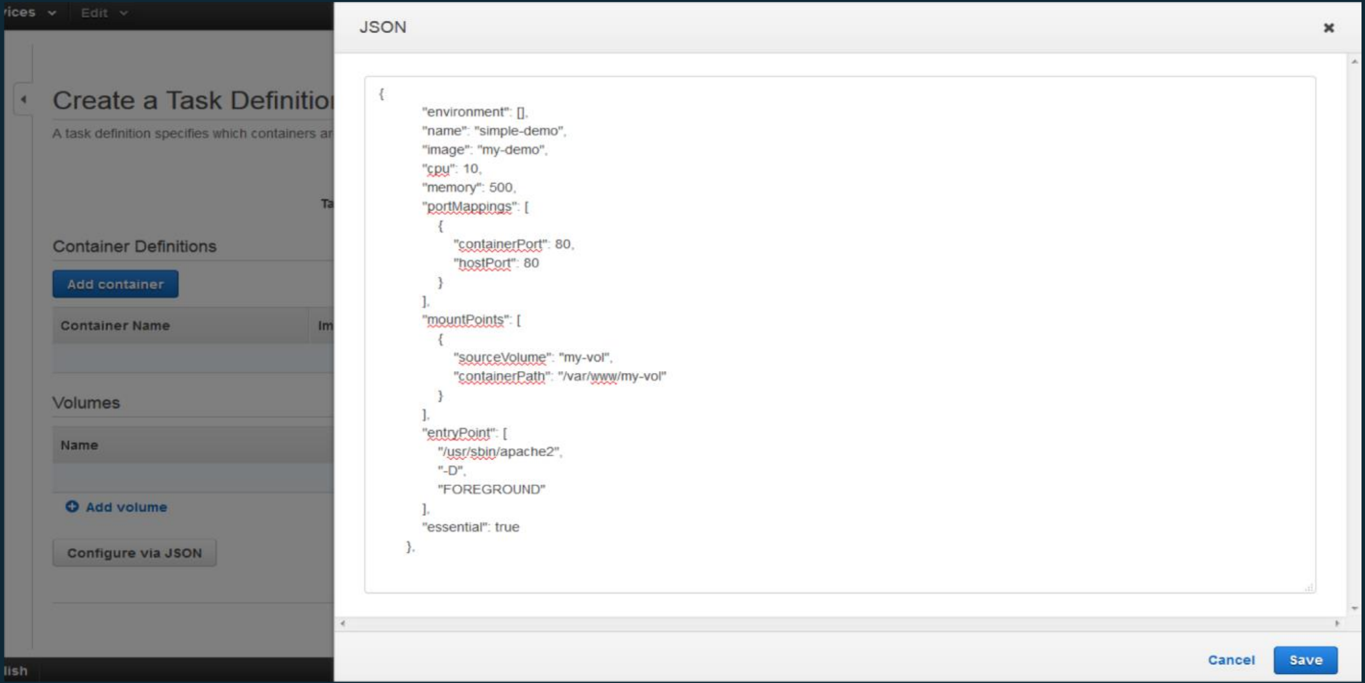
CPU units

Essential ✓

* Required

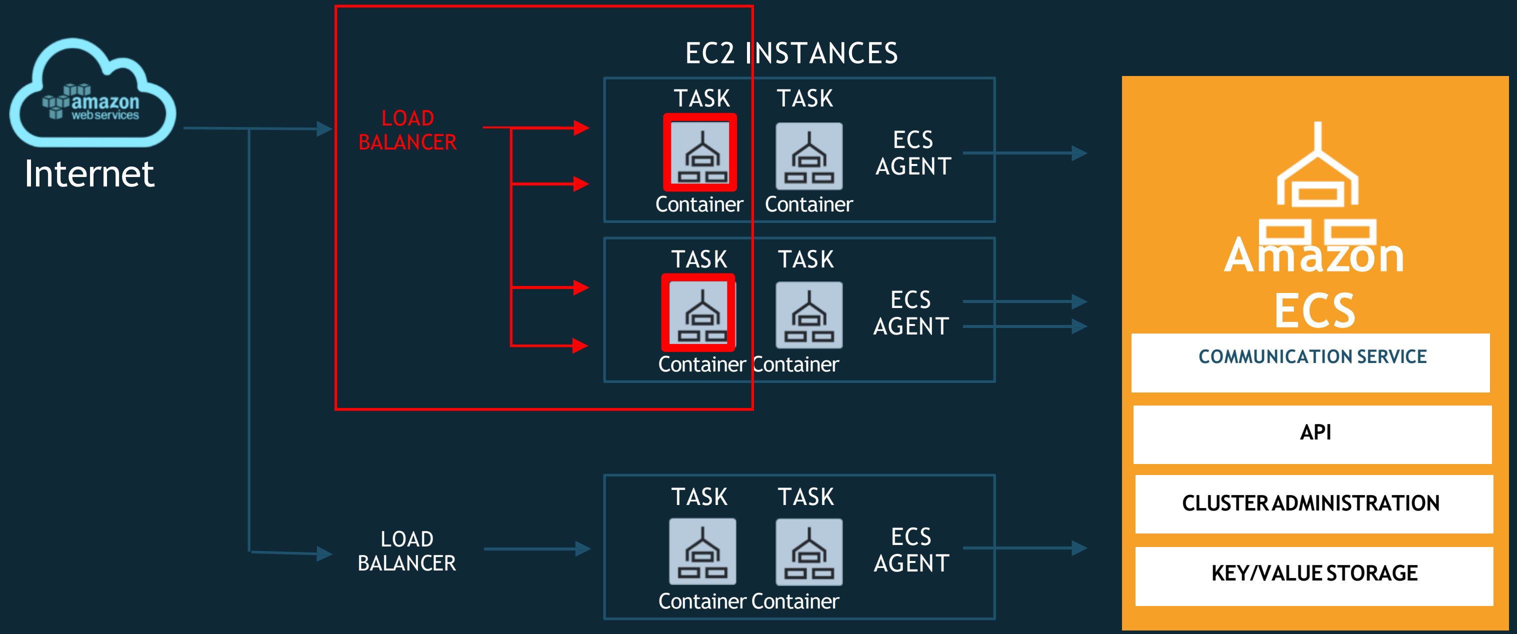
Cancel · Add

Task Definitions



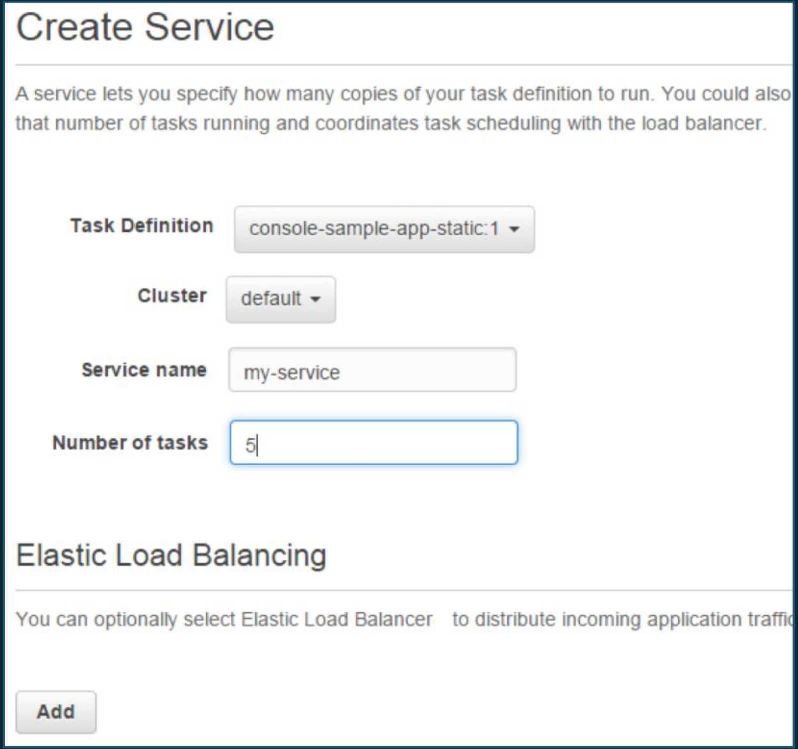
```
{
        "environment": [],
        "name": "simple-demo",
        "image": "my-demo",
        "cpu": 10,
        "memory": 500,
        "portMappings": [
          {
            "containerPort": 80,
            "hostPort": 80
          }
        ],
        "mountPoints": [
          {
            "sourceVolume": "my-vol",
            "containerPath": "/var/www/my-vol"
          }
        ],
        "entryPoint": [
          "/usr/sbin/apache2",
          "-D",
          "FOREGROUND"
        ],
        "essential": true
    },
```

Amazon ECS - Service



Create a Service

For long duration applications



Create Service

A service lets you specify how many copies of your task definition to run. You could also that number of tasks running and coordinates task scheduling with the load balancer.

Task Definition	console-sample-app-static:1
Cluster	default
Service name	my-service
Number of tasks	5

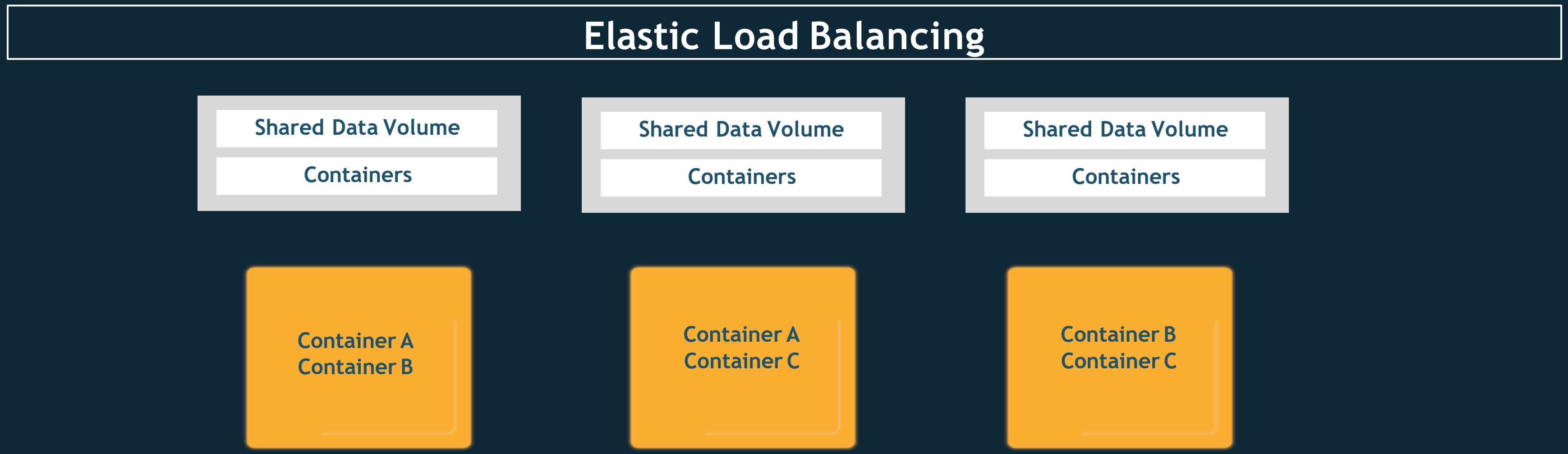
Elastic Load Balancing

You can optionally select Elastic Load Balancer to distribute incoming application traffi

Create a Service

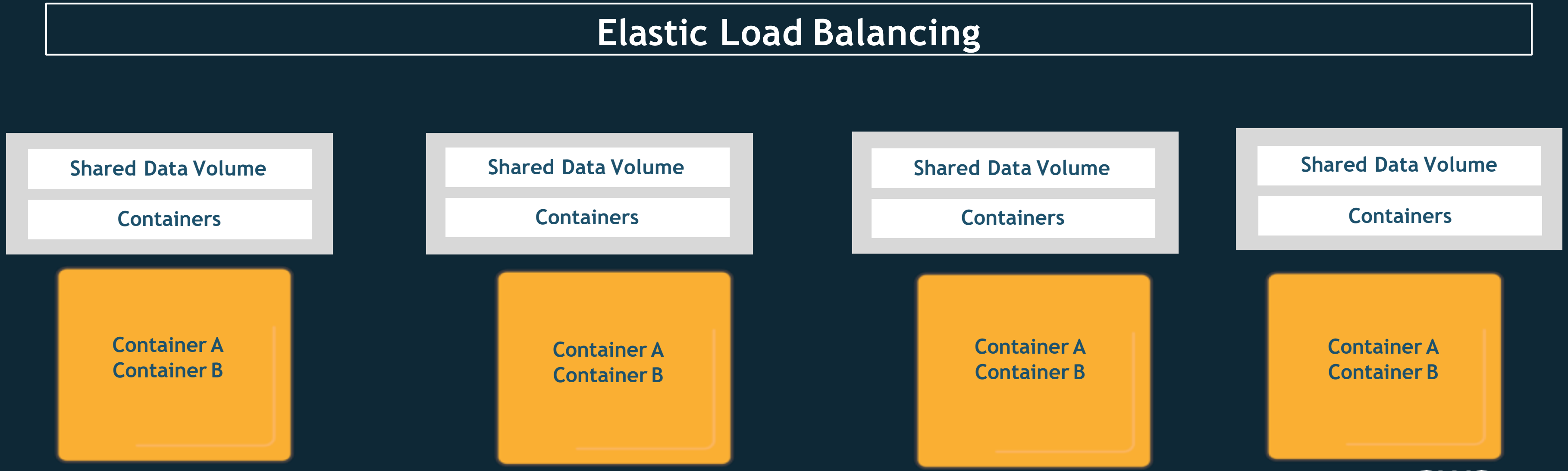
Load balancing between containers

Automatic failover in case of failure



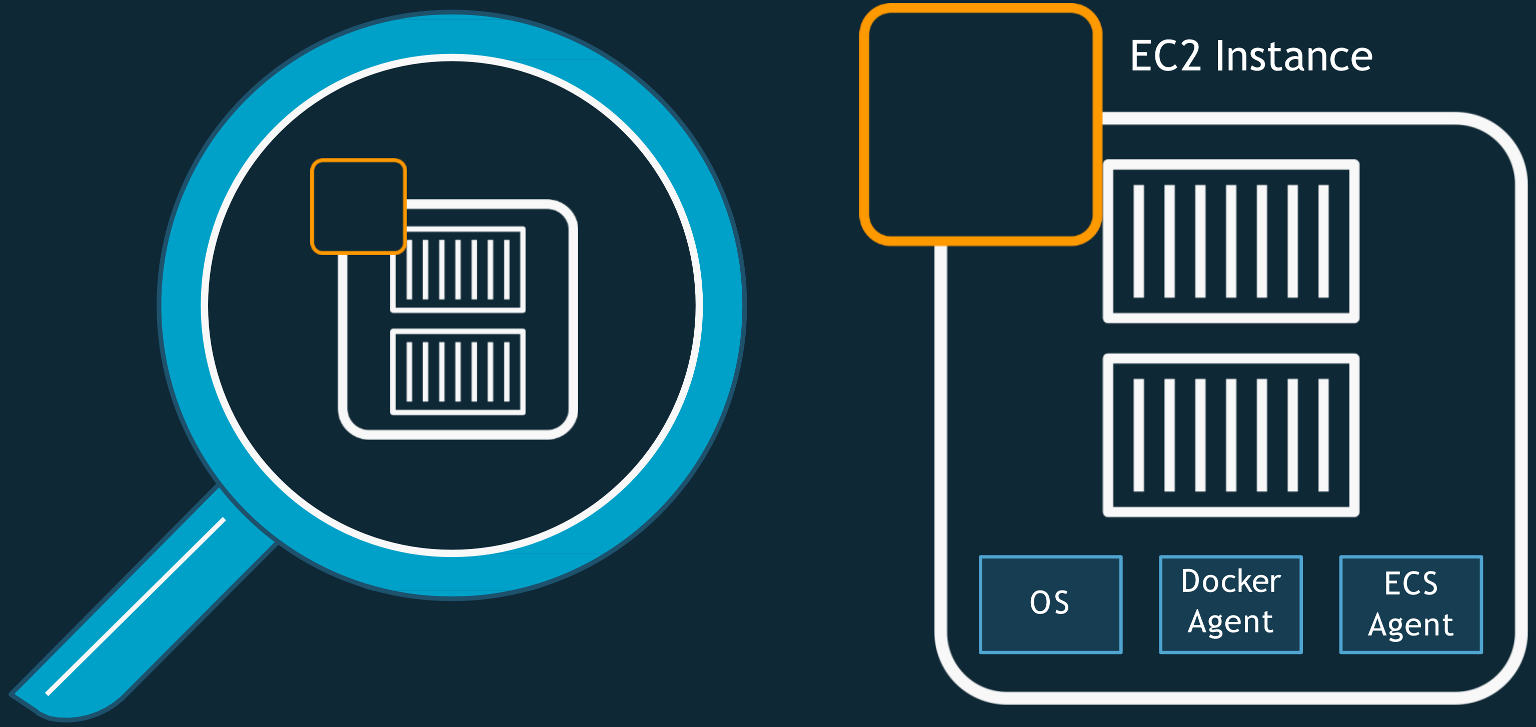
Scale a Service

Scale out and Scale in



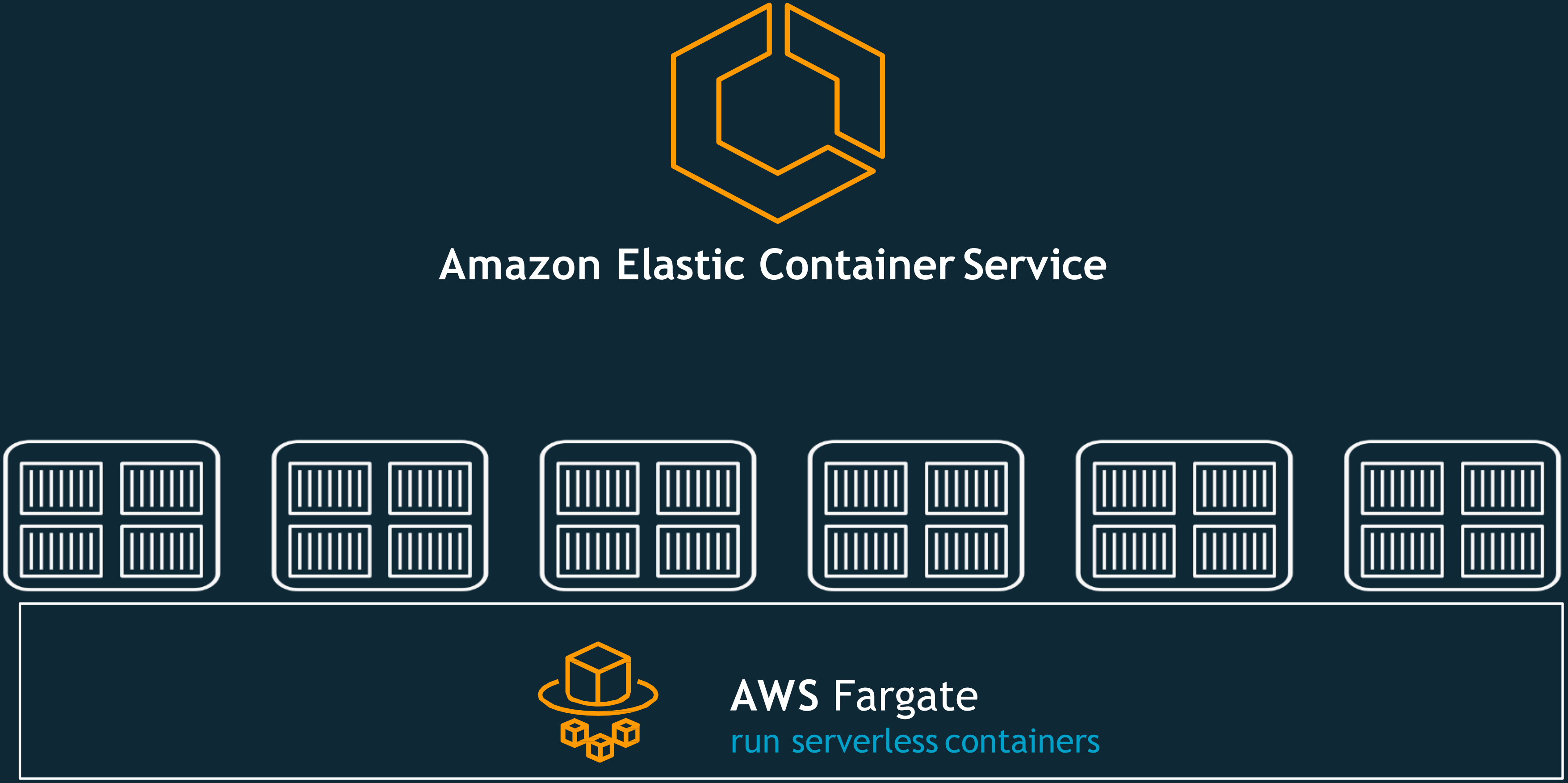
AWS Fargate

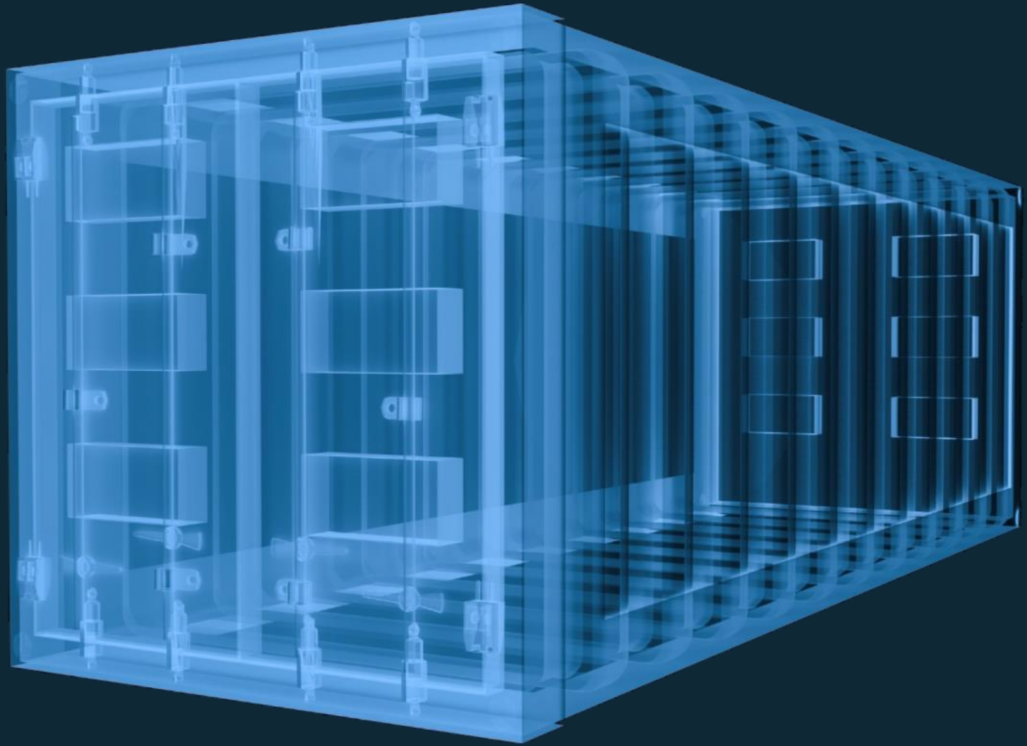
Without Fargate, you end up managing more than just containers



- Patching and Upgrading OS, agents, etc.

- Scaling the instance fleet for optimal utilization


Amazon Elastic Container Service
AWS Fargate
run serverless containers



Run Serverless Containers with Fargate

AWS Fargate

Your containerized applications

Managed by AWS
No EC2 Instances to provision, scale or manage

Elastic
Scale up & down seamlessly. Pay only for what you use

Integrated
with the AWS ecosystem: VPC Networking, Elastic Load Balancing, IAM Permissions, CloudWatch and more

Fully managed container environment with AWS ECS + Fargate

Bring existing code

No changes required of existing code, works with existing workflows and microservices built on Amazon ECS

Production ready

ISO, PCI, HIPAA, SOC compliant. Launch ten or tens of thousands of containers in seconds in 9 global regions (+7 in 2018)

Powerful integrations

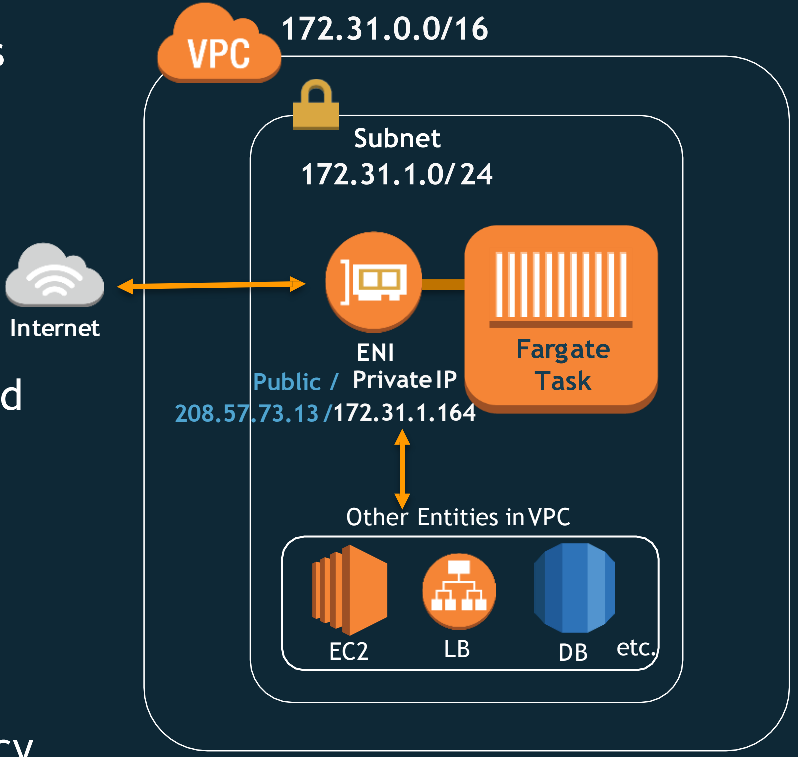
Native AWS integrations for networking, security, CICD, monitoring, and tracing

Fargate runs tens of millions of containers for AWS customers every week

Networking

VPC INTEGRATION

- *AWS VPC* Networking Mode – each task gets its own interface
- All Fargate Tasks run in customer VPC and subnets
- Configure security groups to control inbound & outbound traffic
- Public IP support
- Spread your application across subnets in multiple Availability Zones (AZs) for resiliency



Containers on AWS

Infomach

Vinícius Lima
vinicius@infomach.com.br