# KUBERNETES NODES

The machines that make up a Kubernetes cluster are called **nodes**.

Nodes in a Kubernetes cluster may be physical, or virtual.

There are two types of nodes:

* A Master-node type, which makes up the [Control Plane](https://eksworkshop.com/introduction/architecture/architecture_control), acts as the “brains” of the cluster.
* A Worker-node type, which makes up the [Data Plane](https://eksworkshop.com/introduction/architecture/architecture_worker), runs the actual container images (via pods).

# K8S OBJECTS OVERVIEW

Kubernetes objects are entities that are used to represent the state of the cluster.

An object is a “record of intent” – once created, the cluster does its best to ensure it exists as defined. This is known as the cluster’s “desired state.”

Kubernetes is always working to make an object’s “current state” equal to the object’s “desired state.” A desired state can describe:

* What pods (containers) are running, and on which nodes
* IP endpoints that map to a logical group of containers
* How many replicas of a container are running

# K8S OBJECTS DETAIL

### [Pod](https://kubernetes.io/docs/concepts/workloads/pods/pod/)

* A thin wrapper around one or more containers

### [DaemonSet](https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/)

* Implements a single instance of a pod on a worker node

### [Deployment](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/)

* Details how to roll out (or roll back) across versions of your application

### [ReplicaSet](https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/)

* Ensures a defined number of pods are always running

### [Job](https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/)

* Ensures a pod properly runs to completion

### [Service](https://kubernetes.io/docs/concepts/services-networking/service/)

* Maps a fixed IP address to a logical group of pods

### [Label](https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/)

* Key/Value pairs used for association and filtering

# KUBERNETES ARCHITECTURE

In this section, we’ll cover the following topics:

* [Architectural Overview](https://eksworkshop.com/introduction/architecture/architecture_control_and_data_overview/)
* [Control Plane](https://eksworkshop.com/introduction/architecture/architecture_control/)
* [Data Plane](https://eksworkshop.com/introduction/architecture/architecture_worker/)
* [Kubernetes Cluster Setup](https://eksworkshop.com/introduction/architecture/cluster_setup_options/)

# ARCHITECTURAL OVERVIEW

# 

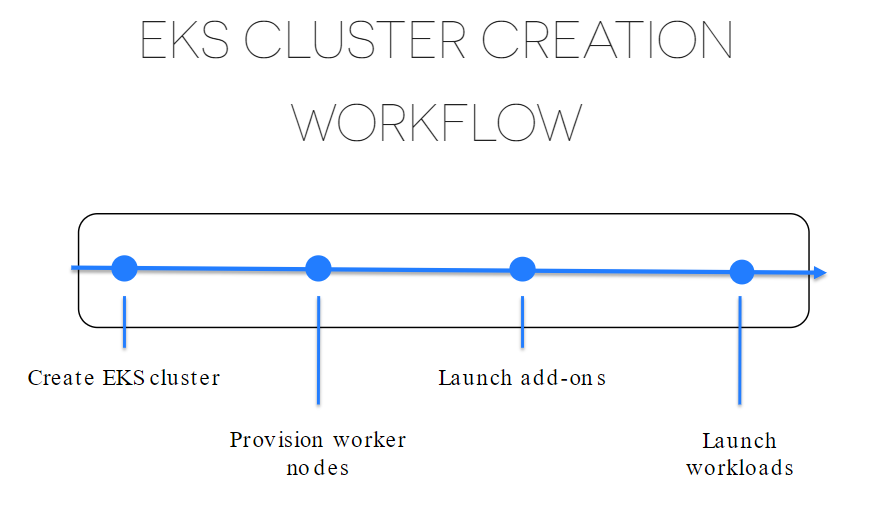
# 

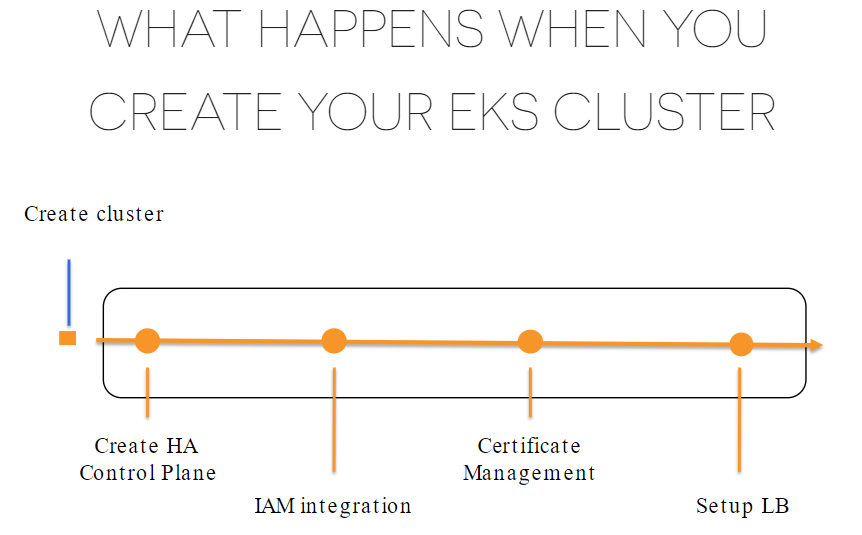
# 

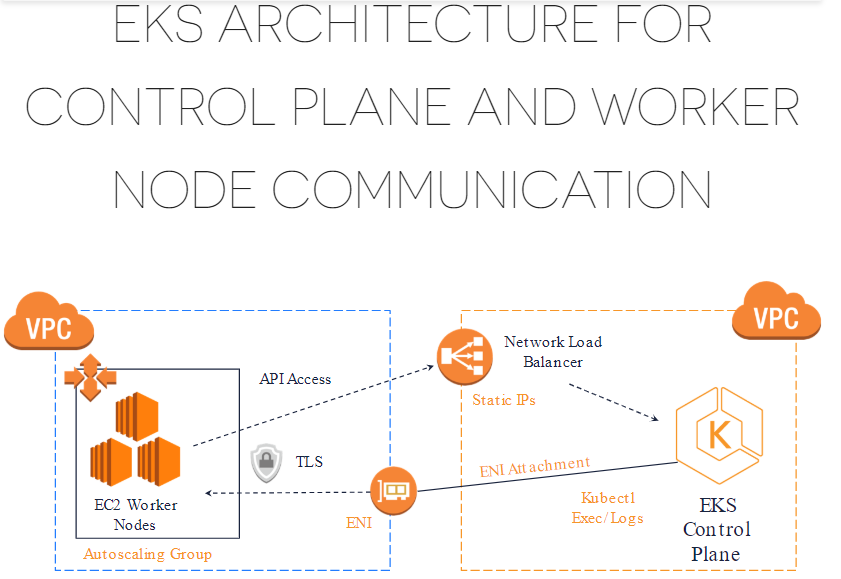
# KUBERNETES CLUSTER SETUP

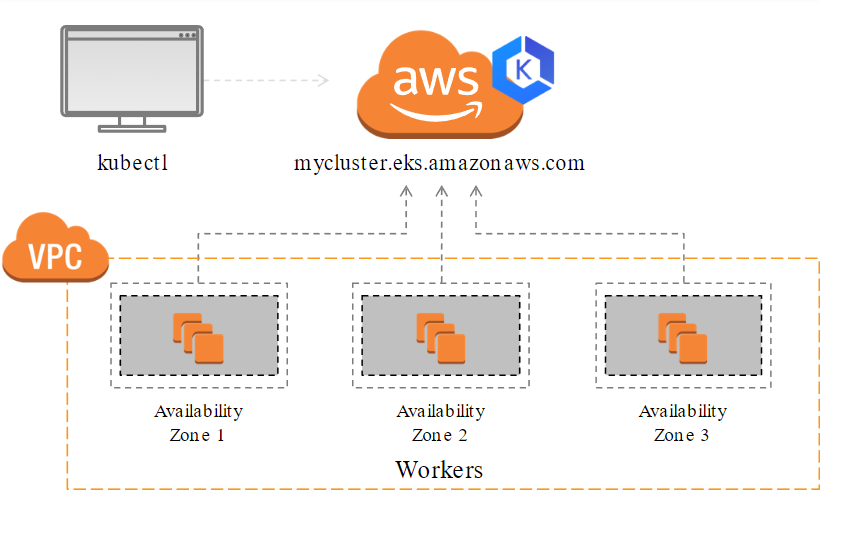
In addition to the managed Amazon EKS solution, there are many tools available to help bootstrap and configure a self-managed Kubernetes cluster. They include:

* [Minikube](https://kubernetes.io/docs/setup/minikube/) – Development and Learning
* [Kops](https://github.com/kubernetes/kops) – Learning, Development, Production
* [Kubeadm](https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/) – Learning, Development, Production
* [Docker for Mac](https://docs.docker.com/docker-for-mac/#kubernetes) - Learning, Development
* [Kubernetes IN Docker](https://github.com/kubernetes-sigs/kind) - Learning, Development

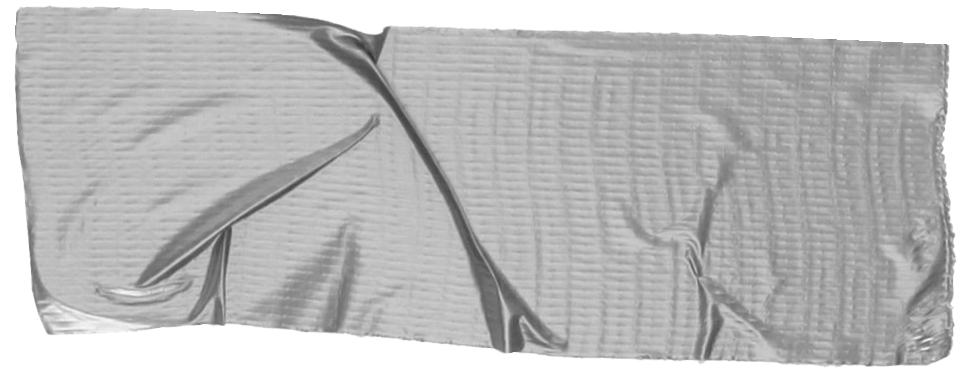
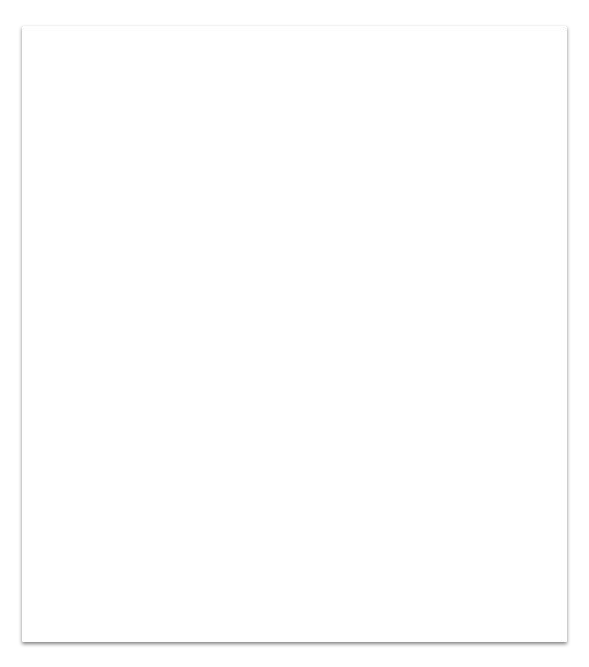








1. **Introduction to AWS EKS (Small Overview)**
2. **Set up EKS Clusters Using eksctl**
3. **Deploy MEAN Stack Web App using kubectl**
4. **Autoscaling: Horizontal Pod autoscaling and Cluster Nodes Autoscaling**
5. **Kubernetes Cluster Monitoring using Prometheus and Grafana**



### Contents

➔ [Introduction to AWS EKS](#_bookmark0)

**➔** [**Beneﬁts**](#_bookmark1)

**➔** [**Cost**](#_bookmark2)

**➔** [**AWS EKS Architecture**](#_bookmark3)

**➔** [**Kubernetes and VPC Networking**](#_bookmark4)

**➔** [**Companies Using AWS EKS**](#_bookmark5)

**➔** [**Let's Get Started & Few notes before that**](#_bookmark6)

**➔** [**Install Essential Tools**](#_bookmark7)

**➔** [**Create AWS EKS Cluster**](#_bookmark8)

**➔** [**Setting up Kubernetes Dashboard**](#_bookmark9)

**➔** [**Setting up Role Based Access Control**](#_bookmark10)

**➔** [**Build The Docker Image to Containerize MEAN App**](#_bookmark11)

**➔** [**Publishing Docker Image in AWS ECR**](#_bookmark12)

**➔** [**Finally Deploy The Application**](#_bookmark13)

**➔** [**Auto Scaling: Horizontal Pod Autoscaler (HPA)**](#_bookmark14)

**➔** [**Auto Scaling: Cluster Autoscaler (CA)**](#_bookmark15)

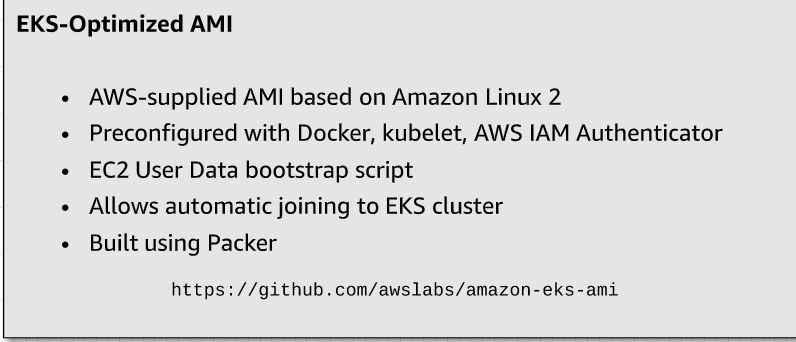
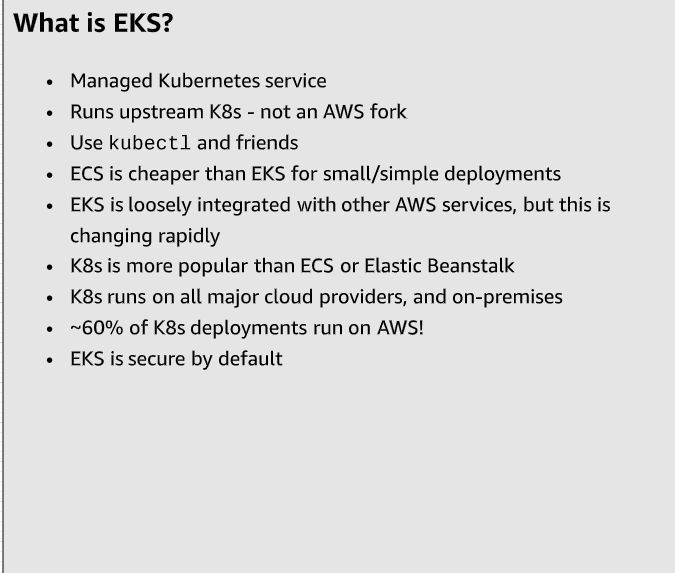
**➔** [**Monitoring EKS Cluster: Prometheus & Grafana**](#_bookmark16)

**➔** [**Final Grafana Dashboard Setup**](#_bookmark17)

**➔** [**Clean Up Everything**](#_bookmark18)

**➔** [**Few More thing You Must Learn**](#_bookmark19)

**➔** [**Contact me if any issue**](#_bookmark20)

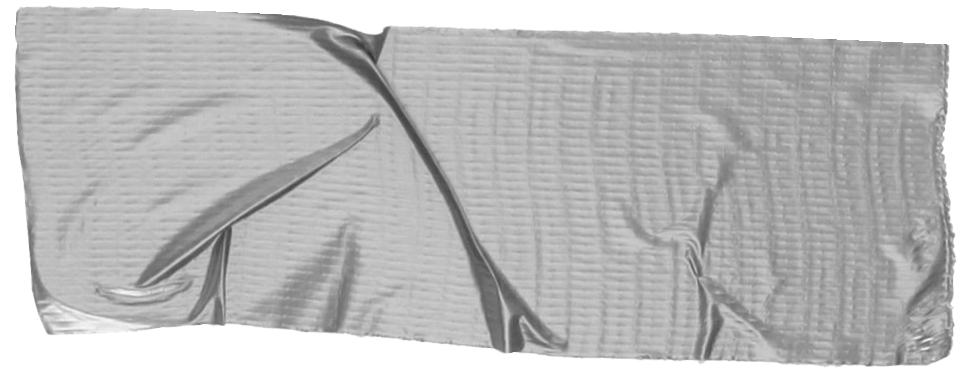
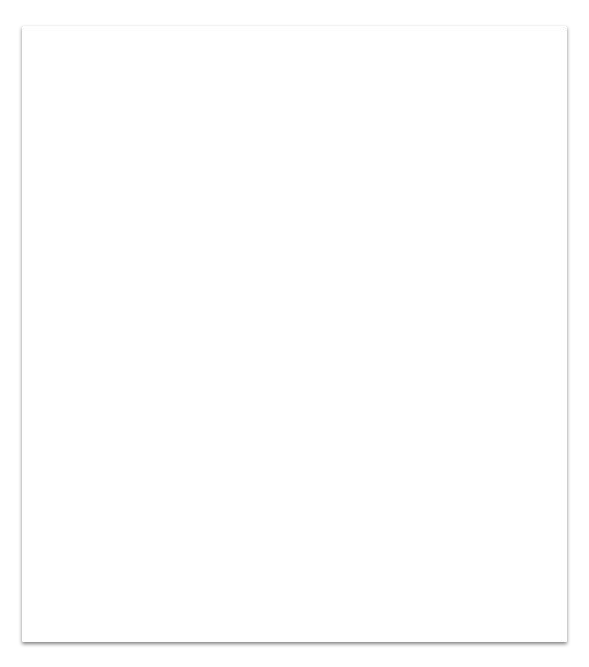


**AWS EKS**

Amazon Elastic Kubernetes Service (Amazon EKS) makes it easy to deploy, manage, and scale containerized applications using [Kubernetes on AWS](https://aws.amazon.com/kubernetes/).

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.



**Beneﬁts**

**Let’s discuss in short what features it provides**

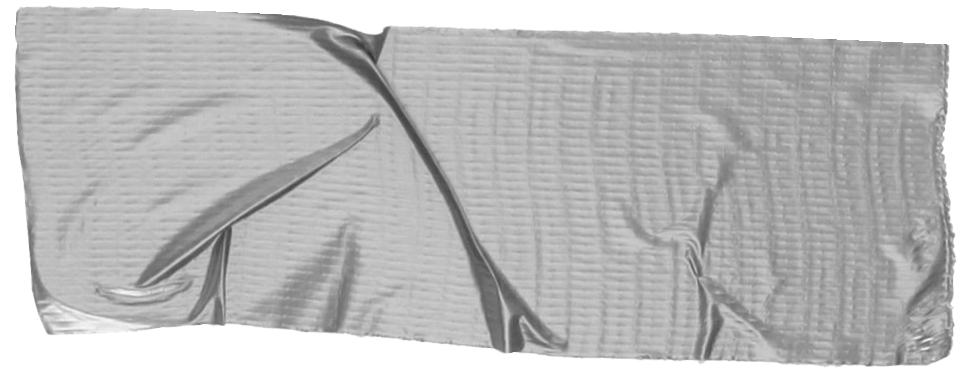
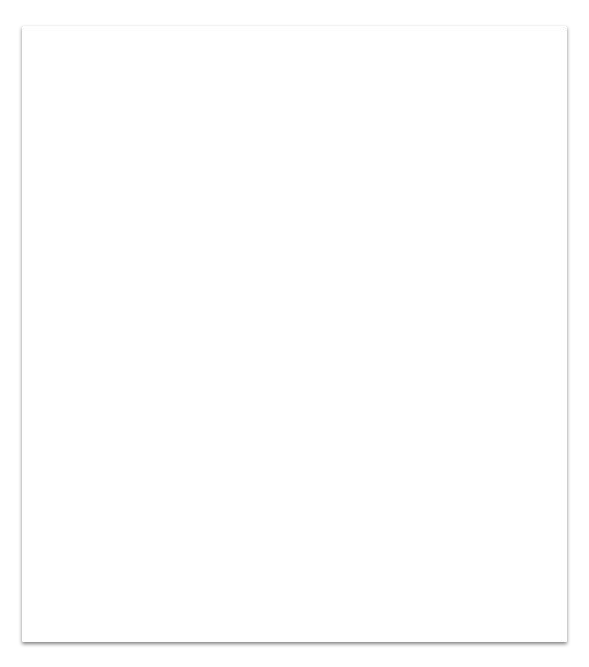
**➔ No Control plane to manage**

**➔ Secure by default**

**➔ Comformant and Compatible**

**➔ Optimized for cost**

**➔ Build with the community**



**Cost**

**Let’s discuss about the cost**

**➔ Each EKS Cluster Cost**

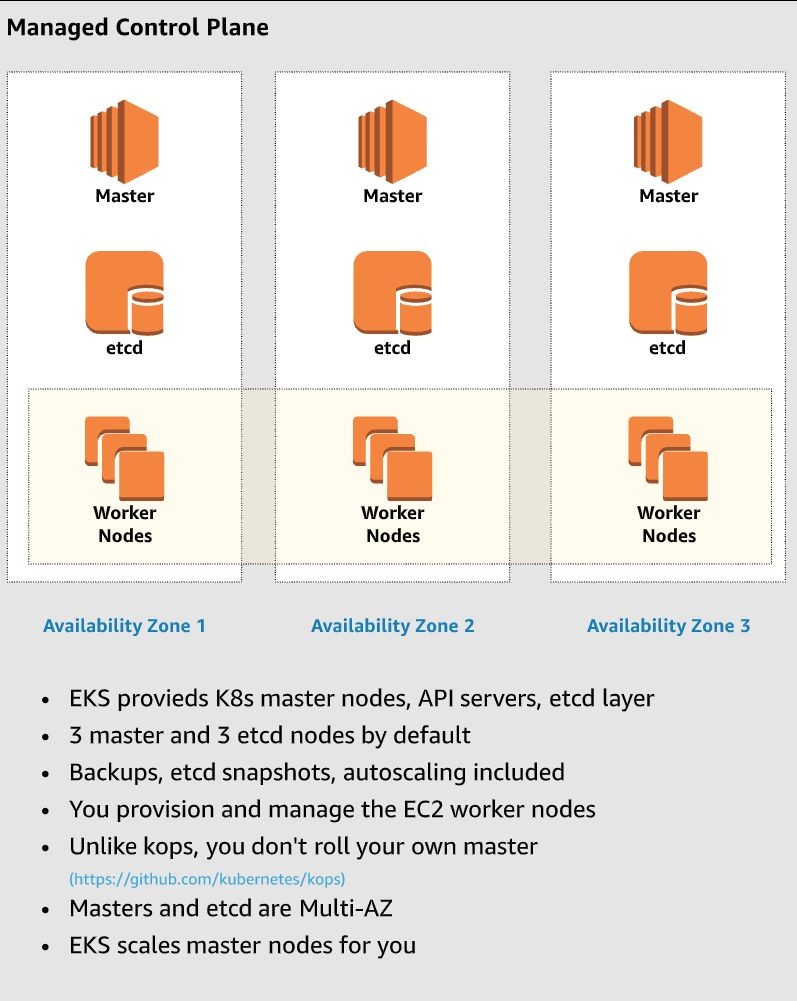
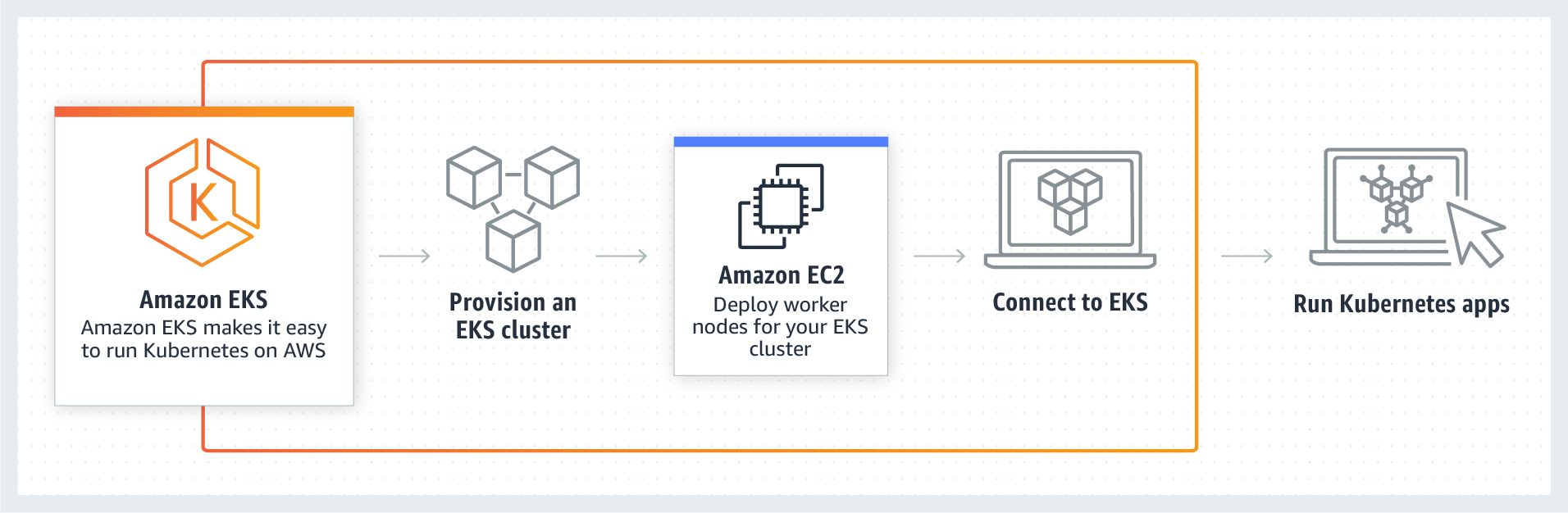
**$0.20/hour i.e. $144/month**

You can use a single Amazon EKS cluster to run multiple applications by taking advantage of Kubernetes namespaces and IAM security policies.

**➔ EC2 Instances & EBS volumes used in AS Worker Nodes**

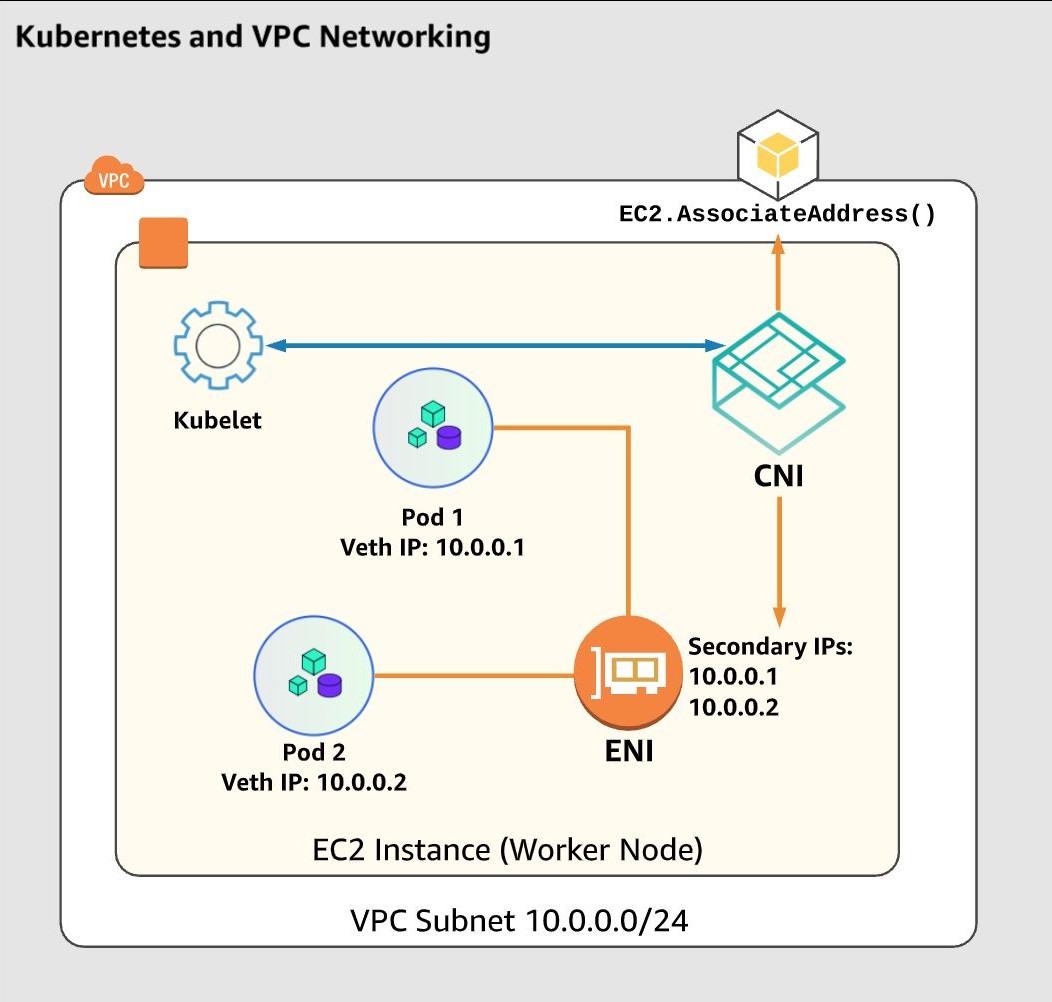
You pay for AWS resources , you create to run your Kubernetes worker nodes. You only pay for what you use, as you use it; there are no minimum fees and no upfront commitments.

**EKS**



**Architecture**

Kubernetes and VPC Networking



**Companies Using AWS EKS**



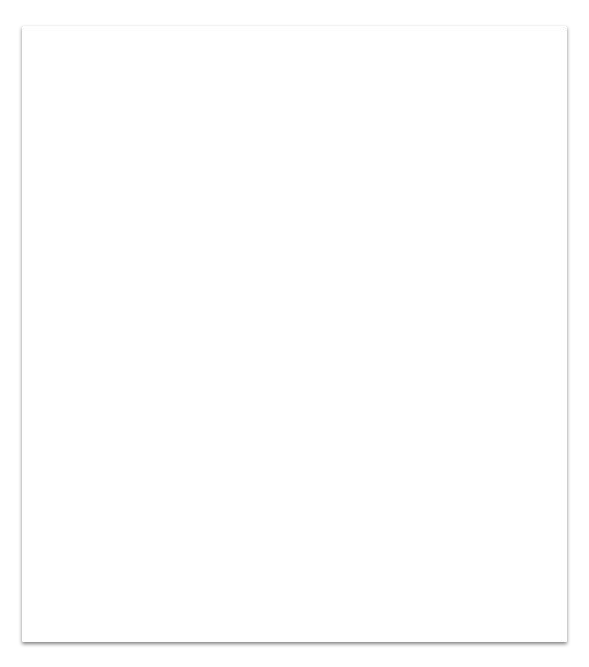
# Let’s Start an AWS EKS Cluster and Deploy A MEAN Stack Application

### For best security perspective, I will suggest utilizing Amazon Linux v2 micro instance with ec2 role and not using personal computer system for any kind of cluster operation or kubernetes related operations to avoid unwanted issues related to security ,project management or confusion / conﬂict with other projects . You should use personal computer for development purpose only.

**I will be using** [**Amazon Linux 2 EC**](https://aws.amazon.com/amazon-linux-2/) **instance with admin IAM role attached to it for entire demo. Don’t forget to assign administrator IAM role to the instance or if you are doing set-up in personal machine then make sure to conﬁgure aws cli locally.**

**Project git:** <https://github.com/cncfkol/mean_eks_demo>

### All Kubernetes related conﬁg ﬁles are in “k8s” folder



**Install Essential Tools**

##### We will need below tools to be installed

**➔ Install aws cli**

[https://docs.aws.amazon.com/cli/latest/userg](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html) [uide/cli-chap-install.html](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html)

**➔ Install kubectl**

[https://docs.aws.amazon.com/eks/latest/user](https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html) [guide/install-kubectl.html](https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html)

**➔ Install aws-iam-authenticator**

[https://docs.aws.amazon.com/eks/latest/user](https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html) [guide/install-aws-iam-authenticator.html](https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html)

**➔ Install eksctl**

[https://docs.aws.amazon.com/eks/latest/user](https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html) [guide/getting-started-eksctl.html](https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html) or <https://eksctl.io/introduction/installation/>

**Create AWS EKS Cluster**

Create Cluster:

**eksctl create cluster -f cluster.yaml**

**Then check for the message:**

**EKS cluster "<cluster name>" in "<aws region>" region is ready**

**Check that kubectl client get auto set properly or not by: cat /home/ec2-user/.kube/config**

If want to Delete Cluster anytime:

**eksctl delete cluster -f cluster.yaml**

Sample Cluster yml file: <https://gist.github.com/sd031/e72eb9f454340a7c844da31b97716e0a>

**Installing: Dashboard, Heapster, InﬂuxDb**

To Install Kubernetes Dashboard:

Check: <https://github.com/kubernetes/dashboard>

**Below are the commands:**

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml>

**Installing Heapster and InfluxDB**

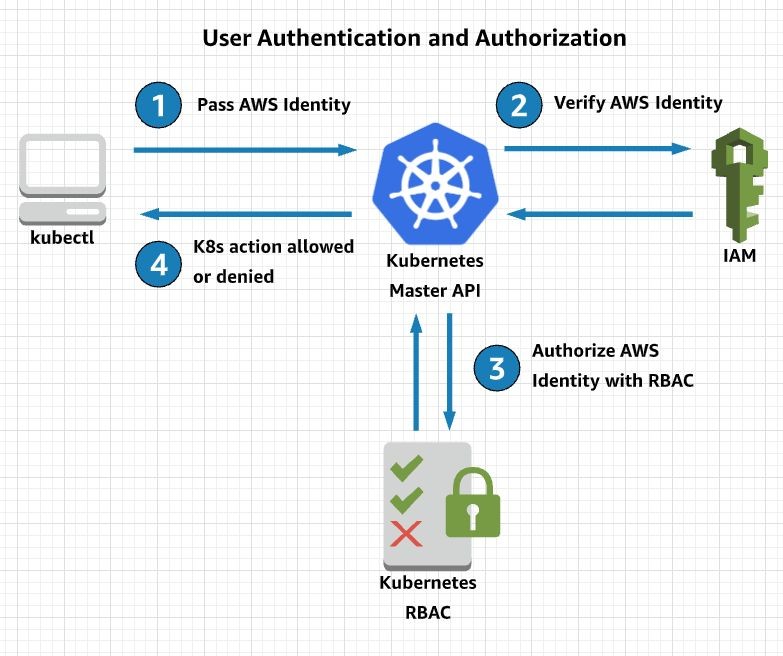
kubectl apply -f https://raw.githubusercontent.com/kubernetes/heapster/master/deploy/kube-config/influxdb/heapster.yaml

kubectl apply -f https://raw.githubusercontent.com/kubernetes/heapster/master/deploy/kube-config/influxdb/influxdb.yaml

kubectl apply -f https://raw.githubusercontent.com/kubernetes/heapster/master/deploy/kube-config/rbac/heapster-rbac.yaml

There is a problem currently, and for fix check: <https://github.com/kubernetes/dashboard/wiki/Accessing-Dashboard---1.7.X-and-above> <https://github.com/kubernetes/dashboard/issues/916>

Just use https to fix it

**Create an administrative account and**

**cluster role binding to access the dashboard**

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

<https://gist.github.com/sd031/37de261794f8fe00d1e277027a862039>

File Source:

kubectl apply -f eks-admin-cluster-role-binding.yaml

<https://gist.github.com/sd031/46673ac1d04554504393a95d405863b2>

File Source:

**Start proxy**

kubectl proxy --address 0.0.0.0 --accept-hosts '.\*' &

**Get a token**

aws-iam-authenticator -i <cluster\_name> token E.g: aws-iam-authenticator -i mean-eks-demo token or

aws-iam-authenticator token -i mean-eks-demo --token-only

**Log in**

http://ip:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#!/login

If any issue check all pods running properly or not: kubectl get pods -o wide --all-namespaces

**Build the docker image using Dockerﬁle**

Run Below command from Project Directory to build the image

Before that install docker: <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html>

E.g.

sudo yum install docker

Sample MEAN Stack APP GIT: <https://github.com/linnovate/mean>

Docker File is already available in the git itself , I have modified a bit:

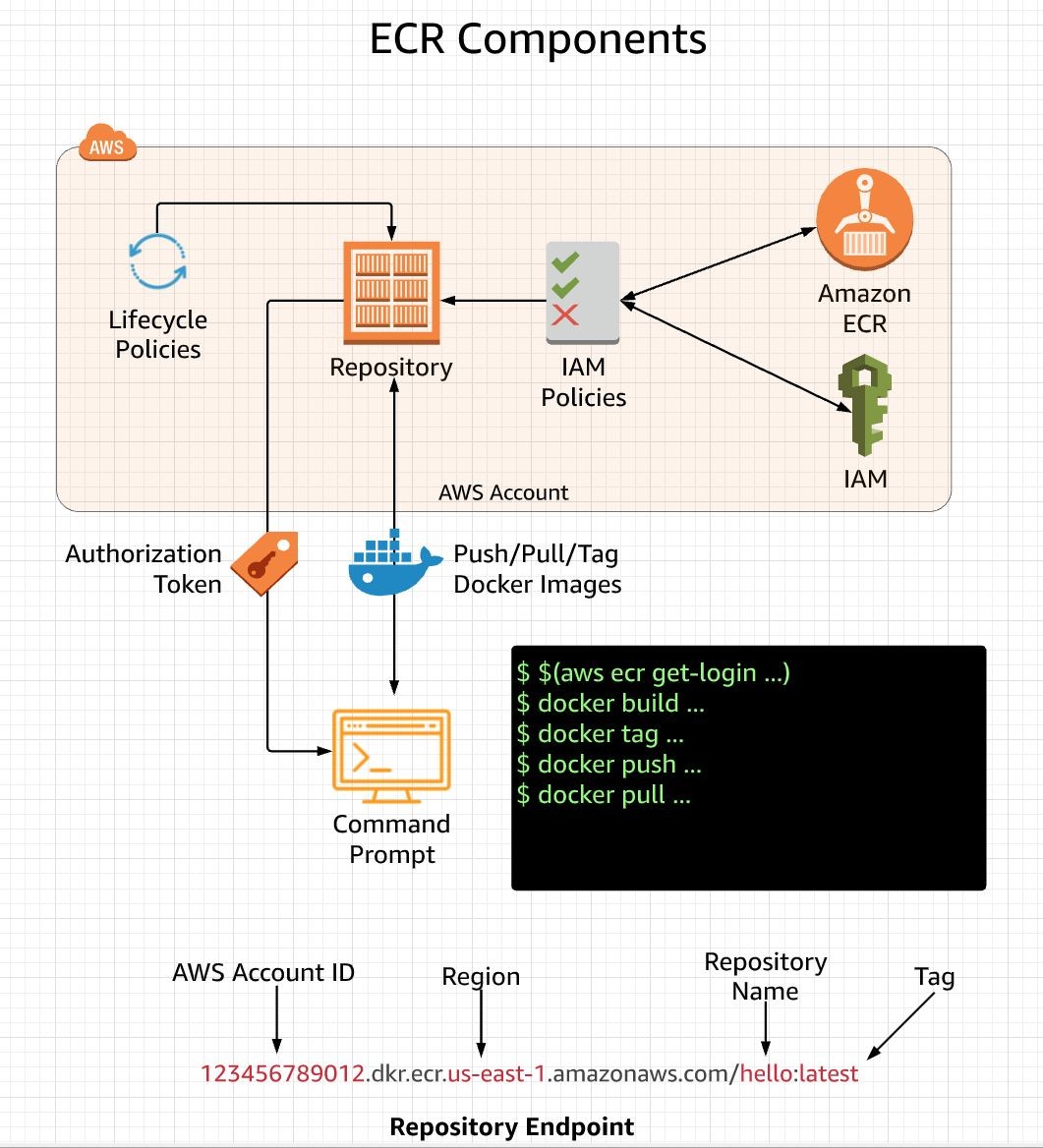
<https://gist.github.com/sd031/a54bd5f575279bf88d7b4be0ea34f38a>

docker image build -t <image\_name>:tag . (this dot refer to current directory) e.g.

docker image build -t mean\_demo:v1 .

Test image running fine or not:

docker container run --name mean\_demo -p 4040:4040 mean\_demo:v1

**Publishing Image to AWS ECR**

It’s just like Docker Hub, where we can push and Pull image and we can use it with AWS ECR same way. To get the docker login and auto execute the login the command is:

**$(aws ecr get-login --no-include-email --region region-name)**

* 1. **$(aws ecr get-login --no-include-email --region us-west-2)**

After this create a ECR repo from AWS Management console, it will give a url like Showing in right side diagram, after that tag

The image: (All command are already given in AWS ECR UI, just use the same)

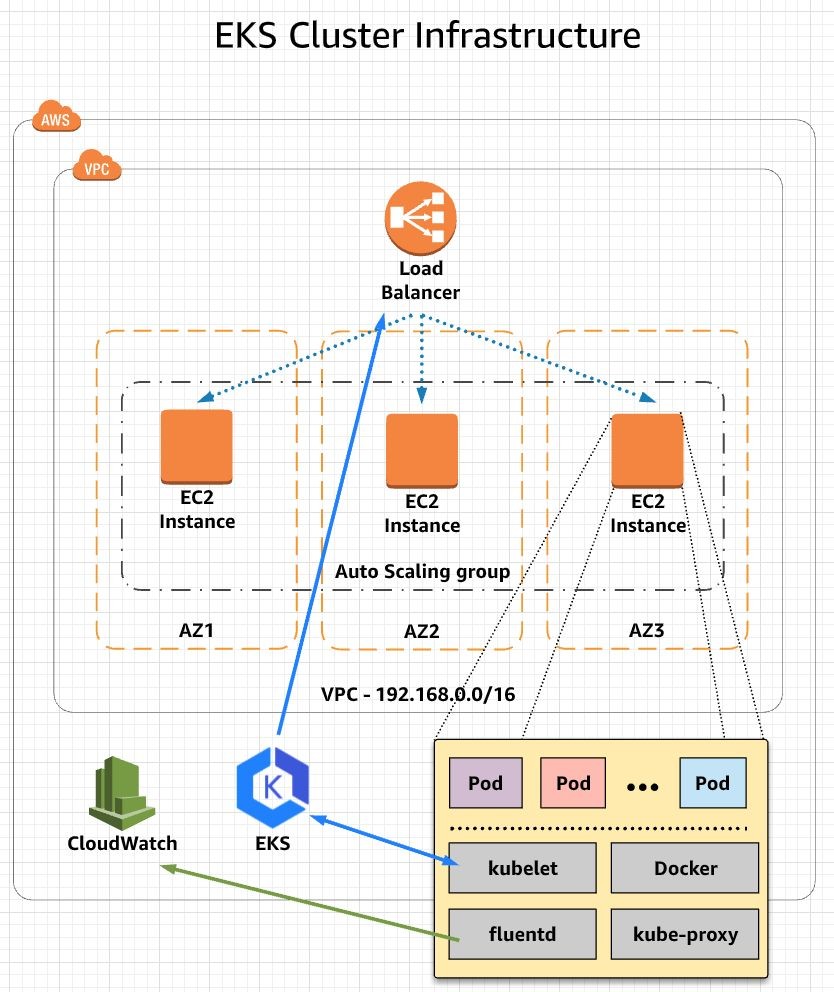
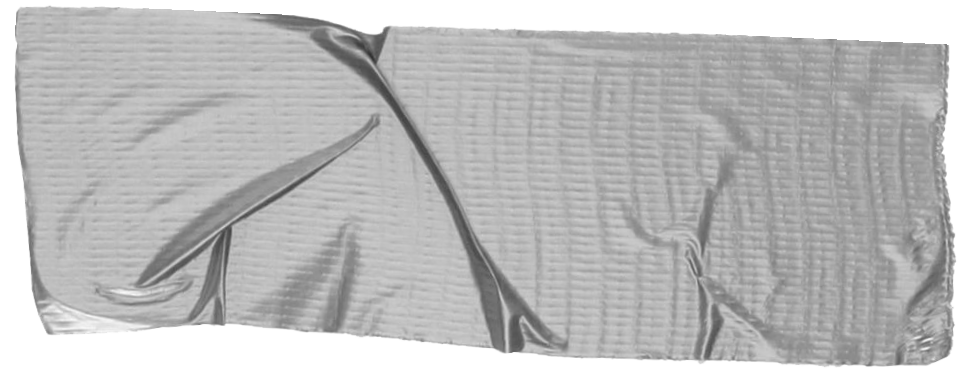
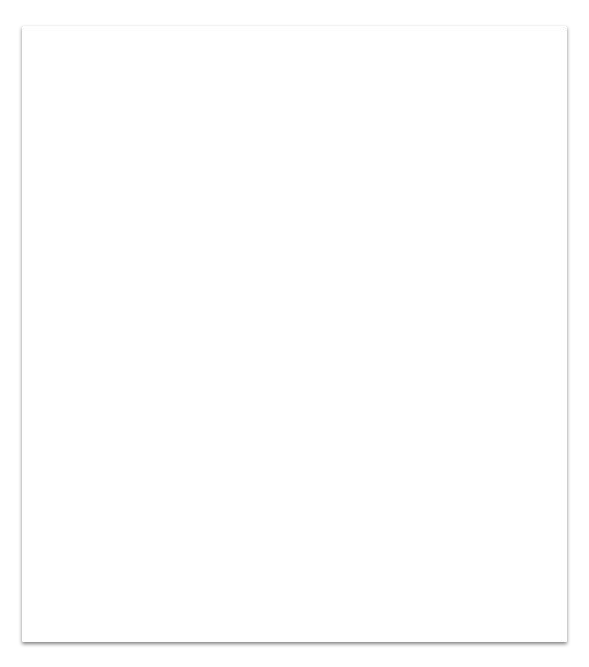
docker tag mean\_demo:v1

123456789012.dkr.ecr.us-west-2.amazonaws.com/mean\_demo:v1

docker push

123456789012.dkr.ecr.us-west-2.amazonaws.com/mean\_demo:v1 Make sure you use the right tags

**Finally Let’s deploy the Application**



[c](https://gist.github.com/sd031/d28f1bd0ca1b0aec7eb109804f966c60)

[a](https://gist.github.com/sd031/e5cd4a3d80813e779a7bb20b6cc3a8ea)

#### kubectl apply -f deployment.yaml deployment.yaml File sample-url:

[https://gist.github.com/sd031/d28f1bd0ca1b0aec7eb109804f966](https://gist.github.com/sd031/d28f1bd0ca1b0aec7eb109804f966c60)

[60](https://gist.github.com/sd031/d28f1bd0ca1b0aec7eb109804f966c60)

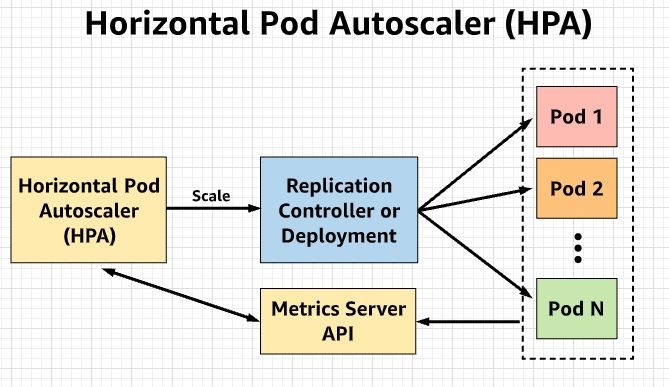
[8ea](https://gist.github.com/sd031/e5cd4a3d80813e779a7bb20b6cc3a8ea) **Tip**

Stories become more credible when they use concrete details such as the speciﬁc complex moves Alberto learned through Translate and his 30 goals in 21 games performance stats.

kubectl apply -f kubernetes/service.yaml service.yaml File Sample: [https://gist.github.com/sd031/e5cd4a3d80813e779a7bb20b6cc3](https://gist.github.com/sd031/e5cd4a3d80813e779a7bb20b6cc3a8ea)

If any issue check logs: kubectl get pods

kubectl describe pod pod\_name\_here

**Auto Scaling: Horizontal Pod Autoscaler (HPA)**

Install Helm

curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get > get\_helm.sh chmod +x get\_helm.sh

./get\_helm.sh

`kubectl apply -f tiller-rbac.yaml` Tiller-rbac.yaml file:

<https://gist.github.com/sd031/6d0207d5920b00cc475475174eed36a0>

### Install Helm using Tiller Service Account

`helm init --service-account tiller` ### Install Metrics Server

```bash

helm install stable/metrics-server --name metrics-server --version 2.0.4 --namespace metrics` kubectl get apiservice v1beta1.metrics.k8s.io -o yaml

### Autoscale the Deployment

`kubectl autoscale deployment mean-demo-deployment --cpu-percent=50 --min=2 --max=10` Check status: kubectl get hpa or kubectl get hpa -w

To run load test: Deploy Apache/PHP

`kubectl run php-apache --image=k8s.gcr.io/hpa-example --requests=cpu=200m --expose --port=80`

`kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10` kubectl run -i --tty load-generator --image=busybox /bin/sh

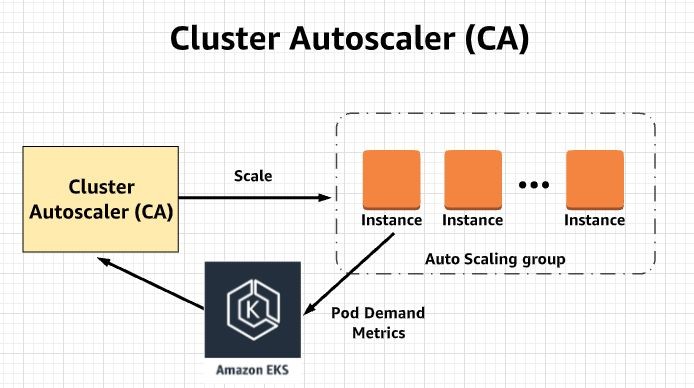
while true; do wget -q -O - http://php-apache; done

## Auto Scaling: Cluster Autoscaler (CA)

Find Auto Scaling group using the AWS Management Console, noting its name. Edit the ASG's min/max size to 2 and 8 nodes, respectively.

Edit `cluster\_autoscaler.yaml`, replacing `<AUTOSCALING GROUP NAME>` with the ASG name you found in the console. Cluster\_autoscaler.yaml file: <https://gist.github.com/sd031/3f0f4c89559e0c4bf026d44db4855ad3>

Optionally change the `AWS\_REGION` to something other than `us-east-1` if you are working in a different region.

We need to configure an inline policy and add it to the EC2 instance profile of the worker nodes Policy json: <https://gist.github.com/sd031/8d21542f81b85d551462a86d84b6b926>

Deploy the autoscaler:

`kubectl apply -f cluster\_autoscaler.yaml` Watch the logs:

`kubectl logs -f deployment/cluster-autoscaler -n kube-system`

Scale out test:

kubectl apply -f nginx.yaml Nginx.yaml file:

<https://gist.github.com/sd031/95b966269a4d99ce0cd45b79df3195a3>

kubectl get deployment/nginx-scaleout

kubectl scale --replicas=10 deployment/nginx-scaleout

## Monitoring an EKS Cluster

## Configure Storage Class

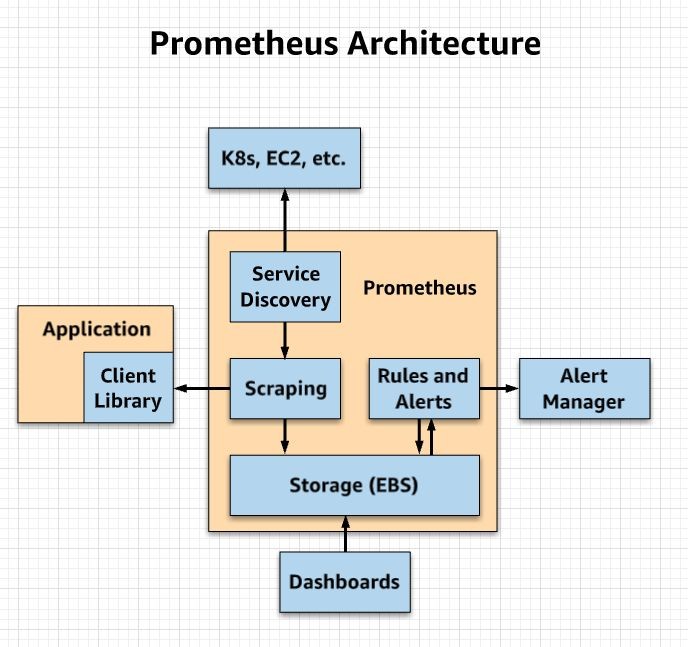
kubectl create -f prometheus-storageclass.yaml

prometheus-storageclass.yaml File Link: <https://gist.github.com/sd031/e094d23fe0e817bec82901b9a93dc5b1>

## Deploy Prometheus

helm install -f prometheus-values.yaml stable/prometheus --name prometheus --namespace prometheus Prometheus-values.yaml file: <https://gist.github.com/sd031/1cd40bb5a5b14c39e87242845a92d1f3>

kubectl get all -n prometheus

You should see all the Prometheus pods, services, daemonsets, deployments, and replica sets are all ready and avail You can access Prometheus server URL by going to any one of your Worker node IP address and specify port `:3090 Remember to open port 30900 in your Worker nodes Security Group. In the web UI, you can see all the targets and m

able.

0/targets` (for ex, `52.12.161.128:30900/targets`. etrics being monitored by Prometheus

## Install Grafana

kubectl create namespace grafana Execute Script:

<https://gist.github.com/sd031/2c54da7bb4476045a4ae7d3fda90e707> Run the following command to check if Grafana is deployed properly: kubectl get all -n grafana

Get Load Balancer url:

export ELB=$(kubectl get svc -n grafana grafana -o jsonpath='{.status.loadBalancer.ingress[0].hostname}') echo "http://$ELB"

When logging in, use the username admin and get the password hash by running the following:

kubectl get secret --namespace grafana grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo

## Final Grafana Dashboard Set-up

###### Create Dashboards:

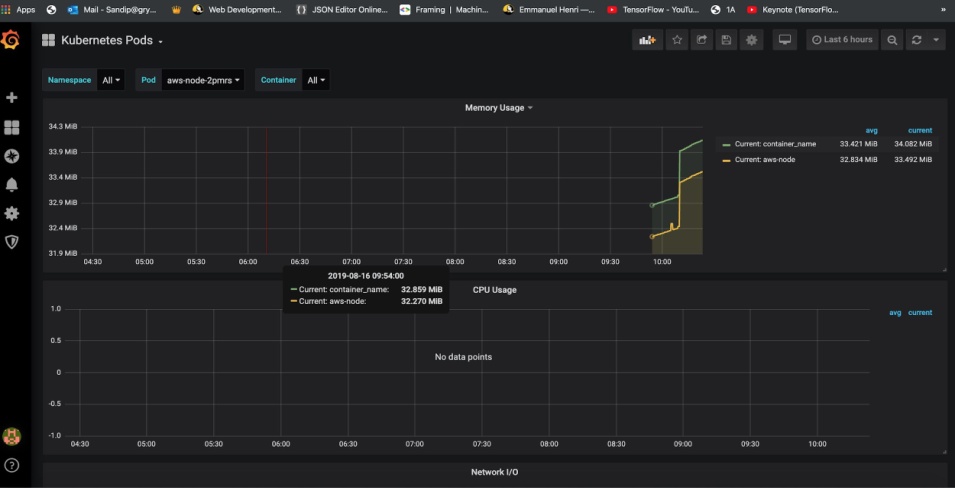
Login into Grafana dashboard using credentials supplied during configuration

You will notice that ‘Install Grafana’ & ‘create your first data source’ are already completed. We will import community created dashboard for this tutorial

Click ‘+’ button on left panel and select ‘Import’

Enter 3131 dashboard id under Grafana.com Dashboard & click ‘Load’.

Leave the defaults, select ‘Prometheus’ as the endpoint under prometheus data sources drop down, click ‘Import’. This will show monitoring dashboard for all cluster nodes

For creating dashboard to monitor all pods, repeat same process as above and enter 3146 for dashboard id, I personally like: 7249, 3091, 1860 You can find more dashboards and plugins in: <https://grafana.com/grafana/dashboards/> - but as long term goal, learn how to do it by yourself

## Clean up everything

Since we have used eksctl, it’s a lot easier,

Before doing anything, remove the inline policy from cluster instance role, just got to instance worker node, then check details, you will find the role link: IAM role e.g. eksctl-mean-eks-demo-nodegroup-ng…..

Click on that, and you will see, under permission the policy you had added, e.g. CA, just delete that. Then remove autoscaling, deployment , services, prometheus , grafana etc etc e.g.

kubectl delete -f cluster\_autoscaler.yaml kubectl delete -f nginx.yaml

Delete the horizontal pod autoscaler and load test: kubectl delete hpa,svc php-apache

kubectl delete deployment php-apache load-generator kubectl delete -f deplyment.yaml

helm delete prometheus helm del --purge prometheus helm delete grafana

helm del --purge grafana

Now finally delete the cluster: Check how many are running: eksctl get clusters

Delete cluster:

eksctl delete cluster -f cluster.yaml

Behind the scene the cloud formation stack will get deleted and accordingly resources will be deleted as well, must do it if you are doing in development or test as a temporary deployment otherwise it will cost you a lot

# There are few more things you need to know

#### This demo is just the start points and there is a lot more out there, the more you use it , the more experience you will gather, so I will highly suggest try by yourself and deploy your own AWS EKS Cluster.

After trying the basic app deployments , the next thing you might be interested to learn are:

* + 1. [Using Spot instances](https://eksctl.io/usage/spot-instances/) with Kubernetes and save 90% of cost
    2. [Types of Deployments](https://www.weave.works/blog/kubernetes-deployment-strategies) available in [Kubernetes Deployment](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/) and how to conﬁgure it
    3. [Types of Services](https://blog.cloudthat.com/3-types-of-cluster-networking-in-kubernetes/) available in [Kubernetes Service](https://kubernetes.io/docs/concepts/services-networking/service/#multi-port-services) and how to conﬁgure it.
    4. [CI/CD with Kubernetes](https://www.linkedin.com/posts/sandip-das-developer_ci-cd-with-kubernetes-activity-6561889388572266496-K8vF)
    5. [Logging with Cloud Trail](https://eksctl.io/usage/cloudwatch-cluster-logging/)
    6. [Logging to Cloudwatch with Fluentd](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Container-Insights-setup-logs.html)
    7. [Complex Authentication , Roles](https://docs.aws.amazon.com/eks/latest/userguide/add-user-role.html) etc

There are lot of resources available online you can learn (check the linked resources) or maybe you want another meet-up to dig deeper into any of these, Let [Me](https://www.linkedin.com/in/sandip-das-developer/) or [Chirag](https://www.linkedin.com/in/chiragnayyar/) know the same.