

Prep: Clean up (Do not show)

- Segment07:
 - Detach Cluster Auto scaling / Dynamo policy from Node instance role
 - Kubectl delete -f eks-dynamo.yaml
 - Delete ECR Repo
 - terraform destroy
 - sls remove
- Segment06
 - Remove group from anotherUser
 - Delete dashboard, hpa-demo, meks-admin-service-account, metrics-server, role-binding
- Segment05
 - Kubectl delete -f ngx-volumes.yaml
 - Kubectl delete -f default-backend
 - Kubectl delete -f cert-manager/
 - Kubectl delete -f nginx-ingress-controller
- Segment04
 - Kubectl delete -f dnstest/

Prep: Clean up 2 (Do not show)

- Segment03:
 - Eksctl delete-cluster apr08
 - Remove manual eks cluster nodes
 - Remove manual eks cluster
 - Remove stacks in cloud formation
- Segment02
 - Terraform destroy –auto-approve

Prep: Setup (Do not show)

1. Open Firefox to namecheap and login (for DNS)
2. Open two Chrome browsers: 1 in incognito mode
3. Open Tab to github.com resources
4. Two terminals open in the source directory
5. Setup 2nd user in remote machine.
6. Get some water



Welcome!

Kubernetes on AWS

Poll Question: Level of Expertise with Kubernetes?

1. So good that I should be teaching this class instead of you.
I've authored books on the subjects and know all the ins and outs.
2. Pretty good, we run it in production and we've been seen our share of issues.
3. I'm not quite a beginner, but don't have operational experience with it.
4. I'm just getting started
5. What is Kubernetes and where am I?

Who is this guy?

- Kubernetes since 2015
- Engineer @ Weave Grid
- Portland, OR
- Wife, 4 kids
- Kubernetes Contracting Work
- Type 1 Diabetes Research
- Past:
 - IBM, Cisco, other Small companies



What we cover today

- Segment 1: Introduction
- Segment 2: IAM Configuration
- Segment 3: EKS Installation
- Segment 4: Verification
- Segment 5: Running Applications
- Segment 6: Administrative Tasks
- Segment 7: Additional AWS connections



Kubernetes on AWS

Segment 1: Introduction

What is Kubernetes?



Zack Kanter 
@zackkanter

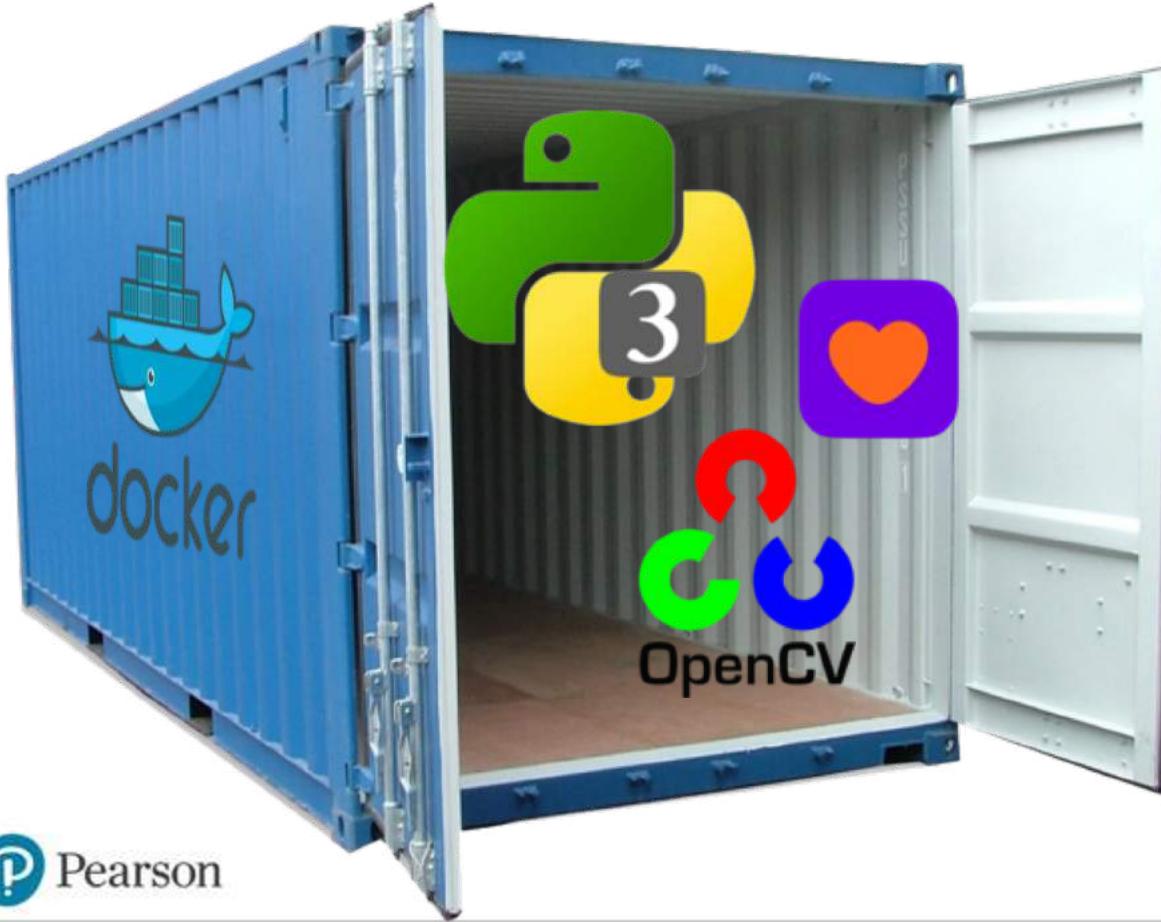


The big antitrust story that no one talks about is that in 2014, Google quietly released a Stuxnet-like virus designed to cripple thousands and thousands of potential competitors from the inside out. It's called Kubernetes.

7:27 PM · Oct 8, 2019 · [Twitter for iPhone](#)

112 Retweets 593 Likes

Why Containers?

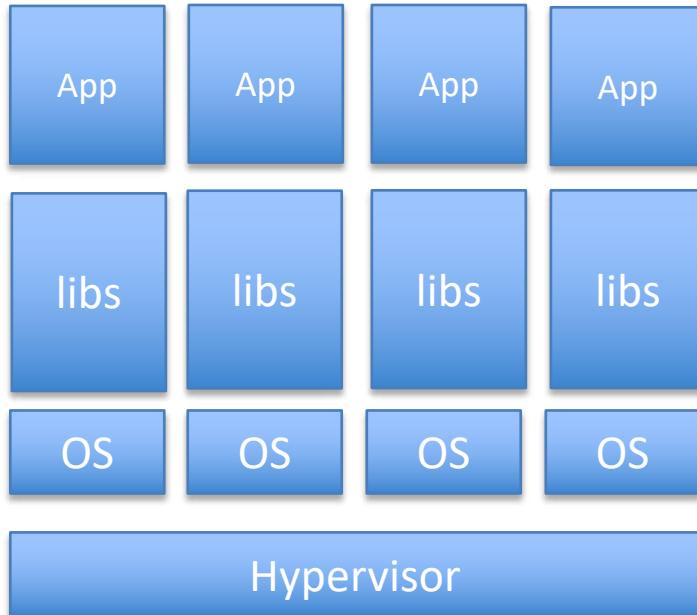


Application ships with dependencies and runtimes

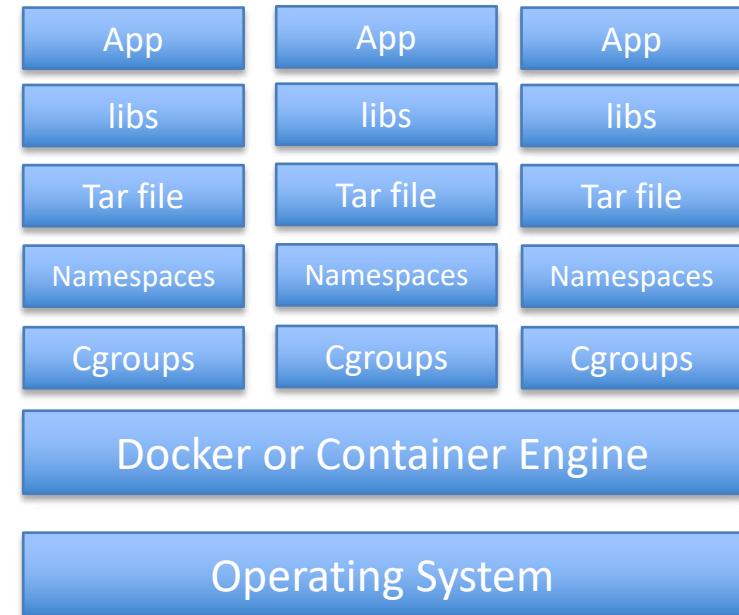
Runs the same in any environment where containers run

Standard way to start applications.

Virtual Machine vs. Container

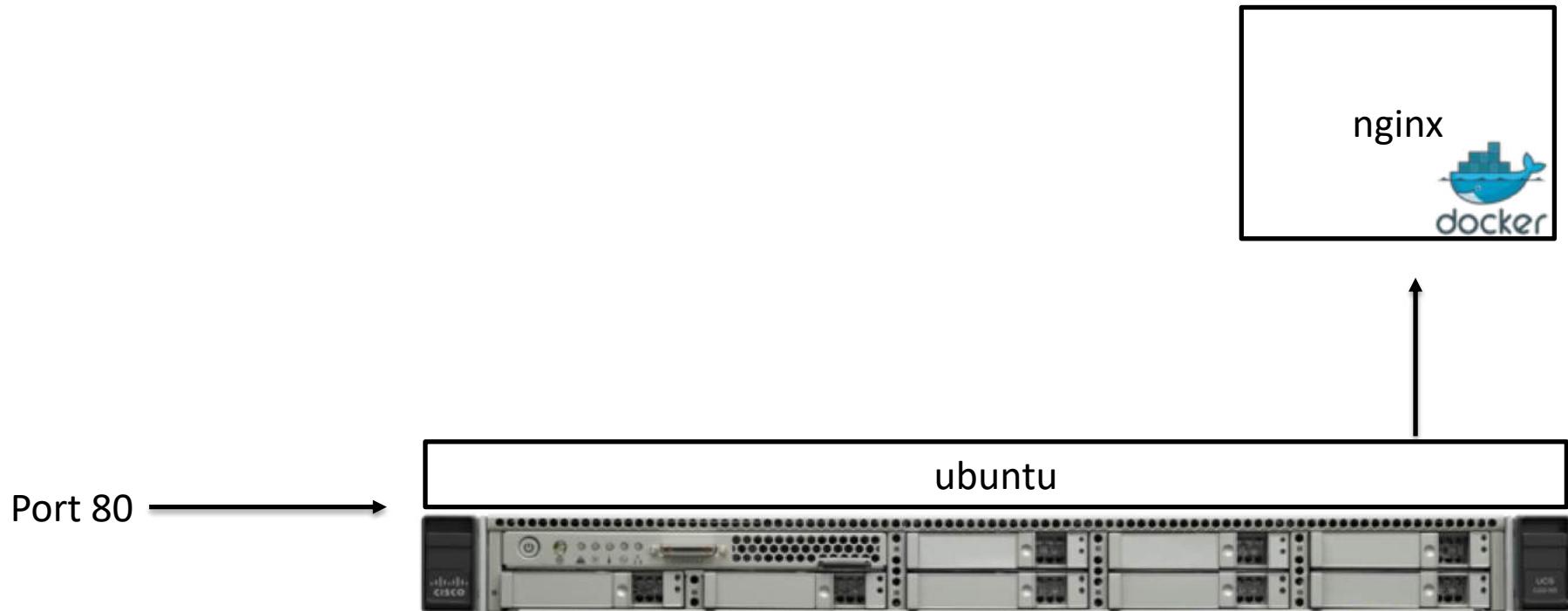


Virtual Machine Stack



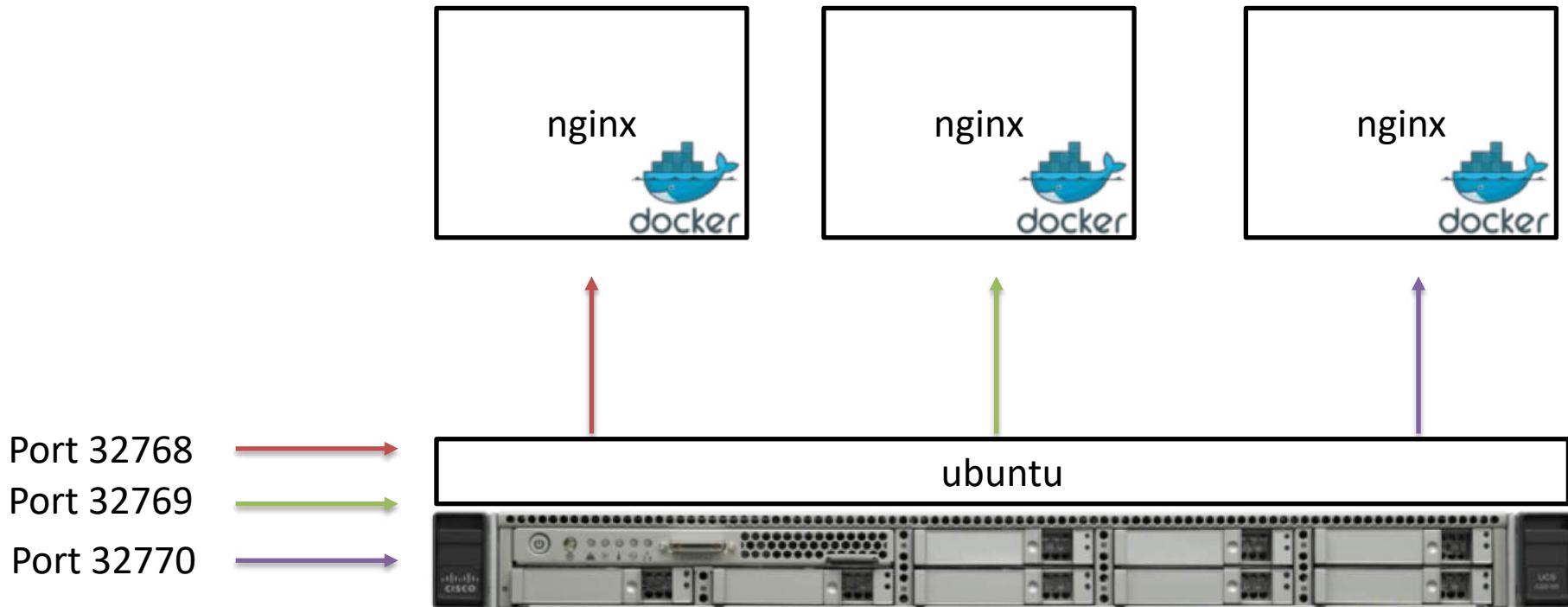
Container Stack

This is easy



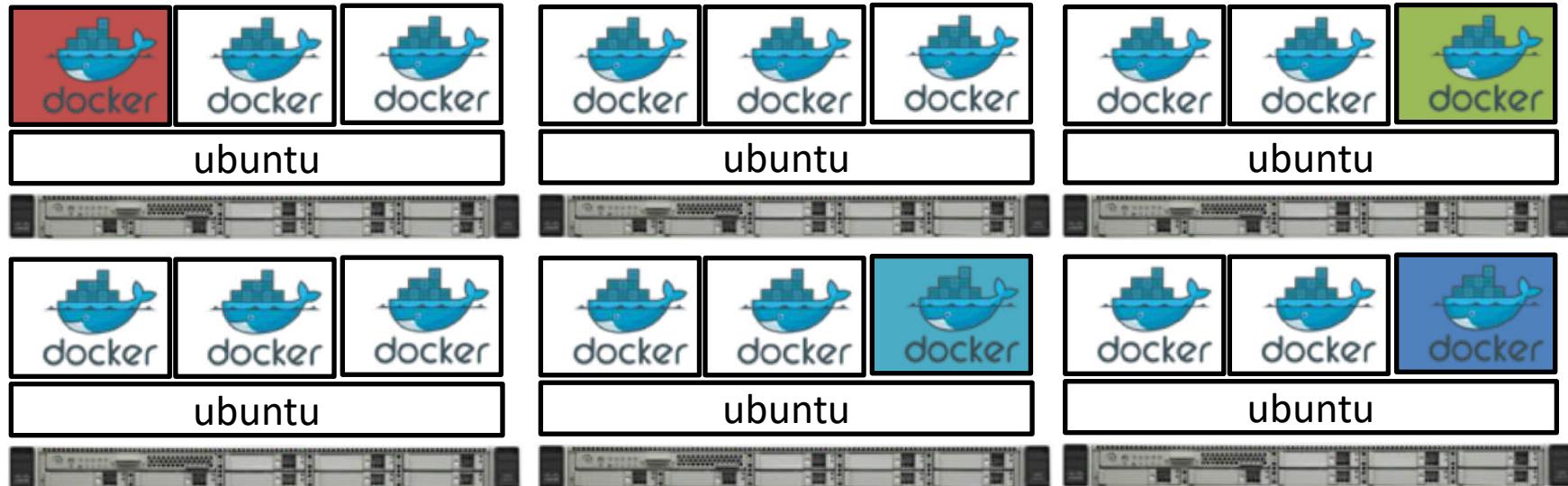
```
docker run -d -p 80:80 nginx
```

This is a little more difficult



```
for i in $(seq 3); do docker run -d -P nginx; done
```

This is starting to become a problem



app-6-3 → app-1-1

```
for i in $(seq 6); do for j in $(seq 3); do ssh node0$i docker  
run -d -P app${i}-${j}; done; done
```

Container Management challenges

How do we keep track of which port goes to which container on which host?

How should we efficiently allocate containers to hosts?

Microservices scale horizontally, so how do we map service dependencies?

Applications are frequently updated, container ports are randomized, how do we account for frequent changes?

Kubernetes

Thursday, April 23, 2015

Borg: The Predecessor to Kubernetes

Google has been running containerized workloads in production for more than a decade. Whether it's service jobs like web front-ends and stateful servers, infrastructure systems like [Bigtable](#) and [Spanner](#), or batch frameworks like [MapReduce](#) and [Millwheel](#), virtually everything at Google runs as a container. Today, we took the wraps off of Borg, Google's long-rumored internal container-oriented cluster-management system, publishing details at the academic computer systems conference [Eurosys](#). You can find the paper [here](#).

Large-scale cluster management at Google with Borg

Abhishek Verma[†] Luis Pedrosa[†] Madhukar Korupolu
David Oppenheimer Eric Tune John Wilkes
Google Inc.

Abstract

Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines.

It achieves high utilization by combining admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. It supports high-availability applications with runtime features that minimize fault-recovery time, and scheduling policies that reduce the probability of correlated failures. Borg simplifies life for its users by offering a declarative job specification language, name service integration, real-time job monitoring, and tools to analyze and simulate system behavior.

We present a summary of the Borg system architecture and features, important design decisions, a quantitative analysis of some of its policy decisions, and a qualitative examination of lessons learned from a decade of operational experience with it.

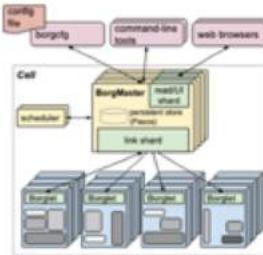


Figure 1: The high-level architecture of Borg. Only a tiny fraction of the thousands of worker nodes are shown.

cluding with a set of qualitative observations we have made from operating Borg in production for more than a decade.

Oh, by the way,
we've been doing
this for 10 years.

Google

<https://ai.google/research/pubs/pub43438>

Kubernetes in the Cloud

Google GKE

The screenshot shows the Google Cloud Compute Products page. The 'Products' tab is selected. A section titled 'KUBERNETES ENGINE' is displayed, with a sub-section 'Reliable, efficient, and secured way to run Kubernetes clusters'. Below this, there's a 'View' button and a Microsoft Azure interface window showing the 'Azure Kubernetes Service (AKS)' blade.

Microsoft AKS

Azure Kubernetes Service (AKS)

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline.

AWS EKS

The screenshot shows the Amazon Elastic Container Service for Kubernetes (Amazon EKS) page. The top navigation bar includes 'Overview', 'Features', 'Pricing', 'Getting Started', 'FAQs', 'Customers', and 'Partners'. The main heading is 'Amazon Elastic Container Service for Kubernetes'. Below it, a sub-headline reads 'Highly available, scalable, and secure Kubernetes service'. A prominent yellow button says 'Start using Amazon EKS'.

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

Amazon EKS runs the Kubernetes management infrastructure for you across multiple AWS availability zones to eliminate a single point of failure. Amazon EKS is certified Kubernetes conformant so you can use existing tooling and plugins from partners and the Kubernetes community. Applications running on any standard Kubernetes environment are fully compatible and can be easily migrated to Amazon EKS.

Amazon EKS is generally available for all AWS customers.

Poll Question: Where do you run Containers today?

1. EKS!
2. AKS
3. GKS
4. Bare Metal
5. VMware
6. Instances I manage
7. Just on my own laptop
8. Some other solution: Nomad, Rancher, OpenShift...

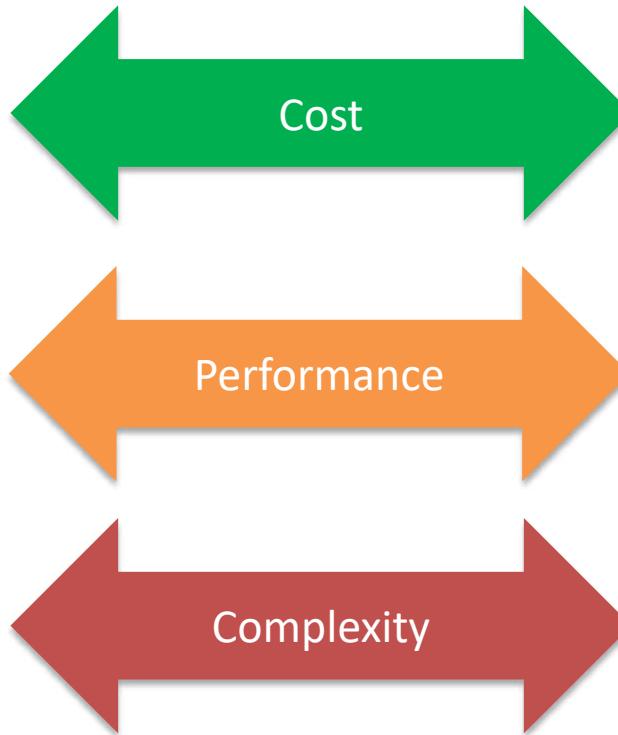
Cloud vs. OnPrem

Cloud

Cloud more expensive if operational costs are not considered.

Can scale up to hundreds of machines so if more performance is needed scale horizontally

Fully managed so many issues such as storage, load balancing, network is all taken care of.



OnPrem

Running in existing datacenter could be less but need more expertise for operational cost.

Can use more powerful hardware of any choice including special TPUs/GPUs

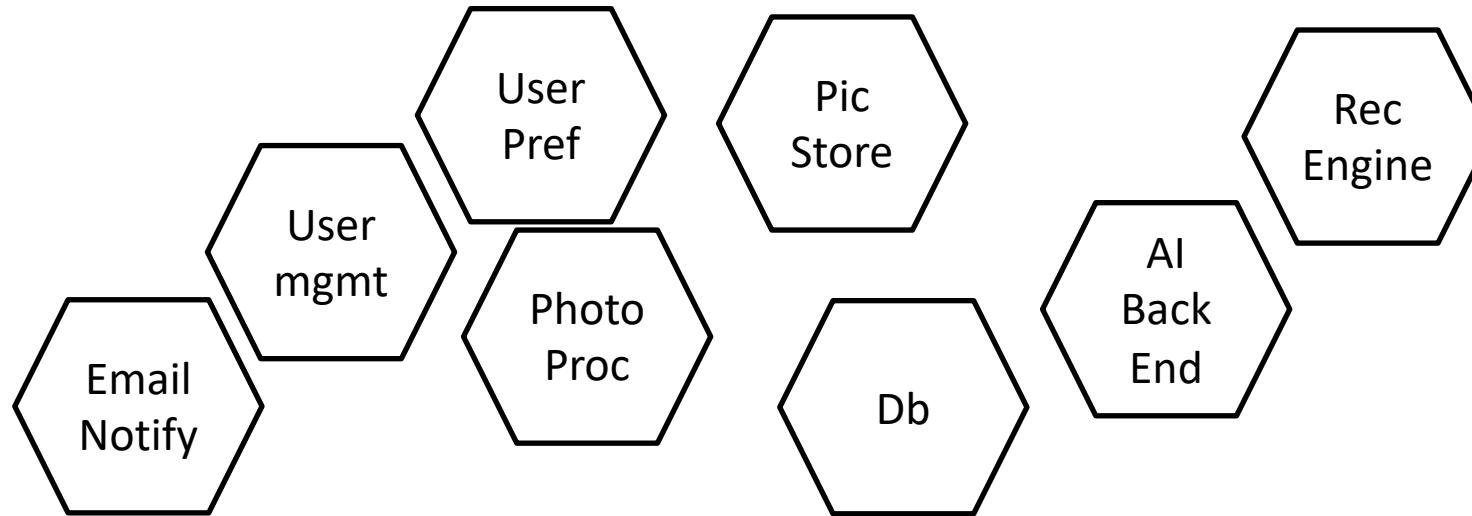
Requires deep expertise to run system which may be undifferentiated expertise for business.



The Cloud

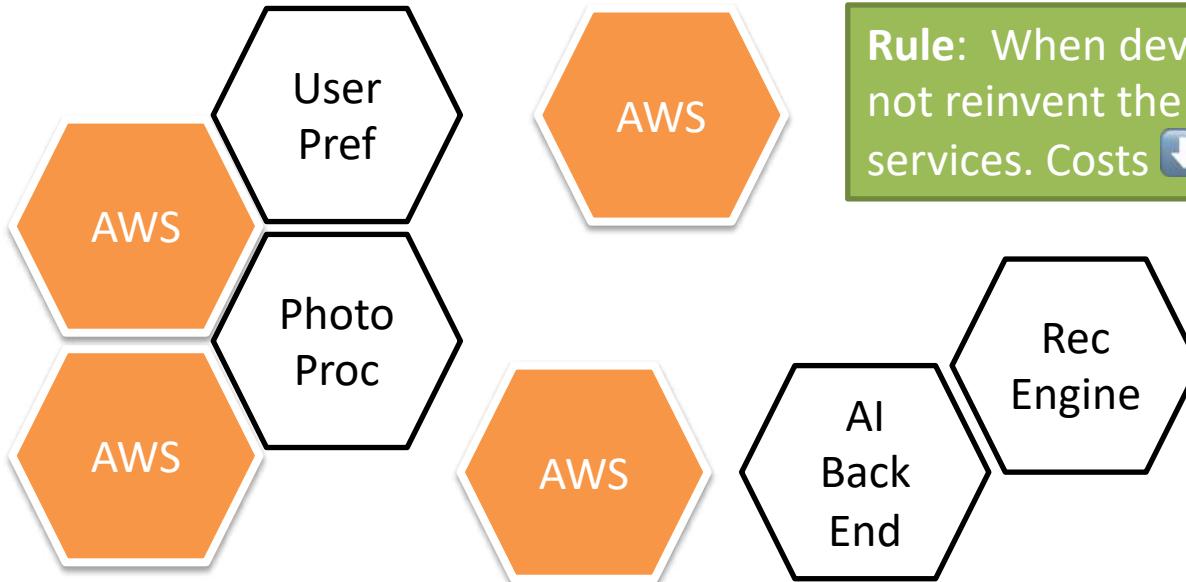
The cloud is not just self service of virtual machines

Microservices architecture has changed application development



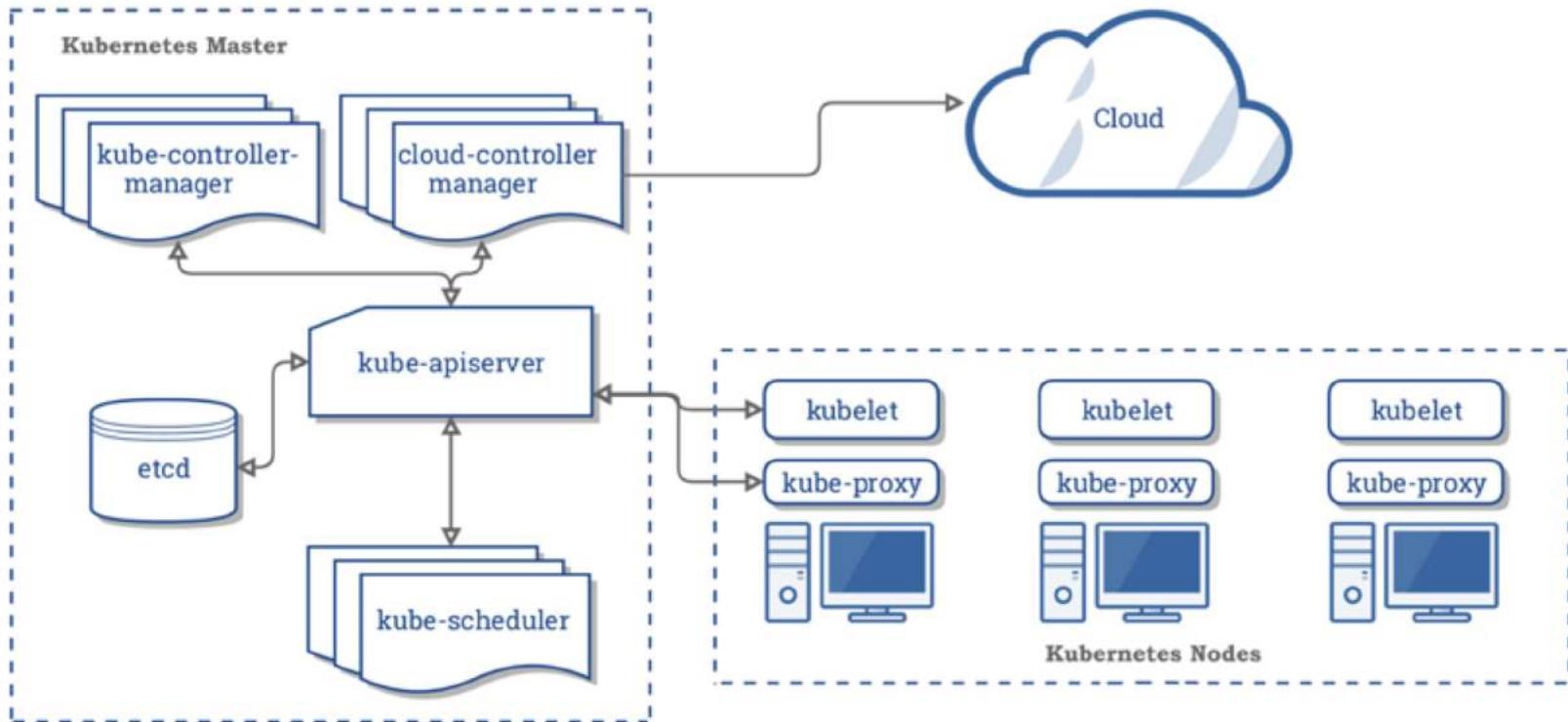
Microservices

Microservices are black boxes to each other.
To increase speed, use existing black boxes



Rule: When developing applications do not reinvent the wheel. Use existing cloud services. Costs , Feature Velocity .

Kubernetes Architecture



Source: <https://kubernetes.io/docs/concepts/overview/components/>

What is EKS?



Contact Sales Support ▾ English ▾ My Account ▾ Sign In to the Console

re:Invent Products Solutions Pricing Documentation Learn Partner Network AWS Marketplace Customer Enablement Events Explore More Q

Amazon EKS

[Overview](#)

Features

Pricing

Getting Started

FAQs

Customers

Partners

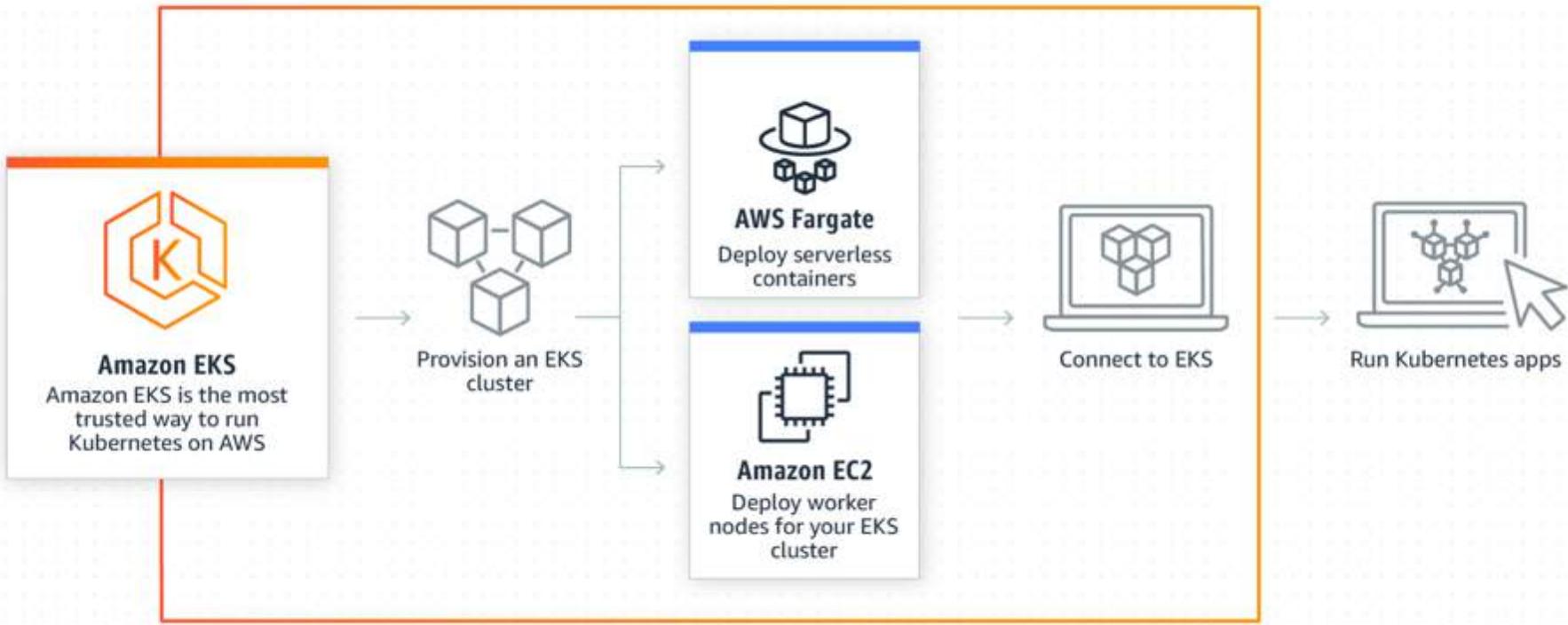
Container Day

Amazon Elastic Kubernetes Service

The most trusted way to run Kubernetes

[Get Started with Amazon EKS](#)

What is EKS?



Source: <https://aws.amazon.com/eks/>

EKS vs. RYO AWS Instances

EKS

\$0.10/hour + 3 x t3.large
\$0.24 hour

Only supported versions
can be run: 1.12, 1.13, 1.14
as of Dec 29, 2019

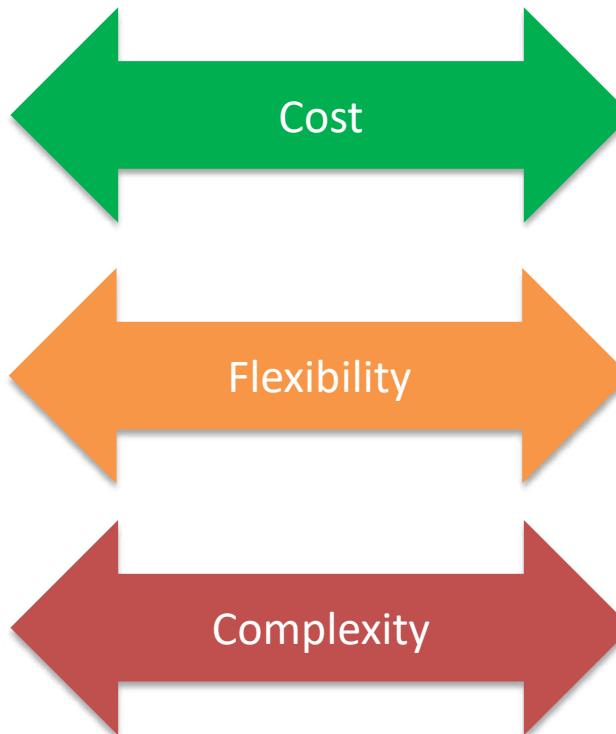
Control plane managed,
redundant, upgrades built
in.

RYO

$3 \times \text{t3.large} = \$0.08 \times 3 = \$0.24/\text{hour}$, or one master:
\$0.08 for testing.

Can run any version,
software, plugins, etc.

Requires build your own
control plane, manage
upgrades, more work.



Other AWS Container Services

ECS

- Proprietary Docker Management service developed first to compete with Kubernetes
- Uses JSON task definition

Fargate

- Container service, no EC2 instances required. Takes place of EC2 layer
- No scaling management
- JSON task definition
- Can be used with EKS and ECS

Benefits of EKS on AWS

- No control plane management
- Built in Load Balancing, Networking, Volume Storage
- Easy to turn off and on
- Integrations with other AWS components to build applications (S3, Cognito, RedShift, RDS, Lambda, etc)

New to EKS (Dec 2020)

- Amazon EKS Distro – an open source Kubernetes distribution used by Amazon EKS
 - <https://github.com/aws/eks-distro>
- EKS Anywhere (Multicloud)
 - <https://aws.amazon.com/eks/eks-anywhere/>
- Amazon Proton for CI/CD



Initial laptop setup

- kubectl
 - <https://github.com/vallard/EKS-Training/blob/master/segment01-intro/kubectl.md>
- AWS command line tool
 - <https://github.com/vallard/EKS-Training/blob/master/segment02-iam/aws-creds.md>
- [eksctl tool](#)
- [Tab completion for aws commands](#)



Kubernetes on AWS

Segment 2: IAM User Accounts

Big Picture

- Create IAM Policies
- Create IAM Group
- Create IAM Roles
- Create IAM User
- Get AWS command line tools
- Login via console and command line

Root Accounts

- Don't use root accounts
- Use root account to create other users: eksdude

Always getting error: You must be logged in to the server (Unauthorized) EKS

Asked 1 month ago Viewed 112 times

Over 400 Billion Feature Flags Served Daily

Deploy code now. Release when ready.

START FREE TRIAL



I am currently playing around with AWS EKS But I always get error: You must be logged in to the server (Unauthorized) when trying to run kubectl cluster-info command.

3

I have read a lot of AWS documentation and look at lots of similar issues who face the same problem. Unfortunately, none of them resolves my problem.



So, this is what I did

1. install all required packages

Blog

This Week #StackOverflow Infinity, Internet-Speak, and...

New post notices: Improving Stack Overflow questions

Featured on Meta

Upvotes on questions will be the same as upvotes on a...

Create EKS Full Access Policy

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "VisualEditor0",  
            "Effect": "Allow",  
            "Action": "eks:*",  
            "Resource": "*"  
        }  
    ]  
}
```

Call the Policy EKSFullAccess

Create policy

1 2

Review policy

Name* EKSFullAccess

Use alphanumeric and '+-,@-_.' characters. Maximum 128 characters.

Description

Maximum 1000 characters. Use alphanumeric and '+-,@-_.' characters.

Summary

Q Filter

Service	Access level	Resource	Request condition
Allow (1 of 221 services) Show remaining 220			
EKS	Full access	All resources	None

Create New Group

Create New Group Wizard

Step 1 : Group Name

Step 2 : Attach Policy

Step 3 : Review

Set Group Name

Specify a group name. Group names can be edited any time.

Group Name:

EKSDemoGroup

Example: Developers or ProjectAlpha

Maximum 128 characters

Attach Policies

Users Permissions **Access Advisor**

Managed Policies ^

The following managed policies are attached to this group. You can attach up to 10 managed policies.

Attach Policy

Policy Name	Actions
AmazonEC2FullAccess	Show Policy Detach Policy Simulate Policy
AmazonS3FullAccess	Show Policy Detach Policy Simulate Policy
AmazonSNSReadOnlyAccess	Show Policy Detach Policy Simulate Policy
AmazonVPCFullAccess	Show Policy Detach Policy Simulate Policy
IAMReadOnlyAccess	Show Policy Detach Policy Simulate Policy
EKSFullAccess	Show Policy Detach Policy Simulate Policy
AWSCloudFormationFullAccess	Show Policy Detach Policy Simulate Policy

Inline Policies ▾

Attach Inline Policy

- Allow EKS user to pass it's role to the new EKS Service Role

The screenshot shows the AWS IAM Groups page for the group 'EKSDemoGroup'. A red arrow points from the text 'Allow EKS user to pass it's role to the new EKS Service Role' to the 'Managed Policies' section. The 'Managed Policies' section displays a list of managed policies attached to the group, including 'AmazonEC2FullAccess', 'AmazonS3FullAccess', 'AmazonDynamoDBFullAccess', 'AmazonVPCFullAccess', 'IAMReadOnlyAccess', and 'AWSCloudFormationFullAccess'. Each policy has 'Show Policy', 'Detach Policy', and 'Simulate Policy' options. Below this section, there is a note: 'There are no inline policies to show. To create one: [click here](#)'.

Policy Name	Actions
AmazonEC2FullAccess	Show Policy Detach Policy Simulate Policy
AmazonS3FullAccess	Show Policy Detach Policy Simulate Policy
AmazonDynamoDBFullAccess	Show Policy Detach Policy Simulate Policy
AmazonVPCFullAccess	Show Policy Detach Policy Simulate Policy
IAMReadOnlyAccess	Show Policy Detach Policy Simulate Policy
AWSCloudFormationFullAccess	Show Policy Detach Policy Simulate Policy

Attach Inline Policy

- Pass ability to let EKS clusters modify roles created by this user.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "PolicyAtatementToAllowUserToPassAnyRole",  
            "Effect": "Allow",  
            "Action": [  
                "iam:PassRole",  
                "iam>CreateServiceLinkedRole",  
                "iam>CreateRole",  
                "iam>DeleteRole",  
                "iam:AttachRolePolicy",  
                "iam:DetachRolePolicy",  
                "iam:PutRolePolicy",  
                "iam>DeleteRolePolicy"  
            ],  
            "Resource": "arn:aws:iam::188966951897:role/*"  
        }  
    ]  
}
```

Policy for eksctl

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "PolicyAtatementToAllowUserToPassAnyRole",  
            "Effect": "Allow",  
            "Action": [  
                "iam:PassRole",  
                "iam>CreateServiceLinkedRole",  
                "iam>CreateRole",  
                "iam>DeleteRole",  
                "iam:AttachRolePolicy",  
                "iam:DetachRolePolicy",  
                "iam:PutRolePolicy",  
                "iam>DeleteRolePolicy",  
                "iam>CreateInstanceProfile"  
            ],  
            "Resource": "arn:aws:iam::188966951897:role/*"  
        }  
    ]  
}
```

Replace with Your organization

Create a new user

Add user

1 2 3 4 5

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

Add another user

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Access type* Programmatic access

Enables an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools.

AWS Management Console access

Enables a password that allows users to sign-in to the AWS Management Console.

Console password* Autogenerated password

Custom password

Require password reset User must create a new password at next sign-in

Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

* Required

Cancel

Next: Permissions

Add user to EKS Group

Add user

1 2 3 4 5

Set permissions

Add user to group Copy permissions from existing user Attach existing policies directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Add user to group

[Create group](#) [Refresh](#)

Group	Attached policies
<input type="checkbox"/> admins	IAMFullAccess and 2 more
<input checked="" type="checkbox"/> EKSDemoGroup	AmazonEC2FullAccess and 7 more

Showing 2 results

Set permissions boundary

[Cancel](#) [Previous](#) [Next: Tags](#)

Make note of Credentials

Add user

1

2

3

4

5



Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://188966951897.signin.aws.amazon.com/console>

Download .csv

	User	Access key ID	Secret access key	Password	Email login instructions
	<input checked="" type="checkbox"/> eksdude	AKIASX72O3PMW7LIC4MH	CgRMi+a0QZQCL6ij3Rncikjl sphpNulgR4KZdMih	89(#L#L]VE30	Send email

AWS CLI

- Download the AWS CLI tools
 - <https://github.com/vallard/EKS-Training/blob/master/segment1/aws-creds.md>

Automating with Terraform

- Infrastructure as Code
- Automate consistent workflows
- Build across multiple clouds
- Version control infrastructure

`terraform init`

`terraform plan`

`terraform apply`



Use the Outputs from Terraform

- **User Password for Console Sign in:**

```
terraform output password | base64 --decode |  
gpg --decrypt | pbcopy
```

- **AWS Credentials for CLI:**

```
aws configure --profile=eksduude
```

- **Access Key:**

```
terraform output key | pbcopy
```

- **Access Secret:**

```
terraform output secret | base64 --decode | gpg  
--decrypt | pbcopy
```

Alternate Credential storage

	User	Access key ID	Secret access key	Password	Email login instructions
▶	<input checked="" type="checkbox"/> eksdude	AKIASX72O3PMREGYZ2CQ	***** Show	***** Show	Send email ↗

Add Credentials to `~/.aws/credentials` file

```
20 [cr-demouser]
21 aws_access_key_id = AKIASX72O3PMREGYZ2CQ
22 aws_secret_access_key = 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
```

Add configuration to `~/.aws/config` file

```
9 [profile cr-demouser]
10 region=us-west-2
11 output=text
```

Set Command Line User

```
export AWS_PROFILE=eksdue  
aws eks list-clusters  
aws ec2 describe-instances
```

Sign in as new user



Account ID or alias

188966951897

IAM user name

eksduke

Password

.....

Sign In

[Sign-in using root account credentials](#)

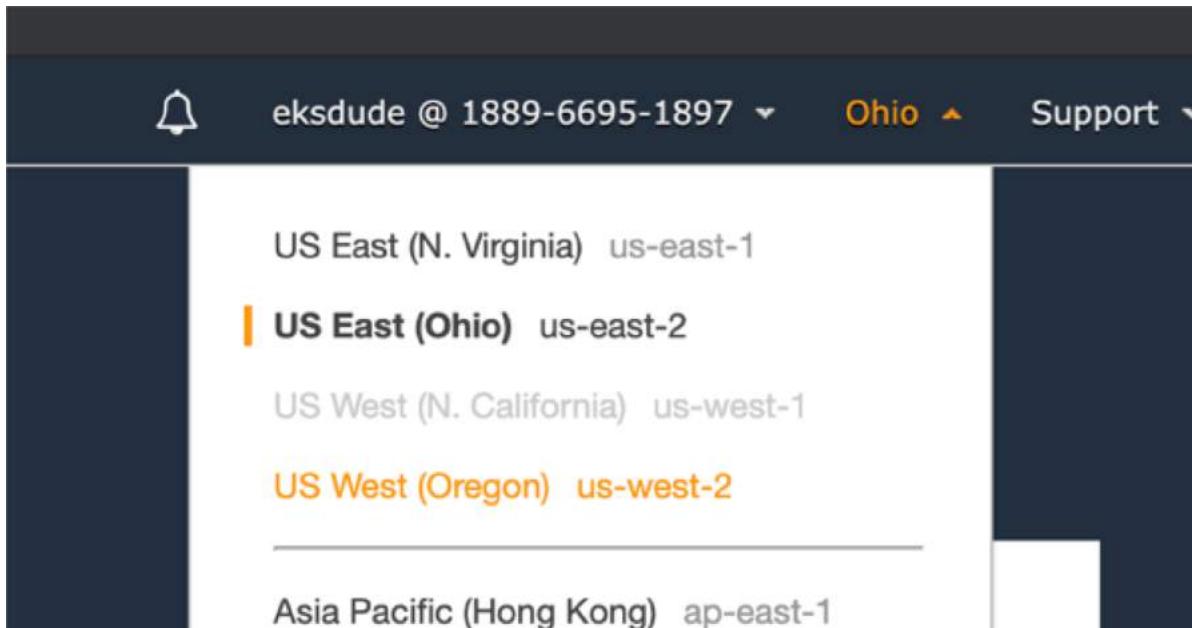
[Forgot password?](#)

AWS Lambda Provisioned Concurrency

Keep your AWS Lambda functions initialized and hyper-ready to respond in double-digit milliseconds, at any scale



Check Proper Region



A large, semi-transparent white play button icon is positioned on the left side of the slide, consisting of a triangle inside a circle.

Kubernetes on AWS

Segment 3: Creating EKS Clusters

Big Picture

- Get eksctl
- Create an EKS Cluster with eksctl
- Create EKS Service Role
- Create EKS Control Plain manually and with Cloud Formation

Different Modes of Running Clusters

AWS Fargate - Linux

- kube-system and default pods run in Fargate
- Fargate Profile Created

Managed Nodes - Linux

- Linux ec2 instances created
- Cloud Formation Stack for EC2 instances created

Self Managed Nodes - Windows

- For running Windows Container Images
- Requires at least 1 Linux instance created to run system services like coreDNS

You can mix and match instance types in the same cluster

You can use eksctl to create clusters

```
eksctl create cluster \
--name aug05 \
--fargate \
--version 1.17 \
--region us-west-2 \
--nodegroup-name standard-
workers \
--node-type t3.medium \
--nodes 3 \
--nodes-min 1 \
--nodes-max 4 \
--managed
```

Hint: If the version is not supported, upgrade eksctl!
brew upgrade eksctl

Use eksctl to Create the Cluster

```
vallards-MacBook-Pro:segment03-install vallard$ ./createEKScctlCluster.sh
+ eksctl create cluster --name aug05 --fargate --version 1.17 --region us-west-2 --nodegroup-name standard
-workers --node-type t3.medium --nodes 3 --nodes-min 1 --nodes-max 4 --managed
[i] eksctl version 0.25.0
[i] using region us-west-2
[i] setting availability zones to [us-west-2b us-west-2c us-west-2d]
[i] subnets for us-west-2b - public:192.168.0.0/19 private:192.168.96.0/19
[i] subnets for us-west-2c - public:192.168.32.0/19 private:192.168.128.0/19
[i] subnets for us-west-2d - public:192.168.64.0/19 private:192.168.160.0/19
[i] using Kubernetes version 1.17
[i] creating EKS cluster "aug05" in "us-west-2" region with Fargate profile and managed nodes
[i] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
[i] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-west-2 --cluster=aug05'
[i] CloudWatch logging will not be enabled for cluster "aug05" in "us-west-2"
[i] you can enable it with 'eksctl utils update-cluster-logging --region=us-west-2 --cluster=aug05'
[i] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "aug05" in "us-west-2"
[i] 2 sequential tasks: { create cluster control plane "aug05", 2 sequential sub-tasks: { create fargate profiles, create managed nodegroup "standard-workers" } }
[i] building cluster stack "eksctl-aug05-cluster"
[i] deploying stack "eksctl-aug05-cluster"
```

	real	21m32.641s
	user	0m0.801s
	sys	0m1.179s



Create cluster manually

- Add EKS Service Role
- CloudFormation to create VPC and entities
- Create EKS Control Plane

Create EKS Service Role

Identity and Access Management (IAM)

AWS Account (188966951897)

Dashboard

Groups

Users

Roles

Create role

Search

Role name

AWSServic

AWSServic

...

Create EKS Service Role (cont.)

API Gateway	CodeBuild	EKS	Kinesis	S3
AWS Backup	CodeDeploy	EMR	Lambda	SMS
AWS Chatbot	CodeStar Notifications	ElastiCache	Lex	SNS
AWS Support	Comprehend	Elastic Beanstalk	License Manager	SWF
Amplify	Config	Elastic Container Service	Machine Learning	SageMaker
AppStream 2.0	Connect	Elastic Transcoder	Macie	Security Hub
AppSync	DMS	Elastic Load Balancing	MediaConvert	Service Catalog
Application Auto Scaling	Data Lifecycle Manager	Forecast	Migration Hub	Step Functions
Application Discovery Service	Data Pipeline	Global Accelerator	OpsWorks	Storage Gateway
Batch	DataSync	Glue	Personalize	Textract
Chime	DeepLens	Greengrass	QLDB	Transfer
CloudFormation	Directory Service	GuardDuty	RAM	Trusted Advisor
CloudHSM	DynamoDB	Inspector	RDS	VPC
CloudTrail	EC2	IoT	Redshift	WorkLink
CloudWatch Application Insights	EC2 - Fleet	IoT Things Graph	Rekognition	WorkMail
CloudWatch Events	EC2 Auto Scaling	KMS	RoboMaker	

Select your use case

EKS

Allows EKS to manage clusters on your behalf.

EKS - Nodegroup

Allow EKS to manage nodegroups on your behalf.

Create EKS Service Role (cont.)

Create role

1 2 3 4

Attached permissions policies

The type of role that you selected requires the following policy.

Policy name	Used as	Description
AmazonEKSClusterPolicy	Permissions policy (1)	This policy provides Kubernetes the permission...
AmazonEKSServicePolicy	Permissions policy (2)	This policy allows Amazon Elastic Container S...

Create EKS Service Role (cont.)

Create role

1 2 3 4

Review

Provide the required information below and review this role before you create it.

Role name* eksServiceRole

Use alphanumeric and '+,-,@-_` characters. Maximum 64 characters.

Role description Allows EKS to manage clusters on your behalf.

Maximum 1000 characters. Use alphanumeric and '+,-,@-_` characters.

Trusted entities AWS service: eks.amazonaws.com

Policies  AmazonEKSClusterPolicy 

 AmazonEKSServicePolicy 

Permissions boundary Permissions boundary is not set

No tags were added.

But wait! We already did this in Terraform

```
resource "aws_iam_role" "EKSServiceRole" {
  name = "EKSServiceRole"
  assume_role_policy = <<-EOF
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Principal": {
          "Service": "eks.amazonaws.com"
        },
        "Effect": "Allow",
        "Sid": ""
      }
    ]
  }
EOF
}

resource "aws_iam_role_policy_attachment" "EKSServiceRoleAttachmentsCluster" {
  role = aws_iam_role.EKSServiceRole.name
  policy_arn  = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
}

resource "aws_iam_role_policy_attachment" "EKSServiceRoleAttachmentsService" {
  role = aws_iam_role.EKSServiceRole.name
  policy_arn  = "arn:aws:iam::aws:policy/AmazonEKSServicePolicy"
}
```

CloudFormation to create Network stack

```
1 ---
2 AWSTemplateFormatVersion: '2010-09-09'
3 Description: 'Amazon EKS Sample VPC'
4
5 Parameters:
6
7   VpcBlock:
8     Type: String
9     Default: 192.168.0.0/16
10    Description: The CIDR range for the VPC. This should be a valid private (RFC 1918) CIDR range.
11
12   Subnet01Block:
13     Type: String
14     Default: 192.168.64.0/18
15     Description: CidrBlock for subnet 01 within the VPC
16
17   Subnet02Block:
18     Type: String
19     Default: 192.168.128.0/18
20     Description: CidrBlock for subnet 02 within the VPC
21
22   Subnet03Block:
23     Type: String
24     Default: 192.168.192.0/18
25     Description: CidrBlock for subnet 03 within the VPC. This is used only if the region has more than 2 AZs.
26
```

Create VPC with Cloud Formation

The screenshot shows the AWS CloudFormation console with the following details:

- CloudFormation > Stacks > vpc20191231**
- vpc20191231** Stack (1) is listed in the Stacks (1) section.
- Events** tab is selected in the navigation bar.
- Events (1)** table:

Timestamp	Logical ID	Status	Status reason
2019-12-31 08:18:41 UTC-0800	vpc20191231	CREATE_IN_PROGRESS	User Initiated
- Filter by stack name**:
- Active**:
- View nested**:
- Change sets**:
- Stack info**, **Resources**, **Outputs**, **Parameters**, **Template** buttons.
- Delete**, **Update**, **Stack actions ▾**, **Create stack ▾** buttons.

Examine Components of new VPC

The screenshot shows the AWS CloudFormation Outputs tab for a stack. The tab bar includes Stack info, Events, Resources, Outputs (which is selected), Parameters, and Template. Below the tab bar, there is a section titled "Change sets". The main area displays the "Outputs (3)" table.

Key	Value	Description	Export name
SecurityGroups	sg-0f3098218f1f0676d	Security group for the cluster control plane communication with worker nodes	-
SubnetIds	subnet-06c96c7bef92ad636, subnet-0a8f2a225aa1478f5, subnet-0ec2e512f5986fb97	All subnets in the VPC	-
VpcId	vpc-0c0c9c4c20e8c03f2	The VPC Id	-

Create EKS Control Plane

The screenshot shows the AWS Amazon Container Services console with the EKS Control Plane creation interface. The top navigation bar includes the AWS logo, Services dropdown, Resource Groups dropdown, a bell icon, user name 'castlerock', a red box highlighting the 'Oregon' region dropdown, and a Support link.

Amazon Container Services

- Amazon ECS
- Clusters
- Task definitions
- Amazon EKS**
- Clusters

Containers

Elastic Kubernetes Service (Amazon EKS)

Fully managed Kubernetes control plane

Amazon EKS is a managed service that makes it easy for you to use Kubernetes on AWS without needing to install and operate your own Kubernetes control plane.

Create EKS cluster

Cluster name:

Next step

Pricing (US)

EKS Control Plane	\$0.20/hr
Worker nodes	EC2 Pricing

Configure cluster

Cluster configuration Info

Name - *Not editable after creation.*

Enter a unique name for this cluster.

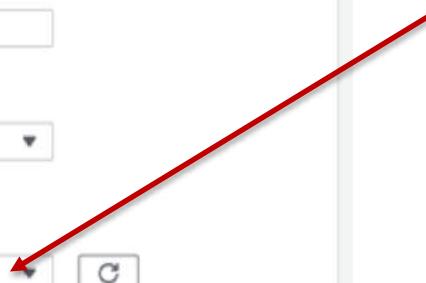
Kubernetes version Info

Select the Kubernetes version for this cluster.

Cluster Service Role Info - *Not editable after creation.*

Select the IAM Role to allow the Kubernetes control plane to manage AWS resources on your behalf.

To create a new role, go to the [IAM console](#).



Created by Terraform

Specify networking

Use values created by Cloud Formation

Networking Info

These properties cannot be changed after the cluster is created.

VPC Info

Select a VPC to use for your EKS Cluster resources.
To create a new VPC, go to the [VPC console](#).

vpc-035d5887a17d7561b | aug05man-vpc-VPC



Subnets Info

Choose the subnets in your VPC where the control plane may place elastic network interfaces (ENIs) to facilitate communication with your cluster.
To create a new subnet, go to the corresponding page in the [VPC console](#).

Select subnets

subnet-0f04b6a34e643bea1 X

subnet-07926e463d7421793 X

subnet-09639dce8576f8536 X



Security groups Info

Choose the security groups to apply to the EKS-managed Elastic Network interfaces that are created in your worker node subnets.
To create a new security group, go to the corresponding page in the [VPC console](#).

Select security groups

sg-03a3145835c0dd4ba X



Cluster endpoint access Info

Configure access to the Kubernetes API server endpoint.

Public

The cluster endpoint is accessible from outside of your VPC. Worker node traffic will leave your VPC to connect to the endpoint.

Public and private

The cluster endpoint is accessible from outside of your VPC. Worker node traffic to the endpoint will stay within your VPC.

Private

The cluster endpoint is only accessible through your VPC. Worker node traffic to the endpoint will stay within your VPC.

► Advanced Settings

Chose endpoint access

Cancel

Previous

Next



Pearson

Additional Settings and Options

Private access
Enable private API server endpoint access
 Disabled

Public access
Enable public API server endpoint access
 Enabled

Logging

CloudWatch log group
EKS automatically creates a CloudWatch log group for you when you enable logging.

API server
Logs pertaining to API requests to the cluster
 Disabled

Audit
Logs pertaining to cluster access via the Kubernetes API
 Disabled

Authenticator
Logs pertaining to authentication requests into the cluster
 Disabled

Controller manager
Logs pertaining to state of cluster controllers
 Disabled

Scheduler
Logs pertaining to scheduling decisions
 Disabled

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources. Tags are optional.

This cluster does not have any tags

[Add tag](#)

Remaining tags available to add: 50

Ensure cluster is being created

ⓘ Cluster creation in progress
aug05man is now being created. This process may take several minutes.

EKS > Clusters > aug05man

aug05man

[Edit](#) [Delete](#)

Cluster configuration

Kubernetes version Info	1.17	Status	Creating
Platform version Info	eks.2		

Break Time!



Segment 4: Verifying EKS

Big Picture

- Create Worker Role
- Get kubectl
- Test EKS cluster

Create AWS Role

Create stack

Prerequisite - Prepare template

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

Template is ready

Use a sample template

Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL

Upload a template file

Upload a template file

Choose file

amazon-eks-nodegroup-role.yaml

JSON or YAML formatted file

S3 URL: <https://s3-us-west-2.amazonaws.com/cf-templates-1se0e9to70ap8-us-west-2/20193655pd-amazon-eks-nodegroup-role.yaml>

[View in Designer](#)

Cancel

Next

Specify stack details

Stack name

Stack name

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

No parameters

There are no parameters defined in your template

Cancel

Previous

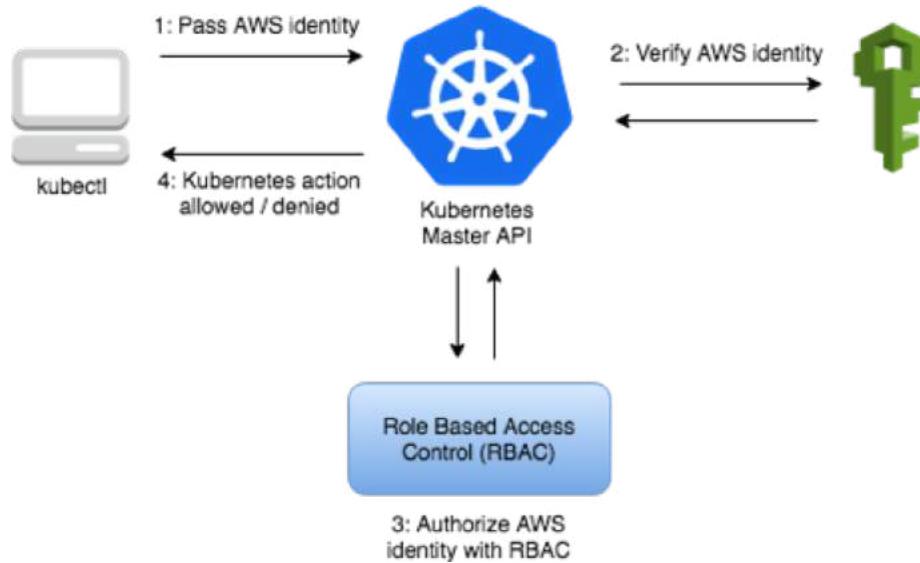
Next

Accept all
defaults

The screenshot shows the AWS CloudFormation console with the stack named "dec3-eks-node-role". The "Events" tab is selected. The table lists the following events:

Timestamp	Logical ID	Status	Status reason
2019-12-03 08:45:41 UTC-0800	dec3-eks-node-role	CREATE_COMPLETE	-
2019-12-03 08:45:39 UTC-0800	NodeInstanc eRole	CREATE_COMPLETE	-
2019-12-03 08:45:23 UTC-0800	NodeInstanc eRole	CREATE_IN_PROGRESS	Resource creation Initiated
2019-12-03 08:45:23 UTC-0800	NodeInstanc eRole	CREATE_IN_PROGRESS	-
2019-12-03 08:45:20 UTC-0800	dec3-eks-node-role	CREATE_IN_PROGRESS	User Initiated

Authentication with EKS



Source: <https://docs.aws.amazon.com/eks/latest/userguide/managing-auth.html>

Connect to cluster

```
export AWS_PROFILE=eksduude  
aws eks list-clusters  
aws eks update-kubeconfig --name <cluster-name>  
kubectl config get-contexts  
kubectl config use-context <new context>  
kubectl get svc
```

```
vallards-MacBook-Pro:segment2 vallard$ aws eks update-kubeconfig --name dec3  
Added new context arn:aws:eks:us-west-2:188966951897:cluster/dec3 to /Users/vallard/.kube/config  
vallards-MacBook-Pro:segment2 vallard$ kubectl config get-contexts  
CURRENT NAME CLUSTER NAMESPACE  
AUTHINFO arn:aws:eks:us-east-1:254056899943:cluster/matomo arn:aws:eks:us-east-1:254056899943:cluste  
r/matomo arn:aws:eks:us-east-1:254056899943:cluster/matomo * arn:aws:eks:us-west-2:188966951897:cluste  
r/dec3 arn:aws:eks:us-west-2:188966951897:cluster/dec3 vallards-MacBook-Pro:segment2 vallard$ kubectl config use-context arn:aws:eks:us-west-2:188966951897:clu  
ster/dec3  
Switched to context "arn:aws:eks:us-west-2:188966951897:cluster/dec3".
```



Interacting

- eksctl get clusters
- aws eks list-clusters

```
vallards-MacBook-Pro:Code vallard$ kubectl get nodes  
No resources found in default namespace.
```

Add Worker Nodes To Console Cluster

Node Groups (0) [Info](#)

Group name	▲	Desired size	▼	Kubernetes version	▼	Status	▼
No managed node groups							
This cluster does not have any managed node groups.							
Nodes that are not part of an Amazon EKS managed node group are not shown in the AWS console.							



Add node group

Configure node group

A node group is a group of EC2 instances that supply compute capacity to your Amazon EKS cluster. You can add multiple node groups to your cluster. [Info](#)

Group configuration

These properties cannot be changed after the node group is created.

Name

Assign a unique name for this node group

wg01

Node IAM Role Name [Info](#)

Select the IAM Role that will be used by the nodes.

dec31-eksrole-NodeInstanceRole-VZVZ8FBO93GW

Role created from
CloudFormation



Subnets [Info](#)

Specify the subnets in your VPC where your nodes will run

Add subnets

subnet-0c23d7de4a0cae3a9 vpc20191231-Subnet01 X

us-west-2a 192.168.64.0/18

subnet-0ffed858554051f8a vpc20191231-Subnet03 X

us-west-2c 192.168.192.0/18

subnet-0742b5586cd70dd4a vpc20191231-Subnet02 X

us-west-2b 192.168.128.0/18

Allow remote access to nodes [Info](#)

Without remote access enabled you will not be able to directly connect to nodes after they are created.

Step 1
Configure node group

Step 2
Set compute
configuration

Step 3
Set scaling
configuration

Step 4
Review and create

Set compute configuration

Node compute configuration

These properties cannot be changed after the node group is created.

AMI type [Info](#)

Select the EKS-optimized Amazon Machine Image for nodes

Amazon Linux 2 (AL2_x86_64)

Instance type [Info](#)

Select the EC2 instance type for nodes

t3.micro

Disk size

Select the size of the attached EBS volume for each node

20

GiB

Cancel

Previous

Next

Verify Nodes

- kubectl get nodes
- kubectl get nodes -o wide

```
vallards-MacBook-Pro:segment04-verify vallard$ kubectl get nodes
NAME                               STATUS  ROLES   AGE     VERSION
ip-192-168-12-72.us-west-2.compute.internal  Ready   <none>  54m    v1.14.7-eks-1861c5
ip-192-168-56-142.us-west-2.compute.internal  Ready   <none>  55m    v1.14.7-eks-1861c5
ip-192-168-67-254.us-west-2.compute.internal  Ready   <none>  54m    v1.14.7-eks-1861c5
```

Namespaces

- Logical separations for applications
- Idea is to group different Kubernetes resources such as Pods, Services, Ingress Rules, and Deployments in logical boxes.

```
$ kubectl get ns
```

Pods

```
$ kubectl get pods
```

```
$ kubectl get po -all-namespaces
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	aws-node-6jjnf	1/1	Running	0	52m
kube-system	aws-node-7cdw6	1/1	Running	0	52m
kube-system	aws-node-g6tkd	1/1	Running	0	52m
kube-system	coredns-84549585c-jj6q5	1/1	Running	0	57m
kube-system	coredns-84549585c-sx2jf	1/1	Running	0	57m
kube-system	kube-proxy-76wgk	1/1	Running	0	52m
kube-system	kube-proxy-f7w2p	1/1	Running	0	52m
kube-system	kube-proxy-ppl9p	1/1	Running	0	52m

Aws-Node Daemon Set

```
$ kubectl describe ds -n kube-system aws-node
```

```
Pods Status: 3 Running / 0 Waiting / 0 Succeeded / 0 Failed
```

```
Pod Template:
```

```
Labels:          k8s-app=aws-node
```

```
Service Account: aws-node
```

```
Containers:
```

```
aws-node:
```

```
Image:      602401143452.dkr.ecr.us-west-2.amazonaws.com/amazon-k8s-cni:v1.5.5
```

```
Port:       61678/TCP
```

```
Host Port: 61678/TCP
```

```
Requests:
```

```
  cpu: 10m
```

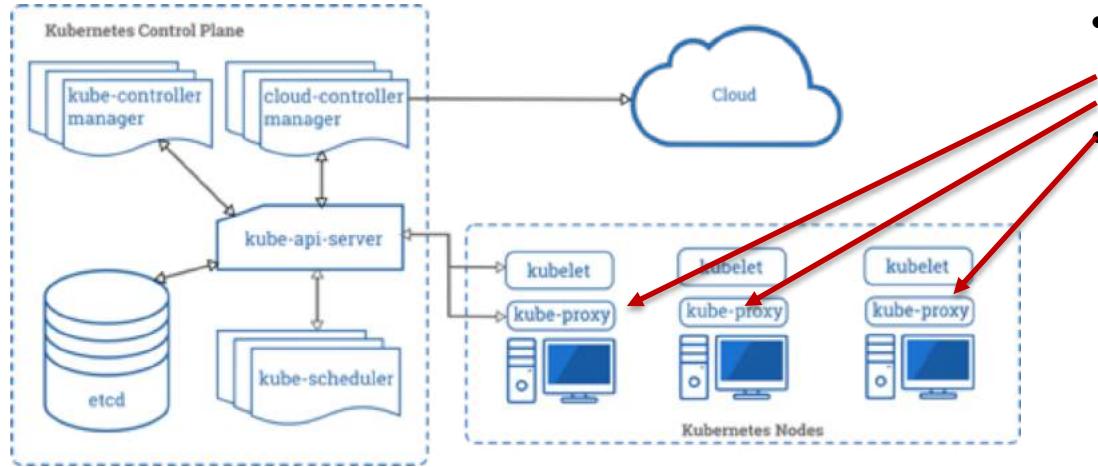
```
Environment:
```

```
  AWS_VPC_K8S_CNI_LOGLEVEL: DEBUG
```

```
  MY_NODE_NAME:           (v1:spec.nodeName)
```



Kube Proxy



- Network Proxy that runs on each node
- Maintains network rules on nodes that allow network communication to Pods

What services run by default?

- What is running by default?

Kubernetes service – API service so internal services can communicate with Kubernetes API service

vallards-MacBook-Pro:segment03-install vallard\$ kubectl get svc --all-namespaces						
NAMESPACE	NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
default	kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP	26m
kube-system	kube-dns	ClusterIP	10.100.0.10	<none>	53/UDP, 53/TCP	26m

DNS service so pods can find other services in the cluster without having to hard code IP addresses

Test kube-dns

```
$ kubectl create -f segment04-verify/dnstablet/  
$ kubectl exec -it bb8-... -- /bin/sh  
$ wget -q -S -O - ngx  
$ wget -q -S -O - ngx.default  
$ wget -q -S -O - ngx.default.svc  
$ wget -q -S -O - ngx.default.svc.cluster.local
```



Segment 5: Running Applications

Big Picture

- LoadBalancers
- Ingress Controller
- R53 Routing
- TLS
- Persistence

Expose a Service

- Currently our service is inward facing. (No access from outside)
- We can access it remotely by changing the Service interface from ClusterIP to LoadBalancer
- This automatically creates a load balancer for us
- We can do this for any service we want to expose

```
vallards-MacBook-Pro:test01 vallard$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP
kubernetes  ClusterIP  10.100.0.1    <none>
nginx     LoadBalancer  10.100.49.193  aada23c992d8611ea87060a79e3e7cb4-855121620.us-west-2.elb.amazonaws.com
                                                 PORT(S)        AGE
                                                 443/TCP       91m
                                                 80:30107/TCP  44m
```

Change to Load balancer

- kubectl edit svc ngx
- Change from:
 - type: ClusterIP
- To:
 - type: LoadBalancer

```
14  name: ngx
15  namespace: default
16  resourceVersion: "14827"
17  selfLink: /api/v1/namespaces/default/services/ngx
18  uid: 283c67cd-00a1-49d4-b410-7b975006c6b0
19 spec:
20   clusterIP: 10.100.45.250
21   ports:
22     - port: 80
23       protocol: TCP
24       targetPort: 80
25   selector:
26     run: ngx
27   sessionAffinity: None
28   type: LoadBalancer
29 status:
30   loadBalancer: {}
```

Load Balancing Types

- Classic Load Balancer
 - Default
 - EC2 Classic Support
- Network Load Balancer
 - Newer improved! TCP, TLS, UDP
 - Static IPs: Elastic IP
- Application Load Balancer
 - HTTP/HTTPS protocol support

```
5 apiVersion: v1
6 kind: Service
7 metadata:
8   annotations:
9     service.beta.kubernetes.io/aws-load-balancer-type: nlb
10    kubectl.kubernetes.io/last-applied-configuration: |
11      {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"name":"nginx","namespace":"default"},"spec":{"ports":[{"port":80,"targetPort":80}],"selector":{"run":"nginx"}}}
12    creationTimestamp: "2020-08-06T17:03:12Z"
```

Multiplexing Load Balancers

- Cost: \$0.0225 / hour => \$16/month for each exposed service
- Need one for each service, when traffic may not be that high
- Solution: Use an ingress controller

Too Many External Services?

<https://docs.castlerock.ai>

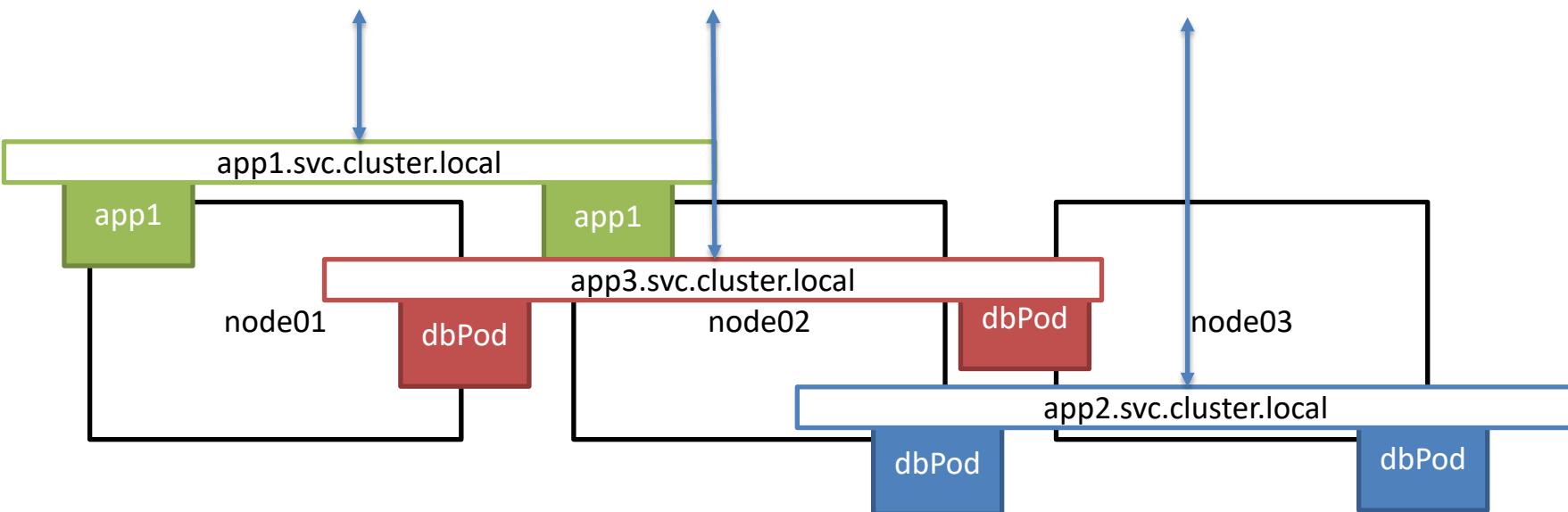
<https://blog.castlerock.ai>

<https://castlerock.ai/support>

LoadBalancer

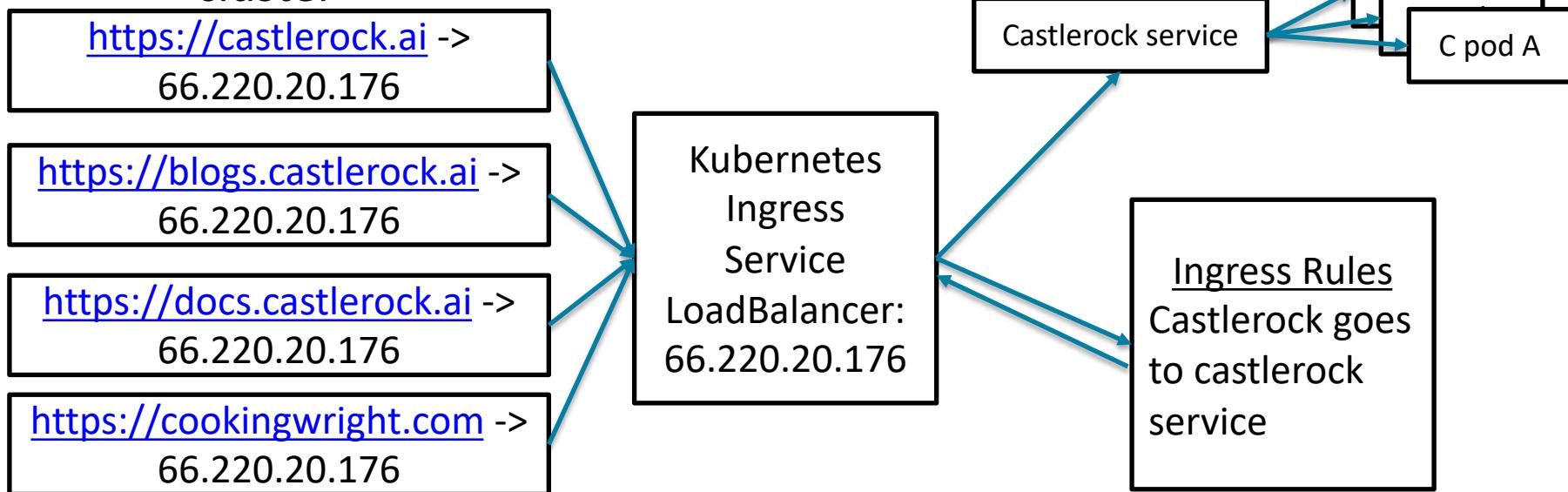
LoadBalancer

LoadBalancer

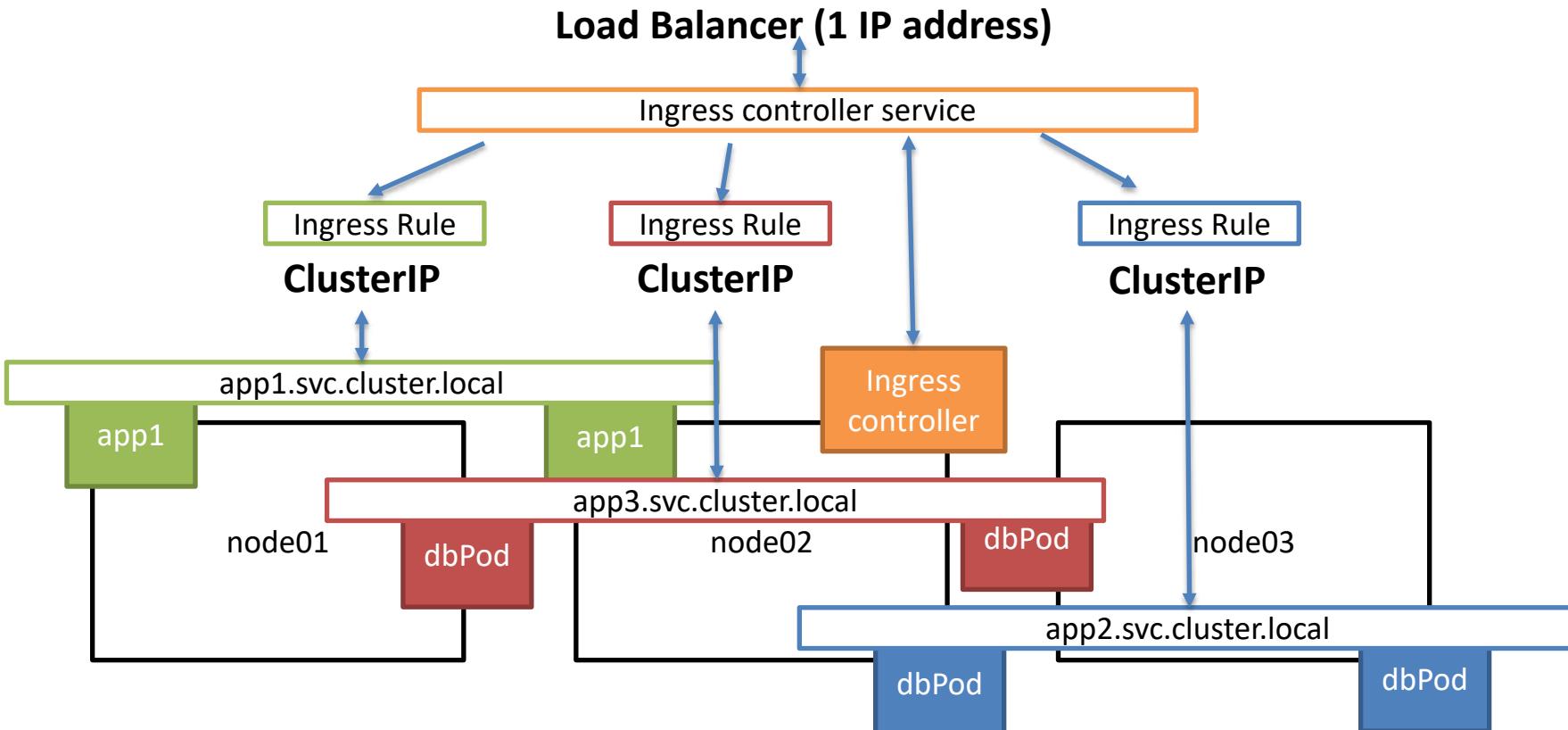


The Need for Ingress Controllers

- Only a fixed amount of external IP addresses
- Might have hundreds of different apps that require URLs in a cluster



Ingress Controller Architecture



Ingress Controller

- A normal Kubernetes Service
- Subscribes to Kubernetes for changes in:
 - Services
 - Ingress rules
- Requires Ingress Rules to route services
- Different Ingress Controllers are available:
 - NGINX
 - HA Proxy
 - Traefik
 - Others

Change nginx back to ClusterIP

- Remove nodePort
- Change type from LoadBalancer to ClusterIP

Ingress Controllers

- Deploy an nginx Load Ingress Controller

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.34.1/deploy/static/provider/aws/deploy.yaml
```

- Examine our new ingress controller service

```
kubectl get svc -n ingress-nginx
```

- Note the NLB address and open browser

503 Service Temporarily Unavailable

nginx/1.19.1

What is in an Ingress Rule?

- Host name: e.g.: www.foo.example.com
- A list of paths /blog/2019-12-11

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: default-backend
spec:
  rules:
    - host: castlerock.ai
      http:
        - paths:
            - backend:
                serviceName: cr-homepage
                servicePort: 80
```

Match Host name
'castlerock.ai'

Accept all paths

Configure Default Backend

- If traffic hits the ingress controller and no rules are specified then we have a default backend.
- Make your customized 404 page

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: default-backend
spec:
  backend:
    serviceName: default-backend
    servicePort: 80
```

No rules, so will catch all other requests.

Alternative would be to have a 'page not found by nginx'

Ingress Rule options

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: default-backend
spec:
  rules:
  - host: castlerock.ai
    http:
      - path: /docs
        backend:
          serviceName: cr-docs
          servicePort: 80
      - path: /blog
        backend:
          serviceName: wordpress
          servicePort: 80
```

Match Host name
'castlerock.ai'

Docs route to docs
app.

Blog path routes
to wordpress
service

Create Default Backend

- `kubectl apply -f segment05-applications/default-backend.yaml`

The screenshot shows a web application interface. At the top, a red bar contains a warning message: "WARNING: This server may expose sensitive and secret information. Be careful." Below the bar, the title "default-backend-856f5d6bd-vlhn6" is displayed in large, bold, dark gray font. Underneath the title, it says "Demo application version v0.8.7-7" and "Serving on 192.168.61.198". On the left side, there is a vertical sidebar with a light gray background containing several tabs: "Request Details", "Server Env", "Memory", "Liveness Probe", "Readiness Probe", "DNS Query" (which is highlighted in dark gray), "KeyGen Workload", "MemQ Server", and "File system browser". The main content area has a white background and displays a "Server side DNS query" form. It includes fields for "DNS Type A", "Name", and a "QUERY" button.

Map to a Domain

- Create a new Route 53 Resource

The screenshot shows the AWS Route 53 service in the AWS Management Console. The left sidebar navigation bar includes links for Dashboard, Hosted zones (which is the current page), Health checks, Traffic flow, Traffic policies, Policy records, Domains, Registered domains, Pending requests, Resolver, VPCs, Inbound endpoints, Outbound endpoints, and Rules.

The main content area has a header with "Create Hosted Zone", "Go to Record Sets", and "Delete Hosted Zone" buttons. A search bar and a "All Types" dropdown are also present. Below this is a table header with columns: Domain Name, Type, Record Set Count, Comment, and Hosted Zone ID. The table body displays the message "You have no hosted zones".

A modal window titled "Create Hosted Zone" is open on the right. It contains descriptive text about what a hosted zone is, followed by input fields for "Domain Name" (set to "k8s.castlerock.ai"), "Comment" (set to "Kubernetes Class Dom"), and "Type" (set to "Public Hosted Zone"). A note below the type field states: "A public hosted zone determines how traffic is routed on the internet."

Map to a Domain

- Create a new Route 53 Resource

	Name	Type	Value	Evaluate Target Hea
<input type="checkbox"/>	k8s.castlerock.ai.	NS	ns-493.awsdns-61.com. ns-811.awsdns-37.net. ns-1344.awsdns-40.org. ns-1585.awsdns-06.co.uk.	-
<input type="checkbox"/>	k8s.castlerock.ai.	SOA	ns-493.awsdns-61.com. awsdns-hostmaster.amazon	-

Add NS Record to Domain Service

<input type="checkbox"/>	NS Record	k8s	ns-493.awsdns-61.com.	Automatic	
<input type="checkbox"/>	NS Record	k8s	ns-811.awsdns-37.net.	Automatic	
<input type="checkbox"/>	NS Record	k8s	ns-1344.awsdns-40.org.	Automatic	
<input type="checkbox"/>	NS Record	k8s	ns-1585.awsdns-06.co.uk.	Automatic	

 ADD NEW RECORD

 SHOW LESS

Create Alias Record

Define simple record

Record name
To route traffic to a subdomain, enter the subdomain name. For example, to route traffic to blog.example.com, enter blog. If you leave this field blank, the default record name is the name of the domain.

blog .k8s.castlerock.ai

Valid characters: a-z, 0-9, ! " # \$ % & ' () * + , - / ; < = > ? @ [\] ^ _ { } . ~

Value/Route traffic to
The option that you choose determines how Route 53 responds to DNS queries. For most options, you specify where you want to route internet traffic.

Alias to Network Load Balancer

US West (Oregon) [us-west-2]

Q 7945aa9abd45ae3deca3f-3345cc15f67e0899.elb.us-west-2.amazonaws.com X

Record type
The DNS type of the record determines the format of the value that Route 53 returns in response to DNS queries.

A – Routes traffic to an IPv4 address and some AWS resources

Choose when routing traffic to AWS resources for EC2, API Gateway, Amazon VPC, CloudFront, Elastic Beanstalk, ELB, or S3. For example: 192.0.2.44.

Evaluate target health
Select Yes if you want Route 53 to use this record to respond to DNS queries only if the specified AWS resource is healthy.

Yes

[Cancel](#) [Define simple record](#)

Map our nginx service to ingress

- `kubectl apply -f ngx-ing.yaml`

Test Name Resolution



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Let's Encrypt

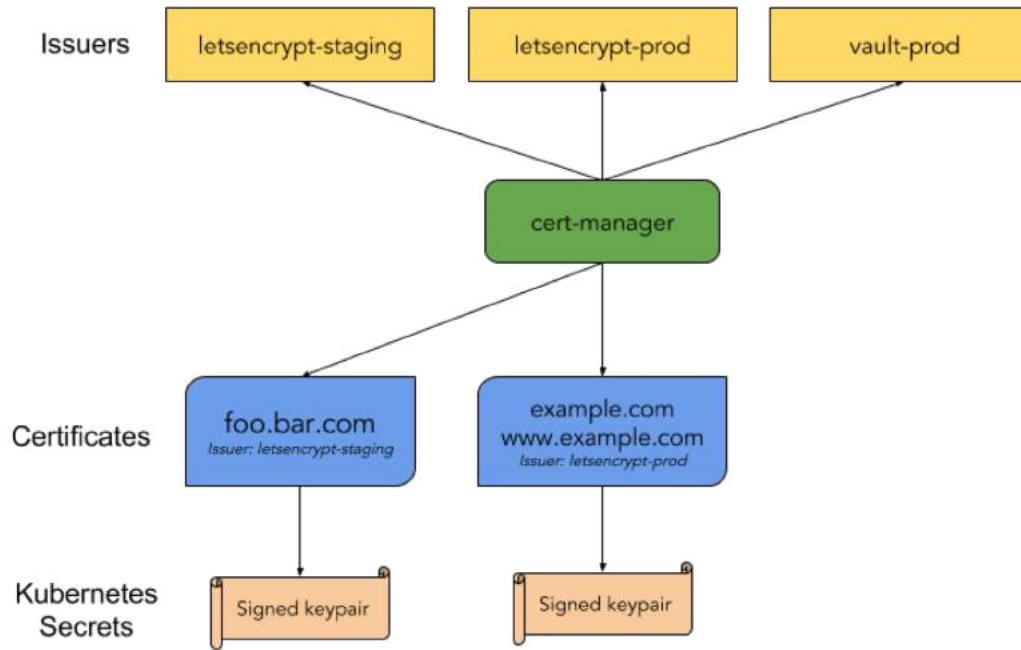


- Free, automated, and open certificate authority (CA) from Internet Security Research Group (ISRG)
- Goal to create a more secure Internet
- Automated service

Let's Encrypt with Kubernetes

- Cert-manager
 - Open source Kubernetes native certificate manager
 - Works with different issuers including Let's Encrypt
 - Fast way to get free TLS certifications

Cert-manager Architecture



Source: <https://github.com/jetstack/cert-manager> accessed 05/27/2019

Create Manifests

```
kubectl create -f cert-manager.yaml
```

```
vallards-MacBook-Pro:cert-manager vallards$ kubectl create -f cert-manager.yaml --validate=false
customresourcedefinition.apiextensions.k8s.io/certificaterequests.cert-manager.io created
customresourcedefinition.apiextensions.k8s.io/certificates.cert-manager.io created
customresourcedefinition.apiextensions.k8s.io/challenges.acme.cert-manager.io created
customresourcedefinition.apiextensions.k8s.io/clusterissuers.cert-manager.io created
customresourcedefinition.apiextensions.k8s.io/issuers.cert-manager.io created
customresourcedefinition.apiextensions.k8s.io/orders.acme.cert-manager.io created
serviceaccount/cert-manager-cainjector created
serviceaccount/cert-manager created
serviceaccount/cert-manager-webhook created
clusterrole.rbac.authorization.k8s.io/cert-manager-cainjector created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-cainjector created
role.rbac.authorization.k8s.io/cert-manager-cainjector:leaderelection created
rolebinding.rbac.authorization.k8s.io/cert-manager-cainjector:leaderselection created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-webhook:auth-delegate created
rolebinding.rbac.authorization.k8s.io/cert-manager-webhook:webhook-authentication-reader created
clusterrole.rbac.authorization.k8s.io/cert-manager-webhook:webhook-requester created
role.rbac.authorization.k8s.io/cert-manager:leaderelection created
rolebinding.rbac.authorization.k8s.io/cert-manager:leaderselection created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-issuers created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-clusterissuers created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-certificates created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-orders created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-challenges created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-ingress-shim created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-issuers created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-clusterissuers created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-certificates created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-orders created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-challenges created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-ingress-shim created
clusterrole.rbac.authorization.k8s.io/cert-manager-view created
clusterrole.rbac.authorization.k8s.io/cert-manager-edit created
service/cert-manager created
service/cert-manager-webhook created
deployment.apps/cert-manager-cainjector created
deployment.apps/cert-manager created
```

Validate

```
kubectl get pods --namespace cert-manager
```

NAME	READY	STATUS	RESTARTS	AGE
cert-manager-598bfb5ddb-jn4k4	1/1	Running	0	2m17s
cert-manager-cainjector-594fd9cc45-rr8gd	1/1	Running	0	2m17s
cert-manager-webhook-785ff8fc78-xtsbz	1/1	Running	0	2m17s

Create Cluster Issuers

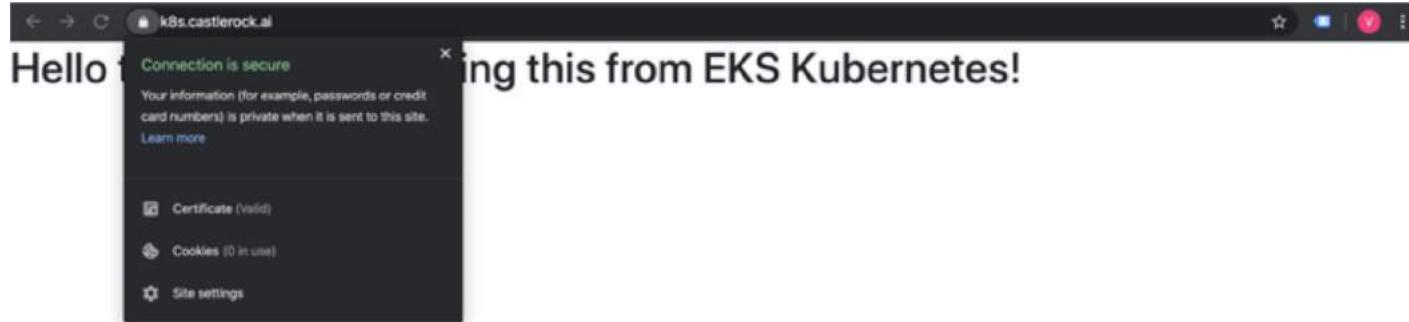
```
kubectl apply -f staging-issuer.yaml  
kubectl describe clusterissuer letsencrypt-staging
```

```
kubectl apply -f prod-issuer.yaml  
kubectl describe clusterissuer letsencrypt-prod
```

Apply to ngx

Edit the Ingress, or apply new definition

```
kubectl apply -f ngx-ing.yaml
```



See: <https://cert-manager.io/docs/tutorials/acme/ingress/>

“Nothing lasts forever,
Even cold November Rain”

- W. Axl Rose



Showing Ephemerality

```
$ cd segment05-applications
```

```
$ kubectl cp webpage/index.html ngx-5f858c479c-hgmqp:/usr/share/nginx/html/
```

Look at the page: <https://k8s.castlerock.ai>

Hello there! We are serving this from EKS Kubernetes!

Kill the pod on another screen

```
$ kubectl delete pod ngx-5f85...
```

Look at the page again <https://k8s.castlerock.ai>

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

Storage Classes

EBS

volume	gp2	io1	st1	stc1
Description	SSD	SSD fast!	HDD	HDD slow
Use Case	General workloads	Critical, Databases, etc	Big Data, log processing, warehouses	Large volumes infrequently accessed

EFS

FSx for Lustre



Examine Storage classes

- `kubectl edit sc gp2`

```
5 apiVersion: storage.k8s.io/v1
6 kind: StorageClass
7 metadata:
8   annotations:
9     kubectl.kubernetes.io/last-applied-configuration: |
10       {"apiVersion":"storage.k8s.io/v1","kind":"StorageClass"
11         "name":"gp2"}, "parameters": {"fsType": "ext4", "type": "gp2"}, "pro
12       storageclass.kubernetes.io/is-default-class: "true"
13   creationTimestamp: "2020-02-13T17:25:34Z"
14   name: gp2
15   resourceVersion: "169"
16   selfLink: /apis/storage.k8s.io/v1/storageclasses/gp2
16   uid: d7ae0bf8-4e85-11ea-8bba-0a0a49091766
17 parameters:
18   fsType: ext4
19   type: gp2
20 provisioner: kubernetes.io/aws-ebs
21 reclaimPolicy: Delete
22 volumeBindingMode: WaitForFirstConsumer
```

Persistent Volume Claim

How we access

How much storage we want

Storage Class we are requesting from

```
5 apiVersion: v1
6 kind: PersistentVolumeClaim
7 metadata:
8   name: nginx-pv-claim
9   labels:
10    app: nginx
11 spec:
12   accessModes:
13     - ReadWriteOnce
14   resources:
15     requests:
16       storage: 5Gi
17   storageClassName: gp2
```

Persistent Volume in our Deployment

Append to our existing pod definition

```
kubectl apply -f ngx-volumes.yaml
```

```
21 apiVersion: extensions/v1beta1
22 kind: Deployment
23 metadata:
24   labels:
25     run: ngx
26   name: ngx
27   namespace: default
28 spec:
29   replicas: 1
30   selector:
31     matchLabels:
32       run: ngx
33   template:
34     metadata:
35       labels:
36         run: ngx
37     spec:
38       containers:
39         - image: nginx
40           imagePullPolicy: Always
41           name: ngx
42           volumeMounts:
43             - name: vol
44               mountPath: /usr/share/nginx/html/
45   volumes:
46     - name: vol
47       persistentVolumeClaim:
48         claimName: ngx-pv-claim
```





Kubernetes on AWS

Segment 6: Administrative Tasks

Big Picture

- Cluster Autoscaling
- Horizontal Pod Autoscaling
- Kubernetes Dashboard
- Add additional Users
- Cluster Upgrades

Autoscaling

- Cluster Autoscaling
 - Update the number of nodes in the Kubernetes cluster
- Horizontal Pod Autoscaling
 - Update the number of pods running in a cluster
- Vertical Pod Autoscaling
 - Update the CPU and memory reservations for the pods

Cluster Autoscaling

- Create policy for nodes to autoscale
- Apply policy to node role that servers are under
- Install Metrics Server
- Install Autoscaler
- Configure Autoscaler (give it the right cluster name)
- Scale Demo

Cluster Autoscaling

Create policy

1 2

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

Visual editor

JSON

Import managed policy

```
1 * {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Action": [
6                 "autoscaling:DescribeAutoScalingGroups",
7                 "autoscaling:DescribeAutoScalingInstances",
8                 "autoscaling:DescribeLaunchConfigurations",
9                 "autoscaling:DescribeTags",
10                "autoscaling:SetDesiredCapacity",
11                "autoscaling:TerminateInstanceInAutoScalingGroup",
12                "ec2:DescribeLaunchTemplateVersions"
13            ],
14            "Resource": "*",
15            "Effect": "Allow"
16        }
17    ]
18 }
```

Note: This policy was already created by Terraform

Character count: 356 of 6,144.

Cancel

Review policy

Cluster Autoscaling

Create policy

Review policy

Name* EKSClusterAutoscaling
Use alphanumeric and '+', '=', '-' characters. Maximum 128 characters.

Description For nodes to Autoscale
Maximum 1000 characters. Use alphanumeric and '+', '=', '-' characters.

Summary

Service	Access level	Resource	Request condition
Allow (2 of 222 services) Show remaining 220			
EC2	Limited: List	All resources	None
EC2 Auto Scaling	Full: Read Limited: List, Write	All resources	None

Note: This policy was already created by Terraform

* Required

Cancel Previous Create policy

Cluster Autoscaling Role Policy

Roles > eksctl-aug05-nodegroup-standard-w-NodeInstanceRole-8XGWUSFCR8N

Delete role

Summary

Policy EKSClusterAutoscaling has been attached for the eksctl-aug05-nodegroup-standard-w-NodeInstanceRole-8XGWUSFCR8N.

Role ARN	arn:aws:iam::188966951897:role/eksctl-aug05-nodegroup-standard-w-NodeInstanceRole-8XGWUSFCR8N
Role description	Edit
Instance Profile ARNs	arn:aws:iam::188966951897:instance-profile/eks-22b9e23f-df85-e10b-797d-bf0bddacdc81
Path	/
Creation time	2020-08-06 08:59 PDT
Last activity	2020-08-06 13:41 PDT (Today)
Maximum session duration	1 hour Edit

Permissions Trust relationships Tags Access Advisor Revoke sessions

▼ Permissions policies (4 policies applied)

[Attach policies](#)

[+ Add inline policy](#)

Policy name	Policy type	
▶ AmazonEKSWorkerNodePolicy	AWS managed policy	X
▶ AmazonEC2ContainerRegistryReadOnly	AWS managed policy	X
▶ AmazonEKS_CNI_Policy	AWS managed policy	X
▶ EKSClusterAutoscaling	Managed policy	X

Install Cluster Autoscaler

- **Install Metrics Server**

```
cd segment06-admin/  
kubectl apply -f metrics-server-0.3.6/components.yaml
```

- **Install Cluster Autoscaler**

```
kubectl apply -f  
https://raw.githubusercontent.com/kubernetes/autoscaler/master/cluster-autoscaler/cloudprovider/aws/examples/cluster-autoscaler-autodiscover.yaml
```

- **Allow Annotation**

```
kubectl -n kube-system annotate deployment.apps/cluster-autoscaler cluster-autoscaler.kubernetes.io/safe-to-evict="false"
```

Edit Autoscaler configuration

```
kubectl -n kube-system edit \
deployment.apps/cluster-autoscaler
```

Update with:

...

- --node-group-auto-discovery=asg:tag=k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/dec07
 - --skip-nodes-with-system-pods=false
 - --balance-similar-node-groups
- ...

Test Cluster Autoscaling

- Check nodes

```
kubectl get nodes
```

- Create Deployment

```
kubectl create deployment autoscaler-demo --image=nginx
```

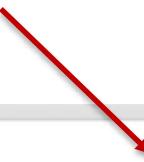
- Scale Deployment

```
kubectl scale deployment autoscaler-demo --replicas=50
```

- Check nodes and logs

```
kubectl get nodes
```

```
kubectl logs -f -n kube-system cluster-autoscaler<tab>
```



<input type="checkbox"/>	i-016f8ba85375ef7df	t3.medium	us-west-2b	● running	hourglass Initializing	None
<input type="checkbox"/>	i-03c50d217e45c4b62	t3.medium	us-west-2d	● running	checkmark 2/2 checks ...	None

Horizontal Pod Autoscaler (HPA)

- Ensure that Metrics Server is deployed
- Create test deployment
- Attach Autoscaler
- Scale load

Add Resource Requests and Limits

- $100m = 100$ millicores
- Limit is the most we will allow for this container (0.2 CPU)

```
5 apiVersion: extensions/v1beta1
6 kind: Deployment
7 metadata:
8   labels:
9     app: hpa-demo
10    name: hpa-demo
11 spec:
12   replicas: 1
13   selector:
14     matchLabels:
15       app: hpa-demo
16   template:
17     metadata:
18       labels:
19         app: hpa-demo
20   spec:
21     containers:
22       - image: httpd
23         imagePullPolicy: Always
24         name: hpa-demo
25         ports:
26           - containerPort: 80
27             protocol: TCP
28         resources:
29           requests:
30             cpu: "100m"
31           limits:
32             cpu: "200m"
```

Add HPA Policy

- Max replicas 10
- Min replicas 1
- Target 50% CPU utilization
 - (100m = 50m)

```
48 apiVersion: autoscaling/v1
49 kind: HorizontalPodAutoscaler
50 metadata:
51   name: hpa-demo
52 spec:
53   maxReplicas: 10
54   minReplicas: 1
55   scaleTargetRef:
56     apiVersion: apps/v1
57     kind: Deployment
58     name: hpa-demo
59   targetCPUUtilizationPercentage: 50
```

HPA – Create and Test

```
kubectl create -f hpa-demo.yaml
```

```
kubectl run apache-bench -i --tty \  
  --rm --image=httpd -- ab -n 500000 \  
  -c 1000 http://hpa-demo.default.svc.cluster.local/
```

vallards-MacBook-Pro:segment06-admin vallard\$ kubectl get hpa -w						
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
hpa-demo	Deployment/hpa-demo	198%/50%	1	10	1	116s
hpa-demo	Deployment/hpa-demo	198%/50%	1	10	4	2m1s
hpa-demo	Deployment/hpa-demo	155%/50%	1	10	4	2m47s
hpa-demo	Deployment/hpa-demo	155%/50%	1	10	8	3m2s

Kubernetes Dashboard

- **Deploy:**

```
kubectl apply -f segment06-admin/dashboard.yaml
```

- **Create Service Account**

```
kubectl apply -f segment06-admin/ eks-admin-service-account.yaml
```

- **Get Credentials**

- ```
kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep eks-admin | awk '{print $1}')
```

- **Connect**

- ```
kubectl proxy
```

Kubernetes Dashboard

- Create the dashboard and service account

```
cd dashboard/
```

```
kubectl apply -f dashboard.yaml
```

```
kubectl apply -f eks-admin-service-account.yaml
```

Get Token:

- `kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep eks-admin | awk '{print $1}')`

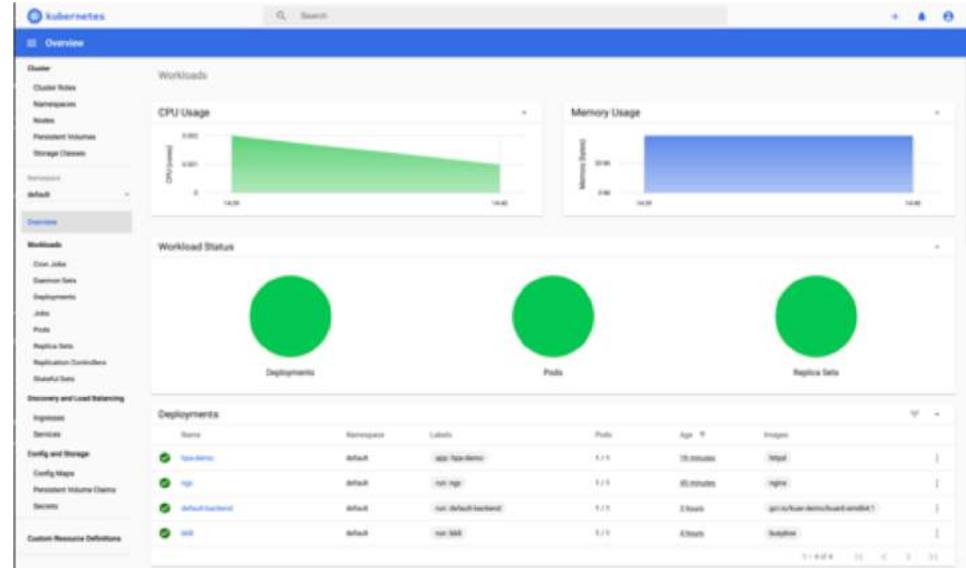
```
Data
-----
ca.crt:    1025 bytes
namespace: 11 bytes
token:     eyJhbGciOiJSUzI1NiIiImtpZC16IiJ9.eyJpc3MiOiJrdWJlcms5LdGVzL3NlcnZpY2VhY2NvdW50Iiwtaz3VizXJuZXRlcyc5pbby9zXJZaWNLYWNjb3VudC9uYW1lc3BhY2U1OiJrdWJlLXNsC3R1bSISImt1YmVybV0ZXMuw8vc2VydmljZWfjY291bnQvc2VjcmV0Lm5hbWUiOj1la3MtYWRtaW4tdG9rZW4tbXhranMlCJrdWJlcms5LdGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoizWtzLWFkbWluIiwiO3ViZXJuZXRlcyc5pbby9zXJZaWNLYWNjb3VudC9zZXJ2aWN1LWFjY291bnQudWlkIjoiyM0ZDNhZmMtMTY2ZC00ZTNmLTk00WUtZDFiNhMSNjZiMjBmIwic3ViIjoic3LzdGVtOnNlcnZpY2VhY2NvdW500mt1YmUtc3LzdGVt0mVrcy1hZG1pbij9.Sh_sJKuKnKC5V_YV378y1roCLUpAhxR778m_vlXQsCEeqUI-21ERAhDBlj4UY58GaaXXAII9xLbZDDKzz10wsZrbIHdEQUIBkkaJBtBTIK5CULvXVJra3ZIjlvrhJtJVbFGBCZ-hYmRSN9hZkQz7hRBBLDIVo-o1j5Av58TaWHXNH3Flux-7pWy_h9XUXl9bimbHHKt7gzQeMw2Rqy7ybU5ecAJBKXji9l5rtr1FNwGm9I6VV_70whp9aSQTzDWAEFwmA8gE0nNu3-haaFlhL1yOGjTBhVoJdQbs47gX6RYB3GoRJKvLWoZ7pu0kaEwtvmTPwezSjZA139c20Kg
```

Access Dashboard

- Localhost via Proxy:

<http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy#!/login>

- May take a few minutes before metrics appear.



Add another AWS User

- By default person who created cluster is the only person who has access to it.
- Other users must authenticate with AWS before getting authorization to run kubectl commands. (Might take 15 min)
- Give other user EKS group name to add permissions
- Other user runs:

```
aws eks update-kubeconfig --name aug05
```

```
aws sts get-caller-identity
```

- Sends caller identity to cluster owner:

```
root@walrus:~# aws sts get-caller-identity
```

```
188966951897 arn:aws:iam::188966951897:user/anotherUser
```

```
AIDASX7203PMVFHROSQHV
```

Add another AWS User

```
root@walrus:~# kubectl get pods  
error: You must be logged in to the server (Unauthorized)
```

- Is logged into AWS but not to Kubernetes
- From EKS cluster creator account run:

```
kubectl get cm -n kube-system
```

```
kubectl edit cm -n kube-system aws-auth
```

```
mapUsers: |  
  - userarn: arn:aws:iam::254056899943:user/judy  
    username: judy  
    groups:  
      - system:masters
```

Limiting Scopes of users

- Create a Role and a Role Binding
- Add User to Role Group in aws-auth Config Map
- Group is “read-pods-role” instead of previous “system:masters”

<https://www.freecodecamp.org/news/adding-limited-access-iam-user-to-eks-cluster/>

```
1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: Role
3 metadata:
4   name: read-pods-role
5 rules:
6 - apiGroups: []
7   resources: [pods]
8   verbs: [list]
9 ---
10 apiVersion: rbac.authorization.k8s.io/v1
11 kind: RoleBinding
12 metadata:
13   name: read-pods-role-binding
14 subjects:
15 - kind: User
16   name: read-pods
17   apiGroup: rbac.authorization.k8s.io
18 roleRef:
19   kind: Role
20   name: read-pods-role
21   apiGroup: rbac.authorization.k8s.io
```

Cluster Upgrades

- Upgrade often, check every 3 months
- Can only migrate one minor version at a time:
 - 1.12 -> 1.13 -> 1.14 -> 1.15 -> 1.16 -> 1.17-> 1.18
- eksctl is easy if you created cluster via eksctl
- Console upgrade process also somewhat simple
 - Upgrade control panel in EKS console
 - Upgrade nodes with cloudformation template



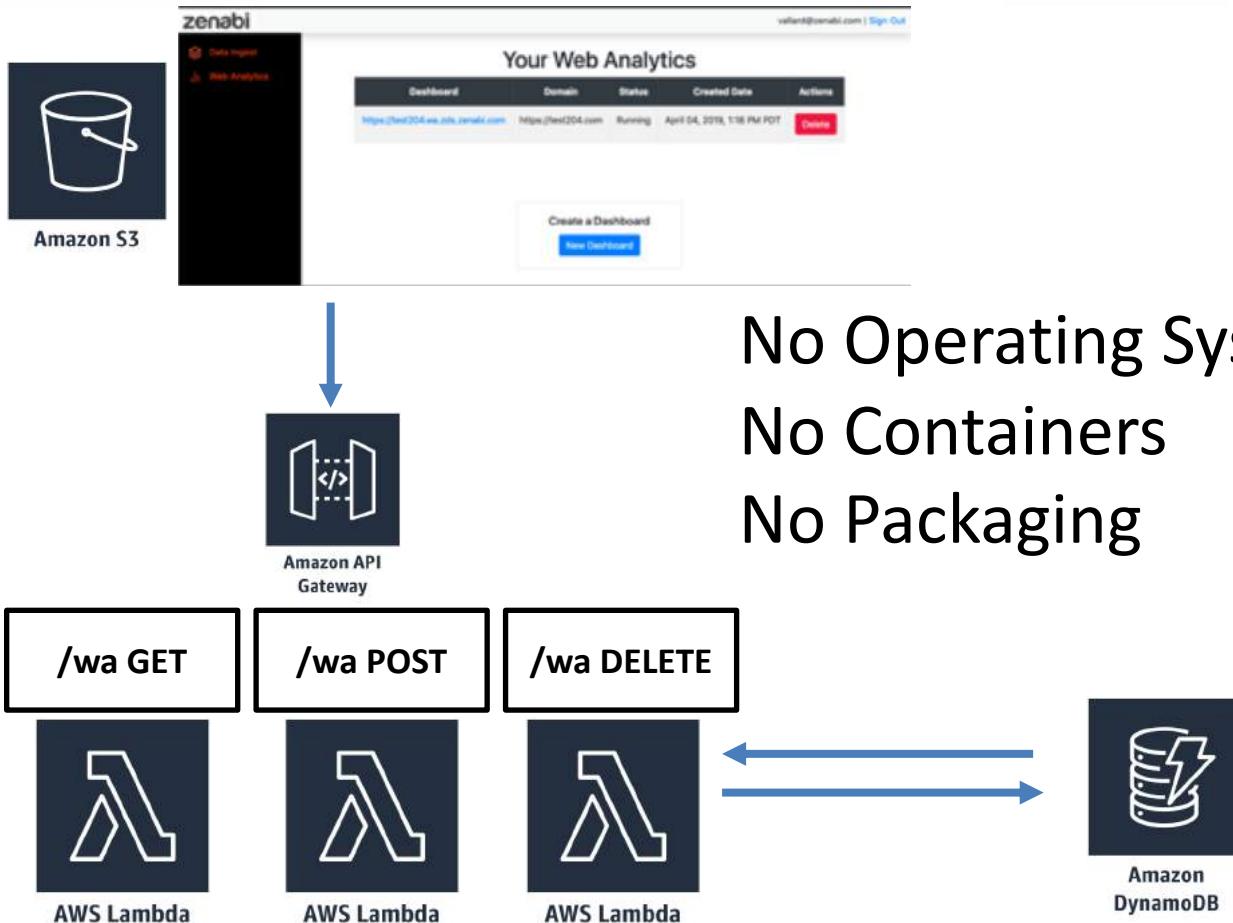
Kubernetes on AWS

Segment 7: Integrations with other AWS Services

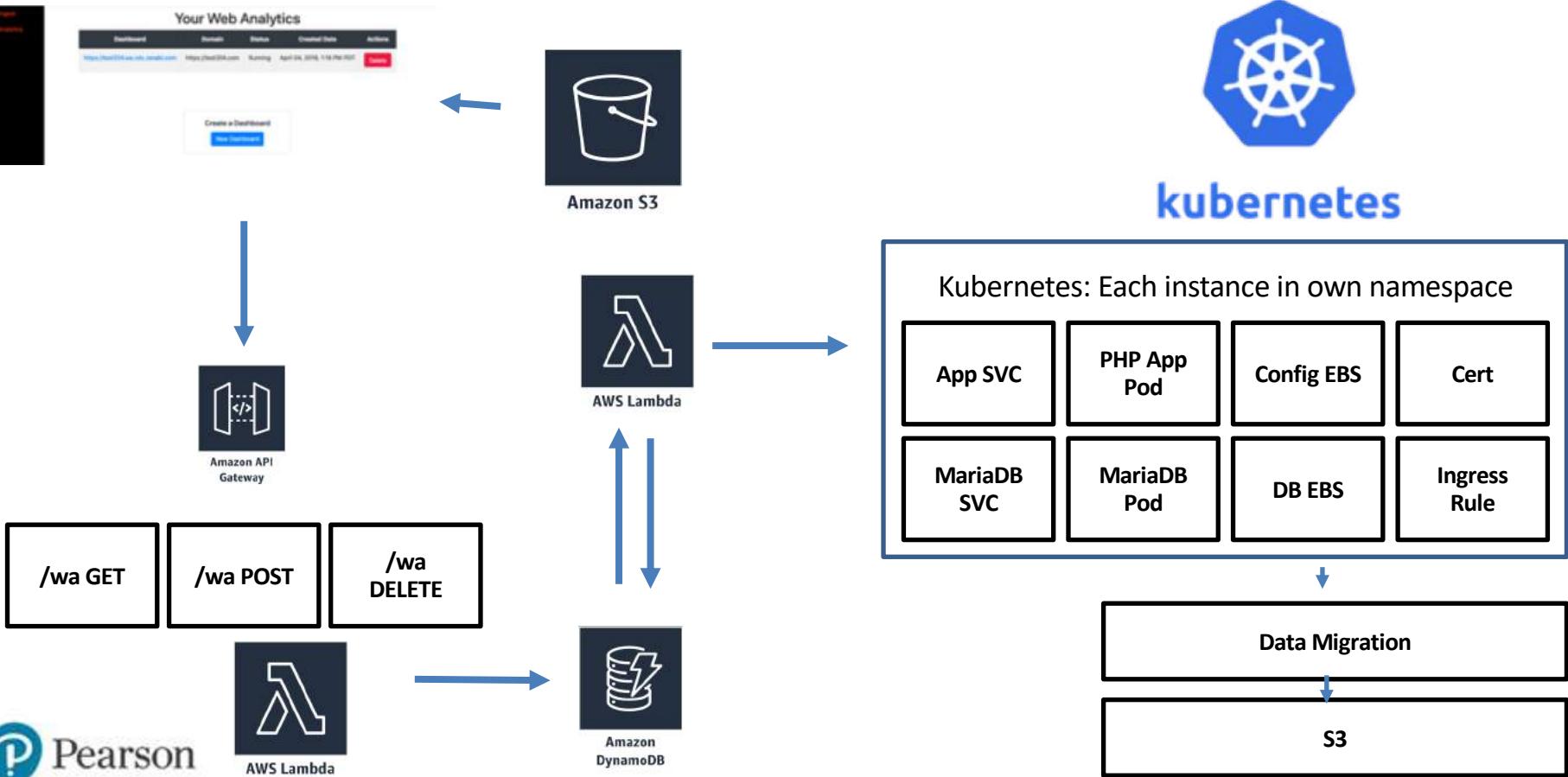
Big Picture

- Serverless
- Accessing resources from Kubernetes pods

Serverless Architecture

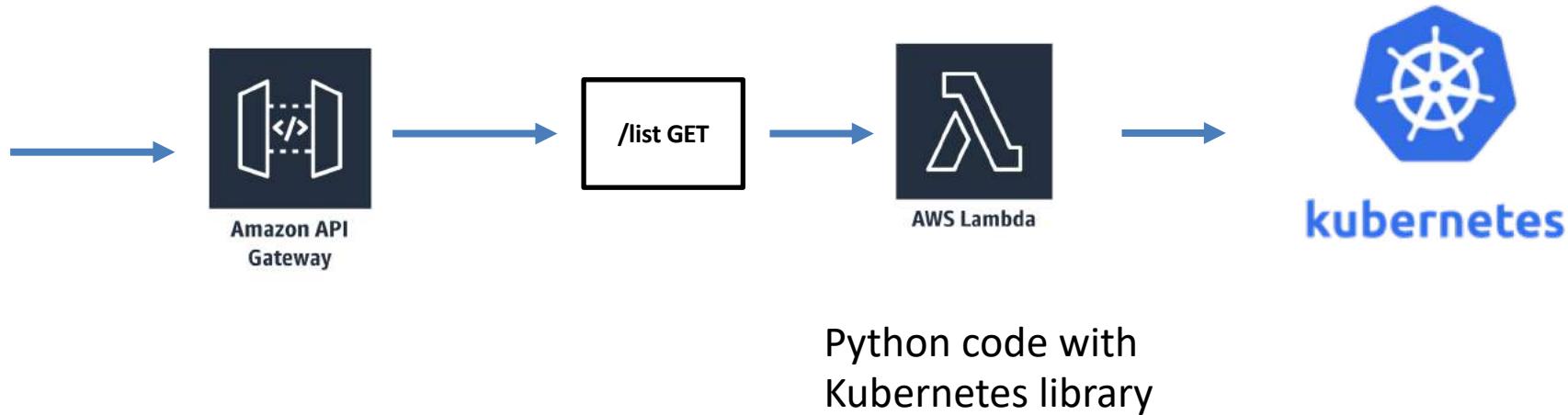


Expanding into EKS



Sample Application

- List the pods in your system



Create Serverless Policy

Add Serverless Policy to EKSDemoGroup

Policy ARN arn:aws:iam::188966951897:policy/Serverless 

Description Ability to create Serverless Functions

Permissions Policy usage Policy versions Access Advisor

Policy summary {} JSON Edit policy ?

Filter

Service	Access level	Resource	Request condition
Allow (3 of 222 services) Show remaining 219			
API Gateway	Full access	All resources	None
CloudWatch Logs	Full access	All resources	None
Lambda	Full access	All resources	None

Kube Lambda Role

Create role

1 2 3 4

Review

Provide the required information below and review this role before you create it.

Role name*

kubeLambda

Use alphanumeric and '+=,.@-_ ' characters. Maximum 64 characters.

Role description

Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=,.@-_ ' characters.

Trusted entities AWS service: lambda.amazonaws.com

Policies Policies not attached

Permissions boundary Permissions boundary is not set

Note: This policy was already created by Terraform

Add Policies to Role

Role ARN arn:aws:iam::188966951897:role/kubeLambda [Edit](#)

Role description Allows Lambda functions to call AWS services on your behalf. [Edit](#)

Instance Profile ARNs [Edit](#)

Path /

Creation time 2020-02-21 16:06 PST

Last activity 2020-02-21 16:47 PST (Today)

Maximum CLI/API session duration 1 hour [Edit](#)

[Permissions](#) [Trust relationships](#) [Tags \(1\)](#) [Access Advisor](#) [Revoke sessions](#)

▼ Permissions policies (3 policies applied)

[Attach policies](#) [Add inline policy](#)

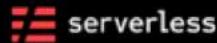
Policy name ▾	Policy type ▾	
▶ AWSLambdaBasicExecutionRole	AWS managed policy	X
▶ EKSFullAccess	Managed policy	X
▶ Serverless	Managed policy	X

Note: This was already created by Terraform

Add to aws-auth ConfigMap

```
7 mapRoles: |  
8   - groups:  
9     - system:bootstrappers  
10    - system:nodes  
11    rolearn: arn:aws:iam::188966951897:role/eksctl-eksctl-2-18-nodegroup-stan-NodeInstanceRole-1H3V6PKC40V5T  
12    username: system:node::{EC2PrivateDNSName}  
13  - rolearn: arn:aws:iam::188966951897:role/kubeLambda  
14    username: kubeLambda  
15    groups:  
16      - system:masters  
17 mapUsers: |  
18  - userarn: arn:aws:iam::188966951897:user/anotherUser  
19    username: read-pods  
20    groups:  
21      - read-pods
```

Serverless



Product Docs Pricing Learn Services More | Contact Sales Sign-in Sign-Up Free



The complete solution for building & operating serverless applications

- sls deploy
- sls invoke local -f list
- sls deploy -f list
- sls invoke -f list
- sls logs -f list
- sls remove

```
curl https://0qtyabihgc.execute-api.us-west-2.amazonaws.com/dev/list | jq
```

Considerations

- This is **wide open**, we would typically add authentication (Cognito) so not everyone could access and list our pods in our cluster.
- <https://serverless-stack.com> for inspiration
- Creating Batch, CronJobs may be cheaper than running on servers
- Scale up / Scale Down with HPA and Cluster Autoscaling makes this a good play.

Accessing AWS Resources in Pods

- One of the most valuable reasons to run EKS is access to other AWS resources
- S3, DynamoDB, other databases, Redshift, are all accessible from your applications that run in pods
- Usually IAM policies and roles are the only tricky parts to worry about when writing these applications.

Add permissions to Node Role

Create role

Delete role



Search

Showing 7 results

Role name ▾

Trusted entities

Last activity ▾

<input type="checkbox"/> AWSServiceRoleForAmazonEKSNodegroup	AWS service: eks-nodegroup (Service-Linked role)	Today
<input type="checkbox"/> AWSServiceRoleForAutoScaling	AWS service: autoscaling (Service-Linked role)	Today
<input type="checkbox"/> AWSServiceRoleForElasticLoadBalancing	AWS service: elasticloadbalancing (Service-Linked role)	Today
<input type="checkbox"/> AWSServiceRoleForSupport	AWS service: support (Service-Linked role)	None
<input type="checkbox"/> AWSServiceRoleForTrustedAdvisor	AWS service: trustedadvisor (Service-Linked role)	None
<input type="checkbox"/> eksctl-eksctl-2-23-cluster-ServiceRole-9I2OH2J...	AWS service: eks	Today
<input type="checkbox"/> eksctl-eksctl-2-23-nodegroup-stan-NodeInstan...	AWS service: ec2	Today

Add DynamoDB Policy to node Role

▼ Permissions policies (4 policies applied)

Policy name	Policy type	
▶ AmazonEKSWorkerNodePolicy	AWS managed policy	✖
▶ AmazonEC2ContainerRegistryReadOnly	AWS managed policy	✖
▶ AmazonDynamoDBFullAccess	AWS managed policy	✖
▶ AmazonEKS_CNI_Policy	AWS managed policy	✖

Now applications running in the node get these permissions

Add DynamoDB Policy to node Role

```
1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Sid": "VisualEditor0",
6              "Effect": "Allow",
7              "Action": [
8                  "dynamodb:GetShardIterator",
9                  "dynamodb:Scan",
10                 "dynamodb:DescribeStream",
11                 "dynamodb:GetRecords"
12             ],
13             "Resource": "arn:aws:dynamodb:us-east-1:254056899943:table/zds/*"
14         },
15         {
16             "Sid": "VisualEditor1",
17             "Effect": "Allow",
18             "Action": "dynamodb>ListStreams",
19             "Resource": "arn:aws:dynamodb:us-east-1:254056899943:table/zds/*"
20         }
21     ]
22 }
```

Make a custom policy to limit access to only certain DBs

Create a Dynamo DB table: dynamoUsers

Create DynamoDB table

Tutorial 

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* 

Primary key* Partition key

Add sort key

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
- Encryption at Rest with DEFAULT encryption type.

 You do not have the required role to enable Auto Scaling by default.
Please refer to documentation.

+ Add tags 

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

[Cancel](#) [Create](#)

Add a Dynamo Entry to dynamoUsers

dynamoUsers [Close](#)

[Overview](#) **Items** [Metrics](#) [Alarms](#) [Capacity](#) [Indexes](#) [Global Tables](#)

[Create item](#) [Actions ▾](#)

Scan: [Table] dynamoUsers: id ^

Scan [Add filter](#)

[Start search](#)

	id	email	name ⓘ
<input type="checkbox"/>	1234	vallard@castlerock.ai	Vallard Benincosa

Deploy Sample Application

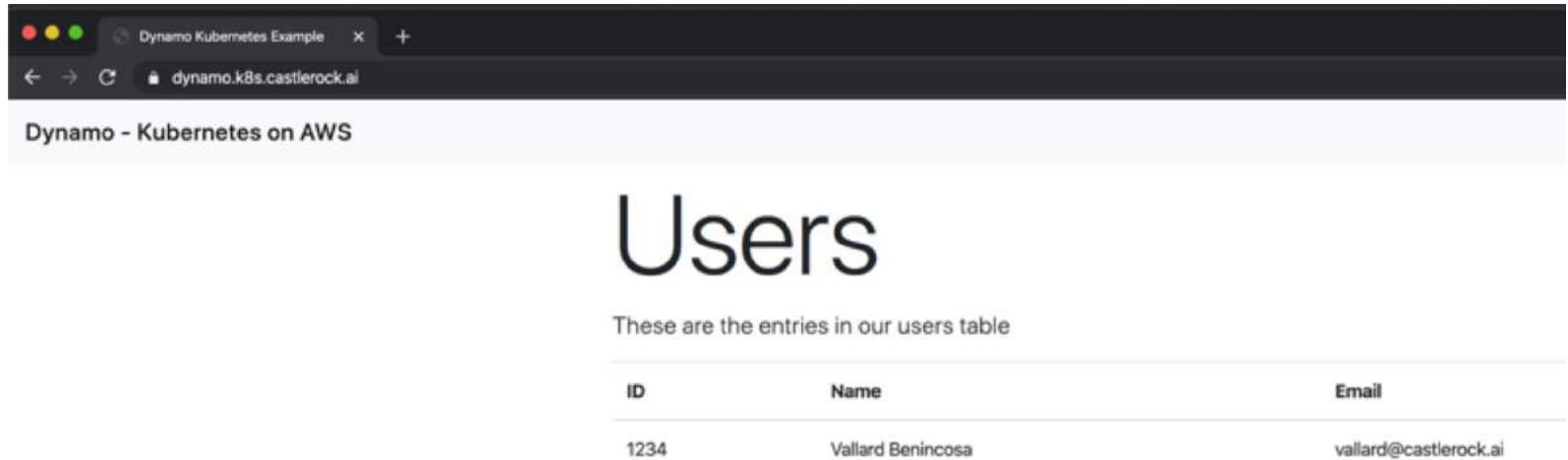
- Build the Application: `make build`
- Put application in ECR if not created yet

The screenshot shows the AWS ECR (Amazon Elastic Container Registry) interface. At the top, there's a breadcrumb navigation: 'ECR > Repositories'. Below that is a search bar labeled 'Find repositories'. A table lists the repository details:

Repository name	URI	Created at	Tag immutability	Scan on push	Encryption type
vallard/eks-dynamo	188966951897.dkr.ecr.us-west-2.amazonaws.com/vallard/eks-dynamo	05/19/20, 02:12:46 PM	Disabled	Disabled	AES-256

- Push the application: `make push`
- Deploy: `kubectl apply -f eks-dynamo.yaml`

Refresh the Application



A screenshot of a web browser window titled "Dynamo Kubernetes Example". The address bar shows the URL "dynamo.k8s.castlerock.ai". The main content area displays the heading "Dynamo - Kubernetes on AWS" followed by a large "Users" title. Below it, a sub-heading states "These are the entries in our users table". A table is shown with three columns: "ID", "Name", and "Email". A single row of data is present, with values "1234", "Vallard Benincosa", and "vallard@castlerock.ai" respectively.

ID	Name	Email
1234	Vallard Benincosa	vallard@castlerock.ai



Kubernetes on AWS

Conclusion

What we covered! 😊

- Brief introduction to Kubernetes, Containers, EKS
- IAM Constructs and Command Line Tools
- Creating EKS Clusters: eksctl & Console
- Verifying and Testing our EKS cluster
- Running Applications in the real world
- Administrative Tasks
- Integrations with other AWS services

What we didn't cover 😞

- Multiple Availability Zones
- Windows Worker nodes
- Fargate for serverless cluster nodes
- App Mesh
- Deep Learning Containers with GPU Instances
- Not too much on Amazon ECR
- Other advanced customizations: Instances, Networking, Volumes



Thank you!

@vallard