

# **LAPORAN PRAKTIKUM VIRTUALISASI KOMPUTER**

## **PEMBUATAN DAN SETUP POD MENGUNAKAN DEPLOYMENT KUBERNETES**



**Agus Pranata Marpaung**

**13323033**

**DIII TEKNOLOGI KOMPUTER**

**INSTITUT TEKNOLOGI DEL  
FAKULTAS VOKASI**

## Judul Praktikum

---

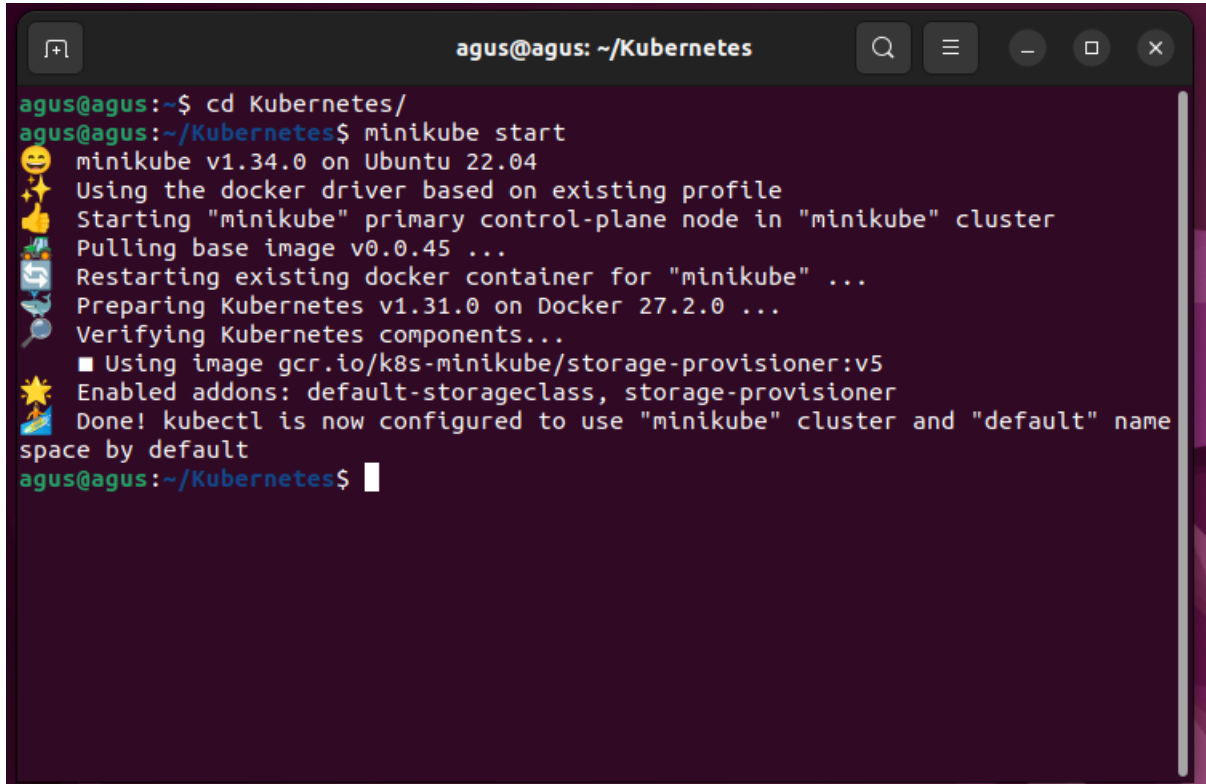
|                            |   |  |
|----------------------------|---|--|
| <b>Minggu/Sesi</b>         | : | XIV/2  |
| <b>Kode Mata Kuliah</b>    | : | 4332103  |
| <b>Nama Mata Kuliah</b>    | : | VIRTUALISASI KOMPUTER  |
| <b>Setoran</b>             | : | Jawaban dalam bentuk <i>softcopy</i>   |
| <b>Batas Waktu Setoran</b> | : | <i>Kamis, 28 November 2024 jam 21:30</i>   |
| <b>Tujuan</b>              | : | 1. Mahasiswa mampu membuat dan melakukan setup pada pod menggunakan Deployment Kubernetes. |

## Petunjuk

## Praktikum

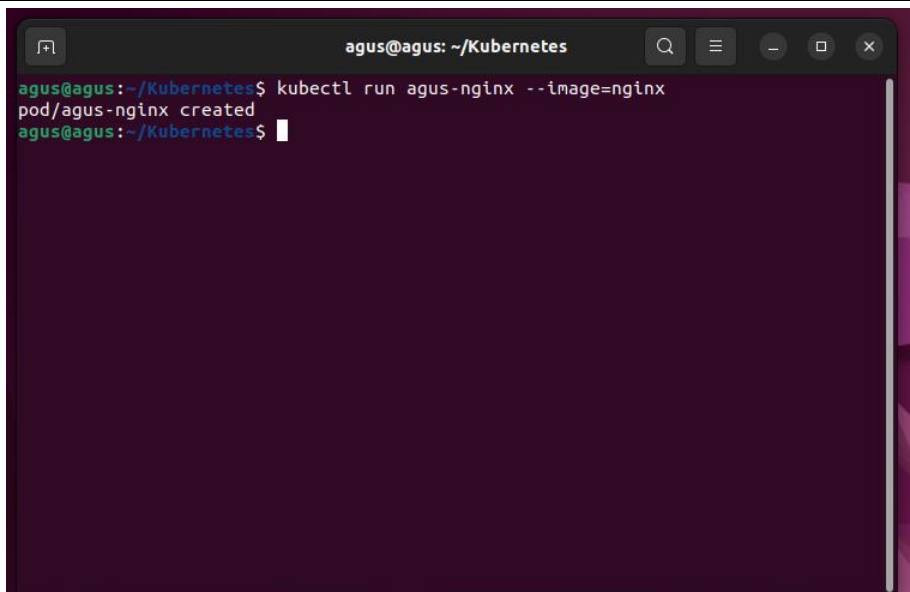
1. Pada praktikum sebelumnya, kita sudah mengenal apa itu pod, dan sekarang kita akan mencoba untuk membuat pod secara manual. Namun terlebih dahulu kita bisa menjalankan minikube dikarenakan defaultnya minikube tidak akan berjalan saat boot.

```
minikube start
```

A terminal window titled 'agus@agus: ~/Kubernetes' showing the execution of 'minikube start'. The output includes several status messages with emojis: 'minikube v1.34.0 on Ubuntu 22.04', 'Using the docker driver based on existing profile', 'Starting "minikube" primary control-plane node in "minikube" cluster', 'Pulling base image v0.0.45 ...', 'Restarting existing docker container for "minikube" ...', 'Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...', 'Verifying Kubernetes components...', 'Using image gcr.io/k8s-minikube/storage-provisioner:v5', 'Enabled addons: default-storageclass, storage-provisioner', and 'Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default'. The prompt returns to 'agus@agus:~/Kubernetes\$'.

2. Agar Kita bisa menjalankan minikube secara otomatis saat boot Kita bisa membuat script pada service minikube tersebut. Kita bisa melakukan eksplor melalui internet.
3. Setelah itu kita bisa membuat pod secara manual dengan menjalankan *command* berikut.

```
kubectl run nginx --image=nginx
```

A terminal window titled 'agus@agus: ~/Kubernetes' showing the execution of 'kubectl run agus-nginx --image=nginx'. The output is 'pod/agus-nginx created'. The prompt returns to 'agus@agus:~/Kubernetes\$'.

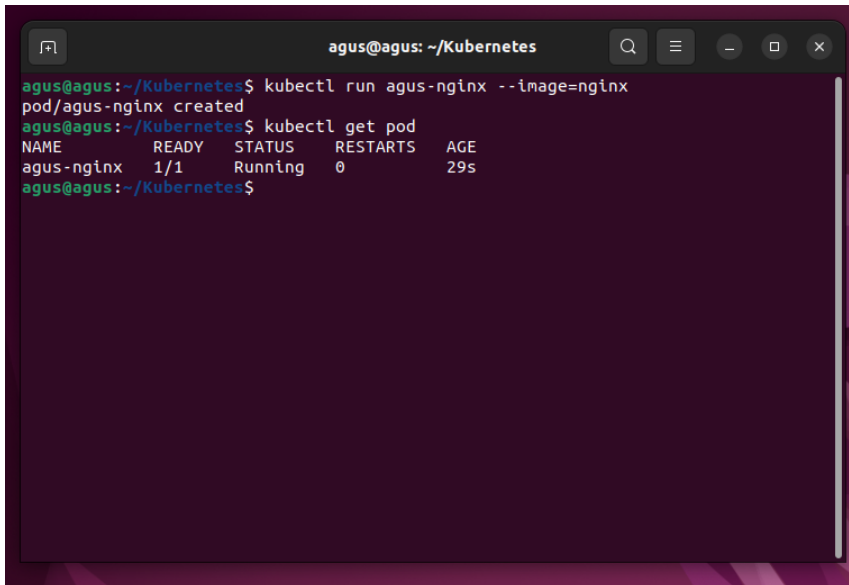
**Keterangan:**

**virkom-nginx** : nama pod yang akan dibuat

**--image=nginx** : image nginx yang digunakan untuk membuat pod

4. Lalu Kita bisa melihat status pod itu sendiri dengan menjalankan *command* berikut.

```
kubectl get pod
```

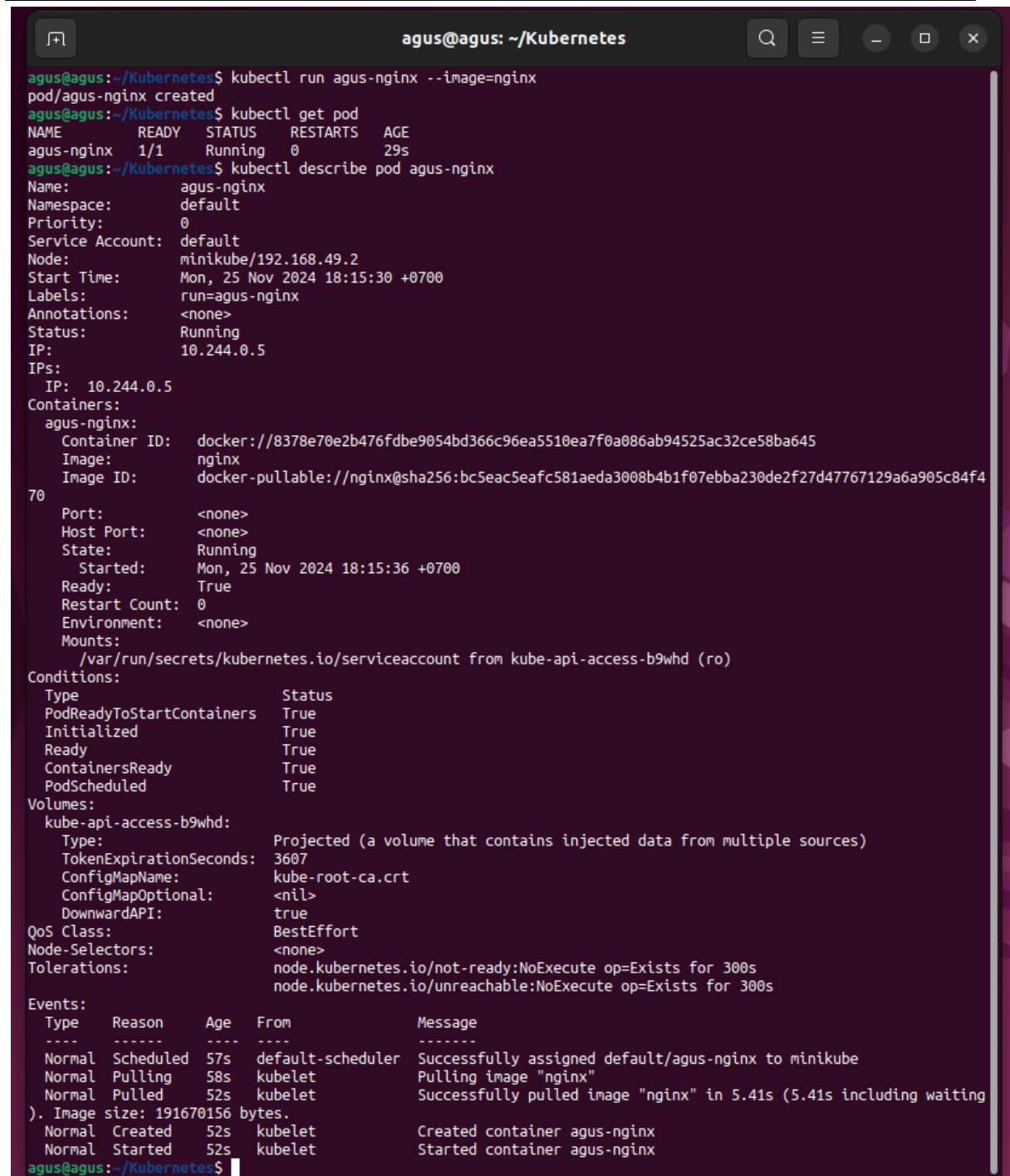


The image shows a terminal window titled 'agus@agus: ~/Kubernetes'. The user enters the command 'kubectl run agus-nginx --image=nginx', which results in 'pod/agus-nginx created'. Then, the user enters 'kubectl get pod', which displays a table with the following information:

| NAME       | READY | STATUS  | RESTARTS | AGE |
|------------|-------|---------|----------|-----|
| agus-nginx | 1/1   | Running | 0        | 29s |

5. Kita juga bisa melihat informasi lengkap mengenai pod yang Kita buat dengan menjalankan *command* berikut

```
kubectl describe pod virkom-nginx
```



```
agus@agus: ~/Kubernetes
agus@agus:~/Kubernetes$ kubectl run agus-nginx --image=nginx
pod/agus-nginx created
agus@agus:~/Kubernetes$ kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
agus-nginx    1/1     Running   0           29s
agus@agus:~/Kubernetes$ kubectl describe pod agus-nginx
Name:          agus-nginx
Namespace:     default
Priority:       0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Mon, 25 Nov 2024 18:15:30 +0700
Labels:        run=agus-nginx
Annotations:    <none>
Status:        Running
IP:            10.244.0.5
IPs:
  IP: 10.244.0.5
Containers:
  agus-nginx:
    Container ID:  docker://8378e70e2b476fdb9054bd366c96ea5510ea7f0a086ab94525ac32ce58ba645
    Image:         nginx
    Image ID:      docker-pullable://nginx@sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d47767129a6a905c84f4
    Port:          <none>
    Host Port:     <none>
    State:         Running
      Started:     Mon, 25 Nov 2024 18:15:36 +0700
    Ready:         True
    Restart Count: 0
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-b9whd (ro)
Conditions:
  Type                               Status
  PodReadyToStartContainers         True
  Initialized                       True
  Ready                             True
  ContainersReady                   True
  PodScheduled                      True
Volumes:
  kube-api-access-b9whd:
    Type:              Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:       kube-root-ca.crt
    ConfigMapOptional:    <nil>
    DownwardAPI:         true
QoS Class:               BestEffort
Node-Selectors:           <none>
Tolerations:              node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                          node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason      Age   From          Message
  ----     ------      -
  Normal   Scheduled   57s   default-scheduler   Successfully assigned default/agus-nginx to minikube
  Normal   Pulling     58s   kubelet          Pulling image "nginx"
  Normal   Pulled      52s   kubelet          Successfully pulled image "nginx" in 5.41s (5.41s including waiting
). Image size: 191670156 bytes.
  Normal   Created     52s   kubelet          Created container agus-nginx
  Normal   Started     52s   kubelet          Started container agus-nginx
agus@agus:~/Kubernetes$
```

6. Lalu, Kita sekarang coba mengecek apakah terdapat container nginx pada node yang sudah Kita buat sebelumnya. Namun Kita terlebih dahulu login kedalam node yang sudah Kita buat sebelumnya dan menjalankan *command* berikut.

```

docker@minikube: ~
Type: Projected (a volume that contains injected data from
multiple sources)
TokenExpirationSeconds: 3607
ConfigMapName: kube-root-ca.crt
ConfigMapOptional: <nil>
DownwardAPI: true
QoS Class: BestEffort
Node-Selectors: <none>
Tolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for
300s
node.kubernetes.io/unreachable:NoExecute op=Exists fo
r 300s
Events:
  Type        Reason         Age   From          Message
  ----        -
Normal       Scheduled      57s   default-scheduler   Successfully assigned default/agus-n
nginx to minikube
Normal       Pulling        58s   kubelet         Pulling image "nginx"
Normal       Pulled         52s   kubelet         Successfully pulled image "nginx" in
5.41s (5.41s including waiting). Image size: 191670156 bytes.
Normal       Created        52s   kubelet         Created container agus-nginx
Normal       Started        52s   kubelet         Started container agus-nginx
agus@agus:~/Kubernetes$ minikube ssh
docker@minikube:~$ docker ps | grep nginx
8378e70e2b47      nginx          "/docker-entrypoint..." 14 minutes ag
o Up 14 minutes   k8s_agus-nginx_agus-nginx_default_8f938494-fcb5-4c19
-a468-f5da6b582c01_0
c521693e471b     registry.k8s.io/pause:3.10 "/pause" 15 minutes ag
o Up 15 minutes   k8s_POD_agus-nginx_default_8f938494-fcb5-4c19-a468-f
5da6b582c01_0
docker@minikube:~$

```

Maka akan terdapat 2 container, yaitu container nginx dan juga container pause yang berguna untuk manajemen pod dan juga memberikan identitas unik pada pod.

7. Kemudian Kita coba untuk masuk kedalam container nginx dengan menjalankan *command* berikut.

```

docker@minikube:~$ docker exec -it 8378e70e2b47 bash
root@agus-nginx:/#

```

8. Setelah masuk, Kita akan mencoba untuk mengakses web server nginx. Untuk itu Kita perlu mengetahui apa IP Address dari container tersebut dengan menjalankan *command* berikut.

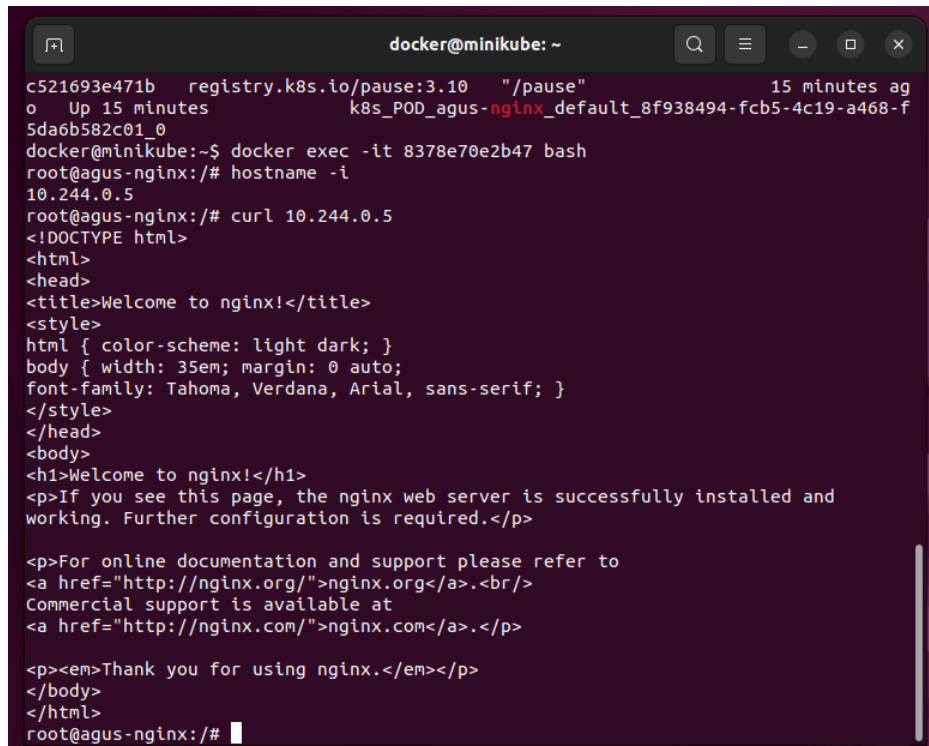
```

hostname -i
root@agus-nginx:/# hostname -i
10.244.0.5
root@agus-nginx:/#

```

9. Lalu Kita sekarang coba mengakses web server dengan menjalankan *command* berikut.

```
curl 10.244.0.5
```

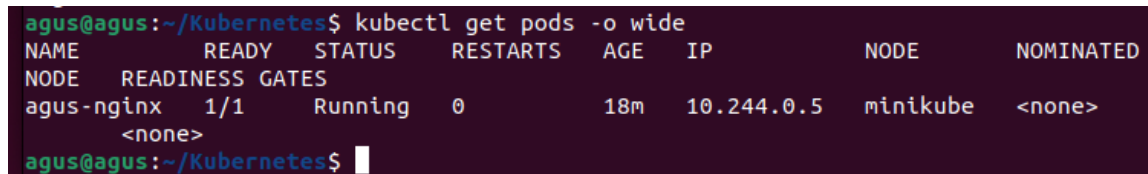


```
docker@minikube: ~  
c521693e471b registry.k8s.io/pause:3.10 "/pause" 15 minutes ago  
Up 15 minutes k8s_POD_agus-nginx_default_8f938494-fcb5-4c19-a468-f  
5da6b582c01_0  
docker@minikube:~$ docker exec -it 8378e70e2b47 bash  
root@agus-nginx:/# hostname -i  
10.244.0.5  
root@agus-nginx:/# curl 10.244.0.5  
<!DOCTYPE html>  
<html>  
<head>  
<title>Welcome to nginx!</title>  
<style>  
html { color-scheme: light dark; }  
body { width: 35em; margin: 0 auto;  
font-family: Tahoma, Verdana, Arial, sans-serif; }  
</style>  
</head>  
<body>  
<h1>Welcome to nginx!</h1>  
<p>If you see this page, the nginx web server is successfully installed and  
working. Further configuration is required.</p>  
  
<p>For online documentation and support please refer to  
<a href="http://nginx.org/">nginx.org</a>.<br/>  
Commercial support is available at  
<a href="http://nginx.com/">nginx.com</a>.</p>  
  
<p><em>Thank you for using nginx.</em></p>  
</body>  
</html>  
root@agus-nginx:/#
```

Dari gambar diatas, dapat disimpulkan bahwa web server berjalan dengan baik.

10. Untuk mendapatkan informasi lebih detail mengenai pod yang sedang berjalan di Kubernetes cluster, Kita bisa menjalankan *command* berikut.

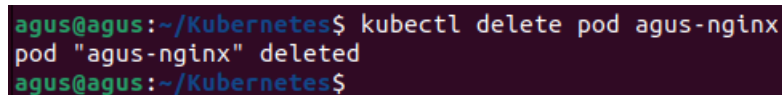
```
kubectl get pods -o wide
```



```
agus@agus:~/Kubernetes$ kubectl get pods -o wide  
NAME          READY   STATUS    RESTARTS   AGE   IP          NODE       NOMINATED  
NODE   READINESS GATES  
agus-nginx    1/1     Running   0           18m   10.244.0.5  minikube   <none>  
<none>  
agus@agus:~/Kubernetes$
```

11. Setelah Kita sudah selesai membuat single pod, Kita bisa menghapus pod tersebut dengan menjalankan *command* berikut.

```
kubectl delete pod virkom-nginx
```



```
agus@agus:~/Kubernetes$ kubectl delete pod agus-nginx  
pod "agus-nginx" deleted  
agus@agus:~/Kubernetes$
```

12. Setelah terhapus, sekarang Kita akan diminta untuk membuat beberapa pod yang dimana Kita bisa melakukan custom konfigurasi, seperti menambahkan jumlah pod dan juga mengurangi jumlah pada pod tersebut. Kita bisa membuat sebuah Deployment untuk melakukan hal tersebut. Untuk membuat Deployment, Kita bisa menjalankan *command* berikut.

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

```
agus@agus:~/Kubernetes$ kubectl create deployment nginx-deployment --image=nginx
deployment.apps/nginx-deployment created
agus@agus:~/Kubernetes$
```

13. Lalu Kita bisa melihat status dari deployment yang sedang berjalan lebih rinci dengan menjalankan *command* berikut.

```
agus@agus:~/Kubernetes$ kubectl get deployments -o wide
NAME                READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES   SE
LECTOR
nginx-deployment    1/1     1            1           69s   nginx        nginx    ap
p=nginx-deployment
agus@agus:~/Kubernetes$
```

14. Kita juga bisa melihat rincian informasi mengenai deployment tertentu dengan menjalankan *command* berikut.

```
agus@agus: ~/Kubernetes
p=nginx-deployment
agus@agus:~/Kubernetes$ kubectl describe deployment nginx-deployment
Name:                nginx-deployment
Namespace:           default
CreationTimestamp:    Mon, 25 Nov 2024 18:36:27 +0700
Labels:              app=nginx-deployment
Annotations:         deployment.kubernetes.io/revision: 1
Selector:             app=nginx-deployment
Replicas:            1 desired | 1 updated | 1 total | 1 available | 0 unavaila
ble
StrategyType:        RollingUpdate
MinReadySeconds:      0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx-deployment
  Containers:
    nginx:
      Image:      nginx
      Port:       <none>
      Host Port:  <none>
      Environment: <none>
      Mounts:      <none>
      Volumes:     <none>
      Node-Selectors: <none>
      Tolerations:  <none>
  Conditions:
    Type           Status  Reason
    ----           -
    Available       True    MinimumReplicasAvailable
    Progressing     True    NewReplicaSetAvailable
  OldReplicaSets:  <none>
  NewReplicaSet:   nginx-deployment-5959b5b5c9 (1/1 replicas created)
  Events:
    Type           Reason             Age           From              Message
    ----           -
    Normal         ScalingReplicaSet   9m57s        deployment-controller  Scaled up replica set n
nginx-deployment-5959b5b5c9 to 1
agus@agus:~/Kubernetes$
```

Pada gambar tersebut Kita melihat bahwa hanya ada 1 pod yang terbentuk.



15. Untuk memastikan apakah hanya 1 pod yang terbentuk, Kita bisa menjalankan *command* berikut.

```
agus@agus:~/Kubernetes$ kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP
  NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-5959b5b5c9-4q8lh  1/1     Running   0           10m   10.244.0.6
  minikube   <none>          <none>
```

16. Dikarenakan masih hanya 1 pod yang terbentuk, Kita perlu menambahkan jumlah replica dari deployment dengan menjalankan *command* berikut.

```
kubectl scale deployment nginx-deployment --replicas=3
```

```
agus@agus:~/Kubernetes$ kubectl scale deployment nginx-deployment --replicas=3
deployment.apps/nginx-deployment scaled
agus@agus:~/Kubernetes$
```

**Keterangan:**

**scale deployment** :mengubah jumlah replica dari suatu deployment

**nginx-deployment** :nama deployment yang ingin diubah jumlah replica dari deployment tersebut

**--replicas=3** : menentukan jumlah replica yang diinginkan, yaitu 3

17. Lalu Kita verifikasi apakah sudah bertambah.

```
agus@agus:~/Kubernetes$ kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP
  NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-5959b5b5c9-4q8lh  1/1     Running   0           12m   10.244.0.6
  minikube   <none>          <none>
nginx-deployment-5959b5b5c9-bcgvp  1/1     Running   0           50s   10.244.0.7
  minikube   <none>          <none>
nginx-deployment-5959b5b5c9-hg5t4  1/1     Running   0           50s   10.244.0.8
  minikube   <none>          <none>
agus@agus:~/Kubernetes$
```

18. Setelah itu, Kita diminta untuk membuat service untuk terhubung ke deployment tertentu menggunakan IP Address tertentu seperti membuat cluster IP. Cluster IP ini akan berperan untuk menghubungkan deployment tertentu dan mendistribusikan load ke berbagai pod yang berbeda di dalam cluster. Untuk itu, Kita bisa menjalankan *command* berikut.

**Note: Cluster IP tidak berfungsi diluar dari Kubernetes cluster.**

```
kubectl expose deployment nginx-deployment --port=8080 --target-port=80
```

```
agus@agus:~/Kubernetes$ kubectl expose deployment nginx-deployment --port=8080 --target-port=80
service/nginx-deployment exposed
agus@agus:~/Kubernetes$
```

**Keterangan:**

**expose deployment** :mengekspos deployment ke luar cluster

**nginx-deployment** :nama deployment yang ingin diekspos

**--port=8080** :port yang akan menerima request dari luar cluster, yaitu 8080

**--target-port=80** :port yang berada di dalam deployment untuk menerima request, yaitu 80

19. Lalu Kita lihat list untuk servicenya.

```
kubectl get services
```

```
agus@agus:~/Kubernetes$ kubectl get services
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP  PORT(S)    AGE
kubernetes          ClusterIP   10.96.0.1       <none>       443/TCP    4d9h
nginx-deployment    ClusterIP   10.104.64.124   <none>       8080/TCP   82s
agus@agus:~/Kubernetes$
```

20. Kita bisa melihat informasi detail mengenai service dari deployment yang sudah Kita buat sebelumnya.

```
kubectl describe service nginx-deployment
```

```
agus@agus:~/Kubernetes$ kubectl describe service nginx-deployment
Name:                nginx-deployment
Namespace:           default
Labels:              app=nginx-deployment
Annotations:         <none>
Selector:            app=nginx-deployment
Type:                ClusterIP
IP Family Policy:    SingleStack
IP Families:         IPv4
IP:                  10.104.64.124
IPs:                 10.104.64.124
Port:                <unset> 8080/TCP
TargetPort:          80/TCP
Endpoints:           10.244.0.6:80,10.244.0.7:80,10.244.0.8:80
Session Affinity:    None
Internal Traffic Policy: Cluster
Events:              <none>
agus@agus:~/Kubernetes$
```

21. Kemudian Kita akan mencoba untuk terhubung ke deployment yang sudah Kita buat menggunakan cluster IP yang tersedia. Namun Kita terlebih dahulu masuk kedalam node yang sudah Kita buat sebelumnya dan menjalankan *command* berikut.

```
curl 10.104.64.124:8080
```

```
agus@agus:~/Kubernetes$ minikube ssh
docker@minikube:~$ curl 10.104.64.124:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
docker@minikube:~$
```

Pada gambar diatas, terdapat keluaran yang disediakan oleh 1 dari 3 pod yang tersedia.

22. Lalu Kita bisa menghapus deployment dan service yang sudah Kita buat sebelumnya dengan menjalankan *command* berikut.

```
agus@agus:~/Kubernetes$ kubectl delete deployment nginx-deployment
deployment.apps "nginx-deployment" deleted
agus@agus:~/Kubernetes$ kubectl delete service nginx-deployment
service "nginx-deployment" deleted
agus@agus:~/Kubernetes$
```

23. Selesai!