

# Kubernetes Fundamental & Deployment Local ke Production

Detail Materi

Konsep dan Arsitektur Kubernetes

Indikator :  
Dapat melakukan Integrasi Kubernetes  
dengan AWS. Melakukan Clustering dan  
Deploymen Container dengan Kubernetes

Service AWS untuk Integrasi  
Kubernetes

Installasi Kubectl dan Kops

Clustering Kubernetes dan  
Web Dashboard

Deploying Container dengan YAML

## **Modul Sekolah DevOps Cilsy**

Hak Cipta © 2020 **PT. Cilsy Fiolution Indonesia**

Hak Cipta dilindungi Undang-Undang. Dilarang memperbanyak atau memindahkan sebagian atau seluruh isi buku ini dalam bentuk apapun, baik secara elektronik maupun mekanis, termasuk mecopy, merekam atau dengan sistem penyimpanan lainnya, tanpa izin tertulis dari Penulis dan Penerbit.

Penulis : Iqbal Ilman F., Muhammad Fakhri A., Adi Saputra, Irfan Herfiandana & Tresna W.

Editor: Taufik Maulana, Iqbal Ilman F., Muhammad Fakhri A., Rizal Rahman & Tresna Widiyaman

**Revisi Batch 9**

Penerbit : **PT. Cilsy Fiolution Indonesia**

Web Site : <https://cilsyfiolution.com> , <https://devops.cilsy.id>

### **Sanksi Pelanggaran Pasal 113 Undang-undang Nomor 28 Tahun 2014 tentang Hak Cipta**

1. Setiap orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam pasal 9 ayat (1) huruf i untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan atau pidana denda paling banyak Rp100.000.000,00 (seratus juta rupiah).
2. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak cipta melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf c, huruf d, huruf f, dan atau huruf h, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah)
3. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf a, huruf b, huruf e, dan atau huruf g, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan atau pidana denda paling banyak Rp1.000.000.000,00 (satu miliar rupiah)
4. Setiap orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan atau pidana denda paling banyak Rp4.000.000.000,00 (empat miliar rupiah)

## Daftar Isi

|   |    |
|---|----|
| Modul Sekolah DevOps Cilsy.....                                 | 2  |
| 9. Kubernetes Fundamental & Deployment Local ke Production..... | 6  |
| Learning Outcomes.....  | 6  |
| Outline Materi.....   | 6  |
| 9.1. Kubernetes.....  | 7  |
| 9.1.1. Apa itu Kubernetes ?.....                                | 7  |
| 9.1.2. Arsitektur Kubernetes.....                               | 8  |
| 9.1.3. Fitur Kubernetes.....                                    | 8  |
| 9.1.4. Istilah - Istilah pada kubernetes.....                   | 9  |
| 9.1.5. Fungsi dan Kegunaan kubernetes.....                      | 11 |
| 9.1.6. Exercise.....  | 11 |
| 9.2. Instalasi Kubernetes (Minikube).....                       | 11 |
| 9.2.1. Install Hypervisor.....                                  | 12 |
| 9.2.2. Install Minikube.....                                    | 12 |
| 9.2.2.1. Install Minikube via direct download.....              | 12 |
| 9.2.2.2. Install Minikube using Homebrew.....                   | 13 |
| 9.2.2.3. Confirm Installation.....                              | 13 |
| 9.2.2.4. Clean up local state.....                              | 13 |
| 9.2.3. Kubernetes Basic Command.....                            | 14 |
| 9.2.3.1. Create.....  | 14 |
| 9.2.3.2. Apply.....   | 14 |
| 9.2.3.3. Describe.....  | 16 |
| 9.2.3.4. Get.....   | 17 |
| 9.2.3.5. Delete.....  | 17 |
| 9.2.3.6. Expose.....  | 17 |
| 9.2.3.7. Exec.....  | 18 |
| 9.2.3.8. Logs.....  | 18 |
| 9.3. Kubernetes Menggunakan Amazon EKS.....                     | 19 |



|   |    |
|---|----|
| 9.3.1.1. Penerapan EKS.....                         | 21 |
| 9.4. Membuat Kubernetes Cluster dengan KubeADM..... | 1  |
| 9.4.1. Persiapan Mesin.....                         | 1  |
| 9.4.1.1. Desain Kubernetes.....                     | 1  |
| 9.4.1.2. Spesifikasi Mesin.....                     | 1  |
| 9.4.1.3. Security Group.....                        | 3  |
| 9.4.2. Persiapan Umum.....                          | 3  |
| 9.4.2.1. Generate Kunci Public.....                 | 3  |
| 9.4.2.2. Install Docker.....                        | 3  |
| 9.4.2.3. Install Kubernetes Tools.....              | 4  |
| 9.4.3. Inisiasi Cluster.....                        | 4  |
| 9.4.3.1. Config Control-Planner.....                | 4  |
| 9.4.3.2. Config Worker.....                         | 6  |
| 9.4.4. Persiapan Workspace.....                     | 6  |
| 9.4.4.1. Install kubectl.....                       | 6  |
| 9.4.5. Setup Network.....                           | 6  |
| 9.4.6. Finalisasi Cluster.....                      | 6  |
| 9.5. Persiapan Instalasi Kubernetes pada AWS.....   | 6  |
| 9.5.1. Persiapan EC2 Instance.....                  | 7  |
| 9.5.2. Persiapan Domain Route53.....                | 7  |
| 9.5.3. Persiapan S3 Bucket.....                     | 8  |
| 9.5.4. Instalasi AWS CLI.....                       | 10 |
| 9.6. Instalasi Kubernetes (Metode Kubectl).....     | 10 |
| 9.6.1. Install kubectl.....                         | 10 |
| 9.6.2. Install Kops.....                            | 11 |
| 9.6.3. Expose environmet dari s3.....               | 11 |
| 9.7. Clustering Kubernetes.....                     | 11 |
| 9.7.1. Membuat Cluster Configuration.....           | 11 |
| 9.7.2. Editing Cluster Configuration.....           | 12 |
| 9.7.3. Update Build Clustering.....                 | 14 |
| 9.8. Instalasi Dashboard Kubernetes.....            | 15 |



|  |    |
|--|----|
| 9.8.1. Perintah Kubernetes Lainnya.....    | 16 |
| 9.8.2. Exercise.....                       | 17 |
| 9.9. Konfigurasi Kubernetes.....           | 17 |
| 9.9.1. File Yaml.....                      | 17 |
| 9.9.2. Setting Deployment dengan YAML..... | 18 |
| 9.9.3. Setting Service.....                | 20 |
| 9.9.4. Deployment Wordpress.....           | 22 |
| 9.9.5. Excercise.....                      | 25 |
| 9.10. Summary.....                         | 25 |

## 9.

# Kubernetes Fundamental & Deployment Local ke Production

## Learning Outcomes

Setelah selesai mempelajari bab ini, peserta mampu :

1. Memahami Konsep dan Arsitektur Kubernetes
2. Mempersiapkan Service AWS untuk Integrasi Kubernetes
3. Melakukan Instalasi Kubernetes menggunakan beberapa metode
4. Mempraktikan Clustering Kubernetes dan Web Dashboard
5. Melakukan Deploying Container dengan YAML

## Outline Materi

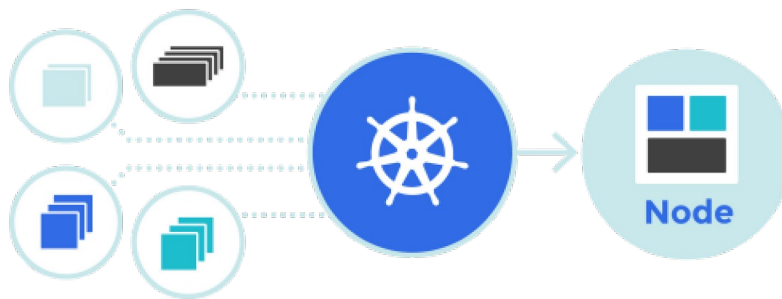
1. Konsep dan Arsitektur Kubernetes
2. Persiapan Instalasi Kubernetes
3. Instalasi Kubernetes
4. Clustering Kubernetes
5. Instalasi Dashboard Kubernetes
6. Konfigurasi YAML
7. Deployment Container



## 9.1. Kubernetes

### 9.1.1. Apa itu Kubernetes ?

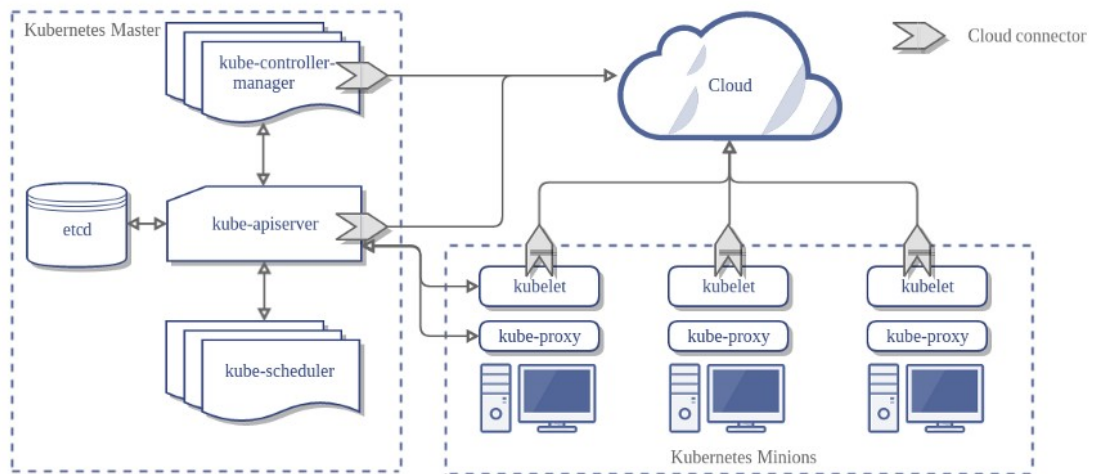
Kubernetes adalah sebuah cluster management open source yang di gunakan untuk mengelola container. Aplikasi ini berasal dari aplikasi internal yang digunakan Google untuk mengelola cluster. Secara bisnis, Kubernetes adalah senjata andalan Google untuk mendongkrak peringkatnya di pasar cloud hosting atau google cloud platform.



*Ilustasi Kubernetes*

Kubernetes berfungsi sebagai mesin untuk menjadwalkan dan menjalankan docker pada server phisical atau virtual server. Kubernetes memberikan infrastruktur kontainer-sentris maksudnya semua aplikasi berjalan dalam kontainer atau docker.

### 9.1.2. Arsitektur Kubernetes



*Arsitektur Kubernetes*

Pada gambar diatas kita bisa lihat diagram arsitektur kubernetes yang memiliki beberapa komponen yang berbeda dan saling terintegrasi satu sama lain diantaranya adalah

1. Kubelet
2. Kubernetes controller manager
3. Kubernetes API server

### 9.1.3. Fitur Kubernetes

#### 1. Automatic Binpacking

Otomatis menempatkan kontainer berdasarkan kebutuhan sumber daya yang dibutuhkan dan tidak mengobarkan ketersediaan resource. mengabungkan kritikal dan beban kerja terbaik untuk meningkatkan pemanfaatan dan menghemat lebih banyak sumber daya.

#### 2. Horizontal Scaling

Melakukan scale up dan down aplikasi menggunakan perintah sederhana, dengan UI, atau dengan otomatis berdasarkan penggunaan cpu.





### **3. Self-healing**

Merestart container yang gagal, menggantikan dan menjadwalkan ulang container saat node worker mati. mematikan container yang tidak merespon pemeriksaan kesehatan yang dibuat oleh pengguna, dan tidak mengadvertise container ke klien hingga siap untuk digunakan.

### **4. Service discovery and load balancing**

Tidak diperlukan memodifikasi aplikasi untuk menggunakan mekanisme service discovery yang tidak dikenal. kubernetes memberikan alamat ip pada container dan memberikan single DNS untuk sekumpulan container dan dapat load balance diantara container.

### **5. Secret and configuration management**

Deploy dan update secret dan konfigurasi aplikasi tanpa rebuild image anda tanpa memberitahukan secret dalam stack konfigurasi.

### **6. Storage orchestration**

Secara otomatis akan mounting ke sistem penyimpanan yang sudah dipilih, baik penyimpanan local, public cloud storage seperti GCP atau AWS dan network storage seperti NFS, iSCSI, Gluster, Ceph, Cinder atau Flocker.

## **9.1.4. Istilah - Istilah pada kubernetes**

Pada aplikasi kubernetes sendiri terdapat beberapa istilah yang mungkin akan sering kita dapatkan, berikut merupakan beberapa istilah beserta arti dari istilah tersebut.

### **1. Deployment**

Deployment menyediakan update deklaratif untuk Pod dan ReplicaSet. Kita bisa mengatur bagaimana pod dibangun, berapa banyak replika dari pod dan sebagainya.



## 2. Replicasets

Replika dari pod yang didefinisikan pada Deployment

## 3. Pod

Pod merupakan grup container instance. Kita bisa menjalankan beberapa container (misalnya aplikasi web + Database) dalam satu pod. Di antara container dalam satu pod bisa saling mengakses dengan menggunakan alamat localhost, bisa di katakan pod seperti laptop yang kita pakai coding.

## 4. Node

Node adalah penyebutan dari mesin-mesin yang di gunakan. Mesin ini bisa saja mesin virtual (seperti VPS atau Instance) atau mesin fisik.

## 5. Service

Service merupakan mekanisme untuk mengekspos pod kita ke dunia luar. Aplikasi kita yang berjalan dalam pod tidak memiliki alamat IP yang tetap. Agar bisa diakses oleh aplikasi lain atau oleh user, kita perlu alamat IP yang tetap. Service menyediakan alamat IP yang tetap, yang nantinya akan kita arahkan ke pod kita dengan menggunakan selector.

## 6. Label

Label adalah seperangkat informasi metadata untuk mencari pod tertentu.

## 7. App-Test

Kita buat label app yang isinya adalah nama aplikasi. Semua container, pod, dan service yang menjadi bagian dari aplikasi test kita beri label app=test.

## 8. Stage-production



Label stage bisa kita gunakan untuk menentukan berbagai konfigurasi environment deployment aplikasi kita, misalnya development, testing, performancetest, securitytest, dan production.

## **9. Jenis-frontend**

kita bisa membuat label jenis aplikasi, misalnya frontend, cache, database, filesaver, dan sebagainya.

## **10. Selector**

Selector adalah filtering menggunakan label. Misalnya kita ingin mencari semua instance database untuk aplikasi test yang berjalan di production.

### **9.1.5. Fungsi dan Kegunaan kubernetes**

Kubernetes dapat menjadwalkan dan menjalankan kontainer aplikasi pada kelompok mesin fisik atau virtual. Kubernetes memberikan keuntungan dan manfaat penuh terhadap Container. Kubernetes menyediakan infrastruktur untuk membangun lingkungan pengembangan yang benar-benar kontainer-sentris.

### **9.1.6. Exercise**

1. Jelaskan menurut kalian perbedaan antara kubernetes dengan Docker !
2. Jelaskan peran aplikasi Kubernetes dalam dunia devops !

## **9.2. Instalasi Kubernetes (Minikube)**

Minikube adalah alat yang dapat dengan mudah menjalankan Kubernetes di komputer lokal. Minikube menjalankan node cluster Kubernetes di mesin virtual (VM) di laptop untuk pengguna yang ingin mencoba atau mengembangkan Kubernetes.



Meski minikube dijalankan pada lokal, fitur yang disediakan cukup merepresentasikan Kubernetes pada implementasi nyatanya. Fitur tersebut antara lain :

- DNS
- NodePort
- Configmap
- *Dashboard*
- *Container runtime* : Docker, CRI-O, dan containerd
- CNI (Container Network Interface)
- Ingress

### 9.2.1. Install Hypervisor

Jika kita belum menginstal hypervisor, install salah satunya sekarang:

- KVM, yang juga menggunakan QEMU
- VirtualBox

Minikube juga mendukung opsi **--vm-driver = none** yang menjalankan komponen Kubernetes di host dan bukan di VM. Menggunakan driver ini membutuhkan Docker dan lingkungan Linux tetapi bukan hypervisor.

### 9.2.2. Install Minikube

Pastikan kita sudah menginstall kubectl pada server-server

#### 9.2.2.1. Install Minikube via direct download

Kita akan install minikube via direct download kita bisa download stand alone binary menggunakan perintah berikut

```
$ curl -Lo minikube
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 \
&& chmod +x minikube
```

Berikut cara mudah untuk menambahkan Minikube



```
$ sudo mkdir -p /usr/local/bin/
$ sudo install minikube /usr/local/bin/
```

#### **9.2.2.2. Install Minikube using Homebrew**

Sebagai alternatif lain, kita dapat menginstal Minikube menggunakan Linux Homebrew dengan perintah berikut

```
$ brew install minikube
```

#### **9.2.2.3. Confirm Installation**

Untuk mengkonfirmasi pemasangan hypervisor dan Minikube yang berhasil, kita dapat menjalankan perintah berikut untuk memulai local Kubernetes cluster:

```
$ minikube start --vm-driver=<driver_name>
```

Setelah minikube mulai selesai, jalankan perintah di bawah ini untuk memeriksa status cluster:

```
$ minikube status
```

Jika cluster kita berjalan, output dari status minikube sebagai berikut

```
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
timeToStop: Nonexistent
```

untuk memberhentikan service dari minikube kita bisa masukan perintah berikut :

```
$ minikube stop
```

#### **9.2.2.4. Clean up local state**

Jika sebelumnya telah menginstal Minikube, dan jalankan perintah berikut

```
$ minikube start
```

dan setelah melakukan perintah diatas muncul error berikut



machine does not exist

kita perlu melakukan clear minikube local state dengan perintah berikut

```
$ minikube delete
```

### 9.2.3. Kubernetes Basic Command

Kita akan berkenalan dengan salah satu tool yang Kubernetes berikan untuk manage dan mengontrol kluster **Kubernetes** kita. Ya, namanya adalah **kubectl**. Kubectl sering digunakan untuk membuat resource untuk kluster seperti membuat sebuah service, deployment, pod, scaling, persistent volume, hingga melihat info dari resource tersebut. Kita juga dapat menggunakan kubectl untuk menkonfigurasi sebuah resource dari file berekstensi yaml.

#### 9.2.3.1. Create

Perintah yang paling umum adalah **kubectl create** dimana perintah ini digunakan untuk membuat resource pada Kubernetes. Contohnya adalah deployment.

Kubectl create deployment [nama deployment] --image=[container image]

```
controlplane $ kubectl create deployment nginx-depl --image=nginx
deployment.apps/nginx-depl created
```

perintah tersebut akan membuat **deployment** dengan nama nginx-depl, **replicasets** dan **pod** dengan image nginx.

#### 9.2.3.2. Apply

Perintah lainnya adalah kubectl **apply**. Perintah ini digunakan untuk menerapkan sebuah konfigurasi ke resource cluster menggunakan file konfigurasi atau stdin. Kegunaan perintah ini hampir mirip dengan **kubectl create** yaitu membuat suatu resource kluster, namun jika resource tersebut membutuhkan opsi atau atribut yang tidak dapat dilakukan oleh perintah



**kubectl create**, maka kita gunakan **kubectl apply**. Perintah ini sangat direkomendasikan untuk production. Misalnya saya ingin membuat deployment dengan file bernama **nginx-prod.yaml**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-prod
  labels:
    app: nginx-prod
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-prod
  template:
    metadata:
      labels:
        app: nginx-prod
    spec:
      containers:
        - name: nginx-prod
          image: nginx
          ports:
            - containerPort: 80
```

Script dapat dilihat di

<https://gist.github.com/sdcilsy/ae521c9b488429fae8409a5b0c788540>

kita bisa menambahkan resource tersebut dengan mengeksekusi perintah

**kubectl apply -f nginx-prod.yaml**.

```
controlplane $ kubectl apply -f nginx-prod.yaml
deployment.apps/nginx-prod created
```



### 9.2.3.3. Describe

Perintah selanjutnya adalah **kubectl describe** dimana perintah ini digunakan untuk memberikan detail resource pada kluster. Misalnya adalah detail **deployment** seperti berikut.

Kubectl describe [resource] [resource name]

```
controlplane $ kubectl describe deployment nginx-depl
Name:                nginx-depl
Namespace:           default
CreationTimestamp:    Wed, 24 Mar 2021 19:17:28 +0700
Labels:              app=nginx-depl
Annotations:         deployment.kubernetes.io/revision: 1
Selector:            app=nginx-depl
Replicas:            1 desired | 1 updated | 1 total | 1 available | 0
unavailable
StrategyType:        RollingUpdate
MinReadySeconds:     0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx-depl
  Containers:
    nginx:
      Image:      nginx
      Port:       <none>
      Host Port:  <none>
      Environment: <none>
      Mounts:      <none>
      Volumes:     <none>
Conditions:
  Type           Status  Reason
  ----           -
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
```





```
OldReplicaSets: <none>
NewReplicaSet:  nginx-depl-5c8bf76b5b (1/1 replicas created)
Events:
  Type            Reason              Age             From              Message
  ----            -
  Normal          ScalingReplicaSet   7m14s          deployment-controller  Scaled up replica set nginx-depl-5c8bf76b5b to 1
```

### 9.2.3.4. Get

Perintah lainnya yang tak kalah penting adalah **kubectl get**. Perintah ini digunakan untuk menampilkan resource pada kluster. Berbeda dengan perintah **describe**, **get** lebih menampilkan resource secara general. Misalnya saya ingin melihat ada berapa pod yang berjalan dengan perintah **kubectl get pod**.

```
controlplane $ kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-depl-84dd4db-22qk8            1/1     Running   0           11m
nginx-depl-84dd4db-8s667            1/1     Running   0           11m
nginx-depl-84dd4db-st8c8            1/1     Running   0           11m
```

### 9.2.3.5. Delete

Jika ingin menghapus resource pada kluster, kita dapat menggunakan perintah **kubectl delete**. Misalnya saya ingin menghapus deployment dengan nama **nginx-depl** dengan perintah **kubectl delete deployment nginx-depl**.

```
controlplane $ kubectl delete deployment nginx-depl
deployment.apps "nginx-depl" deleted
```

### 9.2.3.6. Expose

Pod pada kubernetes secara default memang hanya bisa diakses oleh network pada jaringan kluster Kubernetes saja. Agar pod dapat diakses oleh jaringan



luar. maka kita bisa membuat service menggunakan perintah `kubectl expose [resource] [nama resource] --port=[port fisik] --target-port=[port container]`

```
controlplane $ kubectl expose deployment nginx-depl --port=80 --target-port=80
service/nginx-depl exposed
```

### 9.2.3.7. Exec

Ketika kita ingin mengeksekusi perintah didalam kontainer, maka kita bisa menggunakan perintah **kubectl exec**. Misalnya saya ingin mengeksekusi perintah **date** pada pod dengan nama **nginx-depl-5c8bf76b5b-q4f69**, maka kita bisa menggunakan perintah berikut

```
controlplane $ kubectl exec nginx-depl-5c8bf76b5b-q4f69 -- date
Wed Mar 24 13:19:22 UTC 2021
```

### 9.2.3.8. Logs

Selanjutnya adalah perintah `kubectl logs` yang digunakan untuk menampilkan log dari container pada pod. Misalnya saya ingin melihat log pada pod **nginx-depl-5c8bf76b5b-q4f69** maka kita bisa menggunakan perintah seperti berikut.

```
Controlpanel $ kubectl logs nginx-depl-5c8bf76b5b-q4f69
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to
perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-
default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of
/etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in
/etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-
templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-
processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
```



### 9.3. Kubernetes Menggunakan Amazon EKS

Amazon Elastic Container Services for Kubernetes (Amazon EKS) adalah layanan Kubernetes terkelola penuh. Pelanggan seperti Intel, Snap, Intuit, GoDaddy, dan Autodesk mempercayai EKS untuk menjalankan aplikasi paling sensitif dan misi kritis karena keamanan, keandalan, dan skalabilitasnya.

EKS adalah tempat terbaik menjalankan Kubernetes karena beberapa alasan. Pertama, Kita dapat memilih untuk menjalankan cluster EKS dengan menggunakan AWS Fargate, yang merupakan komputasi tanpa server untuk kontainer. Fargate menghilangkan perlunya menyediakan dan mengelola server, memungkinkan kita menentukan dan membayar sumber daya per aplikasi, dan meningkatkan keamanan melalui isolasi aplikasi sesuai desain. Kedua, EKS sangat terintegrasi dengan layanan seperti Amazon CloudWatch, Auto Scaling Groups, AWS Identity and Access Management (IAM), dan Amazon Virtual Private Cloud (VPC), Kita tidak perlu repot untuk memantau, menskalakan, dan menyeimbangkan beban aplikasi Kita. Ketiga, EKS terintegrasi dengan AWS App Mesh dan menghadirkan pengalaman asli Kubernetes dalam penggunaan fitur-fitur layanan mesh dan menghadirkan observabilitas yang kaya, kontrol trafik, dan fitur keamanan untuk aplikasi. Selain itu, EKS menyediakan bidang kontrol terskala dan sangat tersedia yang beroperasi di beberapa availability zone untuk menghilangkan satu titik kegagalan.

EKS menjalankan Kubernetes hulu dan bersertifikat sesuai dengan Kubernetes sehingga Kita dapat memanfaatkan semua kelebihan tooling sumber terbuka dari komunitas. Migrasi aplikasi Kubernetes standar apa pun ke EKS dapat dilakukan dengan mudah tanpa perlu mengubah kode yang kita buat. Ada beberapa keuntungan untuk ketika kita menggunakan EKS antara lain :

#### 1. Ketersediaan Tinggi

Amazon EKS menjalankan infrastruktur manajemen Kubernetes di beberapa AWS Availability Zone, secara otomatis mendeteksi dan



mengganti simpul bidang kontrol yang tidak sehat, dan memberikan pemutakhiran dan patching sesuai permintaan.

## 2. Opsi tanpa server

EKS mendukung AWS Fargate untuk menyediakan komputasi tanpa server untuk kontainer. Fargate menghilangkan perlunya menyediakan dan mengelola server, memungkinkan Anda menentukan dan membayar sumber daya per aplikasi, dan meningkatkan keamanan melalui isolasi aplikasi sesuai desain.

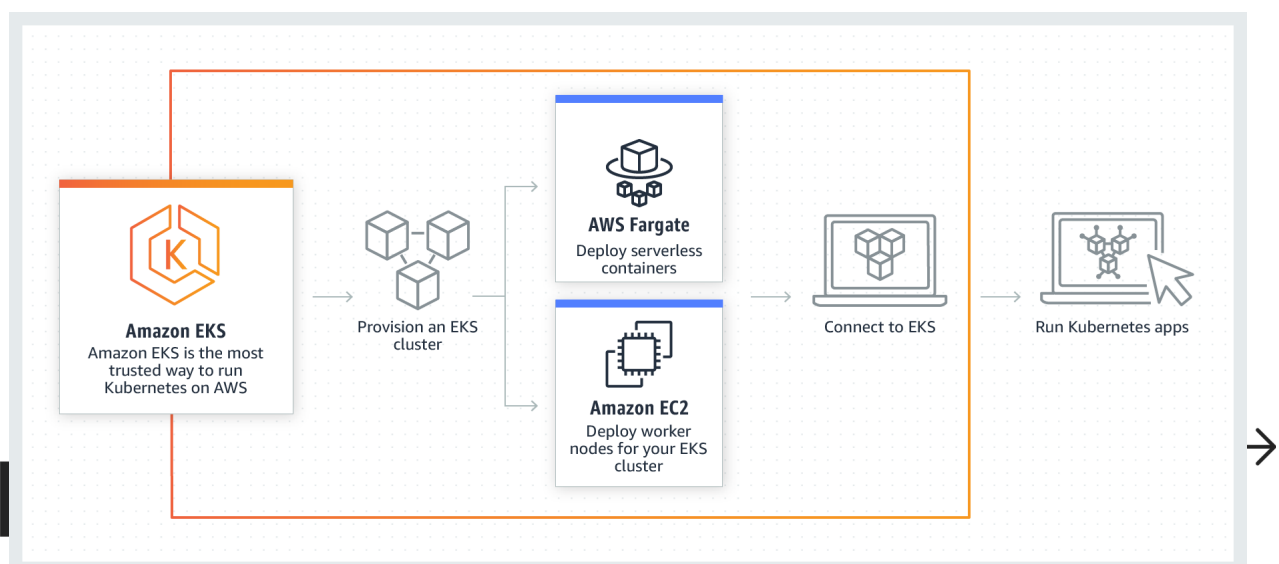
## 3. Aman

EKS secara otomatis menerapkan patch keamanan terbaru untuk bidang kontrol kluster Kita. AWS juga bekerja erat dengan komunitas untuk memastikan masalah keamanan kritis diatasi sebelum rilis dan patch baru diterapkan ke kluster yang ada.

## 4. Dibangun bersama Komunitas

Amazon EKS menjalankan Kubernetes hulu dan bersertifikat sesuai dengan Kubernetes, sehingga aplikasi yang dikelola dengan EKS sepenuhnya kompatibel dengan aplikasi yang dikelola dengan lingkungan Kubernetes standar. AWS aktif bekerja sama dengan komunitas Kubernetes, termasuk berkontribusi pada basis kode Kubernetes yang membantu Anda memanfaatkan layanan dan fitur AWS.

Berikut cara kerja dari EKS :



*Ilustarsi Cara Kerja EKS*

### **9.3.1.1. Penerapan EKS**

Beberapa penerapan EKS antara lain :

#### **1. Penerapan Hibrid**

Kita dapat menggunakan EKS pada AWS Outposts untuk menjalankan aplikasi berkontainer yang mensyaratkan latensi rendah pada sistem lokal. AWS Outposts adalah layanan terkelola penuh yang memperluas infrastruktur AWS, layanan AWS, API, dan alat untuk hampir semua situs yang terhubung. Dengan EKS pada Outposts, Anda dapat mengelola kontainer lokal yang sama mudahnya dengan Anda mengelola kontainer di cloud.

#### **2. Machine Learning**

Kubeflow dengan EKS dapat digunakan untuk pemodelan alur kerja pembelajaran mesin Anda dan menjalankan tugas pelatihan yang didistribusikan secara efisien dengan menggunakan jenis instans EC2 terbaru yang ditenagai GPU. Anda juga dapat menggunakan AWS Deep Learning Containers untuk menjalankan pelatihan dan inferensi dalam TensorFlow pada EKS.

#### **3. Pemrosesan Batch**

Kita dapat menjalankan beban kerja sekuensial atau paralel di kluster EKS dengan menggunakan Kubernetes Jobs API. Dengan EKS, Kita dapat merencanakan, menjadwalkan, dan mengeksekusi beban kerja komputasi batch yang ada di seluruh rentang layanan dan fitur komputasi AWS, seperti Amazon EC2 dan Instans Spot.

#### **4. Aplikasi Web**

Kita dapat membangun aplikasi web yang secara otomatis naik dan turun dan berjalan dalam konfigurasi yang sangat tersedia di beberapa Availability Zone. Dengan berjalan di EKS, aplikasi web yang kita bangun beroleh manfaat dari kinerja, skala, keandalan, dan ketersediaan AWS.



Selain itu, layanan kita mendapatkan integrasi kontan dengan layanan jaringan dan keamanan AWS, seperti Application Load Balancers untuk distribusi beban aplikasi web Anda dan VPC untuk jaringan.

Contoh Perusahaan yang menggunakan EKS:



*Contoh Perusahaan Menggunakan EKS*



## 9.4. Membuat Kubernetes Cluster dengan KubeADM

Bahan bacaan:

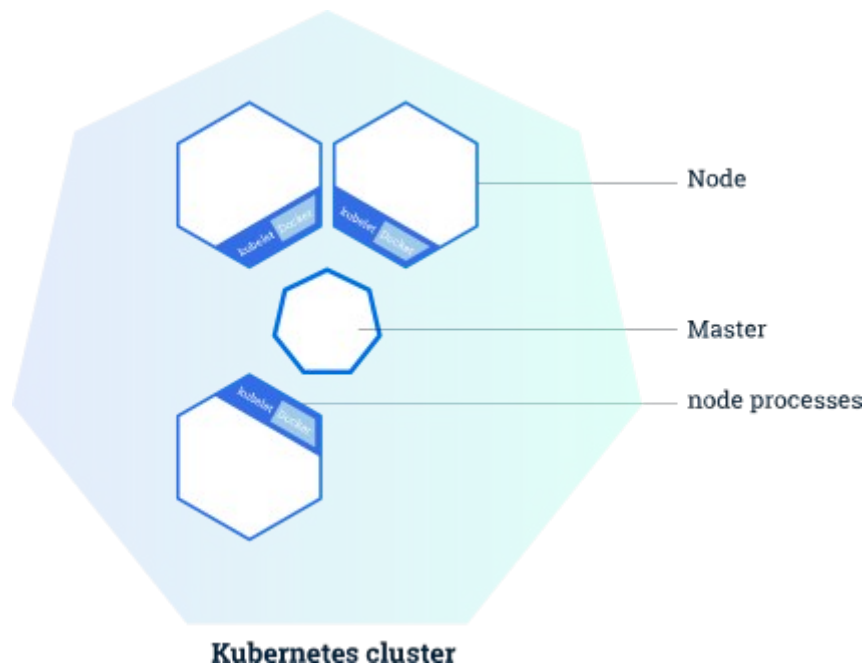
<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/>

### 9.4.1. Persiapan Mesin

\* catatan: jika tidak diminta, silahkan lakukan step-step berikut sebagai user biasa, tanpa tambahan sudo.

Siapkan 3 buah mesin untuk percobaan ini. 1 mesin sebagai control-planner dan 2 mesin sebagai worker.

#### 9.4.1.1. Desain Kubernetes



#### 9.4.1.2. Spesifikasi Mesin

Lakukan penggantian hostname dengan perintah:

```
sudo su  
hostnamectl set-hostname master-01
```



|                |           |
|----------------|-----------|
| Hostname       | master-01 |
| IP (Wajib EIP) |           |
| Core           | 2         |
| RAM            | 4         |
| HDD            | 40        |

|                |         |
|----------------|---------|
| Hostname       | node-01 |
| IP (Wajib EIP) |         |
| Core           | 2       |
| RAM            | 4       |
| HDD            | 40      |

|                |           |
|----------------|-----------|
| Hostname       | master-01 |
| IP (Wajib EIP) |           |
| Core           | 2         |
| RAM (GB)       | 4         |





|     |    |
|-----|----|
|     |    |
| HDD | 40 |

### 9.4.1.3. Security Group

Untuk percobaan kali ini, security group kita **set open all**.

## 9.4.2. Persiapan Umum

Langkah berikut dilakukan di ketiga mesin.

### 9.4.2.1. Generate Kunci Public

Kunci ini akan digunakan untuk semua mesin, jika sudah ada, gunakan yang ada saja.

```
mkdir .ssh
vim .ssh/authorized_keys
vim .ssh/id_rsa
vim .ssh/id_rsa.pub
chmod 600 .ssh/id_rsa
chmod 644 .ssh/id_rsa.pub .ssh/authorized_keys
```

### 9.4.2.2. Install Docker

```
sudo su
apt autoremove -y
apt-get update && apt-get install apt-transport-https ca-certificates curl
software-properties-common -y
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
sudo apt-get update && sudo apt-get install docker-ce=18.06.2~ce~3-0~ubuntu -y
```



```
cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
mkdir -p /etc/systemd/system/docker.service.d
systemctl daemon-reload
systemctl restart docker
usermod -aG docker ubuntu
exit
```

### 9.4.2.3. Install Kubernetes Tools

```
sudo su
dpkg-reconfigure tzdata
apt-get update && apt-get install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
apt-get update
apt-get install -y kubelet kubeadm kubectl nfs-client
apt-mark hold kubelet kubeadm kubectl
swapoff -a
sed -i '/ swap / s/^/#/' /etc/fstab
exit
```

## 9.4.3. Inisiasi Cluster

### 9.4.3.1. Config Control-Planner

Perintah berikut dilakukan di mesin master-01



```
sudo su
kubeadm init --pod-network-cidr=192.168.0.0/16
```

Proses itu akan menghasilkan output kira-kira seperti ini:

```
Your Kubernetes control-plane has initialized successfully!
```

To start using your cluster, you need to run the following as a regular user:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

<https://kubernetes.io/docs/concepts/cluster-administration/addons/>

You can now join any number of the control-plane node running the following command on each as root:

```
kubeadm join 102.167.89.890:6443 --token pjl744.r7x90432jdfsqqf \
    --discovery-token-ca-cert-hash
sha256:14b8dd29ac36bcc3bd04eaefe93175e7fb16dc4a471a7e0556dd1c21e6ba5632 \
    --control-plane --certificate-key
e0018a5b829c5f79961bada5c92559d76c763bf1da06e3b8482d44e4bb60bb5b
```

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!

As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use

"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

Then you can join any number of worker nodes by running the following on each as root:

```
kubeadm join 102.167.89.890:6443 --token pjl744.r7x90432jdfsqqf \
    --discovery-token-ca-cert-hash
sha256:14b8dd29ac36bcc3bd04eaefe93175e7fb16dc4a471a7e0556dd1c21e6ba5632
```

### 9.4.3.2. Config Worker

Perintah ini dilakukan di setiap worker:

```
sudo su
kubeadm join 102.167.89.890:6443 --token pjl744.r7x90432jdfsqqg \
  --discovery-token-ca-cert-hash
sha256:14b8dd29ac36bcc3bd04eaefe93175e7fb16dc4a471a7e0556dd1c21e6ba5632
```

### 9.4.4. Persiapan Workspace

Workspace adalah laptop atau pc yang kita gunakan untuk meremote K8S Cluster.

#### 9.4.4.1. Install kubectl

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/
amd64/kubectl
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl
kubectl version --client
```

Salin file .kube/config dari master-01

```
mkdir ~/.kube
vim ~/.kube/config
```

### 9.4.5. Setup Network

Perintah ini dilakukan dari terminal workspace.

```
kubectl apply -f https://docs.projectcalico.org/v3.11/manifests/calico.yaml
```

### 9.4.6. Finalisasi Cluster

```
kubectl get nodes
```

## 9.5. Persiapan Instalasi Kubernetes pada AWS

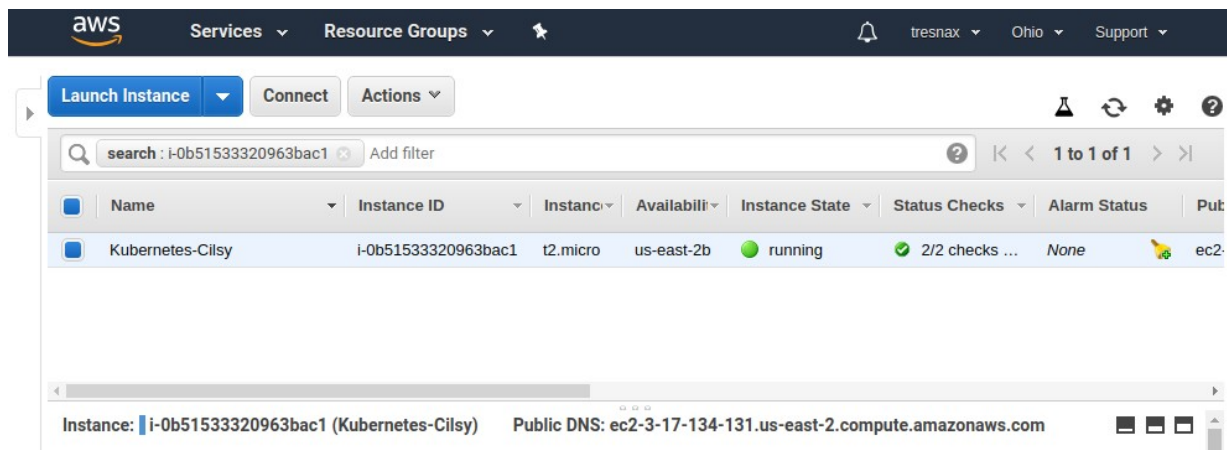
Pada tahap ini kita akan coba untuk melakuakn installasi Kubernetes pada AWS, untuk melakukan installasi kita perlu melakukan persipan terlebih dahulu. Berikut merupakan bagian yang harus kita siapkan :



1. 1 Buah EC2
2. 1 DNS Route 53 pada AWS
3. 1 Buah Bucket S3
4. AWS CLI yang sudah di installkan dikomputer/Instance

### 9.5.1. Persiapan EC2 Instance

Berikut merupakan hasil pembuatan instance, instance yang kita buat disini nantinya akan kita akan installkan kubernetes dan juga beberapa installasi lain didalamnya.



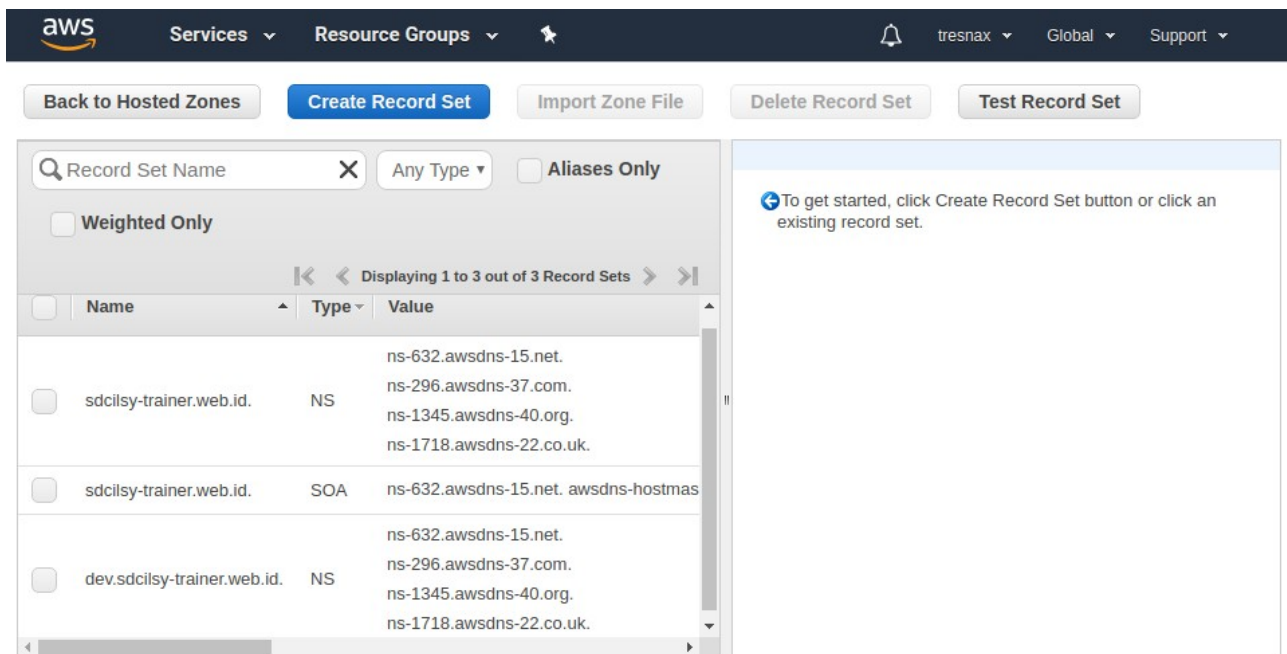
*Persiapan sebuah EC2 Instance*

### 9.5.2. Persiapan Domain Route53

Domain yang akan digunakan disini adalah **sdCILSY-trainer.web.id**, kalian dapat menyesuaikan domain yang kalian miliki baik pribadi maupun yang dengan domain yang cilsy sudah berikan.

Setting domain dan sisipkan Name Server Route53 pada Provider yang menyediakan domain, setelah itu buat subdomain dengan nama **dev.sdCILSY-trainer.web.id** pada Route53. Arahkan subdomain pada NS main domain hingga seperti dibawah ini.





*Hasil Konfigurasi Domain Route53*

Kops menggunakan DNS untuk melakukan discovery didalam clusternya sehingga kita dapat mencapai server API kubernetes dari client. Jika kita menggunakan kops, kita harus menggunakan nama cluster dengan nama DNS yang valid, sehingga kita tidak akan membuat bingung cluster dengan alamat IP Address dan kita dapat membagi cluster secara jelas.

Kops sendiri merekomendasikan kita menggunakan subdomain untuk membagi cluster misalkan **dev.sdccilsy-trainer.web.id**. Ini dikarenakan agar cluster kita nantinya tidak mengganggu domain intinya dan menyebabkan kekacauan.

Kita dapat mengecek NS subdomain yang sudah kita buat tadi di Route53 dengan menggunakan perintah berikut pada instance.

```
dig NS dev.sdccilsy-trainer.web.id
```

Anda akan melihat ada 4 Name Server yang tercatat oleh Route53. Apabila tidak muncul maka anda bisa menunggu prosesnya 5-10 menit kemudian.

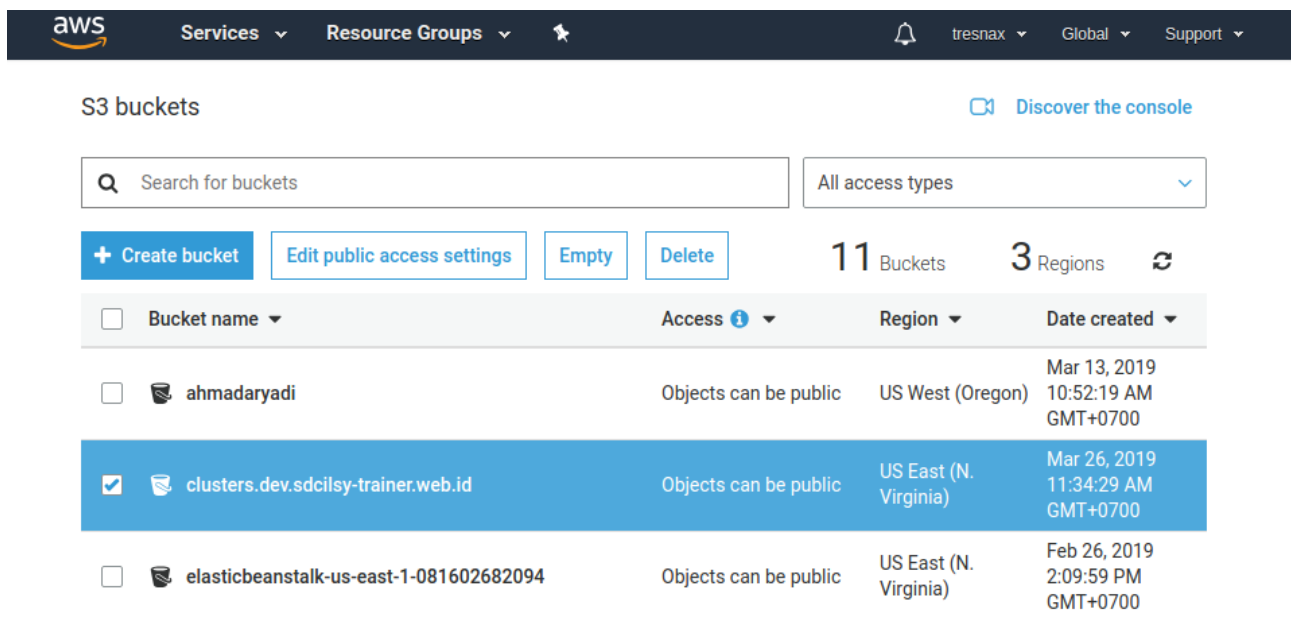
### 9.5.3. Persiapan S3 Bucket

Langkah selanjutnya adalah membuat file penyimpanan config kubernetes pada S3. Tempat penyimpanan config kubernetes sendiri akan di simpan pada



S3, karena EC2 yang kita pakai nanti akan sangat dinamis, dapat berkurang dan bertambah dengan cepat sesuai kebutuhan maka tidak memungkinkan jika kita menyimpan konfigurasi disana.

Disini kita akan membuat S3 Bucket dengan nama yang mengarah kepada domain yang kita miliki, sini kita akan buat bucket tersebut dengan nama **clusters.dev.sdcilsy-trainer.web.id**. Hingga hasilnya bisa kita lihat seperti dibawah ini.



The screenshot shows the AWS Management Console interface for S3 buckets. At the top, there's a navigation bar with 'aws' logo, 'Services', 'Resource Groups', and user information. Below the navigation bar, the 'S3 buckets' section is active. A search bar and a dropdown for 'All access types' are visible. Below these, there are buttons for '+ Create bucket', 'Edit public access settings', 'Empty', and 'Delete'. A summary shows '11 Buckets' and '3 Regions'. A table lists the buckets:

| Bucket name   | Access                | Region                | Date created                      |
|---|-----------------------|-----------------------|-----------------------------------|
| <input type="checkbox"/> ahmadaryadi                                    | Objects can be public | US West (Oregon)      | Mar 13, 2019 10:52:19 AM GMT+0700 |
| <input checked="" type="checkbox"/> clusters.dev.sdcilsy-trainer.web.id | Objects can be public | US East (N. Virginia) | Mar 26, 2019 11:34:29 AM GMT+0700 |
| <input type="checkbox"/> elasticbeanstalk-us-east-1-081602682094        | Objects can be public | US East (N. Virginia) | Feb 26, 2019 2:09:59 PM GMT+0700  |

*Hasil S3 Bucket yang dibuat*

Selain menggunakan console di web AWS, kita bisa juga membuat S3 bucket dengan mudah menggunakan perintah awscli di server yang kita miliki.

```
aws s3 mb s3://clusters.dev.sdcilsy-trainer.web.id
```

Tapi untuk menggunakan perintah aws di terminal, kita harus sudah menginstallkan AWSCLI terlebih dahulu di server kita.

Bucket yang sudah kita buat bisa dipanggil juga dengan nama **s3://clusters.dev.sdcilsy-trainer.web.id**.



### 9.5.4. Instalasi AWS CLI

Langkah selanjutnya dengan menginstall AWS CLI, supaya kita dapat berkomunikasi dengan akun yang kita miliki maka kita harus menkonfigurasi credential pada aws, untuk mendapatkan credential bisa kita lihat dibagian IAM AWS.

Untuk melakukan instalasi AWS CLI bisa menggunakan perintah berikut.

```
#sudo apt-get update
#sudo apt-get install awscli
#aws configure
```

```
ubuntu@ip-172-31-28-164:~$ aws configure
AWS Access Key ID [*****CRZA]: AKIAII4HR022EV7ECRZA
AWS Secret Access Key [*****pZaM]: AthWnkHBegyH6FpQQ5ZklN4QSZTH+M4E09
o4/khq
Default region name [us-east-1a]: us-east-1a
Default output format [None]:
ubuntu@ip-172-31-28-164:~$
```

Setelah langkah-langkah di atas kita lakukan maka hal selanjutnya yang kita lakukan adalah dengan menginstall kubectl dan kops.

## 9.6. Instalasi Kubernetes (Metode Kubectl)

### 9.6.1. Install kubectl

Sekarang masih didalam instance yang kita miliki, kita installkan Kubectl dengan menggunakan perintah berikut.

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/
amd64/kubectl
```

Setelah berhasil di install, kita berikan hak akses pada kubectl lalu pindahkan kubectl ke direktori /usr/local/kubectl seperti dibawah ini.

```
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl
```





### 9.6.2. Install Kops

Pada tahap ini kita coba untuk menginstall bagian Kops, pertama kita download terlebih dahulu, setelah itu berikan hak akses dan pindahkan kops yang sudah didownload ke directory /usr/local/bin/kops seperti dibawah ini.

```
wget https://github.com/kubernetes/kops/releases/download/1.8.1/kops-linux-amd64
chmod +x kops-linux-amd64
sudo mv kops-linux-amd64 /usr/local/bin/kops
```

Setelah menginstall kubectl dan kops langkah selanjutnya adalah dengan membuat cluster kubernetes

### 9.6.3. Expose environmet dari s3

Pada bagian ini kita akan coba untuk membuat sebuah cluster dari kubernetes yang sudah kita installkan tadi, pertama kita akan expose environment-nya terlebih dahulu. Untuk membuatnya kita gundakan perintah berikut dengan memasukan alamat bucket S3 yang sudah kita buat tadi.

```
export KOPS_STATE_STORE=s3://clusters.dev.sdcilsy-trainer.web.id
```

## 9.7. Clustering Kubernetes

### 9.7.1. Membuat Cluster Configuration

Setelah kita melakukan export storage S3, selanjutnya kita akan coba membuat cluser. Tapi sebelum itu kita harus buat konfigurasi pada cluster yang akan kita buat terlebih dahulu.

Untuk sederhananya kita dapat menggunakan perintah berikut untuk membuat konfigurasi kops.

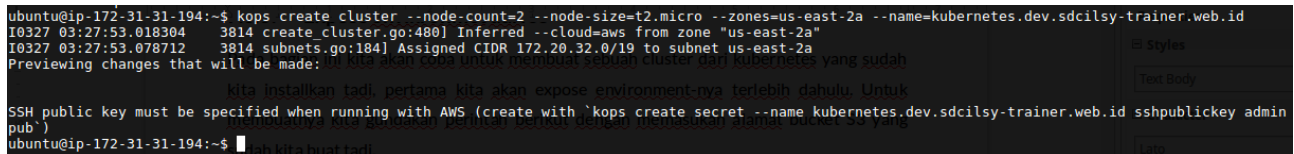
```
kops create cluster --cloud=aws --zones=us-east-2a kubernetes.dev.sdcilsy-trainer.web.id
```

Secara default nantinya dia akan membuat konfigurasi untuk 2 node, 1 master dengan type instance t2.micro untuk node dan t2.medium untuk master. Akan tetapi apabila kita ingin memasukan informasi lebih spesifik seperti jumlah



node yang akan dibuat dan jenis instance yang akan dibuat, kalian dapat menggunakan perintah seperti dibawah ini.

```
kops create cluster --node-count=3 --node-size=t2.micro --master-size=t2.micro
--zones=us-east-2a --name kubernetes.dev.sdcilsy-trainer.web.id
```



### Hasil konfigurasi clusters

Apabila terjadi error pada ssh seperti dibawah ini

Gunakan perintah dibawah ini untuk melakukan generate pada keygennya, setelah itu coba kembali mebuat konfigurasi clusternya.

```
ssh-keygen -t rsa
```

## 9.7.2. Editing Cluster Configuration

Selain membuat konfigurasi, kita juga dapat melihat hasil konfigurasinya dalam bentuk file script. Kita dapat menggunakan perintah berikut.

```
kops edit cluster --name kubernetes.dev.sdcilsy-trainer.web.id
```

Disini kita akan diberikan file konfigurasi yang dibuka menggunakan editor vim, konfigurasinya bisa dilihat seperti dibawah ini.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this
# file will be
# reopened with the relevant failures.
#
apiVersion: kops/v1alpha2
kind: Cluster
metadata:
  creationTimestamp: 2019-03-27T03:27:53Z
```



```

name: kubernetes.dev.sdcilsy-trainer.web.id
spec:
  api:
    dns: {}
  authorization:
    rbac: {}
  channel: stable
  cloudProvider: aws
  configBase: s3://clusters.dev.sdcilsy-trainer.web.id/kubernetes.dev.sdcilsy-
trainer.web.id
  etcdClusters:
    - etcdMembers:
        - instanceGroup: master-us-east-2a
          name: a
          name: main
    - etcdMembers:
        - instanceGroup: master-us-east-2a
          name: a
          name: events
  iam:
    allowContainerRegistry: true
    legacy: false
  kubernetesApiAccess:
    - 0.0.0.0/0
  kubernetesVersion: 1.10.13
  masterInternalName: api.internal.kubernetes.dev.sdcilsy-trainer.web.id
  masterPublicName: api.kubernetes.dev.sdcilsy-trainer.web.id
  networkCIDR: 172.20.0.0/16
  networking:
    kubenet: {}
  nonMasqueradeCIDR: 100.64.0.0/10
  sshAccess:
    - 0.0.0.0/0
  subnets:
    - cidr: 172.20.32.0/19

```



```

name: us-east-2a
type: Public
zone: us-east-2a
topology:
  dns:
    type: Public
  masters: public
  nodes: public

```

script dapat dilihat di

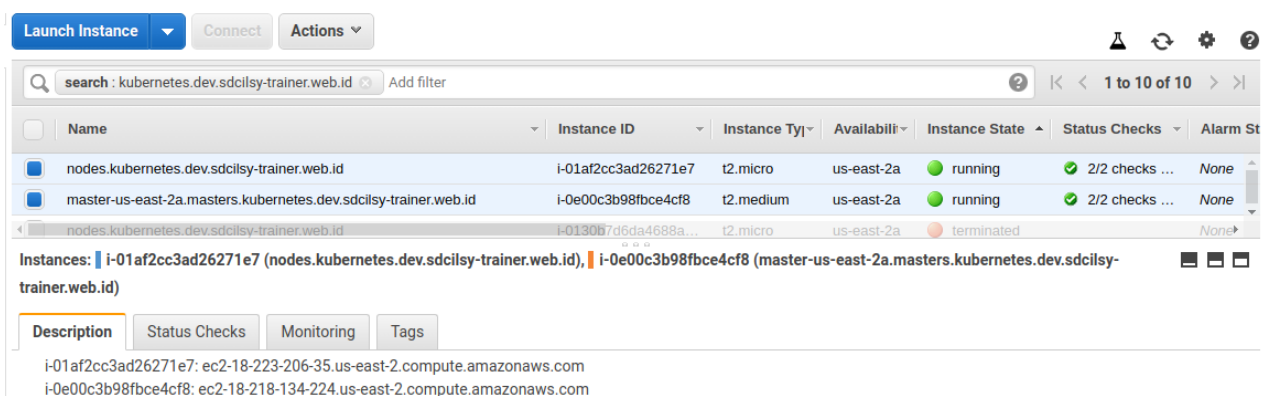
<https://gist.github.com/sdcilsy/7841dbd32e9357391eff6b059f516dfd>

### 9.7.3. Update Build Clustering

Setelah tadi kita buat konfigurasinya, sekarang kita build clusternya dengan melakukan update cluster menggunakan perintah berikut.

```
kops update cluster kubernetes.dev.sdcilsy-trainer.web.id --yes
```

Untuk hasil dari pembuatan cluster tersebut kita bisa langsung buka Console Web AWS dan lihat pada bagian instance yang kita set, disini kami menggunakan region us-east-2 (Oregon).



| Name  | Instance ID          | Instance Type | Availability | Instance State | Status Checks  | Alarm St |
|---|----------------------|---------------|--------------|----------------|----------------|----------|
| nodes.kubernetes.dev.sdcilsy-trainer.web.id                     | i-01af2cc3ad26271e7  | t2.micro      | us-east-2a   | running        | 2/2 checks ... | None     |
| master-us-east-2a.masters.kubernetes.dev.sdcilsy-trainer.web.id | i-0e00c3b98fbce4cf8  | t2.medium     | us-east-2a   | running        | 2/2 checks ... | None     |
| nodes.kubernetes.dev.sdcilsy-trainer.web.id                     | i-0130b7d6da4688a... | t2.micro      | us-east-2a   | terminated     |                | None     |

Instances: i-01af2cc3ad26271e7 (nodes.kubernetes.dev.sdcilsy-trainer.web.id), i-0e00c3b98fbce4cf8 (master-us-east-2a.masters.kubernetes.dev.sdcilsy-trainer.web.id)

Description: i-01af2cc3ad26271e7: ec2-18-223-206-35.us-east-2.compute.amazonaws.com  
i-0e00c3b98fbce4cf8: ec2-18-218-134-224.us-east-2.compute.amazonaws.com

#### Hasil Update Cluster

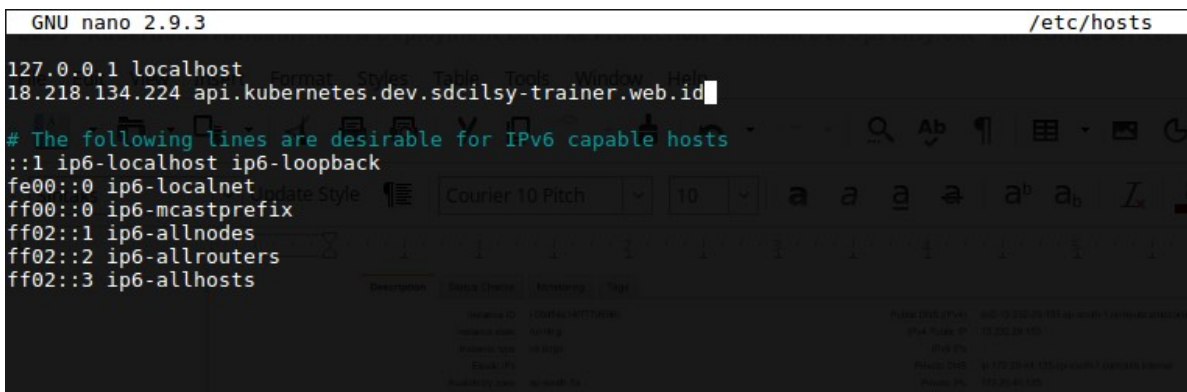
Selanjutnya kita akan melakukan pointing ip public master kubernetes ke server yang kita gunakan. Coba konfigurasi file /etc/hosts dengan ip public



master tersebut, kita bisa menggunakan nano atau vim editor dengan menggunakan perintah berikut.

```
$ sudo nano /etc/hosts
```

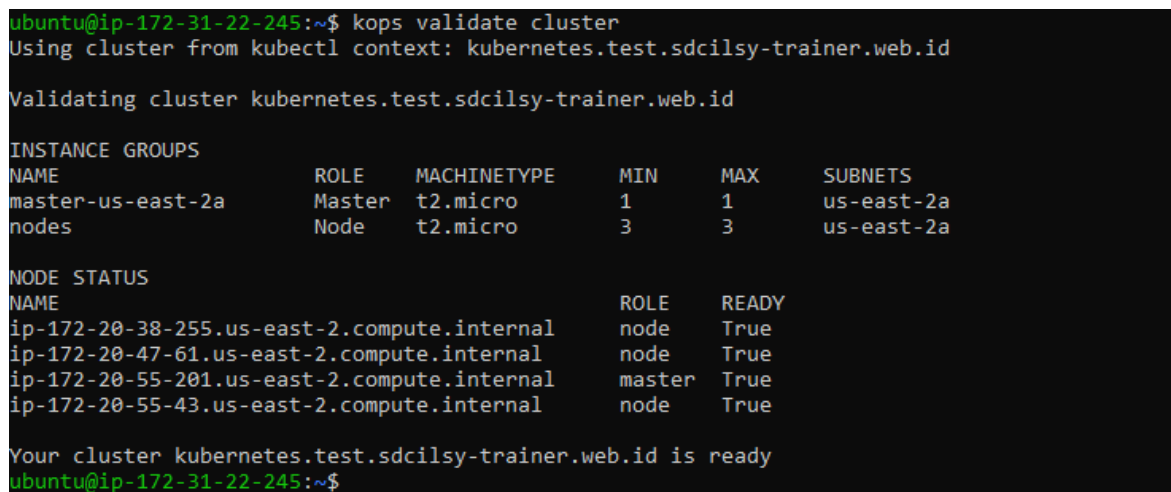
Setelah itu buat konfigurasi dengan menyertakan domain yang ada pada kubernetes seperti gambar dibawah ini. Setelah selesai lalu kita save konfigurasi tersebut.



```
GNU nano 2.9.3 /etc/hosts
127.0.0.1 localhost
18.218.134.224 api.kubernetes.dev.sdcilsy-trainer.web.id
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Setelah Instalasi kubernetes sudah selesai dilakukan, untuk mengecek keberhasilan tersebut kita bisa mengecek menggunakan perintah berikut :

```
kops validate clusters
```



```
ubuntu@ip-172-31-22-245:~$ kops validate cluster
Using cluster from kubectl context: kubernetes.test.sdcilsy-trainer.web.id

Validating cluster kubernetes.test.sdcilsy-trainer.web.id

INSTANCE GROUPS
NAME                ROLE    MACHINETYPE    MIN    MAX    SUBNETS
master-us-east-2a   Master  t2.micro        1      1      us-east-2a
nodes               Node    t2.micro        3      3      us-east-2a

NODE STATUS
NAME                                                         ROLE    READY
ip-172-20-38-255.us-east-2.compute.internal                node    True
ip-172-20-47-61.us-east-2.compute.internal                  node    True
ip-172-20-55-201.us-east-2.compute.internal                  master  True
ip-172-20-55-43.us-east-2.compute.internal                   node    True

Your cluster kubernetes.test.sdcilsy-trainer.web.id is ready
ubuntu@ip-172-31-22-245:~$
```

Hasilnya akan seperti gambar diatas, dimana disana akan ditampilkan detail dari cluster yang kita miliki mulai dari node sampai dengan master. Selanjutnya adalah menjalankan aplikasi tersebut.



## 9.8. Instalasi Dashboard Kubernetes

Untuk mengakses dashboard dari kubernetes maka kita dapat menjalankan perintah seperti di bawah ini.

```
kubectl create -f https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deplo/recommended/kubernetes-dashboard.yaml
```

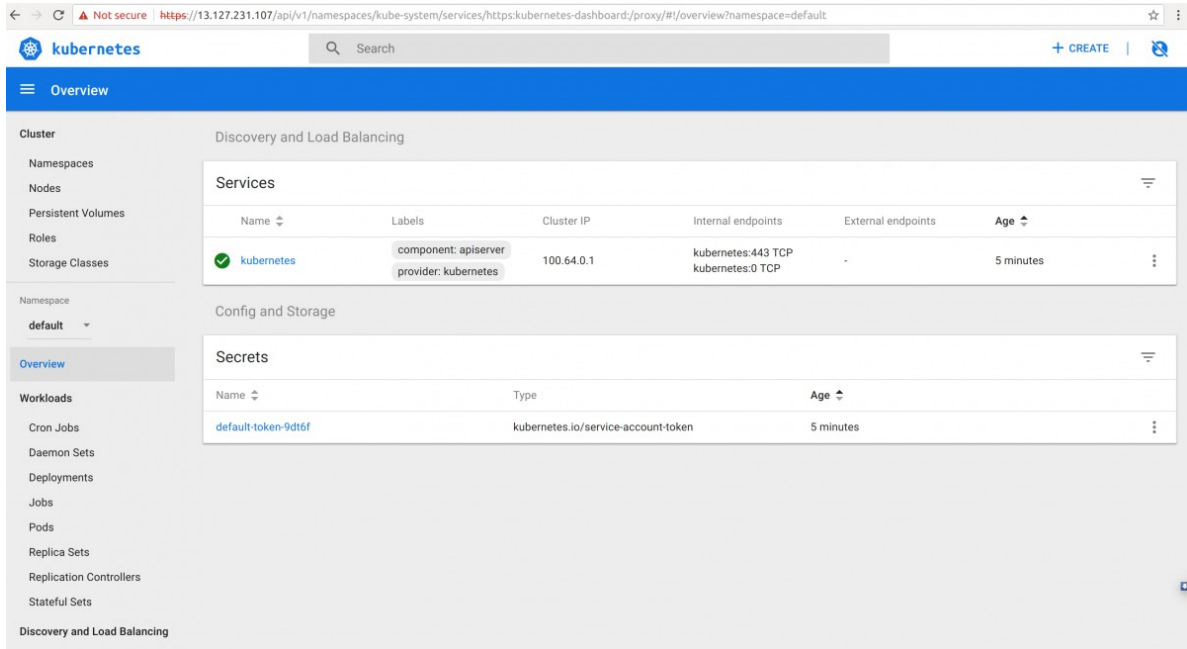
Disini kubernetes baru hanya bisa diakses dari internal, agar bisa diakses dari luar kita dapat menggunakan perintah berikut ini.

```
kubectl proxy &
```

untuk melihat username dan password halaman dashboard maka kita gunakan perintah

```
kubectl config view
```

Berikut merupakan tampilan dashboard dari kubernetes yang sudah berhasil kita installkan.



*Dashboard Web Dashboard Kubernetes*

### 9.8.1. Perintah Kubernetes Lainnya

#### 1. Melihat list cluster



```
kops get cluster
```

## 2. Edit node instance group

```
kops edit ig --name kubernetes.dev.sdcilisy-trainer.web.id nodes
```

## 3. Edit master instance group

```
kops edit ig --name kubernetes.dev.sdcilisy-trainer.web.id master-us-east-1d
```

## 9.8.2. Exercise

Soal Praktek :

1. Install Kubernetes pada server yang kalian miliki
2. Buat 4 Node pada cluster dengan tipe instance t2.micro
3. Buat 1 Master pada cluster dengan tipe instance t2.micro

## 9.9. Konfigurasi Kubernetes

### 9.9.1. File Yaml

File yaml berguna untuk mengatur cluster pada kubernetes. dengan file yaml kita dapat melakukan pengaturan dengan mudah pada cluster kubernetes yang telah kita buat sebelum nya adapun kelebihan menggunakan file yaml adalah sebagai berikut.

1. **Kemudahan,** kita di berikan kemudahan dalam konfigurasi sistem dengan hanya membuat file dan menuliskan kode perubahan maka cluster kubernetes dapat berubah dengan cepat You'll no longer have to add all of your parameters to the command line.
2. **Maintenance,** menggunakan file YAML dapat memaintenance cluster dengan mudah, jika ada perubahan pun akan mudah di cari, berdasarkan perubahan pada file yaml.



3. **Flexibility**, kita dapat membuat struktur dari kubernetes yang kompleks hanya dengan menggunakan file yaml dan di jalankan dengan satu baris command line.

Sebagai contoh kita akan coba untuk membuat sebuah file YAML pada kubernetes, berikut merupakan scriptnya dan simpan dengan nama pod.yaml.

```
apiVersion: v1
kind: Pod
metadata:
  name: rss
  labels:
    app: web
spec:
  containers:
    - name: front-end
      image: nginx
      ports:
        - containerPort: 80
    - name: rss-reader
      image: nginx:latest
      ports:
        - containerPort: 88
```

script dapat dilihat di

<https://gist.github.com/sdcilisy/60a3ab47925a46053040afb9a5d6dc76>

Setelah file disimpan, kita bisa menjalankannya dengan perintah berikut.

```
kubectl create -f pod.yaml
```

### 9.9.2. Setting Deployment dengan YAML

Untuk menjalankan aplikasi maka kita harus mendeploy aplikasi tersebut ke dalam cluster kubernetes, untuk mendeploy kita dapat menggunakan file yaml.





Untuk melakukan deployment kita bisa membuat sebuah file Yaml yang kurang lebih seperti script di bawah ini. Script tersebut digunakan untuk mendeploy nginx ke dalam cluster kubernetes. Simpan dengan nama **deployment.yaml**.

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2          # tells deployment to run 2 pods matching the template
  template:           # create pods using pod definition in this template
    metadata:
      # unlike pod-nginx.yaml, the name is not included in the meta data as a unique
      # name is
      # generated from the deployment name
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```

script dapat dilihat di

<https://gist.github.com/sdcilsy/eed87e5f30dac05f151fd96acd8bbedf>

Setelah itu simpan, lalu jalankan script yang sudah kita buat barusan.

```
kubectl create -f deployment.yaml
```



Jika kita ingin melakukan perubahan misal images docker kita ubah dari nginx:1.7.9 menjadi nginx:latest, kita bisa buat file YAML menggunakan script dibawah ini.

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2          # tells deployment to run 2 pods matching the template
  template:            # create pods using pod definition in this template
    metadata:
      # unlike pod-nginx.yaml, the name is not included in the meta data as a unique
      # name is
      # generated from the deployment name
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - containerPort: 80
```

script dapat dilihat di

<https://gist.github.com/sdcilsy/d78df0ab2de4df1e79db77fa19a6923e>

Setelah disimpan, jalankan konfigurasi berikut.

```
kubectl apply -f deployment.yaml
```

Baiklah kita telah berhasil mendeploy nginx kedalam cluster kubernetes, namun nginx belum dapat kita akses karena kita belum membuat service pada kubernetes untuk nginx. Maka dari itu kita harus melakukan konfigurasi service agar nginx bisa diakses.



### 9.9.3. Setting Service

Pada tahap ini kita akan melakukan konfigurasi service pada kubernetes dengan file yaml, berikut merupakan script file YAML yang dapat kita buat. Simpan file dengan nama **service.yaml**.

```
apiVersion: apps/v1beta2
kind: Service
apiVersion: v1
metadata:
  name: nginx
spec:
  type: LoadBalancer
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

script dapat dilihat di

<https://gist.github.com/sdcilsy/854a1d803cdbf632dbcdd686bddaeced>

Setelah selesai kita simpan, lalu jalankan script berikut.

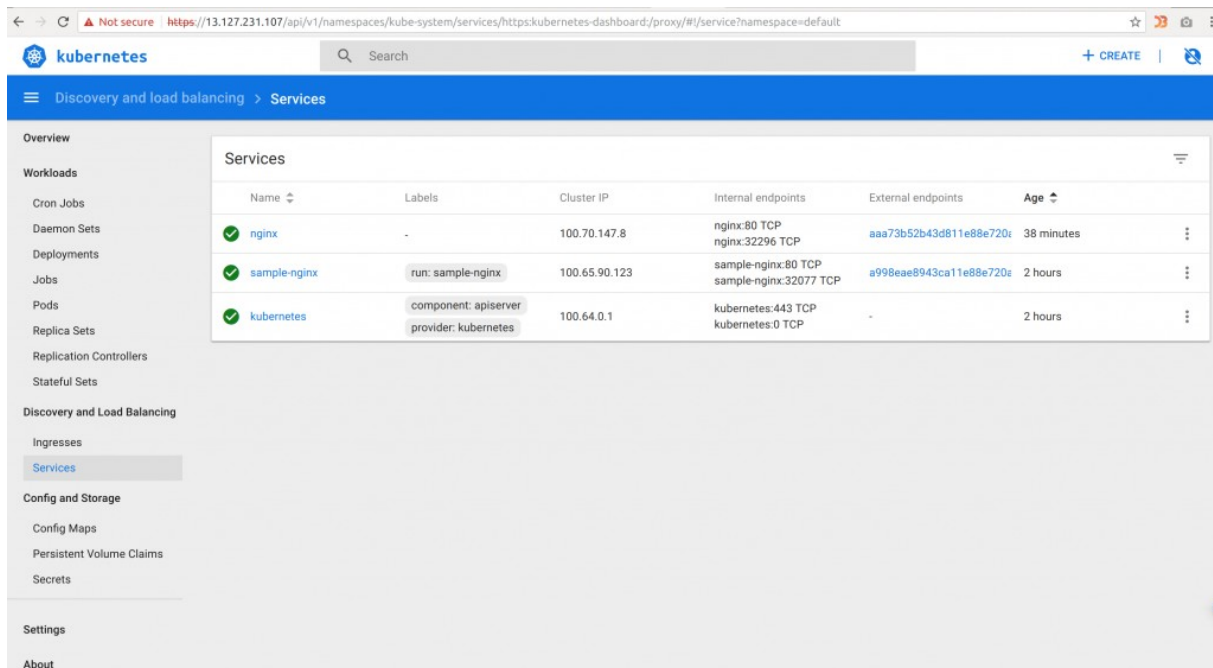
```
kubectl create -f service.yaml
```

Jika ingin mengupdate script maka kita dapat menggunakan perintah berikut.

```
kubectl apply -f service.yaml
```

Setelah semua kita konfigurasi, sekarang kita dapat masuk ke halaman menu service pada dashboard akan muncul nginx seperti di bawah ini.





| Name         | Labels                                       | Cluster IP    | Internal endpoints                            | External endpoints     | Age        |
|--------------|--|---------------|---|------------------------|------------|
| nginx        | -  | 100.70.147.8  | nginx:80 TCP<br>nginx:32296 TCP               | aaa73b52b43d811e88e720 | 38 minutes |
| sample-nginx | run: sample-nginx                            | 100.65.90.123 | sample-nginx:80 TCP<br>sample-nginx:32077 TCP | a998eae8943ca11e88e720 | 2 hours    |
| kubernetes   | component: apiserver<br>provider: kubernetes | 100.64.0.1    | kubernetes:443 TCP<br>kubernetes:0 TCP        | -                      | 2 hours    |

Pada menu service kita dapat mengakses dengan eksternal endpoint. Jika kita akses maka akan muncul seperti di bawah ini

## 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](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

Thank you for using nginx.

*Hasil Deployment nginx*

### 9.9.4. Deployment Wordpress

Pada bagian ini kita akan coba melakukan deployment wordpress pada kubernetes, database yang digunakan pada wordpress tersebut akan menggunakan database dari amazon RDS. Maka dari itu pertama kita siapkan sebuah database mysql di Amazon RDS.

Setelah itu buat sebuah file **wordpress.yaml** dengan script berikut.

```
apiVersion: v1
kind: Service
metadata:
```



```

name: wordpress
labels:
  app: wordpress
spec:
  ports:
    - port: 80
  selector:
    app: wordpress
    tier: frontend
  type: LoadBalancer
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
---
apiVersion: apps/v1beta2 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: frontend

```



```
strategy:
  type: Recreate
template:
  metadata:
    labels:
      app: wordpress
      tier: frontend
  spec:
    containers:
      - image: wordpress:4.8-apache
        name: wordpress
        ports:
          - containerPort: 80
            name: wordpress
```

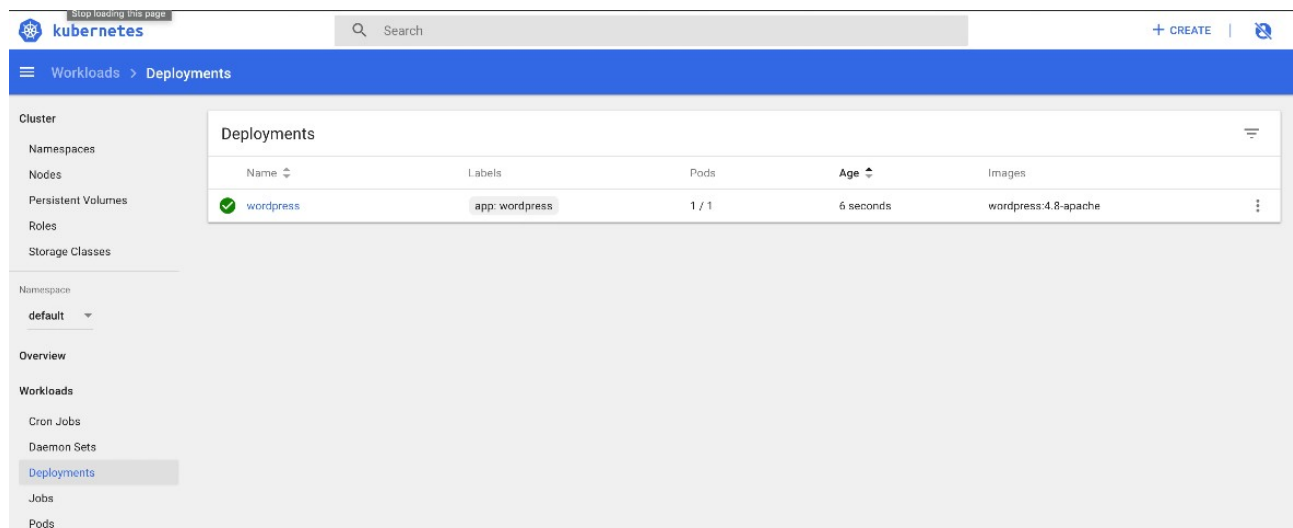
script dapat dilihat di

<https://gist.github.com/sdcilsy/df9f67f843491367e8c7e1ce58d2dc9b>

Setelah itu deploy dengan menggunakan perintah berikut.

```
Kubectl create -f wordpress.yaml
```

Jika berhasil, maka hasilnya dapat kita lihat pada dashboard console kubernetes bagian deployment seperti berikut.




| Name      | Labels         | Pods  | Age       | Images               |
|-----------|----------------|-------|-----------|----------------------|
| wordpress | app: wordpress | 1 / 1 | 6 seconds | wordpress:4.8-apache |

Masuk pada bagian service dan kalian dapat melihat bagian endpoint pada wordpress yang sudah kalian deploykan.



🔍 Search

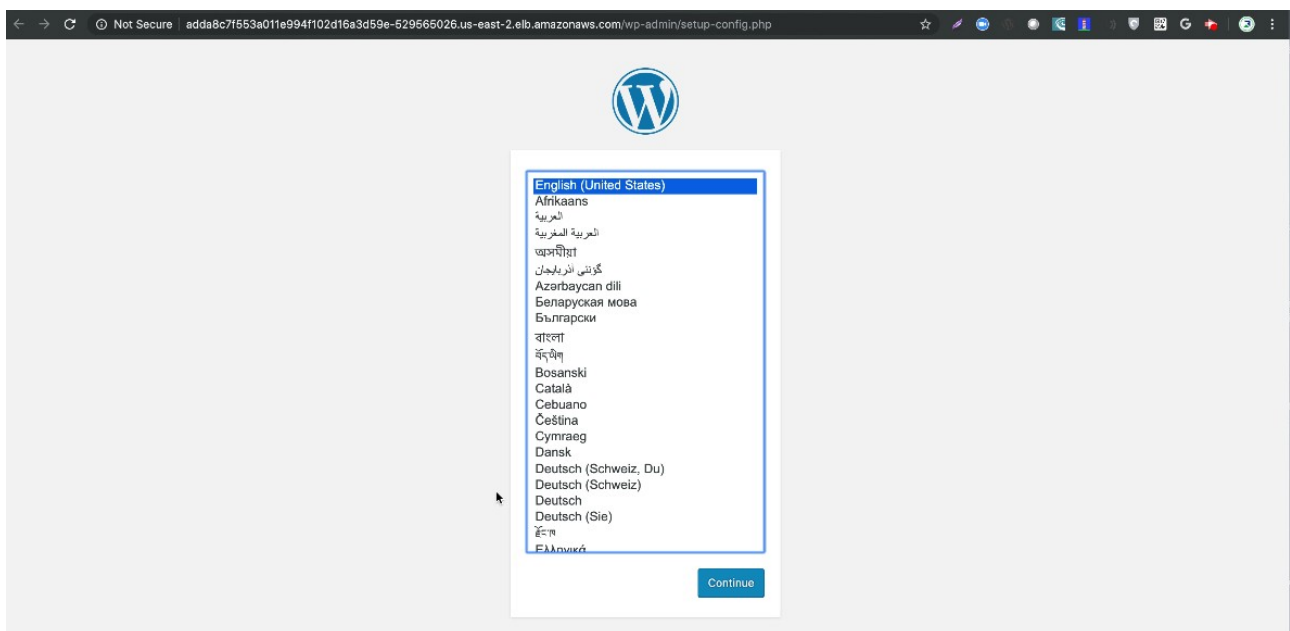
[+ CREATE](#) | 

[Clustering](#) > **Services**

Services

| Name                         | Labels                                       | Cluster IP     | Internal endpoints                      | External endpoints                        | Age        |
|------------------------------|--|----------------|---|---|------------|
| ✔ <a href="#">wordpress</a>  | app: wordpress                               | 100.69.209.217 | wordpress:80 TCP<br>wordpress:32704 TCP | <a href="#">add8c7f553a011e994f102d11</a> | 33 seconds |
| ✔ <a href="#">kubernetes</a> | component: apiserver<br>provider: kubernetes | 100.64.0.1     | kubernetes:443 TCP<br>kubernetes:0 TCP  | -   | 20 minutes |

Klik alamat endpoint tersebut, lalu lakukan setup pada wordpress dengan mengarahkan database pada amazon RDS yang kalian sudah buat sebelumnya. Kalian seharusnya sudah dapat melakukan setup RDS seperti yang sudah dipelajari sebelumnya.



## Wordpress pada kubernetes

### 9.9.5. Exercise

### Soal Praktek :

1. Lakukan Deployment Drupal menggunakan database Amazon RDS.
2. Jalankan hingga dapat kalian akses melalui endpoint public.



3. Cari referensi yang dapat kalian gunakan untuk menginstall drupal pada kubernetes.

## 9.10. Summary

1. Kubernetes adalah sebuah cluster management open source yang di gunakan untuk mengelola container. Aplikasi ini berasal dari aplikasi internal yang digunakan Google untuk mengelola cluster.
2. Dalam arsitektur kubernetes terdapat tiga komponen yang saling berintegrasi yaitu, Kubelet, Kubernetes controller manager dan Kubernetes API server.
3. Untuk melakukan integrasi dengan AWS, kubernetes memerlukan S3 Bucket sebagai storage dan domain yang di parkir di Route53.
4. Domain haruslah valid dan juga kita hanya menggunakan subddomain agar tidak mengakibatkan kekacauan pada domain inti.
5. Pada Clustering kubernetes kita harus melakuakn export S3 untuk storage setelah itu melakuakn konfigurasi cluster dan update cluster untuk melakukan build.
6. Kita dapat melakuakn management Container menggunakan Dashboard Kubernetes yang dapat kita install.
7. File YAML berfungsi untuk mengatur konfigurasi pada kubernetes dan juga untuk melakukan deployment aplikasi.

### Referensi :

<https://kubernetes.io/id/docs/setup/learning-environment/minikube/>

[https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands\\_](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands_)

<https://kubernetes.io/docs/tutorials/hello-minikube/>

