KUBERNETES MASTERCLASS 1

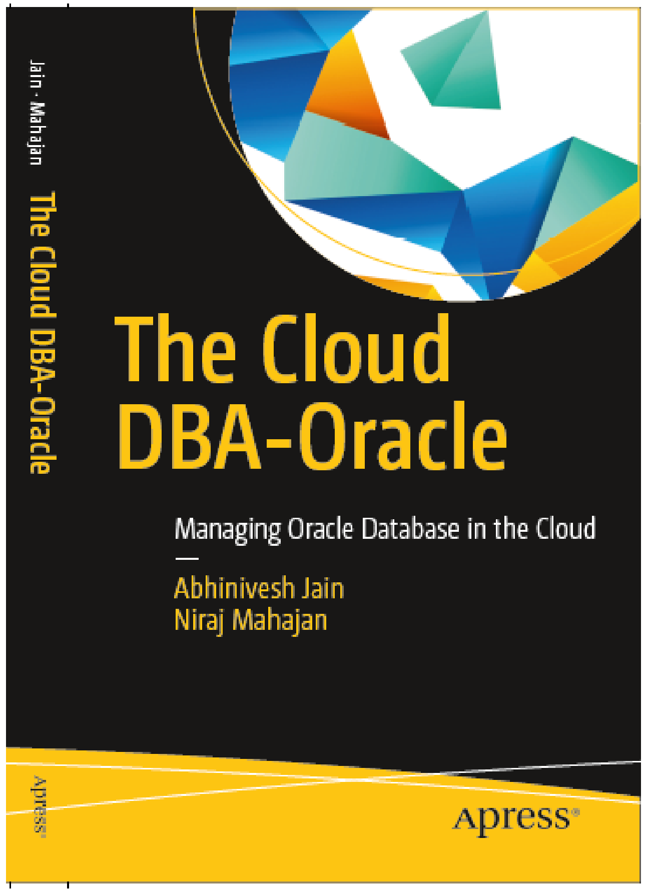
Abhinivesh Jain

ABOUT ME

Author, Speaker and Blogger

Open Source “Contributor”

Working as Distinguished Member of Technical Staff (DMTS)- Senior Member



@AbhiniveshJain

/abhiniveshjain

DMTS
DISTINGUISHED MEMBER OF TECHNICAL STAFF

AGENDA

Container

- History
- Container Image
- Container Image registry
- Container Orchestration
- Sample application deployment

Kubernetes

- History
- Architecture
- Managed k8s Providers
- AWS Offerings
- Launching EKS cluster
- Kubernetes building Blocks

BRIEF HISTORY

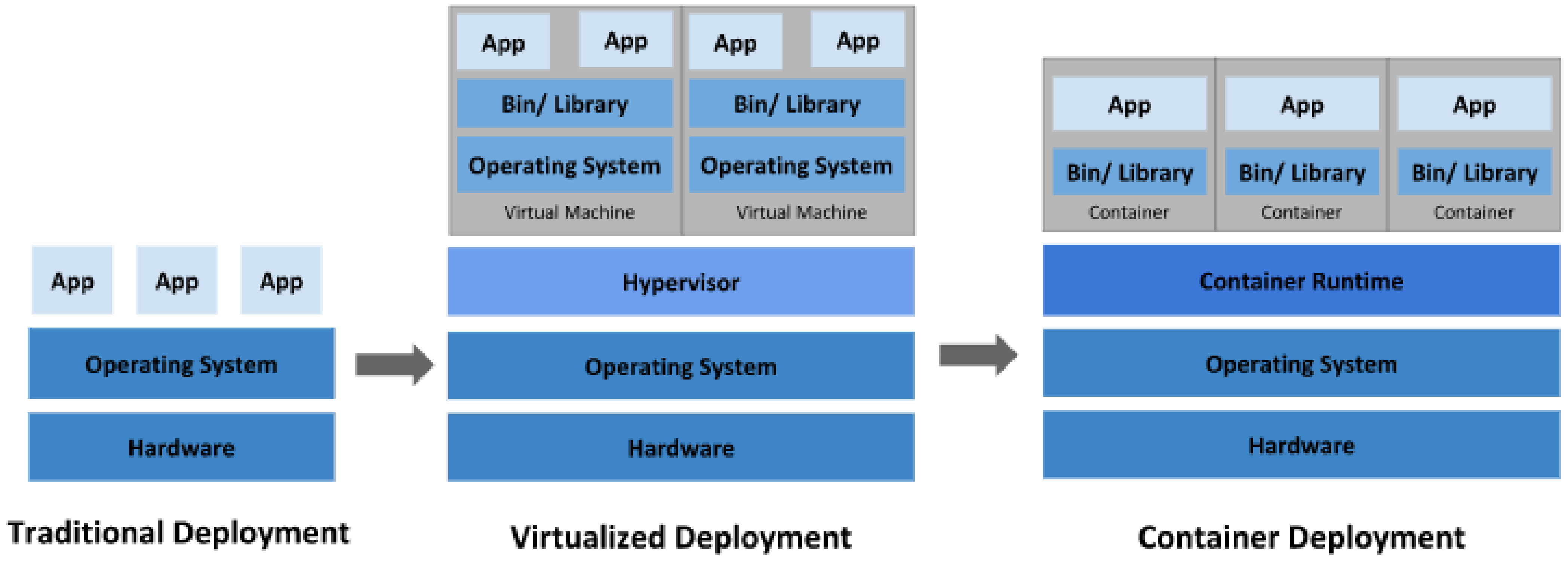
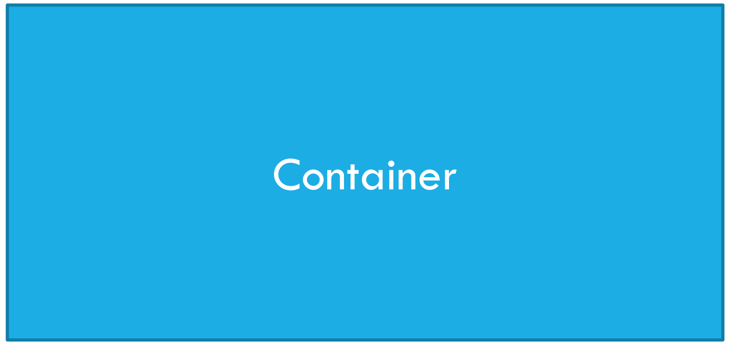


Image source: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

CONTAINER

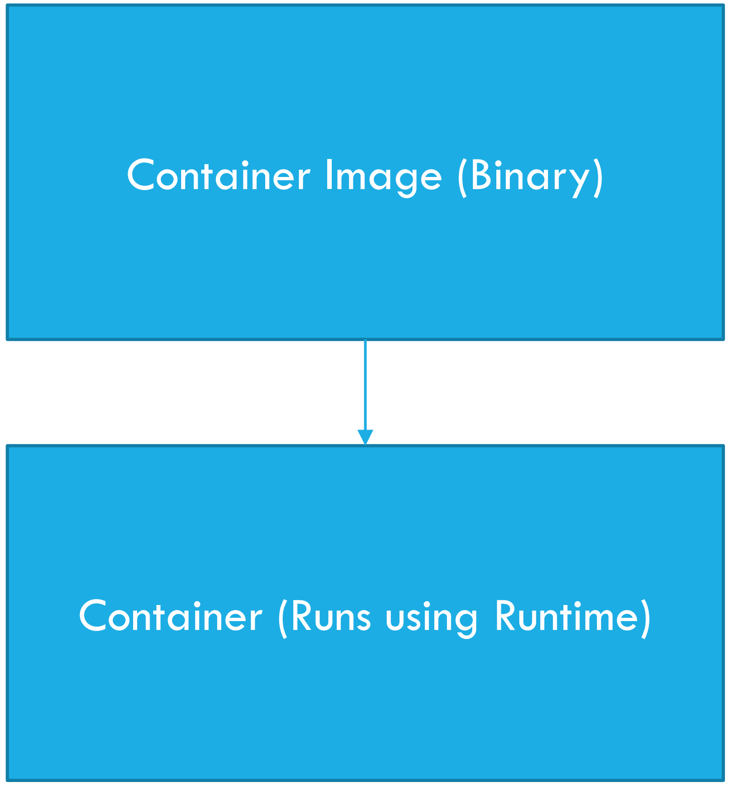
- Container is packaged app with all dependencies
- Runs on a shared Kernel
- Quick deployment in any hosting environment (Baremetal, Virtual Machine, Private cloud, Public cloud)
- No portability challenges like VMs
- Smaller app size

CONTAINER IS SMALLEST UNIT



CONTAINER IS CREATED FROM CONTAINER IMAGE

```
docker run –name mywebserapp nginx
```



CONTAINER IMAGES ARE STORED IN IMAGE REGISTRY

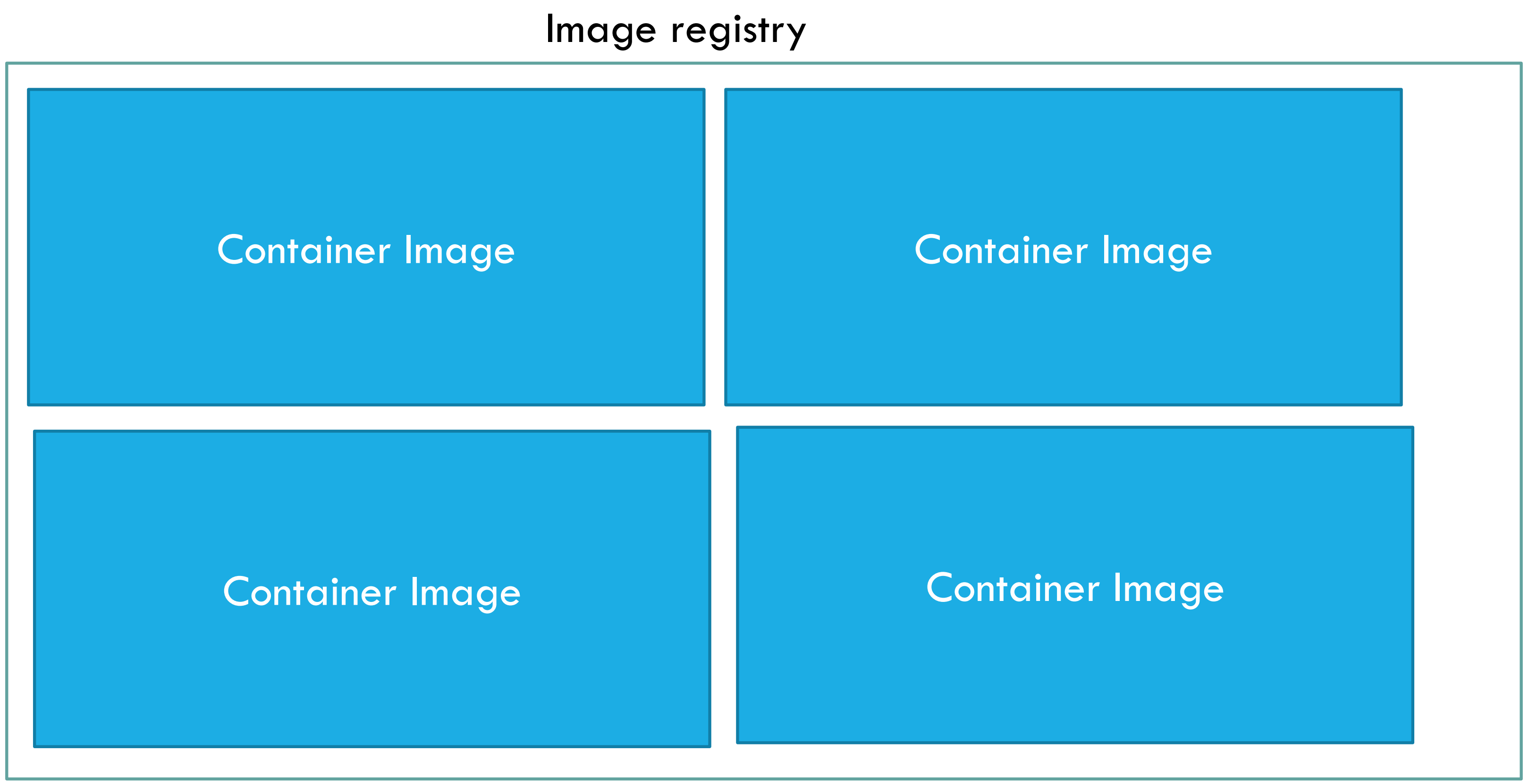


Image registry contains various versions of container images.

CONTAINER IMAGE REGISTRY TYPES

Public Image registry
- Docker hub
- Quay.io

Private Image registry
- Docker registry
- Amazon Elastic Container registry (ECR)

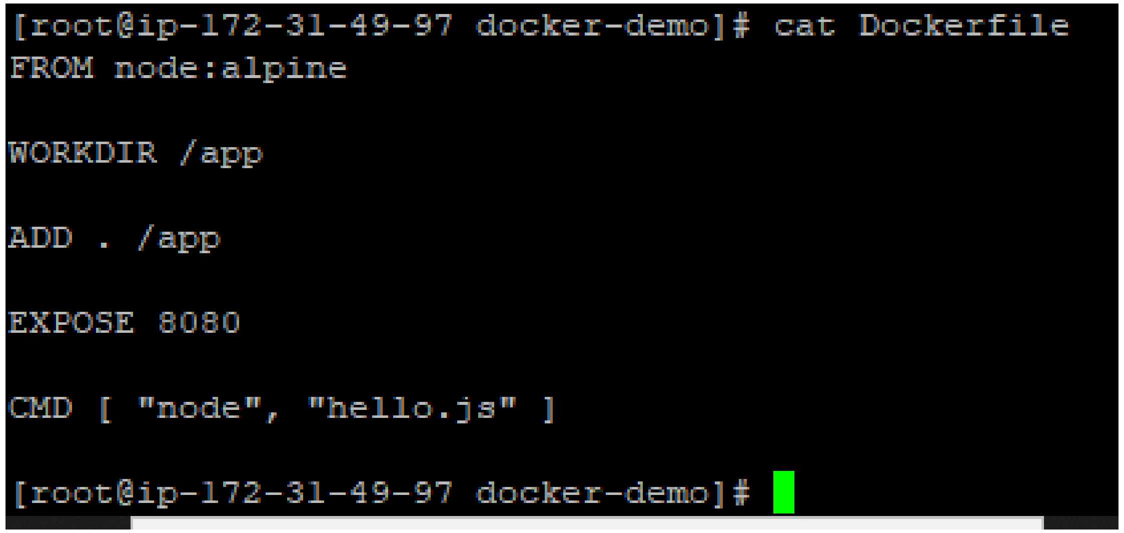
https://hub.docker.com/search?q=&type=image&image_filter=official

DOCKERFILE

Used for defining a container

Writing a dockerfile is first step for app containerization

Next step is to build and test the image



```
[root@ip-172-31-49-97 docker-demo]# cat Dockerfile
FROM node:alpine

WORKDIR /app

ADD . /app

EXPOSE 8080

CMD [ "node", "hello.js" ]

[root@ip-172-31-49-97 docker-demo]#
```

QUIZ TIME

What is the difference between Docker and Container?

ENOUGH THEORY, SHOW ME THE STUFF !!!

Demo Time

Public

DEMO 1

Install Docker

Write “Hello world” program

Containerized your app

- Create Dockerfile
- Perform Docker Build
- Check container image

Run your containerized App

CONTAINER ORCHESTRATION

Platform for managing Container lifecycle

- Create
- Delete
- Schedule
- Scaling
- Self Healing
- Upgrade and Rollback

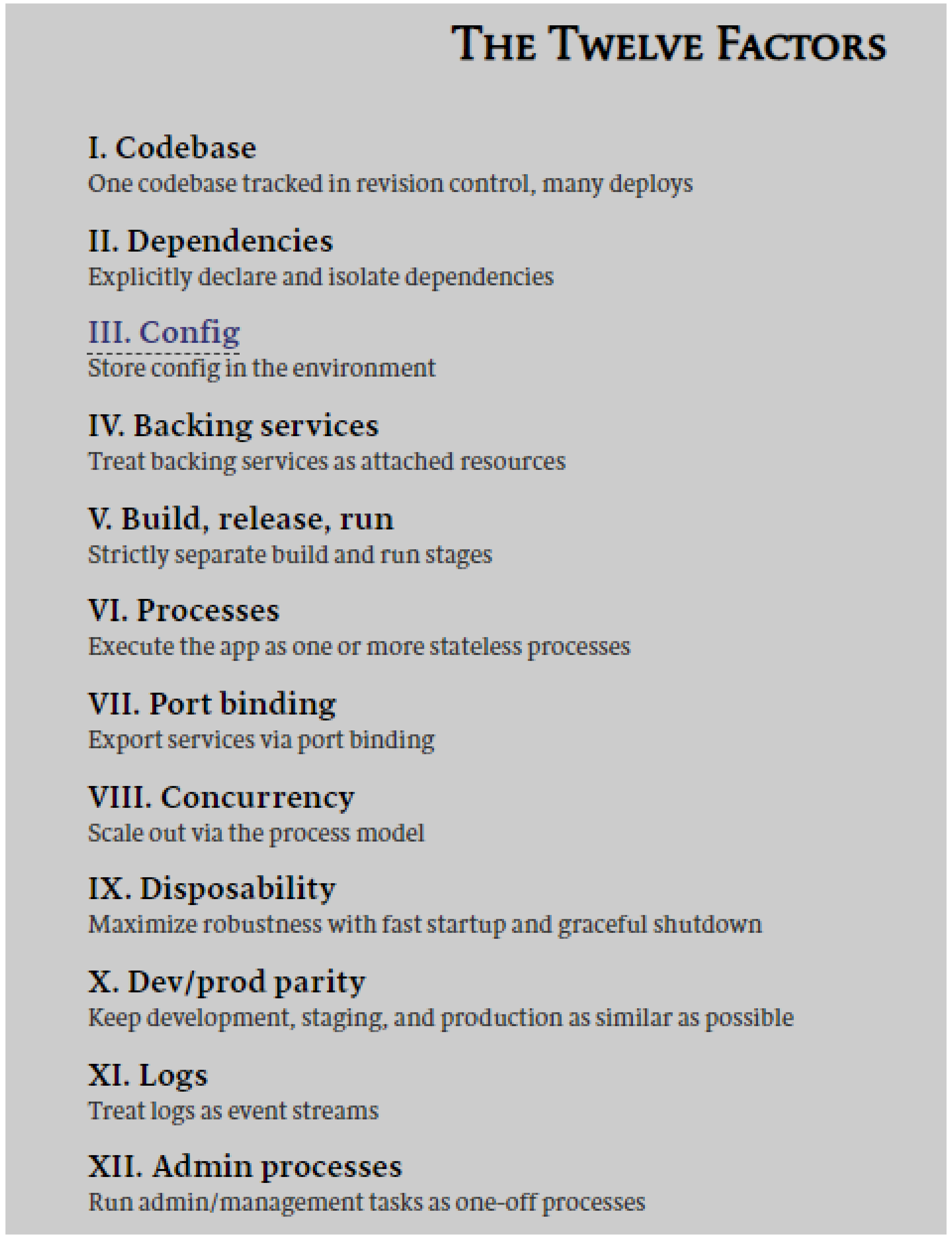
Amazon EKS

Amazon ECS

SOME OTHER RELATED STUFF

Cloud Native application

Kubernetes Native Application



THE TWELVE FACTORS

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

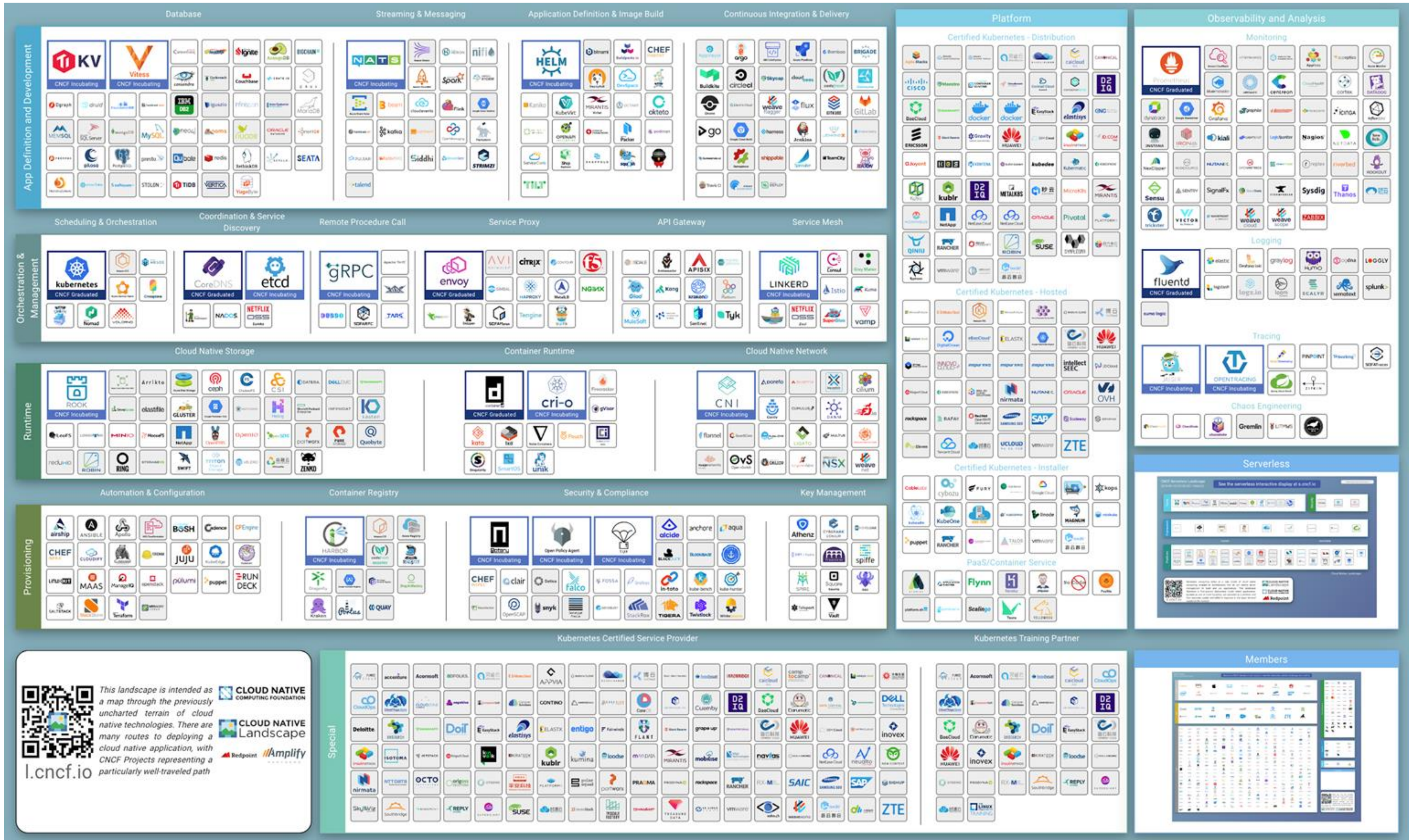
BUT WHY AREN'T WE DISCUSSING ABOUT KUBERNETES?

FUNDAMENTALS
MATTER

LET’S DIVE INTO KUBERNETES WORLD…

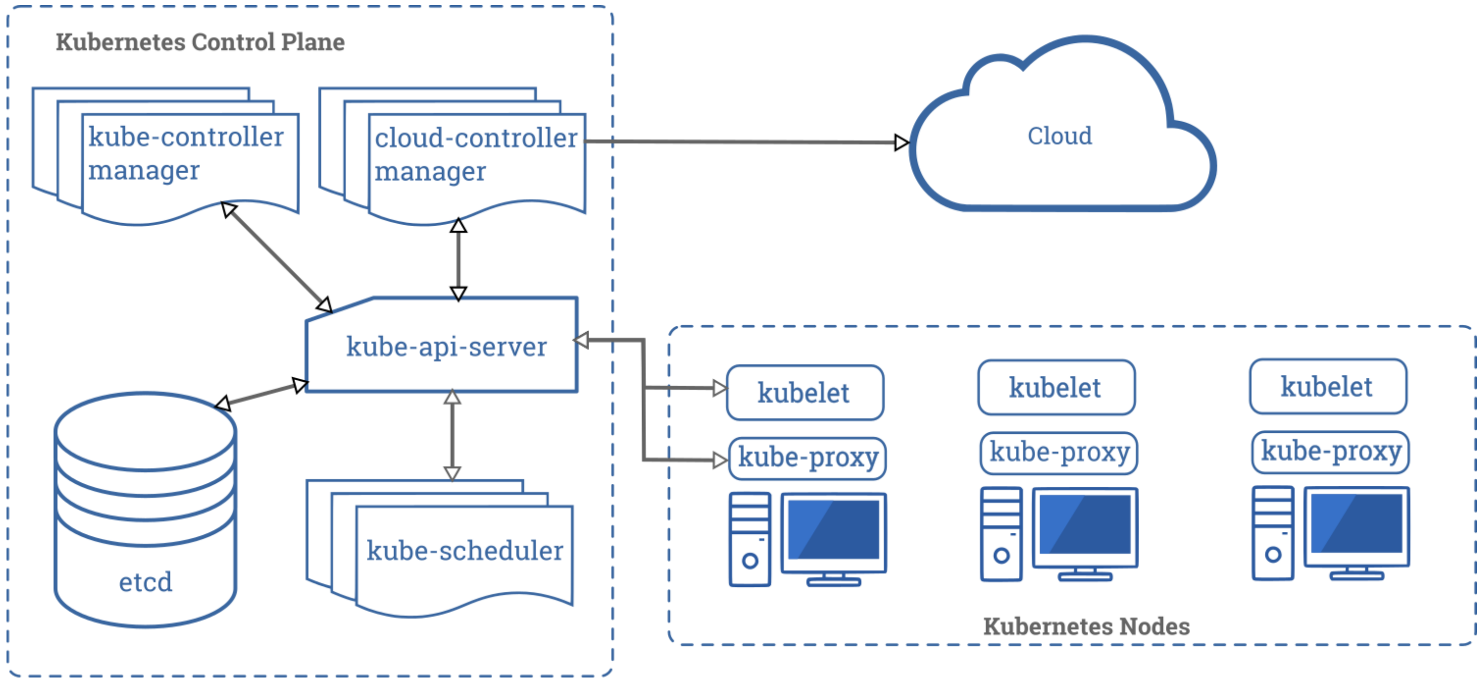
- Primarily an Open source Container Orchestration platform
- Greek word for "helmsman" or "pilot“
- Originally Designed by Google
- Open sourced in 2015 and maintained by CNCF
- One of the key project in CNCF.
- Written in Go language
- Kubernetes short form is k8s where 8 represents the no. of characters in between k and s.
- V1.0 released in July 2015, Latest version V1.18 release on 25th Mar 2020

KUBERNETES POPULARITY


Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
App Definition and Development
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Logging
Tracing
Chaos Engineering
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Orchestration & Management
Cloud Native Storage
Container Runtime
Cloud Native Network
Runtime
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Provisioning
Serverless
Kubernetes Certified Service Provider
Kubernetes Training Partner
Special
Members
kubernetes
CNCF Graduated
CNCF Incubating
l.cncf.io
CLOUD NATIVE COMPUTING FOUNDATION
CLOUD NATIVE Landscape
Redpoint
Amplify

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path

KUBERNETES ARCHITECTURE



Deploying Kubernetes means, you are deploying a cluster. Cluster has more than 1 node except for Minikube which is single node cluster.

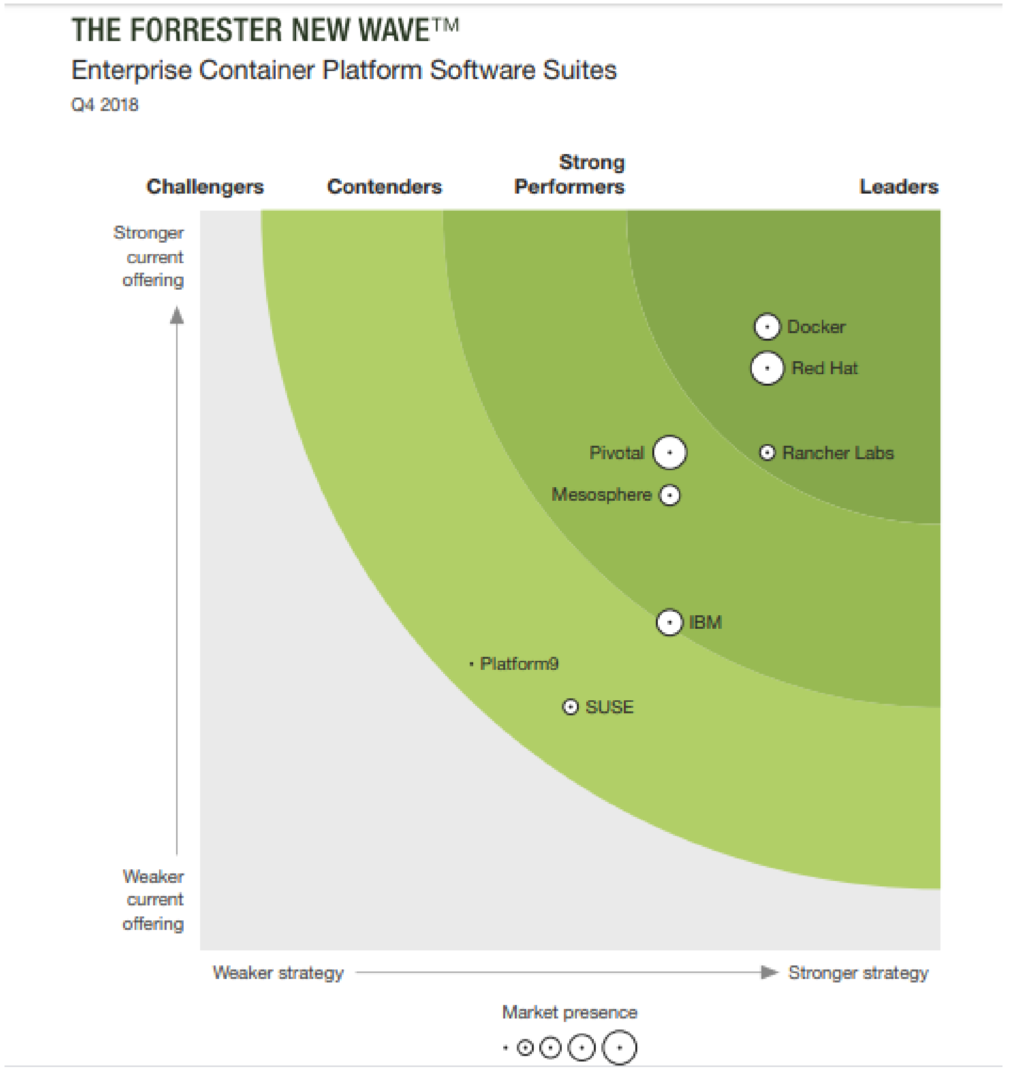
Source: https://kubernetes.io/docs/concepts/overview/components/

KUBERNETES ARCHITECTURE

Architecture component	Purpose
Controller	Control loops to maintain desired state
Scheduler	Responsible for POD scheduling on given node
Etcd	Key value pair based repository/database for cluster data
API server	Common end point for all communication
Kubelet	K8s Agent deployed on all worker nodes
Kube-proxy	Maintains network rules on worker nodes
Container run-time	e.g. Docker, CRI-O
Kubectl	Command line utility for management

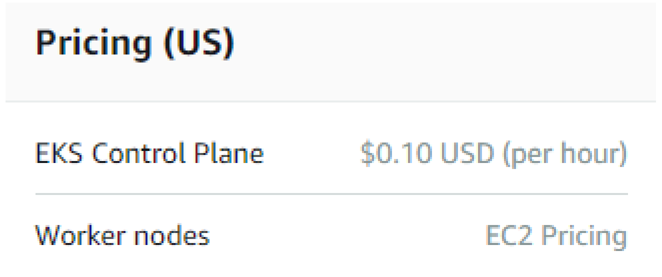
MANAGED KUBERNETES PROVIDERS

Provider	Offering name
AWS	Amazon Elastic Kubernetes Service (EKS)
Google	Google Kubernetes Engine (GKE)
Azure	Azure Kubernetes Service (AKS)
Platform 9	Platform9 Managed Kubernetes (PMK)
Rancher	Rancher Kubernetes Engine (RKE)



AWS OFFERINGS

- AWS is widely used container platform and ~60% deployments of k8s are on AWS.
- 3 Key offerings are-
  - Amazon Elastic Container Service (ECS)
  - Amazon Elastic Kubernetes Service (EKS)
  - Amazon Elastic Container Registry (ECR)
- ECS is cheaper than EKS and suitable for small and simple deployments.
- Spot instances be used as worker nodes of EKS cluster.
- For Worker node, you should use EKS optimized AMI. These come with Amazon Linux 2 OS along with Kubectl, docker etc and can join the cluster automatically.



Pricing (US)

EKS Control Plane	$0.10 USD (per hour)
Worker nodes	EC2 Pricing

https://console.aws.amazon.com/

https://www.cncf.io/blog/2018/08/29/cncf-survey-use-of-cloud-native-technologies-in-production-has-grown-over-200-percent/

AWS OFFERINGS- EKS

- Managed Kubernetes service
- Most popular and widely used k8s platform

3 WAYS TO CONFIGURE AWS EKS

1. Using AWS Console
2. Using eksctl
3. Using AWS CLI

Which one do you think would be most easy one?

QUIZ TIME

1. AWS Console
2. eksctl
3. AWS CLI

Which one do you think would be most easy option to install EKS?

LAUNCHING EKS CLUSTER

Demo Time

https://eksctl.io/

LAUNCHING EKS CLUSTER

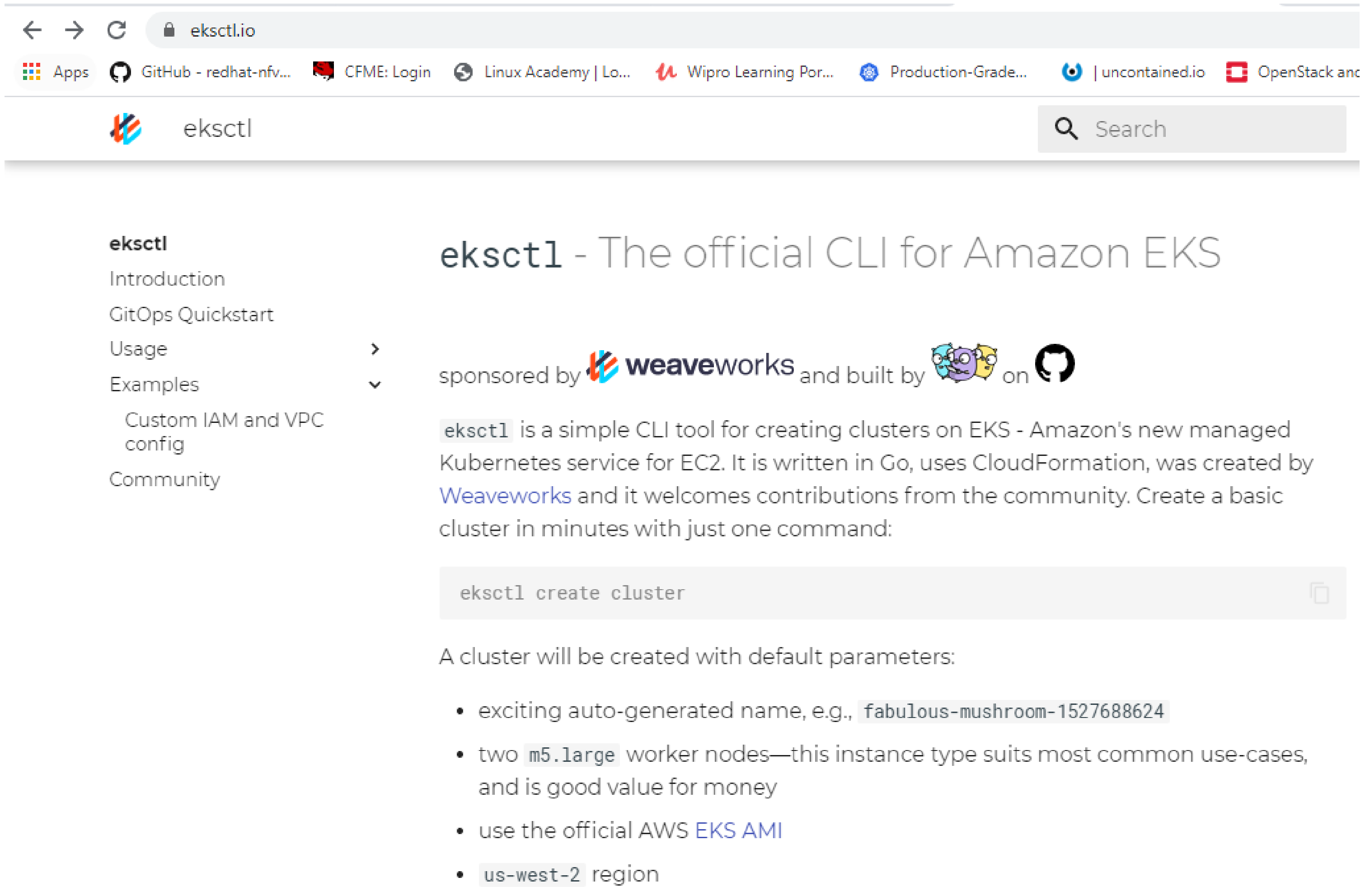
Steps to follow

1. Create an ec2 instance that will act as your jump host
2. Connect to jump host
3. Set your CLI environment using aws configure – atleast v1.18.49
4. Install kubectl (https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html)
5. Install aws-iam-authenticator (https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html)
6. Install eksctl (https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html) – atleast v0.19.0
7. Call eksctl utility to create cluster (this can take 15 min or more)
8. Connect to your cluster using kubectl

LAUNCHING EKS CLUSTER

Steps to follow

1. Create ec2 instance that will act as your jump host
2. Connect to jump host
3. Set your CLI environment using aws configure
4. Call eksctl utility to create cluster (this can take 15 min or more)
5. Connect to your cluster using kubectl



`eksctl` - The official CLI for Amazon EKS

sponsored by weaveworks and built by on

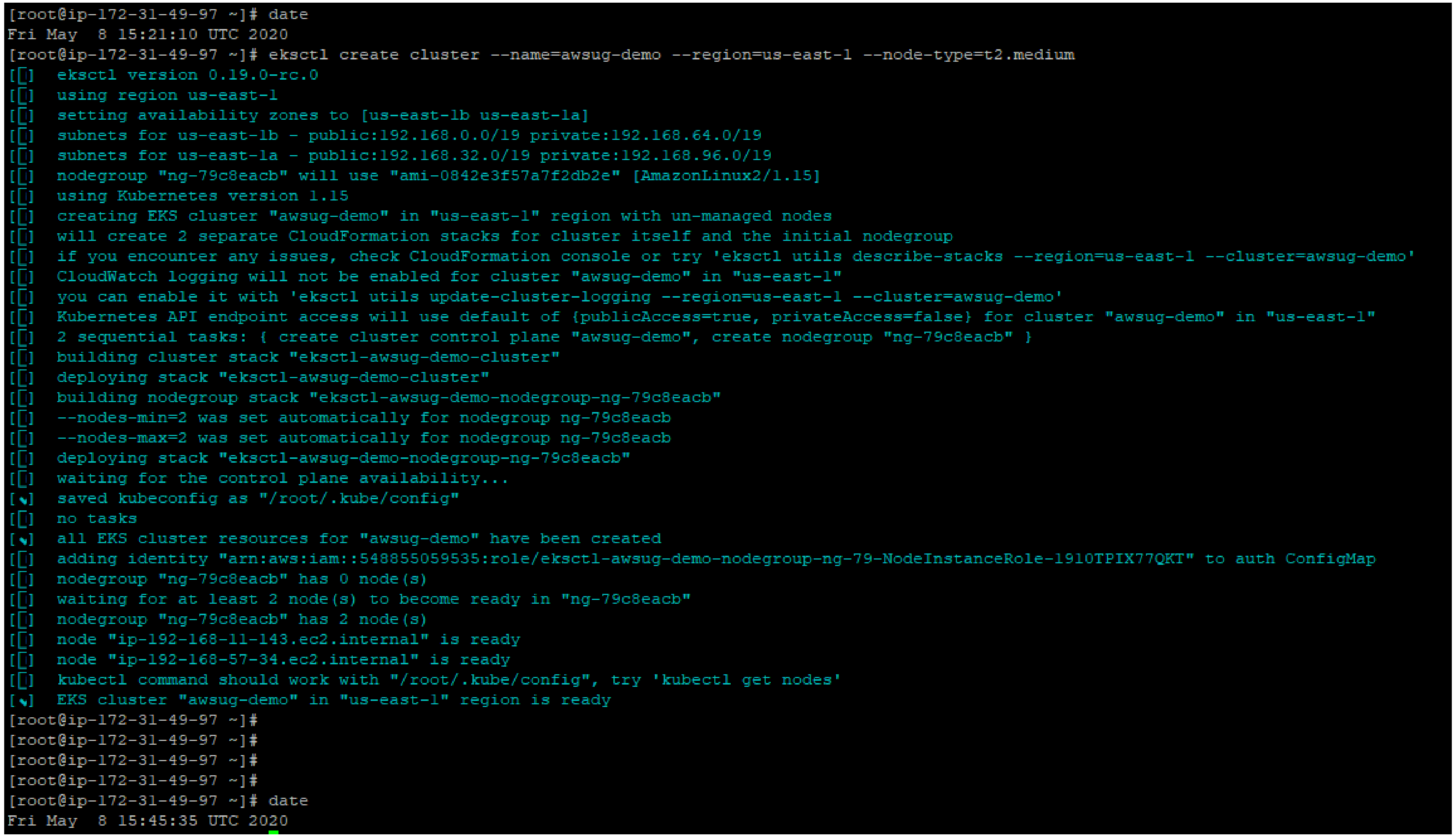
`eksctl` is a simple CLI tool for creating clusters on EKS - Amazon's new managed Kubernetes service for EC2. It is written in Go, uses CloudFormation, was created by Weaveworks and it welcomes contributions from the community. Create a basic cluster in minutes with just one command:

```
eksctl create cluster
```

A cluster will be created with default parameters:

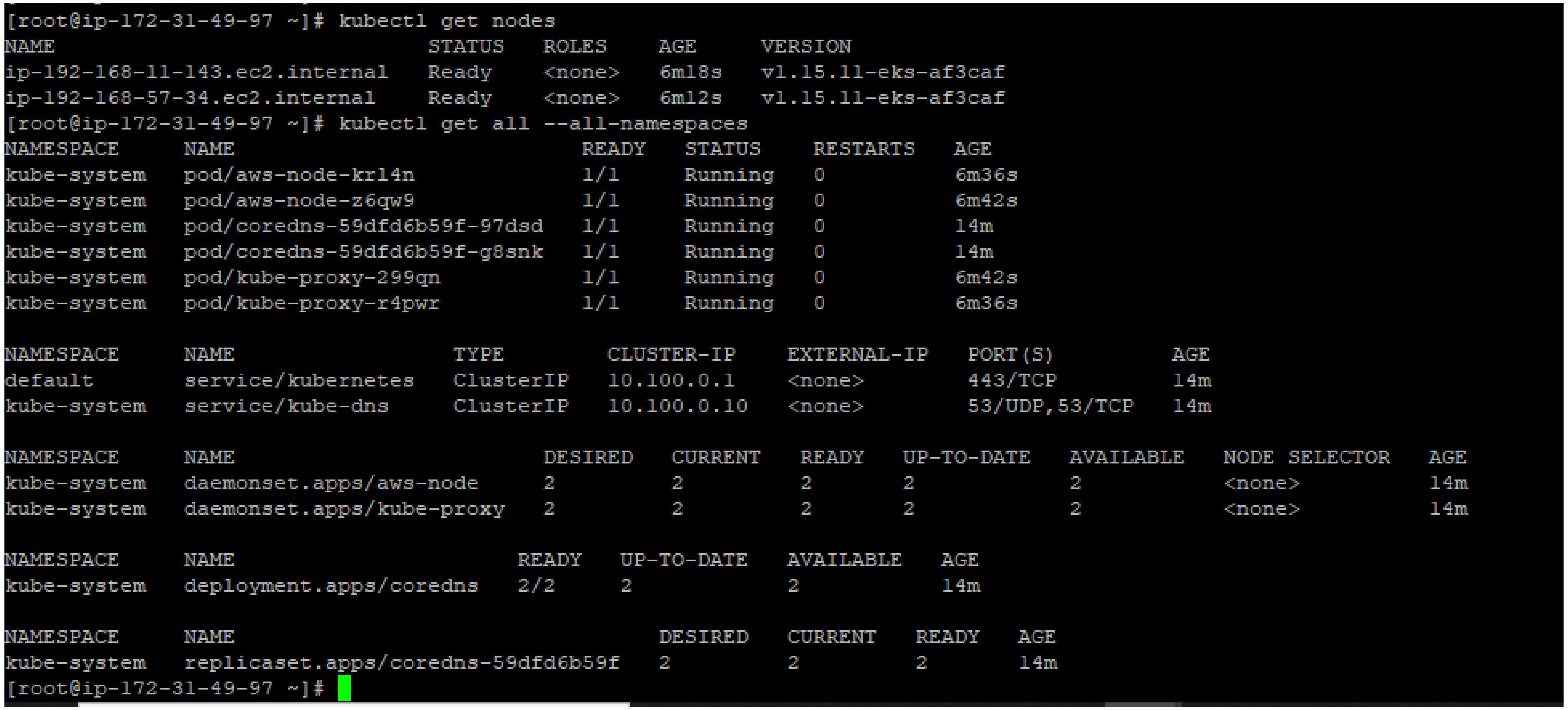
- exciting auto-generated name, e.g., `fabulous-mushroom-1527688624`
- two `m5.large` worker nodes—this instance type suits most common use-cases, and is good value for money
- use the official AWS EKS AMI
- `us-west-2` region

LAUNCHING EKS CLUSTER



```
[root@ip-172-31-49-97 ~]# date
Fri May  8 15:21:10 UTC 2020
[root@ip-172-31-49-97 ~]# eksctl create cluster --name=awsug-demo --region=us-east-1 --node-type=t2.medium
[▯]  eksctl version 0.19.0-rc.0
[▯]  using region us-east-1
[▯]  setting availability zones to [us-east-1b us-east-1a]
[▯]  subnets for us-east-1b - public:192.168.0.0/19 private:192.168.64.0/19
[▯]  subnets for us-east-1a - public:192.168.32.0/19 private:192.168.96.0/19
[▯]  nodegroup "ng-79c8eacb" will use "ami-0842e3f57a7f2db2e" [AmazonLinux2/1.15]
[▯]  using Kubernetes version 1.15
[▯]  creating EKS cluster "awsug-demo" in "us-east-1" region with un-managed nodes
[▯]  will create 2 separate CloudFormation stacks for cluster itself and the initial nodegroup
[▯]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1 --cluster=awsug-demo'
[▯]  CloudWatch logging will not be enabled for cluster "awsug-demo" in "us-east-1"
[▯]  you can enable it with 'eksctl utils update-cluster-logging --region=us-east-1 --cluster=awsug-demo'
[▯]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "awsug-demo" in "us-east-1"
[▯]  2 sequential tasks: { create cluster control plane "awsug-demo", create nodegroup "ng-79c8eacb" }
[▯]  building cluster stack "eksctl-awsug-demo-cluster"
[▯]  deploying stack "eksctl-awsug-demo-cluster"
[▯]  building nodegroup stack "eksctl-awsug-demo-nodegroup-ng-79c8eacb"
[▯]  --nodes-min=2 was set automatically for nodegroup ng-79c8eacb
[▯]  --nodes-max=2 was set automatically for nodegroup ng-79c8eacb
[▯]  deploying stack "eksctl-awsug-demo-nodegroup-ng-79c8eacb"
[▯]  waiting for the control plane availability...
[✔]  saved kubeconfig as "/root/.kube/config"
[▯]  no tasks
[✔]  all EKS cluster resources for "awsug-demo" have been created
[▯]  adding identity "arn:aws:iam::548855059535:role/eksctl-awsug-demo-nodegroup-ng-79-NodeInstanceRole-1910TPIX77QKT" to auth ConfigMap
[▯]  nodegroup "ng-79c8eacb" has 0 node(s)
[▯]  waiting for at least 2 node(s) to become ready in "ng-79c8eacb"
[▯]  nodegroup "ng-79c8eacb" has 2 node(s)
[▯]  node "ip-192-168-11-143.ec2.internal" is ready
[▯]  node "ip-192-168-57-34.ec2.internal" is ready
[▯]  kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'
[✔]  EKS cluster "awsug-demo" in "us-east-1" region is ready
[root@ip-172-31-49-97 ~]#
[root@ip-172-31-49-97 ~]#
[root@ip-172-31-49-97 ~]#
[root@ip-172-31-49-97 ~]#
[root@ip-172-31-49-97 ~]# date
Fri May  8 15:45:35 UTC 2020
```

LAUNCHING EKS CLUSTER

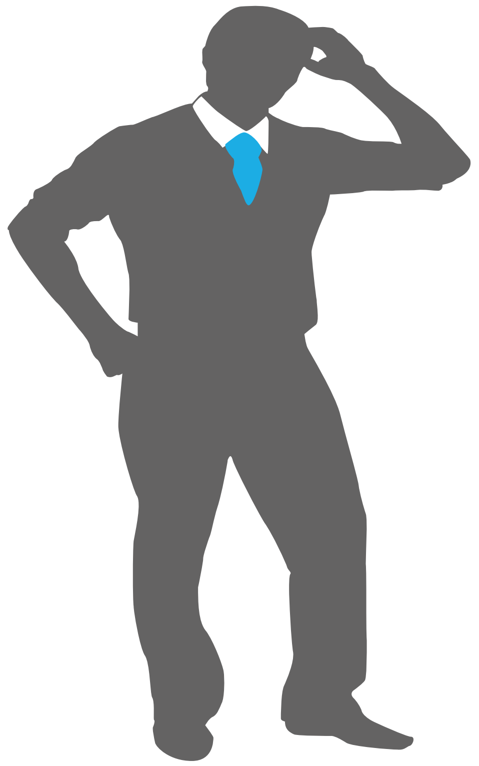


```
[root@ip-172-31-49-97 ~]# kubectl get nodes
NAME                              STATUS   ROLES    AGE     VERSION
ip-192-168-11-143.ec2.internal   Ready    <none>   6m18s   v1.15.11-eks-af3caf
ip-192-168-57-34.ec2.internal    Ready    <none>   6m12s   v1.15.11-eks-af3caf
[root@ip-172-31-49-97 ~]# kubectl get all --all-namespaces
NAMESPACE     NAME                           READY   STATUS    RESTARTS   AGE
kube-system   pod/aws-node-krl4n             1/1     Running   0          6m36s
kube-system   pod/aws-node-z6qw9             1/1     Running   0          6m42s
kube-system   pod/coredns-59dfd6b59f-97dsd   1/1     Running   0          14m
kube-system   pod/coredns-59dfd6b59f-g8snk   1/1     Running   0          14m
kube-system   pod/kube-proxy-299qn           1/1     Running   0          6m42s
kube-system   pod/kube-proxy-r4pwr           1/1     Running   0          6m36s

NAMESPACE     NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)         AGE
default       service/kubernetes   ClusterIP   10.100.0.1    <none>        443/TCP         14m
kube-system   service/kube-dns     ClusterIP   10.100.0.10   <none>        53/UDP,53/TCP   14m

NAMESPACE     NAME                        DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
kube-system   daemonset.apps/aws-node     2         2         2       2            2           <none>          14m
kube-system   daemonset.apps/kube-proxy   2         2         2       2            2           <none>          14m

NAMESPACE     NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
kube-system   deployment.apps/coredns   2/2     2            2           14m

NAMESPACE     NAME                                 DESIRED   CURRENT   READY   AGE
kube-system   replicaset.apps/coredns-59dfd6b59f   2         2         2       14m
[root@ip-172-31-49-97 ~]#
```

REFERENCES

https://redhat-developer-demos.github.io/kubernetes-tutorial/kubernetes-tutorial/pod-rs-deployment.html

QUESTIONS???



THANKS