

# Running Containers in AWS

ANDREW MAY



# About me

---



AWS Certified



Cloud Solutions Lead and Senior Solutions Architect  
at Leading EDJE



AWS Academy instructor  
at Columbus State Community College

# Docker

Orchestration (Services)

Discovery

Transient

Elastic Container  
Service (ECS)

EKS

Elastic Beanstalk

CloudMap

AppMesh

AWS Batch

CodeBuild

EC2

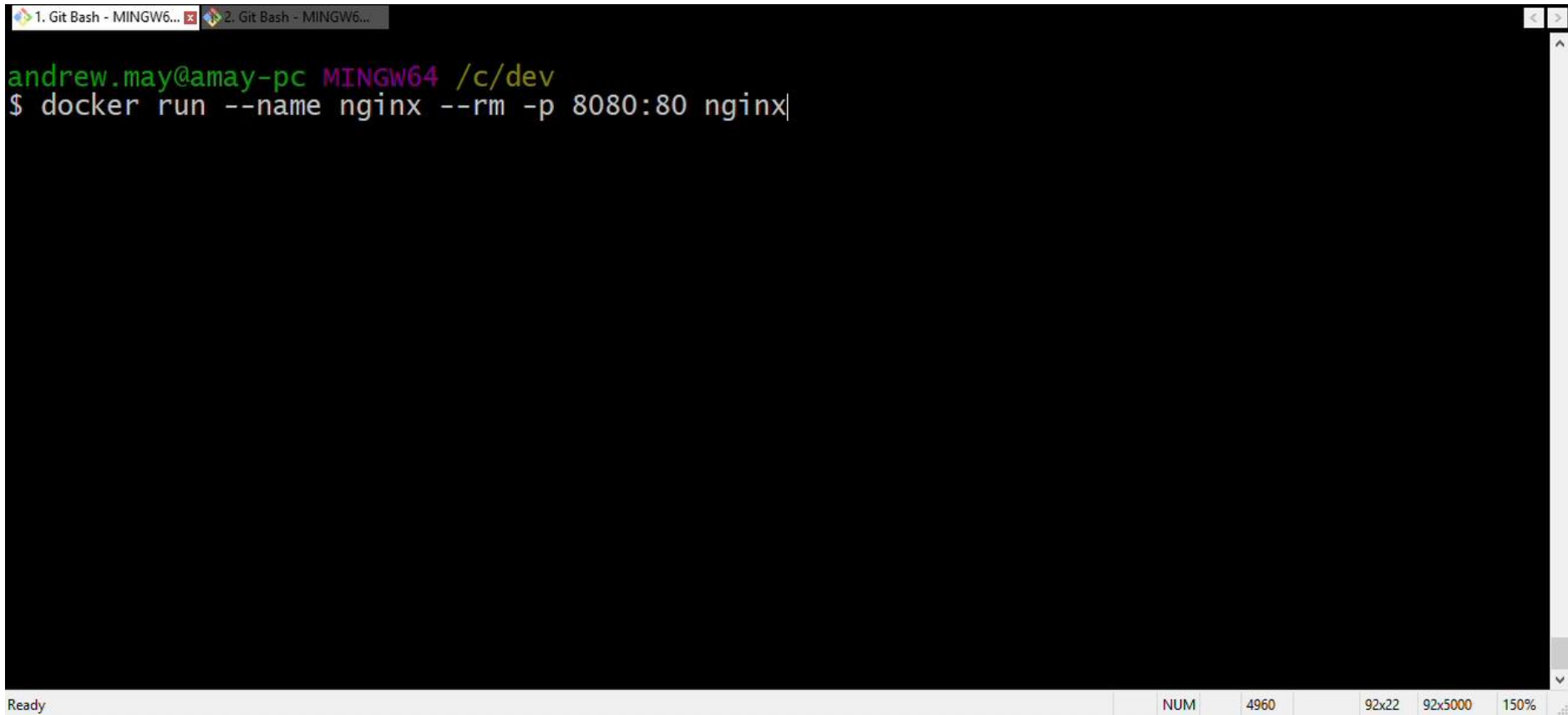
Fargate

Single

Multi

This is not an  
“Intro to Docker”

```
andrew.may@amay-pc MINGW64 /c/dev  
$ docker run --rm docker/whalesay cowsay "Hello AWS Meet-up"
```



The image shows a Windows terminal window with two tabs. The active tab is titled "2. Git Bash - MINGW64...". The terminal displays the prompt "andrew.may@amay-pc MINGW64 /c/dev" and the command "\$ docker run --name nginx --rm -p 8080:80 nginx|". The terminal window has a standard Windows taskbar at the bottom with the "Ready" status and system tray icons.

```
1. Git Bash - MINGW64... 2. Git Bash - MINGW64...
andrew.may@amay-pc MINGW64 /c/dev
$ docker run --name nginx --rm -p 8080:80 nginx|
```

Ready NUM 4960 92x22 92x5000 150%

# ECR

*“Amazon Elastic Container Registry (ECR) is a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images.”*

# Elastic Container Registry (ECR)

---

ECR is a Docker registry hosted within AWS in each region

- Images are stored close to where your containers will run

Images are secured using AWS IAM Policies, the service running the containers, or an AWS IAM User must be granted access

The Elastic Container **Registry** contains a **Repository** for each image, a **Repository** can store many versions of that image

Lifecycle Policies can be used to clean up old versions of images  
(you pay for the amount of storage you use in ECR, and images are often >100 MB)

Login using AWS CLI (`aws ecr get-login --no-include-email --profile ... --region us-east-1`), use standard Docker commands to pull and push

Image names are prefixed by ECR Repository URL (`<account>.dkr.ecr.<region>.amazonaws.com`)



# Repository storing multiple versions

ECR > Repositories > idp/app

## idp/app

View push commands

Images (32)

Find Images

< 1 >

<input type="checkbox"/>	Image tag	Image URI	Pushed at ▼	Digest	Size (MB) ▼
<input type="checkbox"/>	latest, 33	.dkr.ecr.us-east-2.amazonaws.com/idp/app:latest	01/29/19, 6:35:45 PM	sha256:2251cb0c0...	115.16
<input type="checkbox"/>	32	.dkr.ecr.us-east-2.amazonaws.com/idp/app:32	01/29/19, 5:29:39 PM	sha256:c216010ae...	115.16
<input type="checkbox"/>	28	.dkr.ecr.us-east-2.amazonaws.com/idp/app:28	01/28/19, 6:48:23 PM	sha256:2e26861d9...	115.16
<input type="checkbox"/>	27	.dkr.ecr.us-east-2.amazonaws.com/idp/app:27	01/28/19, 6:13:28 PM	sha256:275363304...	115.16
<input type="checkbox"/>	26	.dkr.ecr.us-east-2.amazonaws.com/idp/app:26	01/28/19, 5:51:13 PM	sha256:d9e95b2e6...	115.16

# Orchestration



*When running a production service, it's no longer sufficient to manually start Docker containers.*

*Orchestration is the management of the container lifecycles.*

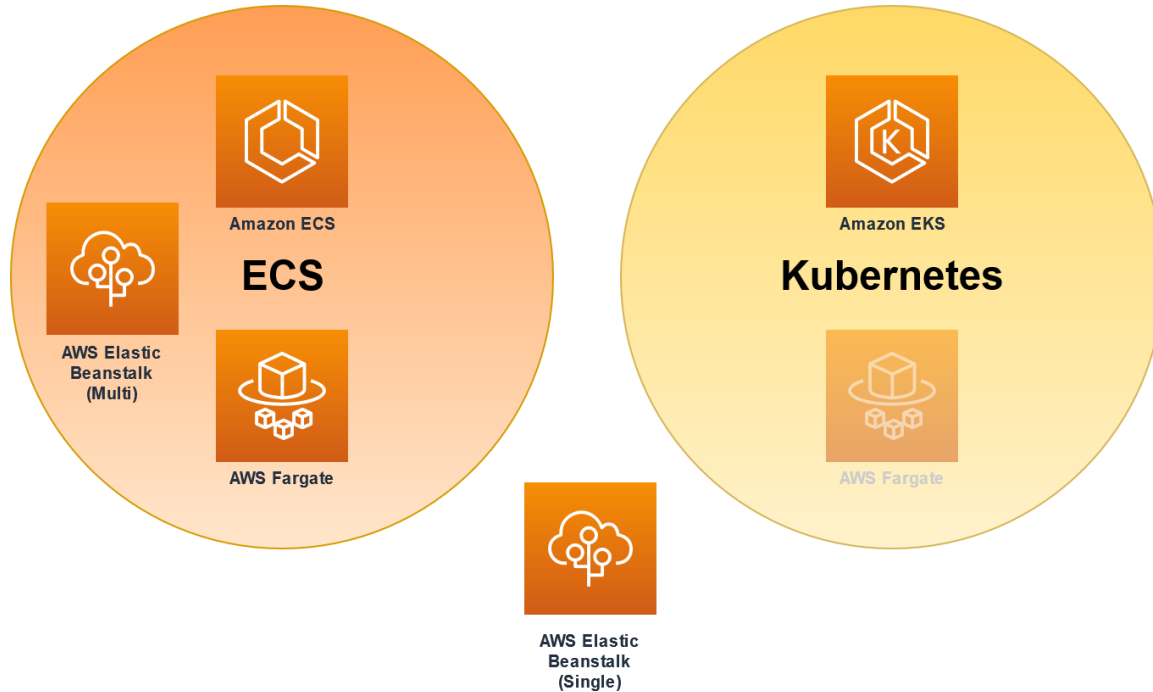
# Orchestration provides:

---

- Configuration
- Scheduling
- Deployment
- Scaling
- Storage (Volume) mapping
- Secret management
- High Availability
- Load balancing integration

# Managed Orchestration options (AWS)

---



# ECS

*“Amazon Elastic Container Service is a highly scalable, high-performance container orchestration service that supports Docker containers and allows you to easily run and scale containerized applications on AWS”*

# Elastic Container Service (ECS)

---

Original AWS Service for running containers

- Before it was made a public service, was the basis for Lambda functions launched a year earlier

ECS Service is free, you only pay for the resources (EC2/Fargate) that are used with ECS

Strong integration with other AWS Services, in particular:

- IAM permissions at container (task) level
- Load balancers (ELB/ALB/NLB)

AutoScaling of containers (similar to EC2 AutoScaling)

## Cluster : development

[Delete Cluster](#)

Get a detailed view of the resources on your cluster.

Status **ACTIVE**

Registered **container instances** 1

Pending **tasks** count 0 Fargate, 0 **EC2**

Running tasks count 0 Fargate, 3 EC2

Active service count 0 Fargate, 3 EC2

Draining service count 0 Fargate, 0 EC2

- ECS can use either EC2 instances or Fargate to run Docker containers
- Container Instances = ECS Instances = EC2 Instances
- Tasks have 1 or more running Containers
- Tasks are defined by Task Definitions

**Services** Tasks ECS Instances Metrics Scheduled Tasks Tags

Create Update Delete Actions ▾

Last updated on February 15, 2019 2:41:37 PM (0m ago) ↺ ⓘ

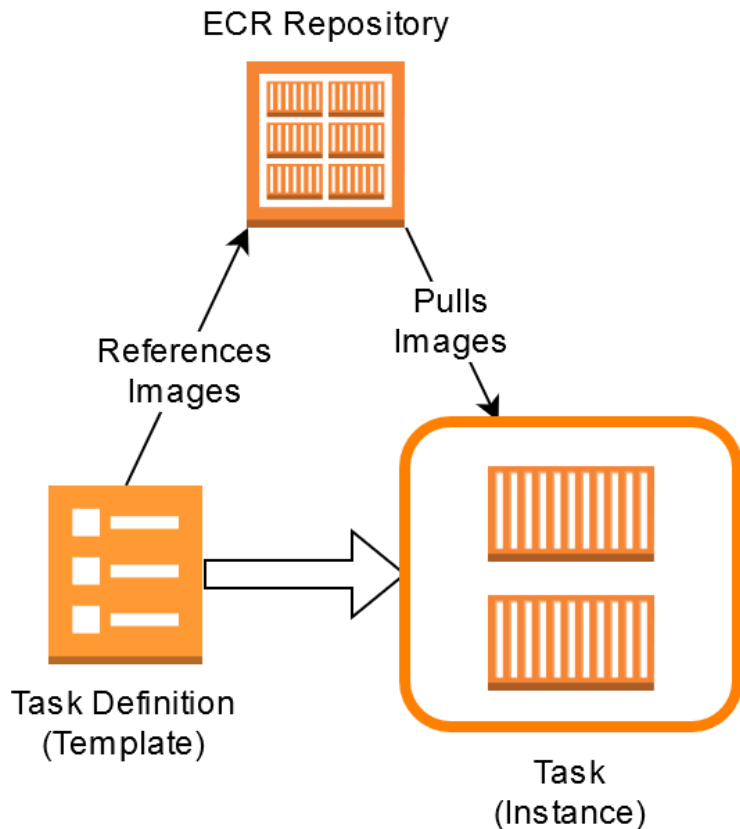
Filter in this page Launch type ALL Service type ALL < 1-3 >

<input type="checkbox"/>	Service Name	Status	Service type	Task Definition	Desired tasks ...	Running tasks ...	Launch type	Platform version
<input type="checkbox"/>	development-ConferenceService-Service-1RPB8N...	ACTIVE	REPLICA	development-Co...	1	1	EC2	--
<input type="checkbox"/>	development-IdpService-Service-117DQYQH11PS9	ACTIVE	REPLICA	development-Idp...	1	1	EC2	--
<input type="checkbox"/>	development-TpsService-Service-XYL3D3QQH5QH	ACTIVE	REPLICA	development-Tp...	1	1	EC2	--



# Task Definition and Tasks

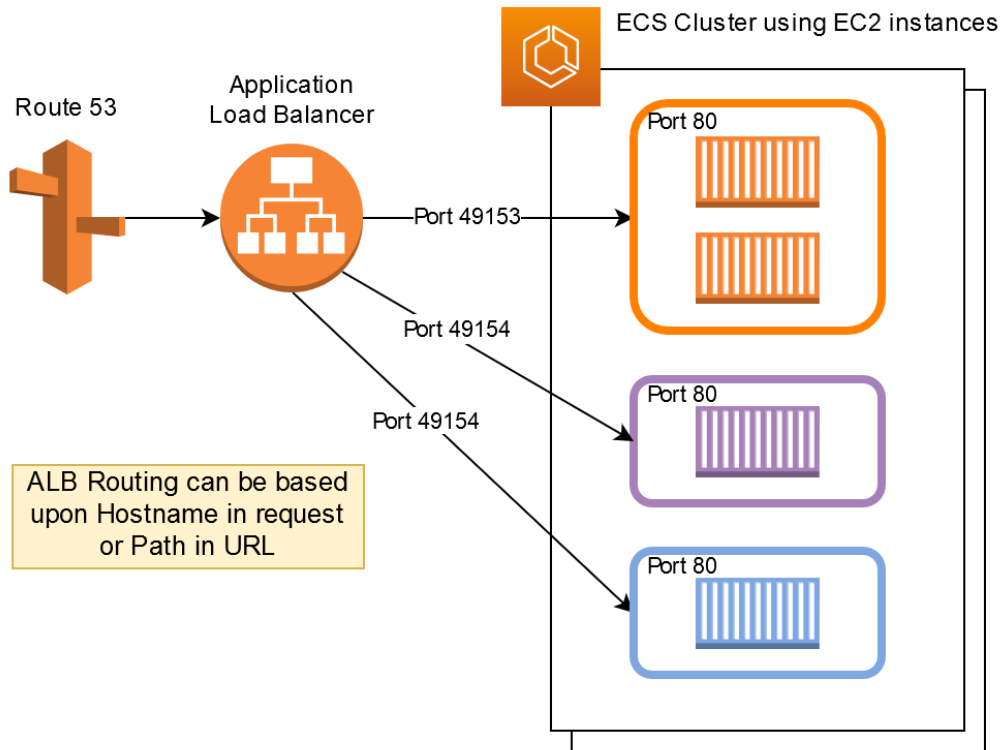
- ❖ The Task Definition is a Template for a Task, that will run one or more Container
- ❖ The Task Definition will reference the Container Image that will be pulled from ECR (or Dockerhub)
- ❖ The ECS Agent pulls this image to start the containers for the Task
- ❖ Also specifies Port mappings, CPU/Memory requirements, Volumes etc.
- ❖ Task Definitions are versioned





# ALB Integration with ECS Services

- ❖ ECS Services are long running collections of Tasks (e.g. Webservers)
- ❖ May run multiple copies
- ❖ Docker Containers have a container port (e.g. 80) and a host port
- ❖ Host ports can be predefined or if zero is specified in Task Definition it will use an ephemeral port
- ❖ ALB Target Group registered with Service will automatically be updated with correct port on EC2 instance when new container starts



# Other ECS Benefits

---

ECS Infrastructure can be created using CloudFormation

Updating a Service to use a new Task Definition will perform a Blue-Green deployment

- New Tasks are started and must be healthy (e.g. via ALB healthcheck) before old Tasks are stopped

Task Placement can distribute across AZs or binpack

- Can also have non homogeneous clusters with different instance types and control where tasks run

DAEMON Services can be used for running agents (e.g. XRay) on each EC2 instance in cluster, rather than including in each Task Definition as a “sidecar”

Metrics collected in CloudWatch, and logs can go to CloudWatch Logs or other destinations

# Fargate

*“AWS Fargate is a compute engine for Amazon ECS that allows you to run containers without having to manage servers”*

# Fargate

---

“Serverless” option for running containers in an ECS Cluster

- Configure desired CPU/Memory and this will be guaranteed for that container
- Compare to using EC2 where you may be over/under provisioned

Configuration uses Task Definitions, with only minor changes from running on EC2

Networking is always *awsvpc* – i.e. it will use an ENI per instance (one of your subnet IPs)

Because there is no shared server, can no longer run DAEMON tasks

- Can still run “sidecar” containers that are defined in the shared task definition

Scaling at a task level is similar to when using EC2 instances, but without the complexity of scaling the underlying EC2 AutoScaling group(s)

# Fargate Pricing

---

Fargate was expensive when launched, but the *Firecracker* VM technology has allowed AWS to reduce costs

- vCPU \$0.04048/hour      Memory \$0.004445/GB hour
- Still expensive compared to well utilized EC2, especially if using reserved instances

Tech10 (<https://www.trek10.com/blog/fargate-pricing-vs-ec2/>) produced a pricing comparison that compares EC2 and Fargate pricing for ECS

- With recent price reductions, Fargate is comparable in price to on-demand EC2 instances with ~70% utilization
- EC2 reserved instances are considerably cheaper, but will typically only be used for baseline load

EKS

*“Amazon Elastic Container Service for Kubernetes (Amazon EKS) makes it easy to deploy, manage, and scale containerized applications using Kubernetes on AWS.”*

# Kubernetes

---

Predominant Docker Orchestration service

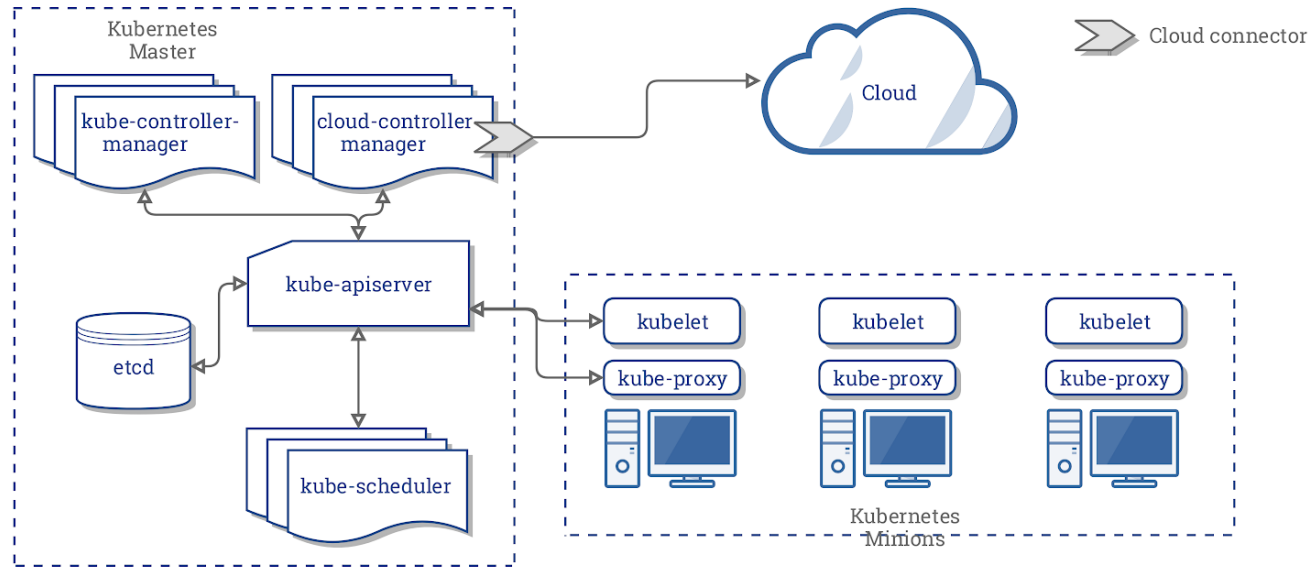
Managed Kubernetes available in:

- Google Cloud Platform (Google Kubernetes Engine)
- Microsoft Azure (Azure Kubernetes Service – AKS)
- Amazon Web Services (EKS)

The GCP offering is the most mature, AKS and EKS were both launched in 2018

Support for running Kubernetes clusters on EC2 and integrating with other services (e.g. ELB) has existed for longer than EKS has been available

# What does EKS manage?



EKS manages three Kubernetes master instances across AZs to provide high availability



# Using EKS

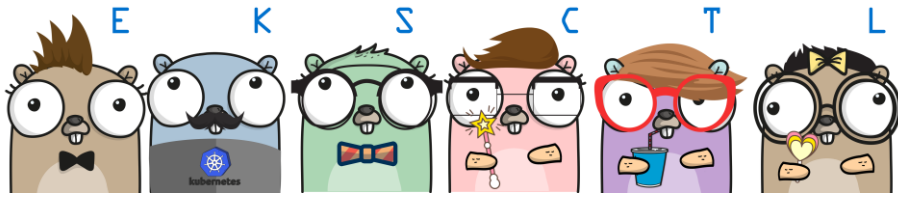
---

Cluster creation can be automated with CloudFormation, and AWS provides sample templates and a quick-start

- Set up VPC
- Create EKS Service IAM Role
- Create EKS Cluster
- Add Node instances to cluster using EKS AMI

Once Cluster created, configure kubectl with access to cluster and use normal Kubernetes tools and templates to manage and deploy to the cluster

EKS Cluster cost \$0.20/hour (about \$140/month)



```
$ eksctl create cluster -p personal -r us-east-2
[ ] using region us-east-2
[ ] setting availability zones to [us-east-2b us-east-2c us-east-2a]
[ ] subnets for us-east-2b - public:192.168.0.0/19 private:192.168.96.0/19
[ ] subnets for us-east-2c - public:192.168.32.0/19 private:192.168.128.0/19
[ ] subnets for us-east-2a - public:192.168.64.0/19 private:192.168.160.0/19
[ ] nodegroup "ng-c94ddd68" will use "ami-04ea7cb66af82ae4a" [AmazonLinux2/1.12]
[ ] creating EKS cluster "floral-painting-1556150853" in "us-east-2" region
[ ] will create 2 separate CloudFormation stacks for cluster itself and the initial nodegroup
[ ] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-2 --
name=floral-painting-1556150853'
[ ] 2 sequential tasks: { create cluster control plane "floral-painting-1556150853", create nodegroup "ng-c94ddd68" }
[ ] building cluster stack "eksctl-floral-painting-1556150853-cluster"
[ ] deploying stack "eksctl-floral-painting-1556150853-cluster"
[ ] building nodegroup stack "eksctl-floral-painting-1556150853-nodegroup-ng-c94ddd68"
[ ] --nodes-min=2 was set automatically for nodegroup ng-c94ddd68
[ ] --nodes-max=2 was set automatically for nodegroup ng-c94ddd68
[X] deploying stack "eksctl-floral-painting-1556150853-nodegroup-ng-c94ddd68"
[X] all EKS cluster resource for "floral-painting-1556150853" had been created
[X] saved kubeconfig as "C:\\Users\\andrew.may.CORP\\.kube/config"
[ ] adding role "arn:aws:iam::648758314004:role/eksctl-floral-painting-1556150853-NodeInstanceRole-4TJQAGSUVG7Q" to auth ConfigMap
[ ] nodegroup "ng-c94ddd68" has 0 node(s)
[ ] waiting for at least 2 node(s) to become ready in "ng-c94ddd68"
[ ] nodegroup "ng-c94ddd68" has 2 node(s)
[ ] node "ip-192-168-25-233.us-east-2.compute.internal" is ready
[ ] node "ip-192-168-79-171.us-east-2.compute.internal" is ready
[ ] kubectrl command should work with "C:\\Users\\andrew.may.CORP\\.kube/config", try 'kubectrl get nodes'
[X] EKS cluster "floral-painting-1556150853" in "us-east-2" region is ready
```

# Integration with other AWS Services

---

Kubernetes (not specifically EKS) supports Classic ELBs and NLBs, but not ALBs

- Can use Nginx Ingress (with ELB) or aws-alb-ingress-controller

Integrating with IAM to provide “pod” level permissions requires installing kube2iam

Registering created load balancers with Route 53 requires installing external-dns

Making logs accessible requires Fluentd or another service to be installed

Metrics collections requires additional services to be installed

You may need to install Helm to install some of these other services

All of these extra service that need to be run, consume memory/CPU on worker nodes

## floral-painting-1556150853

[Update cluster version](#)[Delete](#)

## General configuration

Kubernetes Version

1.12

Platform Version

eks.1

Status

ACTIVE

API server endpoint

<https://3AE628D1E612FF8692AABB7141F07779.yl4.us-east-2.eks.amazonaws.com>

Certificate authority

```
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUN5RENDQWJDZ0F3SUJBZ0lCQURBTklna
3Foa2lHOXcwQkFRc0ZBREFTVjNld0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYR
FRFNU1EUXIOVEF3TVRRME5sb1hEVEk1TURReU1qQXdNVFEwTmxvd0ZURVRNqkVHQTFVRQ
pBeE1LYTNWVpYSnVaWFJsY3pDQ0FTSXdeUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBREND
```

Cluster ARN

arn:aws:eks:us-east-2:648758314004:cluster/floral-painting-1556150853

Role ARN

arn:aws:iam::648758314004:role/eksctl-floral-painting-1556150853-clus-ServiceRole-LCSN1S6NIEOL

## Networking

[Update](#)

VPC

[vpc-03fe50a2345f6b273](#)

Subnets

[subnet-05212937dc0093f63](#) [subnet-0bd0e2140ea08bdc5](#) [subnet-058a5e5eee10ef611](#) [subnet-078b8c1e008d55429](#) [subnet-0a6e6753ccf879007](#) [subnet-0bb9df57758fc9766](#)

Security groups

[sg-0ced8c9ba854080c5](#)

API server endpoint access

Private access

Disabled

Public access

Enabled

# Cloud Portability

---

One of the promises of Kubernetes is the ability to run applications in the same way across different Cloud platforms

There significant variations between what is supported across AWS, GCP and Azure, and what services are preinstalled as part of managed clusters

Running the same service across multiple Cloud platforms requires using platform specific annotations

However, most of those differences are hidden from the running applications that can operate the same way across platforms

# Elastic Beanstalk

*“AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.”*

# Elastic Beanstalk

---

Elastic Beanstalk is a Platform-as-a-Service (PaaS) for a variety of languages and services

Aims to make it easy to migrate applications to AWS

Manages underlying infrastructure, high availability, deployments, logging etc.

- Can also create databases and other related services

# Elastic Beanstalk Docker support

---

## Single Container (version 1)

Launched in 2014 (before ECS)

Runs single container per EC2 instance

Uses Nginx as proxy to container

Upload source code including Dockerfile and it will build and deploy container

OR

Upload configuration file referencing image stored in Docker registry

## Multiple Container (version 2)

Launched in 2015

Runs containers on ECS

Can only reference prebuilt images stored in a Docker registry

Can upload files that are mounted as container volumes as part of the deployment bundle



# AWS CloudMap

*“AWS Cloud Map is a cloud resource discovery service.*

*With Cloud Map, you can define custom names for your application resources, and it maintains the updated location of these dynamically changing resources.”*

# CloudMap

---

## AWS CloudMap supercedes Service Discovery for ECS

- Launched in May 2018, Service Discovery for ECS registered running tasks in a Route 53 namespace (hosted zone), creating A (IP) and SRV (IP and Port) records
- Query DNS for available services
- ECS Tasks must use *awsvpc* networking, either using Fargate or the limited number of ENIs available per EC2 instance

AWS CloudMap builds the previous service to add an API based system that still supports DNS, but is no longer reliant upon it, allowing it to be used for resources where DNS does not apply

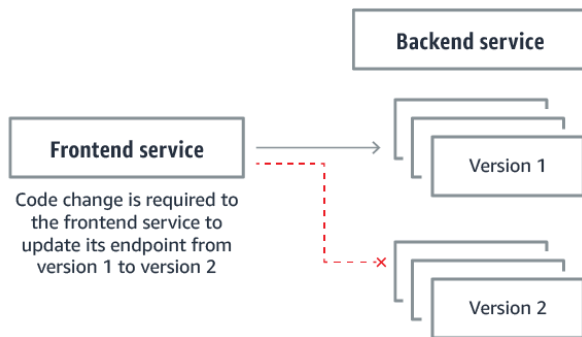
- Supports a variety of services including ECS, EKS, S3, SQS, Lambda, Load Balancers
- ECS Tasks with Service discovery enabled automatically enrolled in CloudMap
- EKS support via External DNS

Similar to the Service Discovery portions of Hashicorp Consul

# CloudMap

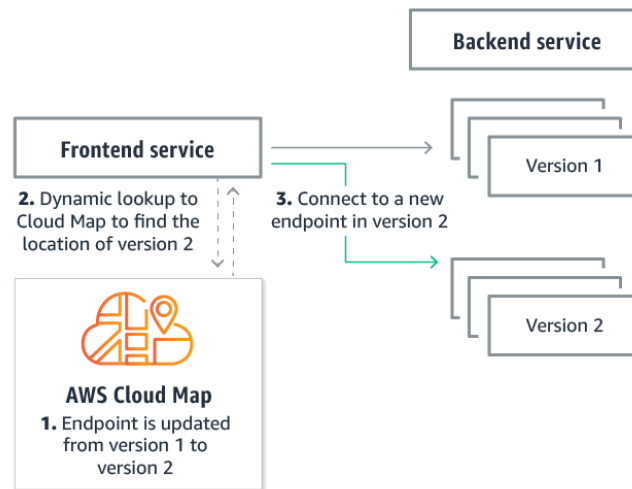
## Without Cloud Map

Endpoints are statically coded into your application

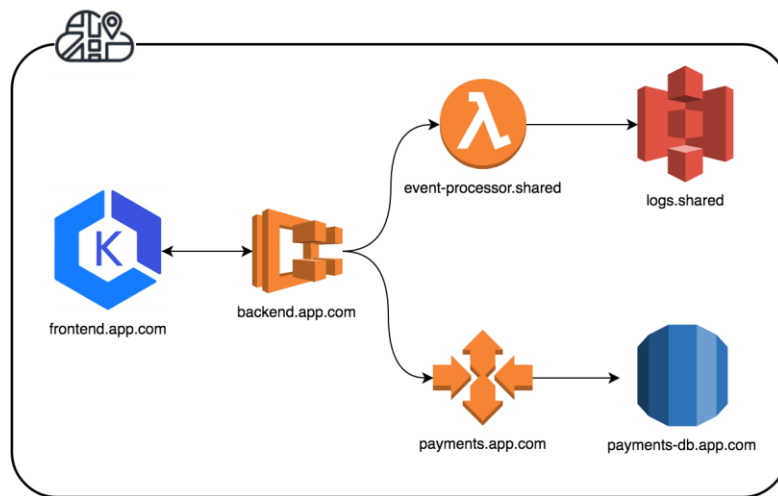
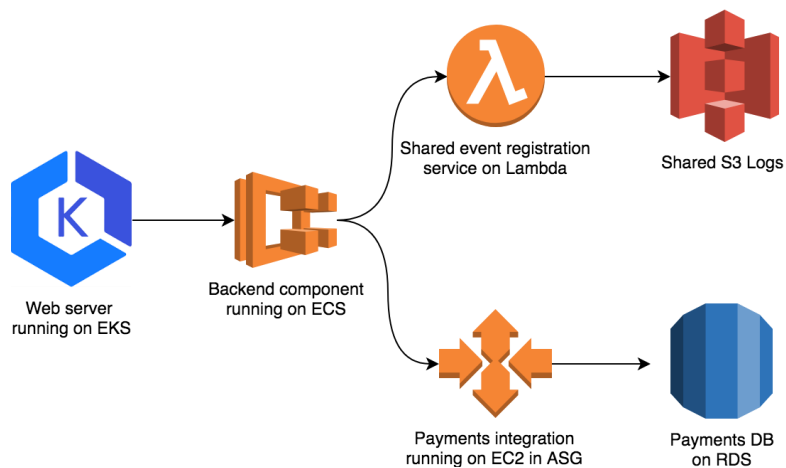


## With Cloud Map

Endpoints are dynamically located



# CloudMap



# AWS AppMesh

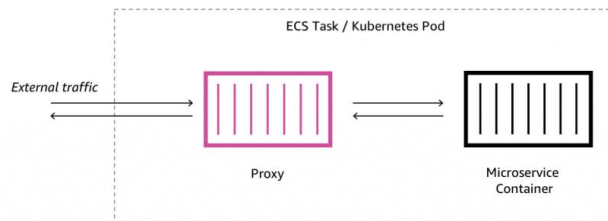
*“AWS App Mesh makes it easy to monitor and control microservices running on AWS.*

*App Mesh standardizes how your microservices communicate, giving you end-to-end visibility and helping to ensure high-availability for your applications.”*

# AppMesh

Recently released, but still in early stages

Service Mesh implementation for AWS, using the Open Source Envoy proxy, but a custom Control Plane (i.e. not Istio)



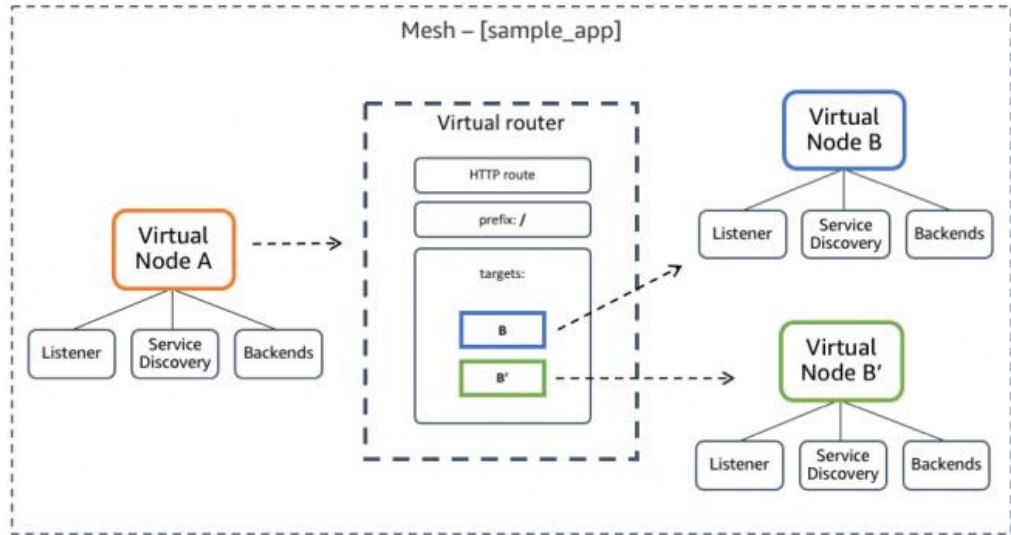
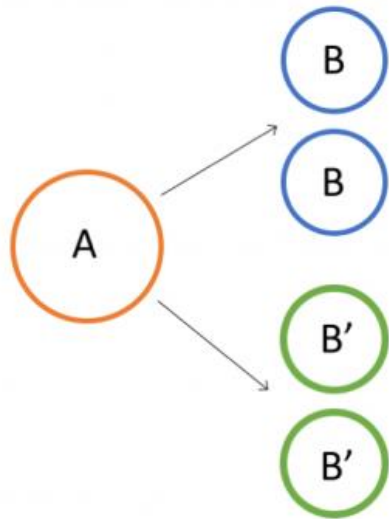
Manage connectivity between microservices, including traffic shaping functionality like:

- Routing: Canary and A/B Testing
- Load Balancing and Service Discovery
- Handling Failures (Retry, Circuit Breaker)

Integration with CloudMap for Service Discovery

Logging and Tracing (CloudWatch, X-Ray)

# AppMesh



# AppMesh Roadmap

In coming soon:

- Circuit Breaker
- Retry

Researching:

- ECS without using awsvpc
- mTLS
- Lambda integration

<div><div>13 Researching</div><div><div>1 AWS Cloud Map selectors</div><div>#47 opened by coultin</div><div>Proposed</div></div><div><div>1 Cookie based routing</div><div>#14 opened by jamsajones</div></div><div><div>1 HTTP Header based routing</div><div>#15 opened by jamsajones</div></div><div><div>1 GRPC routing</div><div>#13 opened by jamsajones</div></div><div><div>1 Use App Mesh for ingress routing</div><div>#37 opened by jamsajones</div></div><div><div>1 Simplify external service egress traffic setup</div><div>#2 opened by bcelenza</div></div><div><div>1 End to end encryption of traffic with customer provided certs</div><div>#38 opened by jamsajones</div></div><div><div>1 End to end encryption of traffic with ACM managed certs</div><div>#39 opened by jamsajones</div><div>Proposed</div></div><div><div>1 Enable custom filters for Envoy</div><div>#41 opened by jamsajones</div></div><div><div>1 Implement Fault Injection</div><div>#36 opened by jamsajones</div></div><div><div>1 Enable ECS with other networking modes</div><div>#40 opened by jamsajones</div><div>ECS Proposed</div></div><div><div>1 authN based on mTLS</div><div>#34 opened by jamsajones</div></div><div><div>1 Integration with AWS Lambda</div><div>#33 opened by alahen</div></div></div>	<div><div>3 We're Working On It</div><div><div>1 Region expansion</div><div>14 of 21</div><div>#1 opened by jamsajones</div></div><div><div>1 Provide the Envoy software in each region where App Mesh is available</div><div>#56 opened by jtoberon</div></div><div><div>1 Open Source the App Mesh Envoy Image build, release, and validation tools</div><div>#5 opened by dastbe</div></div></div>	<div><div>7 Coming Soon</div><div><div>1 Commit X-Ray tracer plugin to Envoy upstream</div><div>#21 opened by jamsajones</div></div><div><div>1 VPC Endpoint/Private Link for App Mesh Envoy xDS API</div><div>#12 opened by ewbankit</div></div><div><div>1 Circuit Breaker Policy</div><div>#6 opened by jamsajones</div></div><div><div>1 ECS workflow integration</div><div>#8 opened by jamsajones</div></div><div><div>1 Retry Policy</div><div>#7 opened by jamsajones</div></div><div><div>1 Bring Envoy from official release</div><div>#10 opened by jamsajones</div></div><div><div>1 Hosted EDS implementation with AWS Cloud Map</div><div>#11 opened by jamsajones</div></div></div>	<div><div>17 Just Shipped</div><div><div>1 TCP routing</div><div>#4 opened by jamsajones</div></div><div><div>1 Resource-based authorization in IAM</div><div>#20 opened by jamsajones</div></div><div><div>1 App Mesh Console</div><div>#22 opened by jamsajones</div></div><div><div>1 Integration with EKS</div><div>#9 opened by jamsajones</div></div><div><div>1 CloudFormation</div><div>#23 opened by jamsajones</div></div><div><div>1 Setup iptables via CNI plugin</div><div>#3 opened by jamsajones</div></div><div><div>1 Tag Based Resources</div><div>#19 opened by jamsajones</div></div><div><div>1 Emit DogStatsD-compatible metrics</div><div>#16 opened by jamsajones</div></div><div><div>1 Access logging</div><div>#18 opened by jamsajones</div></div><div><div>1 Support AWS X-Ray Tracing</div><div>#17 opened by jamsajones</div></div><div><div>1 Commit SigV4 auth addition to Envoy upstream</div><div>#27 opened by jamsajones</div></div><div><div>1 Service call auditing</div><div>#26 opened by jamsajones</div></div><div><div>1 Clarify usage of ServiceNames</div><div>#24 opened by bcelenza</div></div><div><div>1 Implement Health checks</div><div>#28 opened by jamsajones</div></div><div><div>1 Updates to routes have no effect on running envoyos</div><div>#30 opened by kiranmeduri</div></div><div><div>1 Integration with Consul Service Discovery</div><div>#25 opened by jamsajones</div></div><div><div>1 [BUG] Version numbers in Envoy resource names cause connection failures, changing metric names</div><div>#29 opened by bcelenza</div></div></div>
--	---	--	---



# AppMesh Examples

---

<https://github.com/aws/aws-app-mesh-examples/>

Example project contains a sample application that runs in AppMesh within both ECS and EKS

Documents all the steps needed to get started

Includes sidecar injection for EKS

# AWS Batch

*“AWS Batch enables developers, scientists, and engineers to easily and efficiently run hundreds of thousands of batch computing jobs on AWS.”*

# AWS Batch

---

Batch service that automatically provisions computing resources to run Jobs

Job = Unit of work: shell script, Linux executable or Docker container image

Job Definition = How jobs are run: IAM Role, CPU/Memory requirements

- Similar to ECS Task Definition

Job Queue = Queue of jobs to run, can have multiple queues  
(e.g. different priorities)

Jobs are run using Docker containers utilizing ECS technology

- EC2 instances for running containers are automatically managed
- Can use EC2 spot instances to reduce costs

# AWS CodeBuild

*“AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy.”*

# AWS CodeBuild

---

## Docker based build/test environment

- All steps run in docker containers, either provided codebuild containers for common build tools, or custom images from ECR/Dockerhub
- Support for Windows (e.g. .NET Framework) builds using Windows containers on Windows Server

## Commonly integrated with CodePipeline

Scales automatically rather than provisioning agents

## Pay for build minutes

- First 100 build minutes (smallest instance type) per month free

Questions?