

# Automating Docker on AWS

March 29, 2017  
Sydney Docker User Group

Taylor Bertie

Marcus Santos

# Agenda

Docker and AWS

EC2 Container Service (ECS) Overview

ECS Integration with AWS products

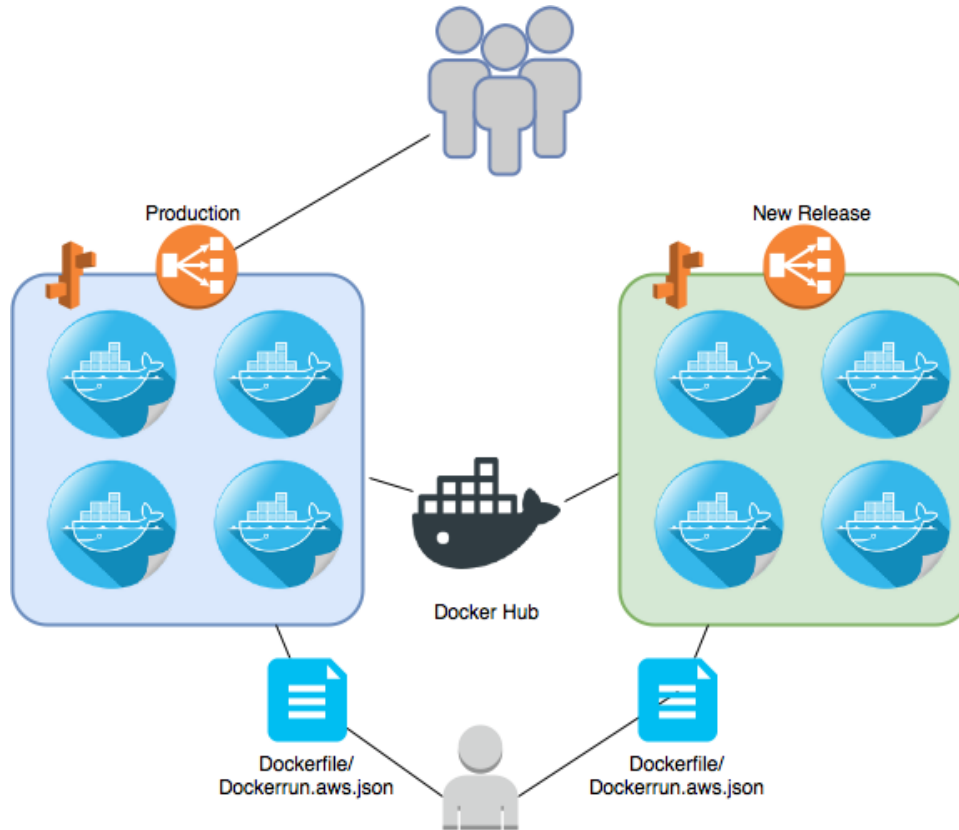
Demo

ECS Best Practices

# Docker Solutions from AWS

- Elastic Beanstalk
- AWS Batch
- EC2 Container Service

# Deploying Docker containers using Elastic Beanstalk



# AWS Batch

## Dashboard

### Job status

Submit job

Queue name (priority) ▾	Submitted ▾	Pending ▾	Runnable ▾	Starting ▾	Running ▾	Failed ▾	Succeeded
test-queue (500)	1	1	1	1	1	1	1
production-queue (1000)	1231	942	12	104020	57	17	8742

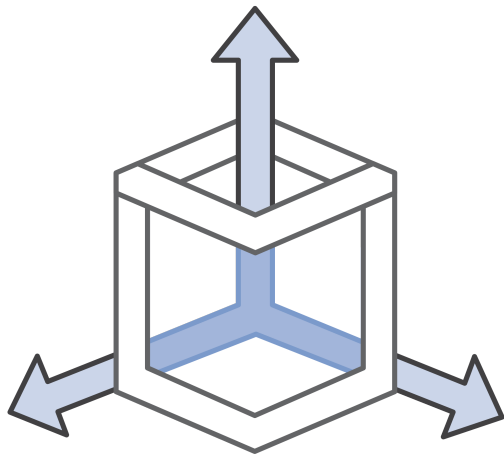
### Job queues

Name ▾	Priority ▾	Pending job count ▾	Running job count ▾	Registered instance count ▾	Total vCPUs
test-queue	500	1	1	1	4
production-queue	1000	942	104202	20	96

### Compute environments

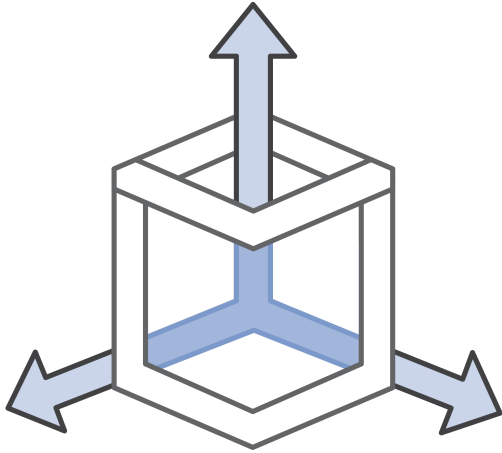
Name ▾	Type ▾	Running job count ▾	Desired vCPUs ▾	Registered instance count
spot-env	managed	1002301	1600	100
production	unmanaged	12	1000	100

# EC2 Container Service



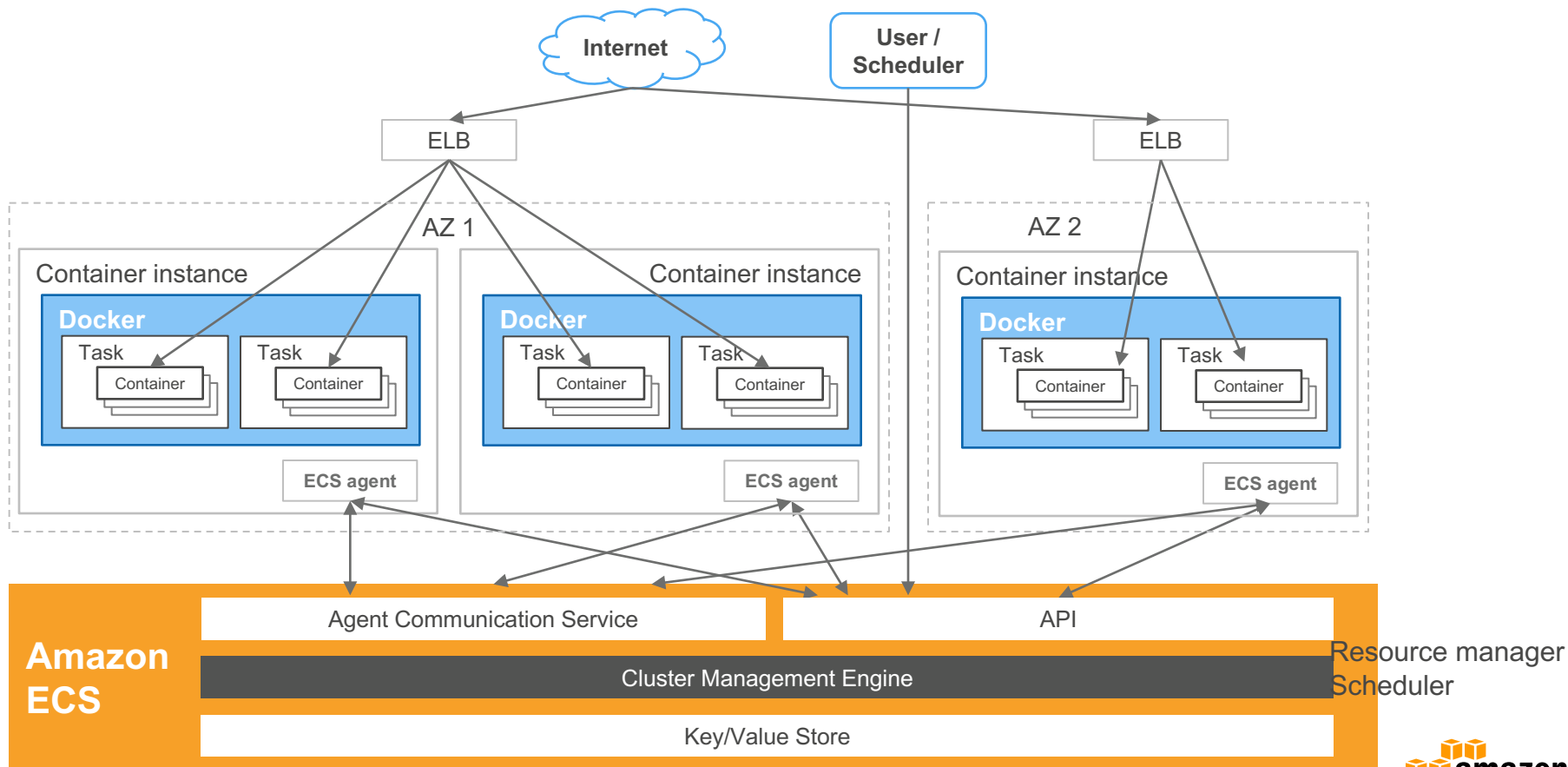
- **Eliminates cluster management software**
- **Manages cluster state**
- **Manages containers**
- **Control and monitoring**
- **Scale from one to tens of thousands of containers**

# EC2 Container Service terminology.



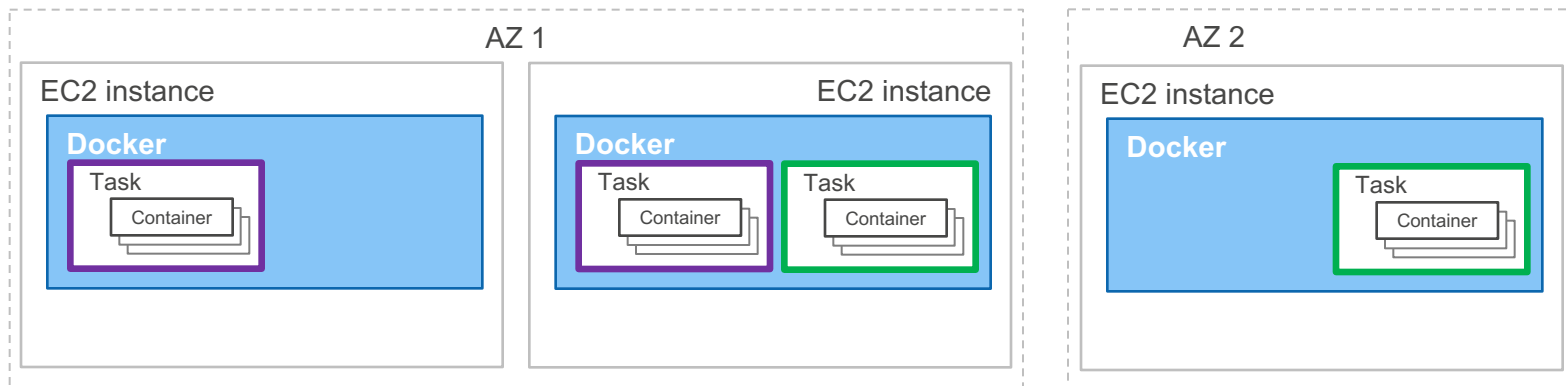
- Task Definition → Docker Compose
- Task → Running Container
- EC2 Container Registry → Docker Hub

# Amazon ECS

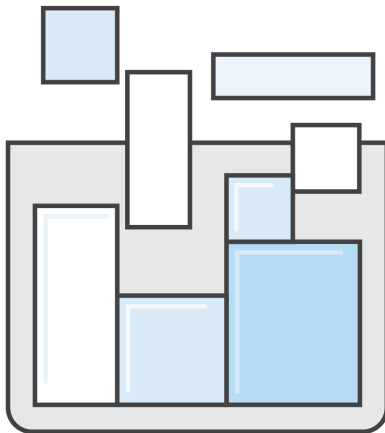




# Cluster management: Scheduling

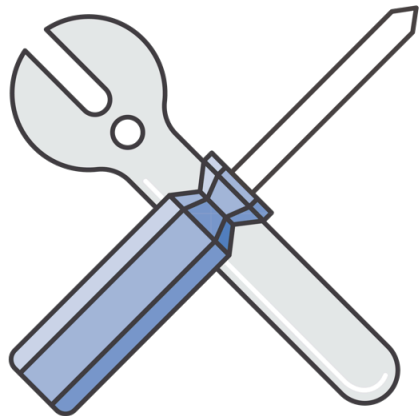


# Flexible container placement



- Applications / Services
- Batch jobs /RunTask
- Multiple schedulers

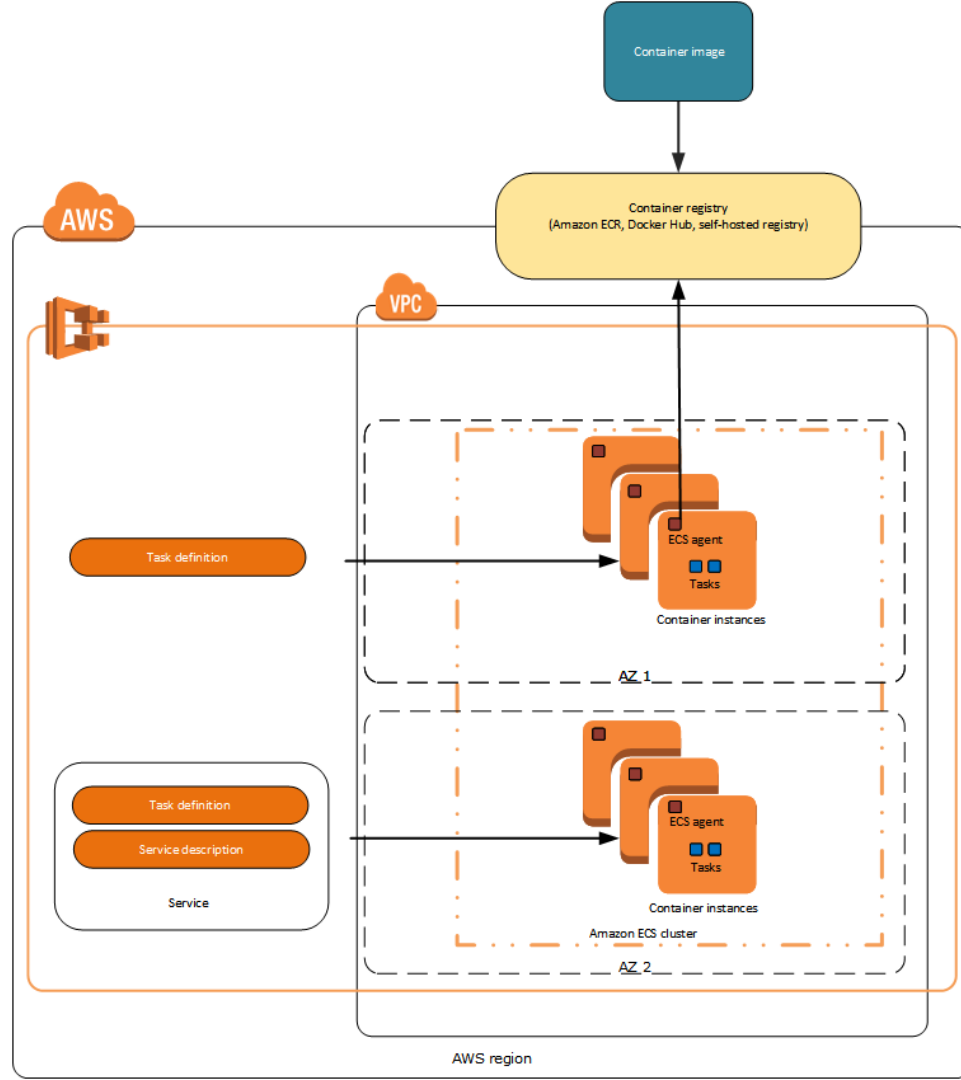
# Extensible



- Comprehensive APIs
- Custom schedulers
- Open source agent and CLI

# ECS Components

- ECS Cluster
- Container Agent
- Task Definitions
- Tasks and Scheduling
- Services and ECR



# ECS Cluster

[dockermeetup01 >](#)

1  
Services

4  
Running tasks  
0  
Pending tasks

0.00%  
CPU Utilization

0.05%  
Memory Utilization

4  
Container instances

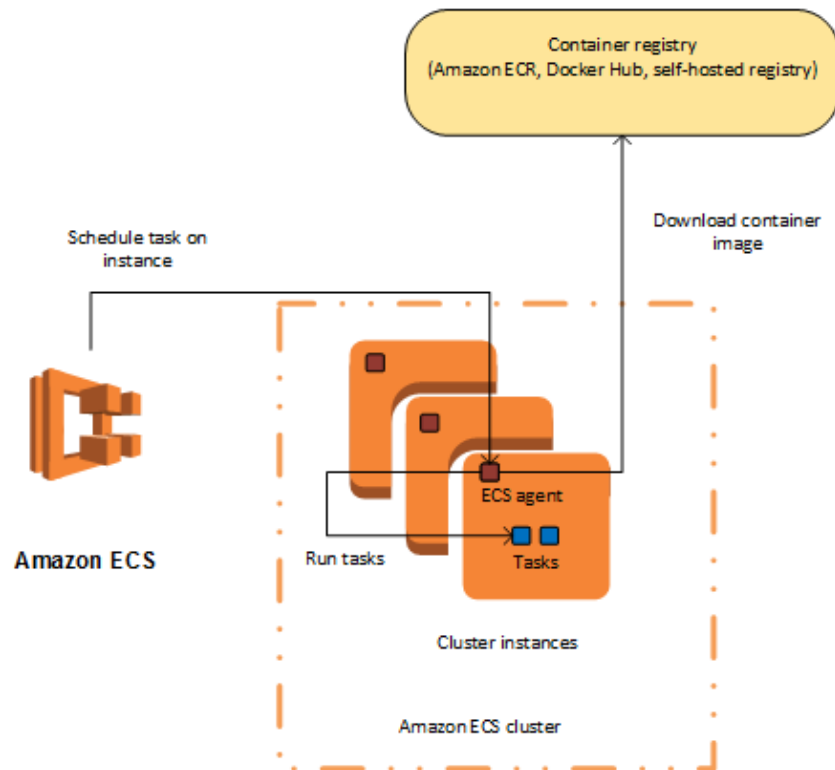
Status: **ALL** ACTIVE DRAINING

Filter in this page

< 1-4 > Page size 50 ▼

<input type="checkbox"/>	Container Instance	EC2 Instance	Availa...	Agent...	Status	Runni...	CPU a...	Memo...	Agent...	Dock...
<input type="checkbox"/>	1a6fc99f-4497-4cf6-995e-...	i-0416dc054db...	us-eas...	true	ACTIVE	1	2048	3255	1.14.0	1.12.6
<input type="checkbox"/>	286a2368-9af0-4402-8eec...	i-0262a411578...	us-eas...	true	ACTIVE	1	2048	3255	1.14.0	1.12.6
<input type="checkbox"/>	cd4f50e1-169e-41df-a01f-...	i-0b11c96c42f5...	us-eas...	true	ACTIVE	1	2048	3255	1.14.0	1.12.6
<input type="checkbox"/>	e261ed44-587e-4177-8c5...	i-04a98a57c7b...	us-eas...	true	ACTIVE	1	2048	3255	1.14.0	1.12.6

# Container Agent



# Key components: Task definitions

The screenshot shows the 'Add container' dialog in the AWS Management Console. The dialog is divided into two main sections: 'Standard' and 'Advanced container configuration'. The 'Standard' section includes fields for 'Container name\*', 'Image\*' (with a custom image format hint), 'Maximum memory (MB)\*' (with a recommendation of 300-500 MB), and 'Port mappings' (with a table for Host port, Container port, and Protocol). The 'Advanced container configuration' section includes a field for 'CPU units' and a checkbox for 'Essential'. The dialog also features a sidebar on the left with options like 'Create a Task Definition', 'Add container', 'Container Name', 'Image', 'Volumes', and 'Add volume'. The bottom of the dialog has 'Cancel' and 'Add' buttons.

**Add container**

▼ Standard

Container name\*

Image\*   
Custom image format: [registry-uri]/[namespace]/[image]:[tag]

Maximum memory (MB)\*   
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
<input type="text" value="80"/>	<input type="text" value="80"/>	<input type="text" value="tcp"/>

[+ Add port mapping](#)

▼ Advanced container configuration

**ENVIRONMENT**

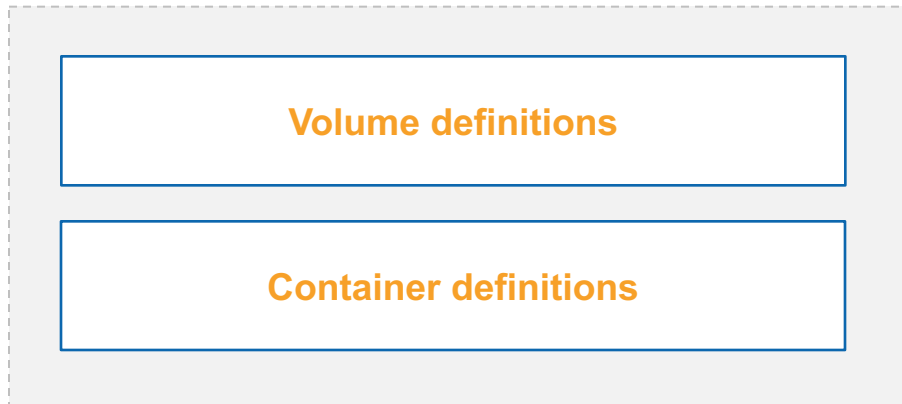
CPU units

Essential ☒

\* Required

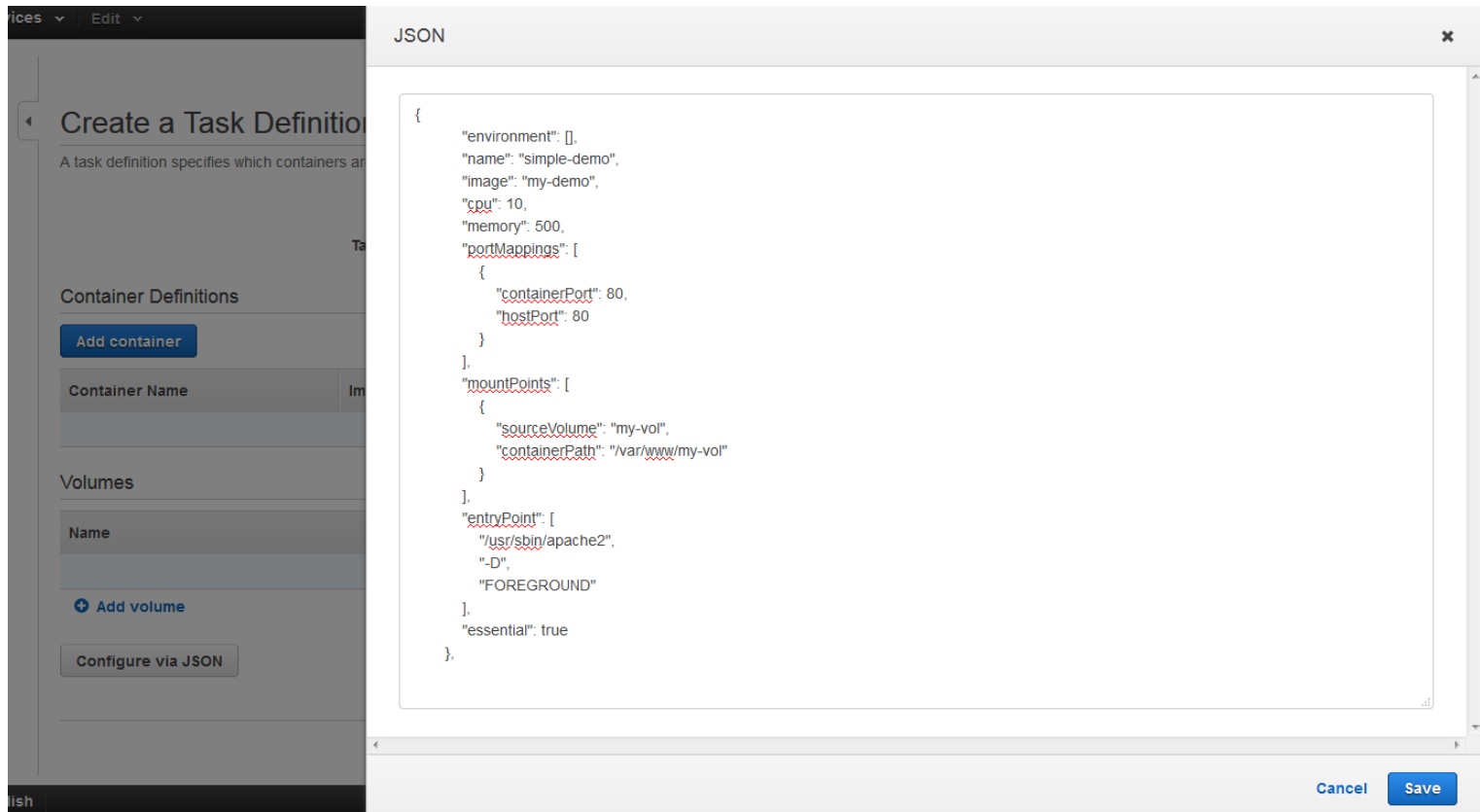
[Cancel](#) [Add](#)

# Task definitions





# Key components: Task definitions

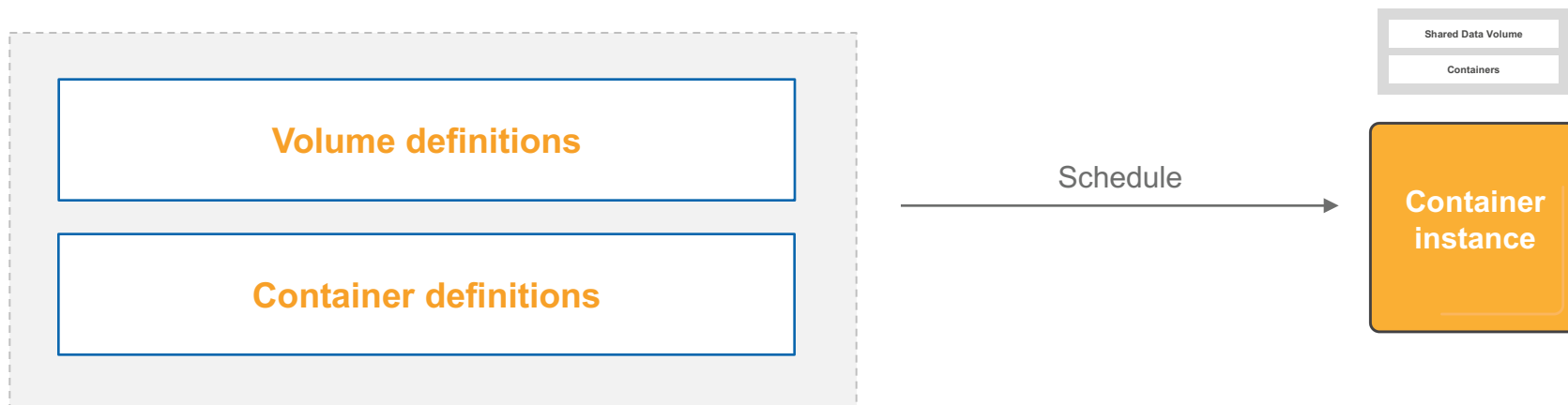


The screenshot shows the AWS ECS console interface for creating a task definition. The left sidebar has a 'Create a Task Definition' section with a description: 'A task definition specifies which containers are used by a task.' Below this are sections for 'Container Definitions' (with an 'Add container' button), 'Volumes' (with an 'Add volume' button), and a 'Configure via JSON' button. The main area is titled 'JSON' and displays a task definition configuration in JSON format. The configuration includes environment variables, container name, image, CPU, memory, port mappings, mount points, entry point, and essential status.

```
{
  "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
}
```

At the bottom right of the console window, there are 'Cancel' and 'Save' buttons.

# Tasks



# Tasks

- Unit of work
- Grouping of related containers
- Run on container instances

# Create service

Good for long-running applications and services

## 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.

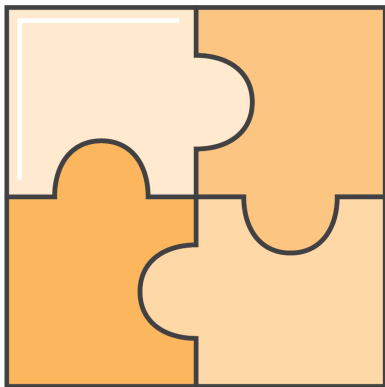
Task Definition	<input type="text" value="console-sample-app-static:1"/>
Cluster	<input type="text" value="default"/>
Service name	<input type="text" value="my-service"/>
Number of tasks	<input type="text" value="5"/>

## Elastic Load Balancing

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

Add

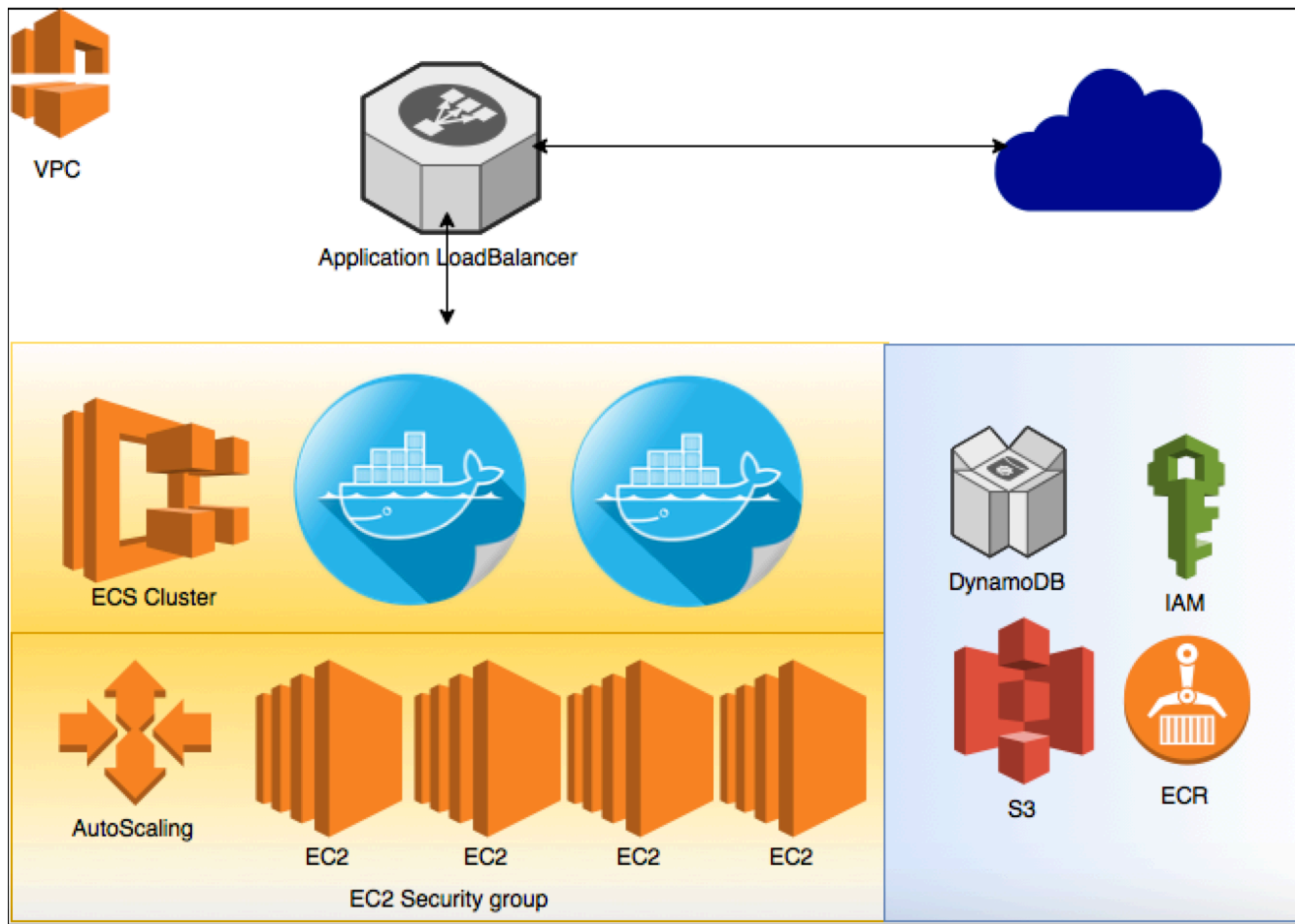
# Integration ECS with AWS products



- Elastic Load Balancing
- AWS Identity and Access Management
- AWS CloudTrail
- Amazon CloudWatch
- EC2 Container Registry (ECR)
- AWS Cloud Formation
- AWS Code\*
- Amazon Elastic Block Store
- Elastic File System
- Amazon Virtual Private Cloud

# Demo

# High Level Architecture



# Amazon Alexa



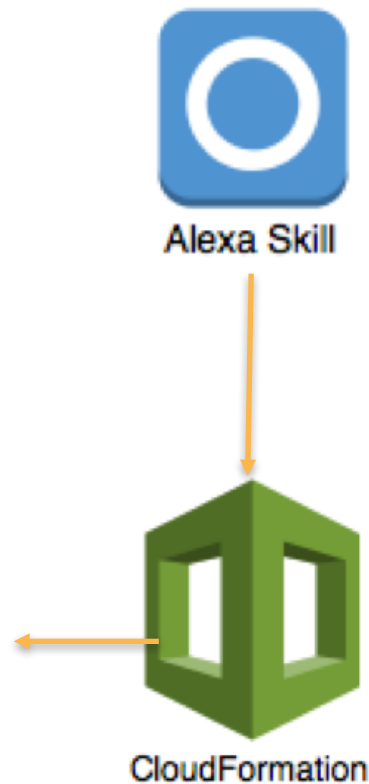
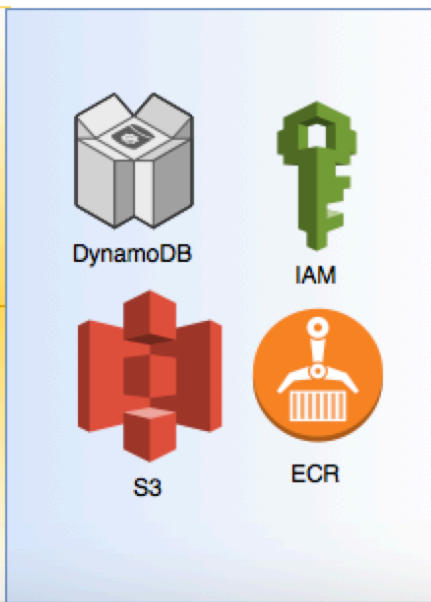
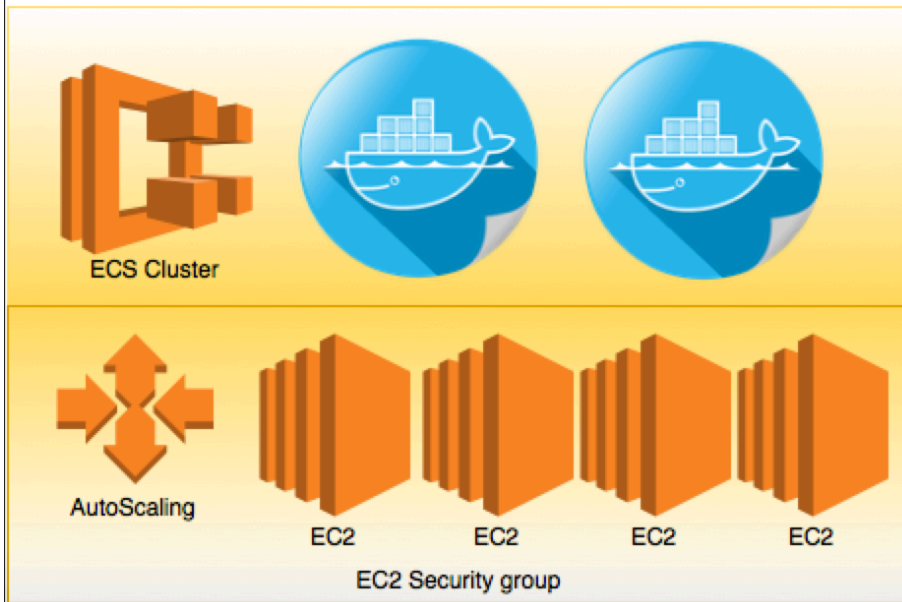
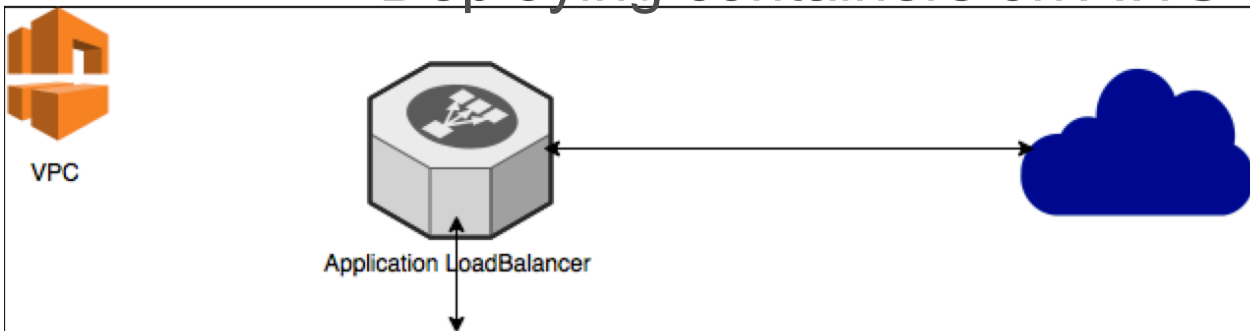
Alexa Skill



CloudFormation

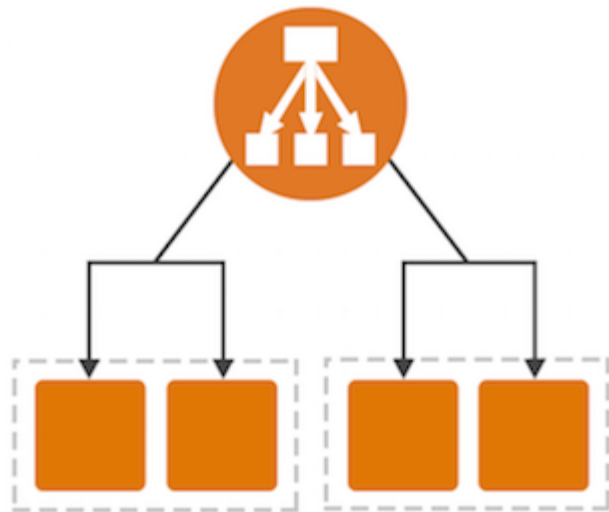


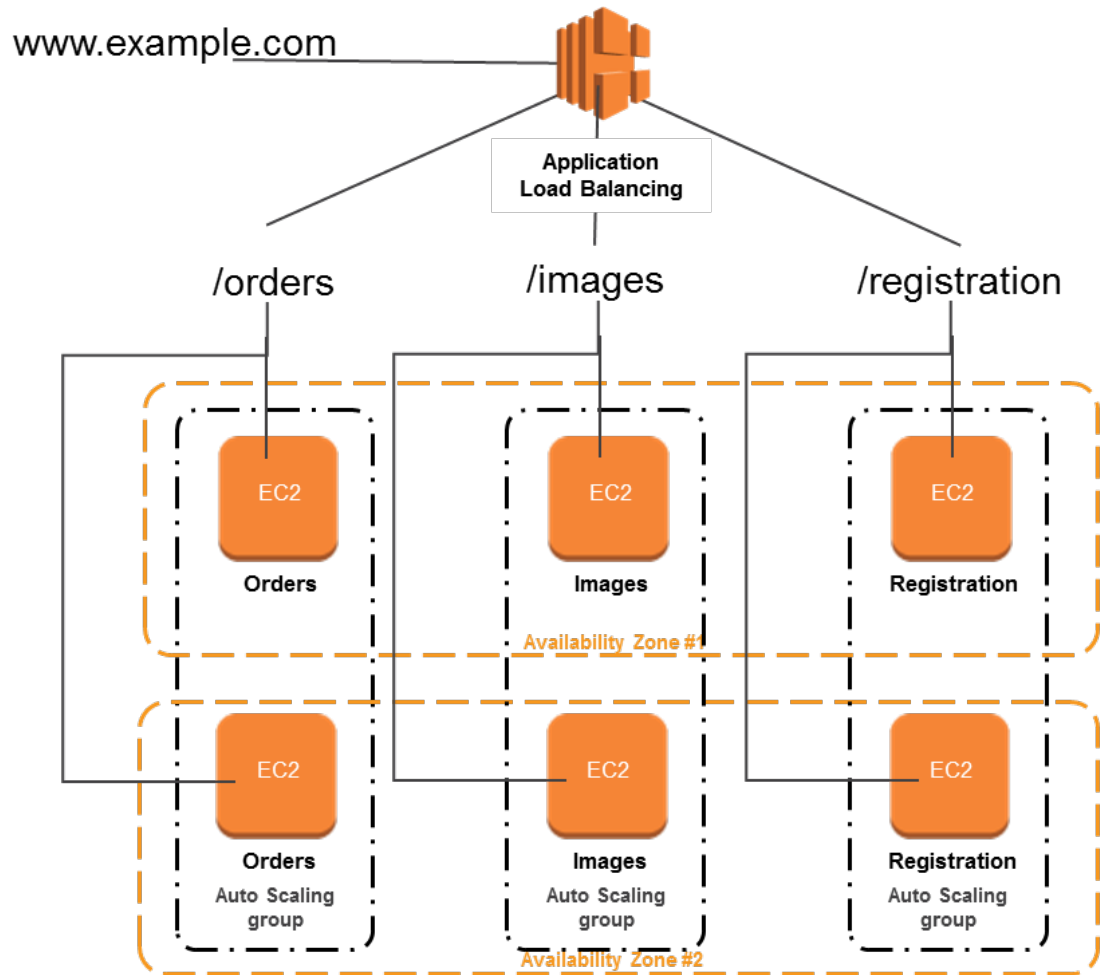
# Deploying containers on AWS



# Application Load Balancer

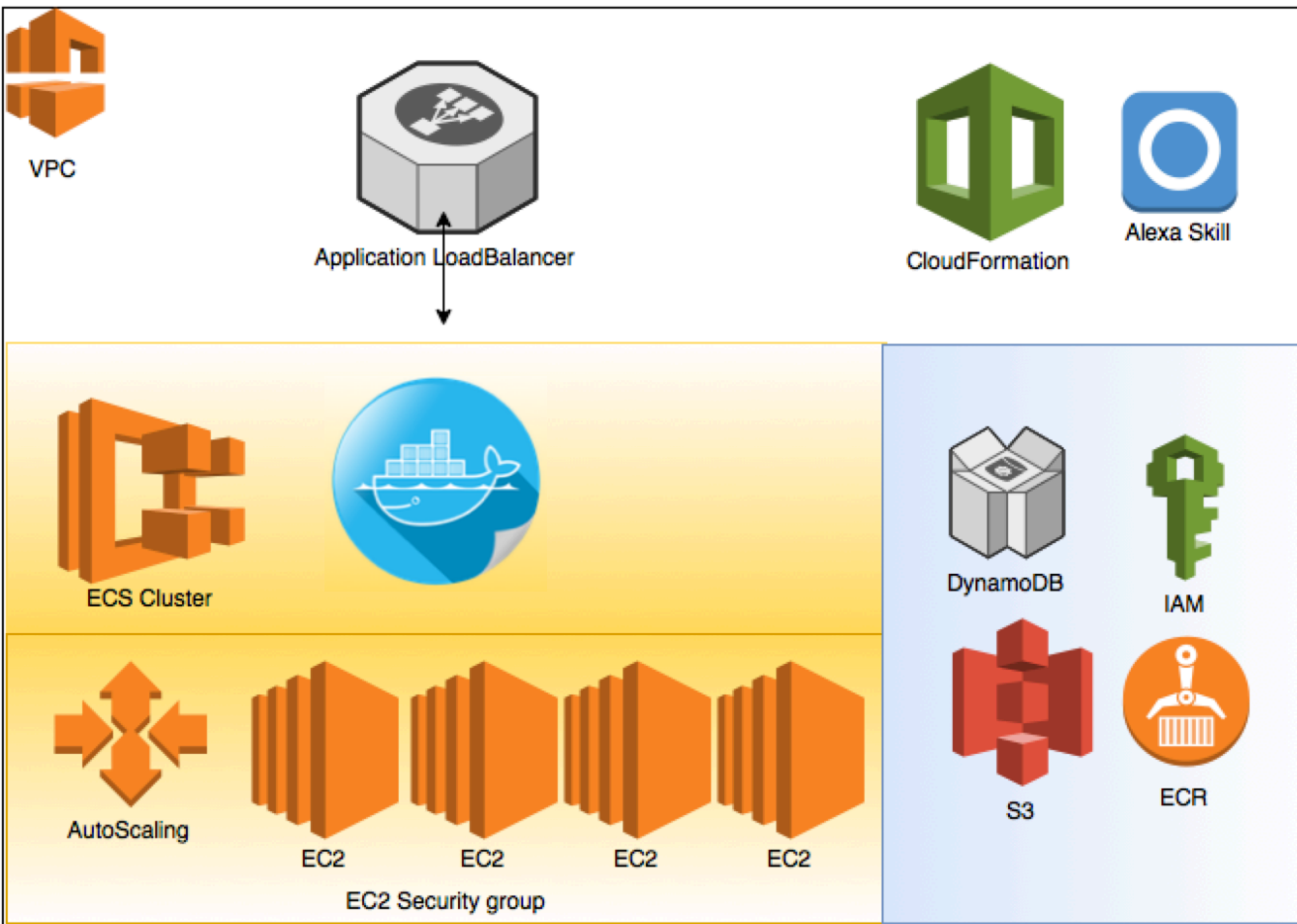
- A Layer 7 load balancer
- HTTP/2 support
- WebSocket support
- IPv6 Support
- Support for Container-Based Applications
- Content-Based Routing



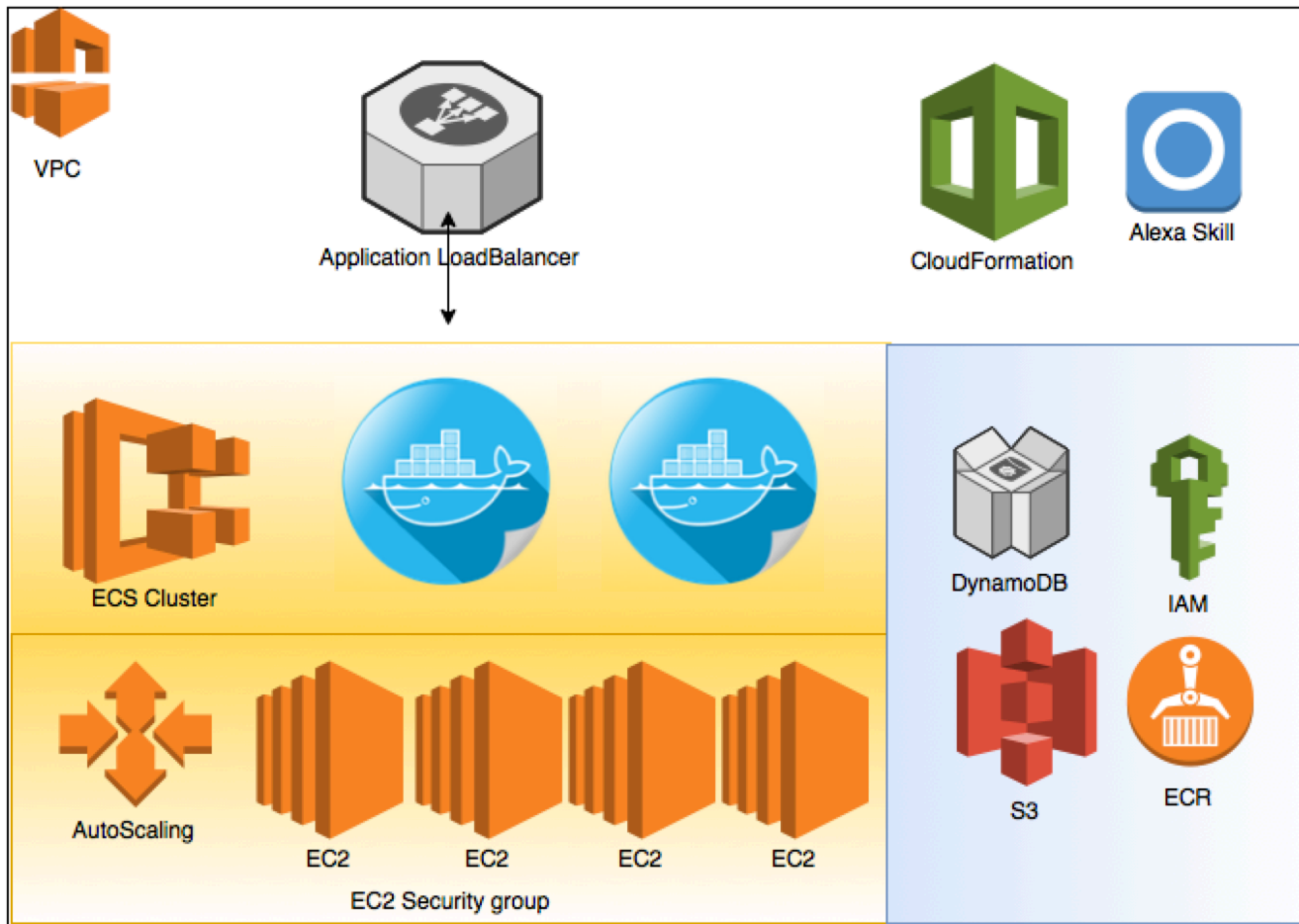


# Building the base infrastructure for ECS

Alexa tell  
demo to  
deploy the  
game

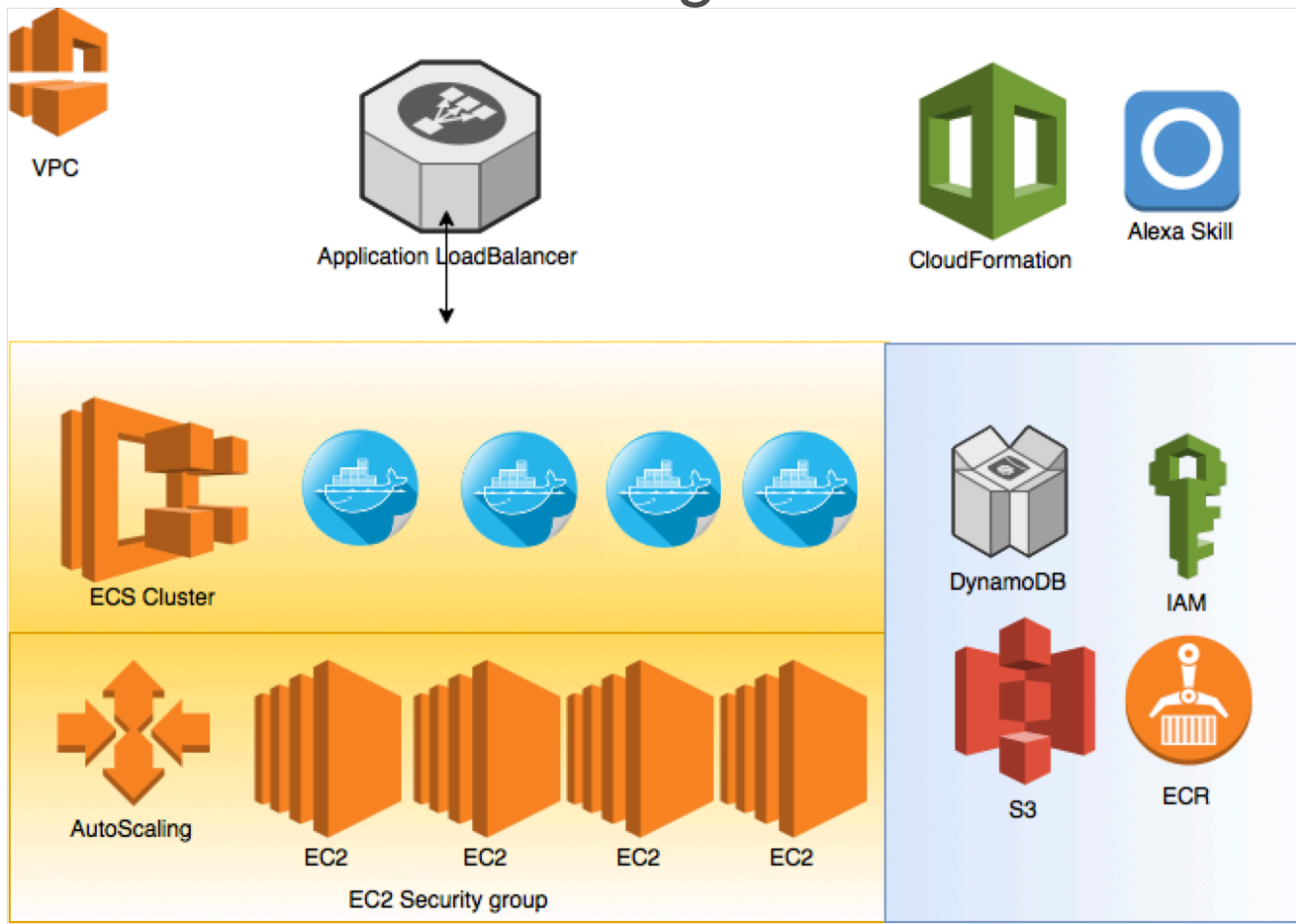


# Updating the Task Definition to add a container



Alexa tell  
demo deploy  
score submit  
container

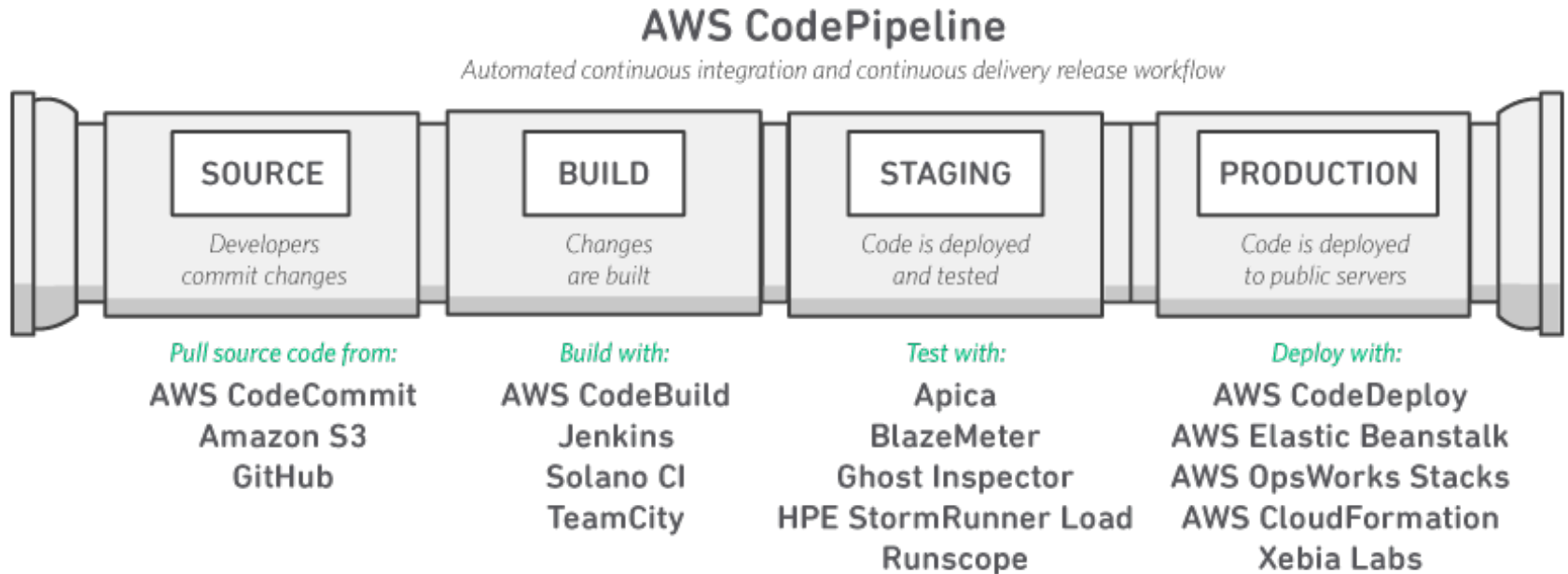
# Creating new Task definitions



Alexa tell  
demo to  
deploy high  
scores page

Alexa tell  
demo to  
deploy info  
page

# Continuous Integration in AWS



# ECS Best Practices

- CloudWatch logs integration
- Check ECS service Limits
- ECS log collector
- Cloud Trails logs and notification
- Autoscaling for scaling your cluster
- Integration with the Code\* series
- ECS credential helper



# WE ARE HIRING!!

- DevOps Engineers
- Software Developers
  - Linux specialists
- Systems Engineers

# Thank you!



<https://aws.nkh.io/info>

<https://github.com/SydneyDockerMeetupAWS>

# Resources:

- Docker and AWS: <https://aws.amazon.com/docker/>
- AWS Batch: <https://aws.amazon.com/batch/use-cases/>
- ECS First Run: <https://aws.amazon.com/getting-started/tutorials/deploy-docker-containers/>
- Scaling ECS: <https://www.youtube.com/watch?v=eun8CqGqdk8>
- AWS Sydney Summit: <https://aws.amazon.com/summits/sydney/agenda/>