

## ปฏิบัติการที่ 7 การบริหารงานพื้นฐานของ Kubernetes

### แบบฝึกหัดที่ 7.1 การกำหนดค่า Replication

เข้าไปที่เครื่อง Linux หลักที่รัน Minikube

1. พิมพ์ตรวจสอบว่า minikube รันอยู่หรือไม่

kubectl get nodes

```
administrator@ubuntu01:~/ubuntu$ kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
minikube      Ready     master   7d    v1.18.3
administrator@ubuntu01:~/ubuntu$
```

2. ตรวจสอบไฟล์กำหนดค่า yaml โดยไปที่ Replication-controller

Cat kubernetes-course/replication-controller/helloworld-repl-controller.yaml

3. สร้าง Replicate จากสคริปต์ โดยจะต้องอยู่ตำแหน่งก่อนโฟลเดอร์ kubernetes-course

kubectl create -f kubernetes-course/replication-controller/helloworld-repl-controller.yaml

```
administrator@ubuntu01:~$ kubectl create -f kubernetes-course/replication-controller/helloworld-repl-controller.yaml
replicationcontroller/helloworld-controller created
administrator@ubuntu01:~$
```

4. ตรวจสอบด้วยคำสั่ง

Kubectl get pods

```
administrator@ubuntu01:~$ kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
busybox                             0/1     Error               0          2d19h
helloworld-controller-2djhm         0/1     ContainerCreating   0          3s
helloworld-controller-wl6rx        0/1     ContainerCreating   0          3s
administrator@ubuntu01:~$
```

จะพบว่ามีการ pods ปรากฏขึ้น

5. เมื่อทำการลบ

kubectl delete pod helloworld-controller-xxxxx (ตามชื่อ)

พบว่าจะมีชื่อใหม่ปรากฏขึ้นมาแทน

6. การกำหนดขยายขนาดโดยใช้คำสั่ง scale

Kubectl scale --replicas=4 -f kubernetes-course/replication-controller/helloworld-repl-controller.yaml

```

administrator@ubuntu01:~$ kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
busybox                             0/1      Error     0           2d19h
helloworld-controller-92dwq         1/1      Running   0           8m1s
helloworld-controller-wfm5t         0/1      ContainerCreating 0           6s
helloworld-controller-zvc6p         1/1      Running   0           6s
nodehelloworld.example.com          1/1      Running   1           8d
administrator@ubuntu01:~$

```

พบว่าจะมีรายการ replica ขึ้น

7. การตรวจสอบค่า Replica controller

kubectl get rc

```

administrator@ubuntu01:~$ kubectl get rc
NAME                DESIRED    CURRENT    READY    AGE
helloworld-controller 4           4           4        20m
administrator@ubuntu01:~$

```

พบว่าจะมีการบอกชื่อของโหนด

8. การตรวจสอบ scale และระบุ replicas ให้เป็น 1

kubectl scale --replicas=1 rc/helloworld-controller

```

VirtualBox
administrator@ubuntu01:~$ kubectl scale --replicas=1 rc/helloworld-controller
replicationcontroller/helloworld-controller scaled
administrator@ubuntu01:~$ kubectl get rc
NAME                DESIRED    CURRENT    READY    AGE
helloworld-controller 1           1           1        23m
administrator@ubuntu01:~$

```

พบว่าเมื่อใช้คำสั่ง kubectl get rc

เครื่องที่สำเนาจะหายไป

9. การลบ replica

kubectl delete rc/helloworld-controller

```

administrator@ubuntu01:~$ kubectl delete rc/helloworld-controller
replicationcontroller "helloworld-controller" deleted
administrator@ubuntu01:~$

```

10. เมื่อเข้าไปตรวจสอบ kubectl get pods พบว่าจะไม่ปรากฏ pods ที่สร้างขึ้น

## แบบฝึกหัดที่ 7.2 การ Deployment ของ Replication set

การจัดเตรียม โดยตรวจสอบค่ากำหนดของไฟล์ helloworld.yml

```
administrator@ubuntu01:~/kubernetes-course$ cat deployment/helloworld.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: helloworld
  template:
    metadata:
      labels:
        app: helloworld
    spec:
      containers:
        - name: k8s-demo
          image: sipadocker/k8s-demo
          ports:
            - name: nodejs-port
              containerPort: 3000
administrator@ubuntu01:~/kubernetes-course$
```

การจัดเตรียมอิมเมจ Tag 2 เพื่อใช้ Rollout

การสร้างอิมเมจที่ 2 (ถ้ามีอยู่แล้วไม่ต้องดำเนินการซ้ำไปได้เลย)

1. เข้าไปใน vagrant ssh โดยอยู่ใน Sub folder ที่ระบุ ในที่นี้คือ Ubuntu
2. Cd docker-demo เพื่อเข้าไปปรับแก้ไขไฟล์
3. ปรับแก้ไขไฟล์ index.js เปลี่ยนที่ Hello World! เป็น Hello World v2!
4. ทำการบันทึกไฟล์

```
vagrant@ubuntu-bionic: ~/docker-demo
File Edit View Search Terminal Tabs Help
vagrant@ubuntu-bionic: ~/docker-demo
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World v2!');
});

var server = app.listen(3000, function () {
  var host = server.address().address;
  var port = server.address().port;

  console.log('Example app listening at http://%s:%s', host, port);
});
```

5. ทำการสร้าง docker ด้วยคำสั่ง

sudo docker build .

ระบบจะรันค่ากำหนดตามสคริปต์เพื่อสร้างไฟล์ ให้ดูค่าแฮช เพื่อใช้ในการดำเนินการส่งข้อมูล

```
vagrant@ubuntu-bionic:~/docker-demo$ sudo docker build .
Sending build context to Docker daemon 212.5kB
Step 1/6 : FROM node:12
---> f5be1883c8e0
Step 2/6 : WORKDIR /app
---> Using cache
---> 45ac3389da76
Step 3/6 : ADD . /app
---> d3b1d2d4cd6a
Step 4/6 : RUN npm install
---> Running in 06e7d80298dc
added 94 packages from 485 contributors and audited 95 packages in 24.002s
found 1 low severity vulnerability
run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container 06e7d80298dc
---> 399c5916c865
Step 5/6 : EXPOSE 3000
---> Running in d357b3fee074
Removing intermediate container d357b3fee074
---> fcb33ecafe56
Step 6/6 : CMD npm start
---> Running in 4160be276d87
Removing intermediate container 4160be276d87
---> c604331c44bb
Successfully built c604331c44bb
vagrant@ubuntu-bionic:~/docker-demo$
```

6. พิมพ์คำสั่ง

```
sudo docker run -p 3000:3000 -it c604331c44bb
```

7. การเข้าไปใช้อิมเมจใน Docker Hub (xxx)

```
sudo docker tag c604331c44bb sipadocker/k8s-demo:2
```

8. ตรวจสอบค่าอิมเมจ

```
sudo docker images
```

```
vagrant@ubuntu-bionic:~$ sudo docker tag 913db8908edf sipadocker/k8s-demo
vagrant@ubuntu-bionic:~$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED
sipadocker/k8s-demo	latest	913db8908edf	50 minutes ago
node	12	f5be1883c8e0	12 hours ago

```
vagrant@ubuntu-bionic:~$
```

9. การส่งไฟล์ขึ้น

```
sudo docker push sipadocker/k8s-demo:2
```

```
vagrant@ubuntu-bionic:~/docker-demo$ sudo docker tag c604331c44bb sipadocker/k8s-demo:2
vagrant@ubuntu-bionic:~/docker-demo$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
sipadocker/k8s-demo	2	c604331c44bb	9 minutes ago	924MB
sipadocker/k8s-demo	abc	913db8908edf	10 days ago	924MB
sipadocker/k8s-demo	latest	913db8908edf	10 days ago	924MB
node	12	f5be1883c8e0	10 days ago	918MB

```
vagrant@ubuntu-bionic:~/docker-demo$ sudo docker push sipadocker/k8s-demo:2
The push refers to repository [docker.io/sipadocker/k8s-demo]
2b24885f3450: Pushed
035b302ac372: Pushed
05f1a0a22041: Layer already exists
dc48ece44f3c: Layer already exists
798326960eac: Layer already exists
dacaab4534e4: Layer already exists
bc17cd405095: Layer already exists
ee854067fbdb: Layer already exists
740ffea5d5c3: Layer already exists
eac9ead92b24: Layer already exists
23bca356262f: Layer already exists
8354d5896557: Layer already exists
2: digest: sha256:ef7a6727d68bec61b4f10ca08253581fe51242d41657304c96d8f2c957a38312 size: 2842
vagrant@ubuntu-bionic:~/docker-demo$
```

## ขั้นตอนการใช้ Deployment

1. เปิด Terminal บนเครื่อง Host เพื่อตรวจสอบค่าของ Deployment

```
kubectl get deployments
```

พบว่ายังไม่มีรายการ deployment

2. การสร้าง deployment

kubectl create -f deployment/helloworld.yml

```
administrator@ubuntu01:~/kubernetes-course$ kubectl create -f deployment/helloworld.yml
deployment.apps/helloworld-deployment created
administrator@ubuntu01:~/kubernetes-course$
```

3. การตรวจสอบการสร้าง deployment

kubectl get deployments

อธิบายแต่ละคอลัมน์

4. การตรวจสอบชุดของ Replicate

kubectl get rs

ตรวจสอบว่ามี deployment และ replicate เท่าไร ซึ่งจะมี 3 รายการ

```
administrator@ubuntu01:~/kubernetes-course$ kubectl get deployments
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
helloworld-deployment   0/3     3             0            29s
administrator@ubuntu01:~/kubernetes-course$ kubectl get rs
NAME                                DESIRED   CURRENT   READY   AGE
helloworld-deployment-d4674668d     3         3         0       46s
administrator@ubuntu01:~/kubernetes-course$
```

5. ตรวจสอบค่า pods

kubectl get pods

```
administrator@ubuntu01:~/kubernetes-course$ kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
busybox                             0/1     Error               0          3d16h
helloworld-deployment-d4674668d-m6n74 0/1     ContainerCreating   0          2m6s
helloworld-deployment-d4674668d-tmfft 0/1     ContainerCreating   0          2m6s
helloworld-deployment-d4674668d-zgnvf 0/1     ContainerCreating   0          2m7s
administrator@ubuntu01:~/kubernetes-course$
```

6. ตรวจสอบค่า pods และแสดงป้ายชื่อ

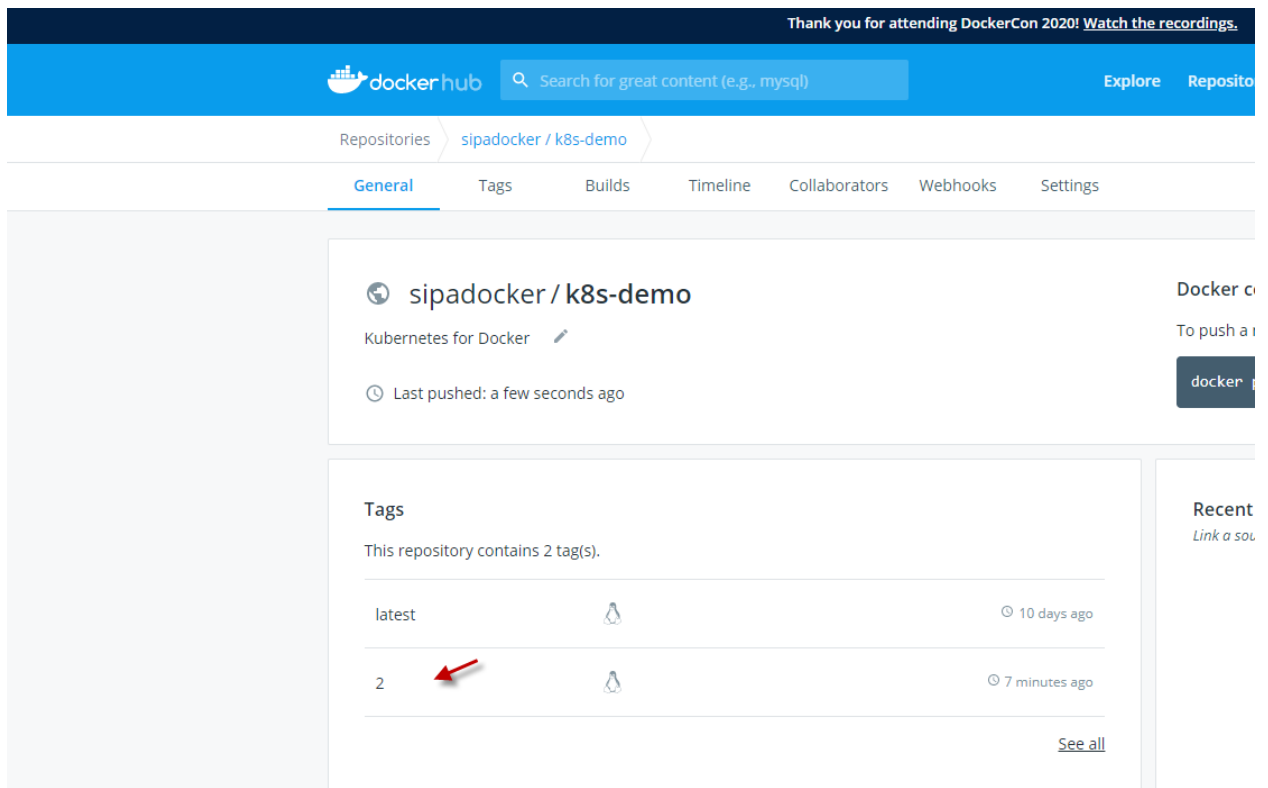
kubectl get pods --show-labels

จะมีป้ายชื่ออธิบายว่ามาจากแอปไหน

```
administrator@ubuntu01:~/kubernetes-course$ kubectl get pods --show-labels
NAME                                READY   STATUS              RESTARTS   AGE   LABELS
busybox                             0/1     Error               0          3d16h  run=busybox
helloworld-deployment-d4674668d-m6n74 0/1     ContainerCreating   0          2m33s  app=helloworld,pod-template-hash=d4674668d
helloworld-deployment-d4674668d-tmfft 0/1     ContainerCreating   0          2m33s  app=helloworld,pod-template-hash=d4674668d
helloworld-deployment-d4674668d-zgnvf 0/1     ContainerCreating   0          2m34s  app=helloworld,pod-template-hash=d4674668d
administrator@ubuntu01:~/kubernetes-course$
```

7. เข้าไปดูในเว็บ docker hub ตรวจสอบในค่า tags

ว่าได้ดำเนินการสร้างเวอร์ชัน 2 ไว้ก่อนแล้ว



8. ดูสถานะของ rollout

kubectl rollout status deployment/helloworld-deployment

9. การสร้าง deployment

kubectl expose deployment helloworld-deployment --type=NodePort

ระบบจะสร้างพอร์ตอัตโนมัติ

10. เข้าไปดูบริการ

kubectl get services

```
Help
administrator@ubuntu01:~/kubernetes-course$ kubectl expose deployment helloworld-deployment --type=NodePort
service/helloworld-deployment exposed
administrator@ubuntu01:~/kubernetes-course$ kubectl get services
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE
helloworld-deployment  NodePort      10.108.205.199 <none>       3000:30052/TCP   51s
helloworld-service    LoadBalancer 10.96.190.117 <pending>    80:30421/TCP     2d23h
kubernetes            ClusterIP     10.96.0.1     <none>       443/TCP          9d
nodehelloworld-service NodePort      10.103.118.123 <none>       3000:30909/TCP   4d12h
administrator@ubuntu01:~/kubernetes-course$
```

11. เข้าไปดูรายละเอียดของบริการที่สร้าง

kubectl describe service helloworld-deployment

```

administrator@ubuntu01:~/kubernetes-course$ kubectl describe service helloworld-deployment
Name: helloworld-deployment
Namespace: default
Labels: <none>
Annotations: <none>
Selector: app=helloworld
Type: NodePort
IP: 10.108.205.199
Port: <unset> 3000/TCP
TargetPort: 3000/TCP
NodePort: <unset> 30052/TCP
Endpoints: 172.17.0.4:3000,172.17.0.5:3000,172.17.0.6:3000
Session Affinity: None
External Traffic Policy: Cluster
Events: <none>
administrator@ubuntu01:~/kubernetes-course$

```

ค่า NodePort จะเป็นค่าสุ่ม 30052

12. การดู url โดยคำสั่ง minikube

minikube service helloworld-deployment --url

```

administrator@ubuntu01:~/kubernetes-course$ minikube service helloworld-deployment --url
http://192.168.99.100:30052
administrator@ubuntu01:~/kubernetes-course$

```

13. เข้าไปในพอร์ตที่ระบุ

curl http://192.168.99.100:30052

```

administrator@ubuntu01:~/kubernetes-course$ curl http://192.168.99.100:30052
Hello World!administrator@ubuntu01:~/kubernetes-course$

```

14. การกำหนดค่าอิมเมจ

kubectl set image deployment/helloworld-deployment k8s-demo=sipadocker/k8s-demo:2

```

Hello World!administrator@ubuntu01:~/kubernetes-course$ kubectl set image deployment/helloworld-deployment
adocker/k8s-demo:2
deployment.apps/helloworld-deployment image updated
administrator@ubuntu01:~/kubernetes-course$

```

เป็นการกำหนดค่าอิมเมจให้เปลี่ยน Tag 2 ซึ่งจะมีข้อความว่า Helloworld v2!

15. การตรวจสอบสถานะหลังจากส่ง deployment

kubectl rollout status deployment/helloworld-deployment

พบว่าจะแจ้งสถานะว่าได้ดำเนินการสำเร็จ

16. ทดสอบโดยการเข้าไปในเวอร์ชันที่ส่งใหม่

curl <http://192.168.99.100:30052>

```

administrator@ubuntu01:~/kubernetes-course$ kubectl set image deployment/helloworld-deployment k8s-demo=sip
adocker/k8s-demo:2
deployment.apps/helloworld-deployment image updated
administrator@ubuntu01:~/kubernetes-course$ kubectl rollout status deployment/helloworld-deployment
deployment "helloworld-deployment" successfully rolled out
administrator@ubuntu01:~/kubernetes-course$ curl http://192.168.99.100:30052
Hello World v2!administrator@ubuntu01:~/kubernetes-course$

```



17. การตรวจสอบค่า deployment ใหม่

kubectl get pods

พบว่าจะมีรายการเก่าถูกลบ และถูกสร้างรายการใหม่ให้สังเกตค่า Hashing ที่ต่อท้าย

```
Hello World v2!administrator@ubuntu01:~/kubernetes-course$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
busybox                             0/1     Error     0           3d22h
helloworld-deployment-945bc564-4qxx6 1/1     Running   0           63s
helloworld-deployment-945bc564-75rcr 1/1     Running   0           70s
helloworld-deployment-945bc564-d4kt8 1/1     Running   0           79s
administrator@ubuntu01:~/kubernetes-course$
```

18. การตรวจสอบประวัติ

kubectl rollout history deployment/helloworld-deployment

```
administrator@ubuntu01:~/kubernetes-course$ kubectl rollout history deployment/helloworld-deployment
deployment.apps/helloworld-deployment
REVISION  CHANGE-CAUSE
1          <none>
2          <none>
administrator@ubuntu01:~/kubernetes-course$
```

ถ้าเรา kubectl create -f deployment/helloworld.yml --record ก็จะขึ้นเพิ่ม revision และมี CHANGE-CAUSE อธิบายว่าทำอะไร

19. ถ้าต้องการกลับไปเวอร์ชันก่อน คือ helloworld! เหยยให้ดำเนินการดังนี้

kubectl rollout undo deployment/helloworld-deployment

ระบบจะเรียกย้อนกลับ

20. ตรวจสอบสถานะโดย

kubectl rollout status deployment/helloworld-deployment

21. เข้าดูรายการ pods

Kubectl get pods

```
administrator@ubuntu01:~/kubernetes-course$ kubectl rollout undo deployment/helloworld-deployment
deployment.apps/helloworld-deployment rolled back
administrator@ubuntu01:~/kubernetes-course$ kubectl rollout status deployment/helloworld-deployment
Waiting for deployment "helloworld-deployment" rollout to finish: 1 old replicas are pending termination..
Waiting for deployment "helloworld-deployment" rollout to finish: 1 old replicas are pending termination..
deployment "helloworld-deployment" successfully rolled out
administrator@ubuntu01:~/kubernetes-course$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
busybox                             0/1     Error     0           3d23h
helloworld-deployment-945bc564-d4kt8 0/1     Terminating 0           27m
helloworld-deployment-d4674668d-6zw8f 1/1     Running      0           46s
helloworld-deployment-d4674668d-8bczw 1/1     Running      0           54s
helloworld-deployment-d4674668d-k8hrn 1/1     Running      0           59s
administrator@ubuntu01:~/kubernetes-course$
```

22. ทดสอบอีกครั้ง

curl <http://192.168.99.100:30052>

```
administrator@ubuntu01:~/kubernetes-course$ curl http://192.168.99.100:30052
Hello World!administrator@ubuntu01:~/kubernetes-course$
```

23. ตรวจสอบเวอร์ชัน

kubectl rollout history deployment/helloworld-deployment

```
Hello World!administrator@ubuntu01:~/kubernetes-course$ kubectl rollout history deployment/helloworld-deployment
deployment.apps/helloworld-deployment
REVISION  CHANGE-CAUSE
2          <none>
3          <none>
```

24. ดีพอยท์จะมีสองรายการถ้าต้องการเพิ่ม

kubectl edit deployment/helloworld-deployment

ได้ replicas: 3 ให้เพิ่มโดยกดคีย์ a แล้วไปที่ด้านท้ายบรรทัด

revisionHistoryLimit: 100

```
resourceVersion: 682400
selfLink: /apis/apps/v1/namespaces/default/deployments/default/helloworld-deployment
uid: 59ccfc18-1c2d-4ea1-95eb-988e48c00c77
spec:
  progressDeadlineSeconds: 600
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: helloworld
  strategy:
    rollingUpdate:
```

กดคีย์ :wq!

25. ย้อนกลับไปเรียกอิมเมจที่ version 2

kubectl set image deployment/helloworld-deployment k8s-demo=sipadocker/k8s-demo:2

26. ย้อนกลับไปเรียกอิมเมจที่ version 1

kubectl set image deployment/helloworld-deployment k8s-demo=sipadocker/k8s-demo:latest

27. ตรวจสอบค่าประวัติ

kubectl rollout history deployment/helloworld-deployment

```
administrator@ubuntu01:~/kubernetes-course$ kubectl rollout history deployment/helloworld-deployment
deployment.apps/helloworld-deployment
REVISION  CHANGE-CAUSE
3          <none>
4          <none>

administrator@ubuntu01:~/kubernetes-course$ kubectl set image deployment/helloworld-deployment k8s-demo=sipadocker/k8s-demo:1
deployment.apps/helloworld-deployment image updated
administrator@ubuntu01:~/kubernetes-course$ kubectl rollout history deployment/helloworld-deployment
deployment.apps/helloworld-deployment
REVISION  CHANGE-CAUSE
3          <none>
4          <none>
5          <none>
```

28. ลองสร้าง Revision ใหม่โดย

```
kubectl set image deployment/helloworld-deployment k8s-demo=sipadocker/k8s-demo:latest
```

29. ดูค่าใน History

```
kubectl rollout history deployment/helloworld-deployment
```

```
administrator@ubuntu01:~/kubernetes-course$ kubectl rollout history deployment/helloworld-deployment
deployment.apps/helloworld-deployment
REVISION  CHANGE-CAUSE
3         <none>
4         <none>
5         <none>
6         <none>
7         <none>
8         <none>
```

30. เรียก Revision เดิมกลับมาโดย

```
kubectl rollout undo deployment/helloworld-deployment --to-revision=3
```

แล้วพิมพ์

Curl <http://192.168.99.100:30052>

```
administrator@ubuntu01:~/kubernetes-course$ kubectl rollout undo deployment/helloworld-deployment --to-revision=3
deployment.apps/helloworld-deployment rolled back
administrator@ubuntu01:~/kubernetes-course$ curl http://192.168.99.100:30052
Hello World!administrator@ubuntu01:~/kubernetes-course$
```

## แบบฝึกหัดที่ 7.3 การสร้าง และการใช้งานงาน Services

ดำเนินการบนเครื่อง Ubuntu ด้านนอก ไม่ใช่เครื่องที่เกิดจาก vagrant

1. เปิด Terminal แล้วเข้าไปในเครื่องที่ลง minikube

Kubectl get node

Kubectl get pods

Kubectl create -f first-app/helloworld.yml

Kubectl get pods

```
administrator@ubuntu01:~/kubernetes-course$ kubectl get node
NAME          STATUS    ROLES    AGE   VERSION
minikube      Ready     master   9d    v1.18.3
administrator@ubuntu01:~/kubernetes-course$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
busybox                             0/1     Error     0           3d23h
helloworld-deployment-d4674668d-frt28 1/1     Running   0           21m
helloworld-deployment-d4674668d-mhtvj 1/1     Running   0           21m
helloworld-deployment-d4674668d-s46ck 1/1     Running   0           21m
administrator@ubuntu01:~/kubernetes-course$ kubectl create -f first-app/helloworld.yml
pod/nodehelloworld.example.com created
administrator@ubuntu01:~/kubernetes-course$ kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
busybox                             0/1     Error               0           3d23h
helloworld-deployment-d4674668d-frt28 1/1     Running             0           21m
helloworld-deployment-d4674668d-mhtvj 1/1     Running             0           21m
helloworld-deployment-d4674668d-s46ck 1/1     Running             0           21m
nodehelloworld.example.com            0/1     ContainerCreating   0           4s
administrator@ubuntu01:~/kubernetes-course$
```

2. การขอรายละเอียด

kubectl describe pod nodehelloworld.example.com

ตรวจสอบป้ายชื่อเลื่อนดูด้านบน สถานะว่าเปิดหรือไม่

```
Events:
  Type     Reason      Age   From              Message
  ----     -
  Terminal Scheduled   6m57s default-scheduler Successfully assigned default/nodehelloworld.example.com to minikube
  Normal   Pulling     6m56s kubelet, minikube Pulling image "sipadocker/k8s-demo"
  Normal   Pulled      6m53s kubelet, minikube Successfully pulled image "sipadocker/k8s-demo"
  Normal   Created     6m52s kubelet, minikube Created container k8s-demo
  Normal   Started     6m52s kubelet, minikube Started container k8s-demo
administrator@ubuntu01:~/kubernetes-course$
```

3. ตรวจสอบค่าไฟล์

cd kubernetes-course

cat first-app/helloworld-nodeport-service.yml

กำหนดค่า nodeport ไว้

ชื่อพอร์ตจะดูจากไฟล์ cat first-app/helloworld.yml ชื่อว่า nodejs-port และมีป้ายชื่อ helloworld

#### 4. การสร้างบริการ

ถ้ามีของเดิมให้ลบก่อนด้วยคำสั่ง `kubectl delete service helloworld-service`

`kubectl create -f first-app/helloworld-nodeport-service.yml`

#### 5. การดู url

`minikube service helloworld-service --url`

ตรวจสอบพอร์ตว่าเป็นไปตามที่กำหนดหรือไม่ ใช้คำสั่ง

```
administrator@ubuntu01:~/kubernetes-course$ kubectl delete service helloworld-service
service "helloworld-service" deleted
administrator@ubuntu01:~/kubernetes-course$ kubectl create -f first-app/helloworld-nodeport-service.yml
service/helloworld-service created
administrator@ubuntu01:~/kubernetes-course$ minikube service helloworld-service --url
http://192.168.99.100:31001
administrator@ubuntu01:~/kubernetes-course$
```

Curl <http://192.168.99.100:31001>

การตรวจสอบค่าบริการ

`Kubectl describe svc helloworld-service`

`Kubectl get svc`

ตรวจสอบค่าบริการที่เราสร้าง helloworld-service

ip ข้างในเป็นของคลัสเตอร์ใน IP:

```
administrator@ubuntu01:~/kubernetes-course$ curl http://192.168.99.100:31001
Hello World!
administrator@ubuntu01:~/kubernetes-course$ kubectl describe svc helloworld-service
Name: helloworld-service
Namespace: default
Labels: <none>
Annotations: <none>
Selector: app=helloworld
Type: NodePort
IP: 10.108.37.171
Port: <unset> 31001/TCP
TargetPort: nodejs-port/TCP
NodePort: <unset> 31001/TCP
Endpoints: 172.17.0.4:3000,172.17.0.7:3000,172.17.0.8:3000 + 1 more...
Session Affinity: None
External Traffic Policy: Cluster
Events: <none>
administrator@ubuntu01:~/kubernetes-course$ kubectl get svc
NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
helloworld-deployment              NodePort    10.108.205.199 <none>         3000:30052/TCP   6h41m
helloworld-service                 NodePort    10.108.37.171 <none>         31001:31001/TCP  92s
kubernetes                         ClusterIP   10.96.0.1     <none>         443/TCP          9d
nodehelloworld-service             NodePort    10.103.118.123 <none>         3000:30909/TCP  4d19h
administrator@ubuntu01:~/kubernetes-course$
```

#### 6. การลบ service และสร้างใหม่เพื่อตรวจสอบหมายเลข IP ว่าเปลี่ยนไปหรือไม่

`Kubectl delete svc helloworld-service`

`kubectl create -f first-app/helloworld-nodeport-service.yml`

`Kubectl describe svc helloworld-service`

```
administrator@ubuntu01:~/kubernetes-course$ kubectl delete svc helloworld-service
service "helloworld-service" deleted
administrator@ubuntu01:~/kubernetes-course$ kubectl create -f first-app/helloworld-nodeport-service.yml
service/helloworld-service created
administrator@ubuntu01:~/kubernetes-course$ kubectl describe svc helloworld-service
Name: helloworld-service
Namespace: default
Labels: <none>
Annotations: <none>
Selector: app=helloworld
Type: NodePort
IP: 10.98.255.228
Port: <unset> 31001/TCP
TargetPort: nodejs-port/TCP
NodePort: <unset> 31001/TCP
Endpoints: 172.17.0.4:3000,172.17.0.7:3000,172.17.0.8:3000 + 1 more...
Session Affinity: None
External Traffic Policy: Cluster
Events: <none>
administrator@ubuntu01:~/kubernetes-course$
```

ในขณะที่พอร์ตจะคงที่เนื่องจากระบุ static เราสามารถที่ระบุ IP เช่นเดียวกัน

## แบบฝึกหัดที่ 7.4 การใส่ป้ายชื่อ

Labels จะเหมือนกับ Tags ใน aws เราสามารถใช้ในการกรองได้ เราดำเนินการบนเครื่องโฮสต์ด้านนอก

1. เปิด Terminal และเริ่ม

Kubectl get nodes

Cat deployment/helloworld-nodeselector.yml

ดูค่า nodeselector ว่าระบุอย่างไร

```
administrator@ubuntu01:~/kubernetes-course$ kubectl get nodes
NAME        STATUS    ROLES    AGE   VERSION
minikube    Ready     master   9d    v1.18.3
administrator@ubuntu01:~/kubernetes-course$ cat deployment/helloworld-nodeselector.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: helloworld
  template:
    metadata:
      labels:
        app: helloworld
    spec:
      containers:
        - name: k8s-demo
          image: sipadocker/k8s-demo
          ports:
            - name: nodejs-port
              containerPort: 3000
          nodeSelector:
            hardware: high-spec
administrator@ubuntu01:~/kubernetes-course$
```

2. ตรวจสอบค่าคำอธิบาย

kubectl get nodes --show-labels

3. การสร้าง deployment บนค่า nodeselector ไฟล์ที่ระบุ

ให้ดำเนินการลบค่า deployment เดิมก่อน

kubectl delete service helloworld-deployment

kubectl delete deployment helloworld-deployment

kubectl create -f deployment/helloworld-nodeselector.yml

ตรวจสอบโดย

kubectl get deployment

```

administrator@ubuntu01:~/kubernetes-course$ kubectl get deployment
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
helloworld-deployment 0/3      3             0            107s
administrator@ubuntu01:~/kubernetes-course$

```

kubectl get pods

```

administrator@ubuntu01:~/kubernetes-course$ kubectl get deployment
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
helloworld-deployment 0/3      3             0            3m18s
administrator@ubuntu01:~/kubernetes-course$ kubectl get pods
NAME                READY    STATUS    RESTARTS    AGE
busybox             0/1      Error     0            4d
helloworld-deployment-568d6bc585-28cwv 0/1      Pending   0            3m22s
helloworld-deployment-568d6bc585-n7jx9 0/1      Pending   0            3m22s
helloworld-deployment-568d6bc585-pj25k 0/1      Pending   0            3m22s
nodehelloworld.example.com 1/1      Running   0            40m
administrator@ubuntu01:~/kubernetes-course$

```

- เราตรวจสอบรายละเอียดด้วย

kubectl describe pod helloworld-deployment-568d6bc585-28cwv

```

Events:
  Type    Reason             Age   From              Message
  ----    -
  Warning FailedScheduling  63s (x6 over 6m50s)  default-scheduler  0/1 nodes are available: 1 node(s) did n't match node selector.
administrator@ubuntu01:~/kubernetes-course$

```

พบว่าไม่ตรง

- เราต้องระบุค่าป้ายชื่อ

kubectl label nodes minikube hardware=high-spec

- เข้าตรวจสอบใหม่

kubectl get nodes

kubectl get nodes --show-labels

kubectl get pods

```

administrator@ubuntu01:~/kubernetes-course$ kubectl get pods
NAME                READY    STATUS             RESTARTS    AGE
busybox             0/1      Error              0            4d
helloworld-deployment-568d6bc585-28cwv 0/1      ContainerCreating   0            7m31s
helloworld-deployment-568d6bc585-n7jx9 0/1      ContainerCreating   0            7m31s
helloworld-deployment-568d6bc585-pj25k 0/1      ContainerCreating   0            7m31s
nodehelloworld.example.com 1/1      Running             0            44m
administrator@ubuntu01:~/kubernetes-course$

```

- ตรวจสอบรายละเอียดใหม่

kubectl describe pod helloworld-deployment-568d6bc585-28cwv



```

Host Port:      0/TCP
State:          Running
  Started:      Mon, 29 Jun 2020 16:48:05 +0700
Ready:          True
Restart Count:  0
Environment:    <none>
Mounts:
  /var/run/secrets/kubernetes.io/serviceaccount from default-token-7xhh9 (ro)
Conditions:
  Type            Status
  Initialized      True
  Ready            True
  ContainersReady  True
  PodScheduled     True
Volumes:
  default-token-7xhh9:
    Type:          Secret (a volume populated by a Secret)
    SecretName:     default-token-7xhh9
    Optional:       false
QoS Class:        BestEffort
Node-Selectors:    hardware=high-spec
Tolerations:       node.kubernetes.io/not-ready:NoExecute for 300s
                   node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type            Reason              Age             From              Message
  ----            -
  Warning         FailedScheduling    66s (x7 over 8m23s)  default-scheduler  0/1 nodes are available: 1 node(s) did
n't match node selector.
  Normal          Scheduled           56s              default-scheduler  Successfully assigned default/hellowor
ld-deployment-568d6bc585-28cwv to minikube
  Normal          Pulling             54s              kubelet, minikube  Pulling image "sipadocker/k8s-demo"
  Normal          Pulled              43s              kubelet, minikube  Successfully pulled image "sipadocker/
k8s-demo"
  Normal          Created             43s              kubelet, minikube  Created container k8s-demo
  Normal          Started             42s              kubelet, minikube  Started container k8s-demo
administrator@ubuntu01:~/kubernetes-course$

```

## แบบฝึกหัดที่ 7.5 การตรวจสอบสุขภาพ

เราตรวจสอบสุขภาพเพื่อตรวจสอบการทำงานของแอปพลิเคชันโดยเฉพาะระบบใช้งานจริง โดยจะมีการระบุการตรวจสอบ

1. เปิด Terminal แล้วตรวจสอบรายละเอียดไฟล์

cat deployment/helloworld-healthcheck.yml

```
administrator@ubuntu01:~/kubernetes-course$ cat deployment/helloworld-healthcheck.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: helloworld
  template:
    metadata:
      labels:
        app: helloworld
    spec:
      containers:
        - name: k8s-demo
          image: sipadocker/k8s-demo
          ports:
            - name: nodejs-port
              containerPort: 3000
          livenessProbe:
            httpGet:
              path: /
              port: nodejs-port
            initialDelaySeconds: 15
            timeoutSeconds: 30
administrator@ubuntu01:~/kubernetes-course$
```

2. การสร้างการ deployment แบบมีการตรวจสอบสุขภาพ

ให้ดำเนินการลบค่า deployment เดิมก่อน

kubectl delete deployment helloworld-deployment

kubectl create -f deployment/helloworld-healthcheck.yml

3. ตรวจสอบค่า pod

kubectl get pods

```
administrator@ubuntu01:~/kubernetes-course$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
busybox                             0/1     Error     0           4d
helloworld-deployment-59fd54fc68-7hjss 1/1     Running   0           5m8s
helloworld-deployment-59fd54fc68-kx5dq 1/1     Running   0           5m8s
helloworld-deployment-59fd54fc68-nrp4k 1/1     Running   0           5m8s
nodehelloworld.example.com           1/1     Running   0           60m
administrator@ubuntu01:~/kubernetes-course$
```

4. ดูรายละเอียดของ pod

kubectl describe pod helloworld-deployment-59fd54fc68-7hjss

```
3f98461bfcae7e5b4a867d
  Port: 3000/TCP
  Host Port: 0/TCP
  State: Running
    Started: Mon, 29 Jun 2020 16:58:14 +0700
  Ready: True
  Restart Count: 0
  Liveness: http-get http://:nodejs-port/ delay=15s timeout=30s period=10s #success=1 #failure=3
  Environment: <none>
  Mounts:
    /var/run/secrets/kubernetes.io/serviceaccount from default-token-7xhh9 (ro)
Conditions:
  Type              Status
  Initialized        True
  Ready              True
  ContainersReady    True
  PodScheduled       True
Volumes:
  default-token-7xhh9:
    Type: Secret (a volume populated by a Secret)
    SecretName: default-token-7xhh9
    Optional: false
QoS Class: BestEffort
Node-Selectors: <none>
Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s
              node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason      Age   From          Message
  ----    -
  Normal  Scheduled   6m31s default-scheduler Successfully assigned default/helloworld-deployment-59fd54fc68-7hjss to minikube
  Normal  Pulling     6m30s kubelet, minikube Pulling image "sipadocker/k8s-demo"
  Normal  Pulled      6m18s kubelet, minikube Successfully pulled image "sipadocker/k8s-demo"
  Normal  Created     6m18s kubelet, minikube Created container k8s-demo
  Normal  Started     6m17s kubelet, minikube Started container k8s-demo
administrator@ubuntu01:~/kubernetes-course$
```

ตรวจสอบค่า liveness แนะนำว่าควรที่

1. การแก้ไขค่ากำหนด

kubectl edit deployment/helloworld-deployment

ตรวจสอบค่าใน Liveness

```
rollingUpdate:
  maxSurge: 25%
  maxUnavailable: 25%
  type: RollingUpdate
template:
  metadata:
    creationTimestamp: null
    labels:
      app: helloworld
  spec:
    containers:
      - image: sipadocker/k8s-demo
        imagePullPolicy: Always
        livenessProbe:
          failureThreshold: 3
          httpGet:
            path: /
            port: nodejs-port
            scheme: HTTP
          initialDelaySeconds: 15
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 30
        name: k8s-demo
        ports:
          - containerPort: 3000
            name: nodejs-port
            protocol: TCP
        resourceRequirements: {}
```

เลื่อนมาดูค่าที่กำหนดที่ปรับแก้ไขได้ ถ้าปรับแก้ไขพิมพ์ :wq!

## แบบฝึกหัดที่ 7.6 การตรวจสอบความพร้อม

เป็นการตรวจสอบความพร้อม liveness จะดูตอนเปิดและรีสตาร์ท ถ้าเป็น Readiness Probe จะตรวจสอบถ้าไม่ทำงานจะทำการลบออก

1. เปิด Terminal บนเครื่องหลัก

cat deployment/helloworld-liveness-readiness.yml

```
      initialDelaySeconds: 15
      timeoutSeconds: 30
administrator@ubuntu01:~/kubernetes-course$ cat deployment/helloworld-liveness-readiness.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld-readiness
spec:
  replicas: 3
  selector:
    matchLabels:
      app: helloworld
  template:
    metadata:
      labels:
        app: helloworld
    spec:
      containers:
        - name: k8s-demo
          image: sipadocker/k8s-demo
          ports:
            - name: nodejs-port
              containerPort: 3000
          livenessProbe:
            httpGet:
              path: /
              port: nodejs-port
              initialDelaySeconds: 15
              timeoutSeconds: 30
          readinessProbe:
            httpGet:
              path: /
              port: nodejs-port
              initialDelaySeconds: 15
              timeoutSeconds: 30
administrator@ubuntu01:~/kubernetes-course$
```

พบว่าจะมีทั้ง liveness และ readinessProbe

2. การสร้าง deployment

ให้ดำเนินการลบค่า deployment เดิมก่อน

kubectl delete deployment helloworld-deployment

kubectl create -f deployment/helloworld-healthcheck.yml && watch -n1 kubectl get pods

ผลที่ได้

```
Every 1.0s: kubectl get pods                                ubuntu01: Mon Jun 29 17
```

NAME	READY	STATUS	RESTARTS	AGE
busybox	0/1	Error	0	4d1h
helloworld-deployment-59fd54fc68-8s978	1/1	Running	0	16s
helloworld-deployment-59fd54fc68-999r2	1/1	Terminating	0	45s
helloworld-deployment-59fd54fc68-fgmh6	1/1	Running	0	16s
helloworld-deployment-59fd54fc68-fs6zg	1/1	Running	0	16s
helloworld-deployment-59fd54fc68-mlbnx	1/1	Terminating	0	45s
helloworld-deployment-59fd54fc68-pwgv1	1/1	Terminating	0	45s
nodehelloworld.example.com	1/1	Running	0	115m

พบว่าระบบจะดำเนินการตรวจสอบ และรีสตาร์ท

### 3. ถ้าใช้ readinessProbe วิธีการใช้

`kubectl create -f deployment/helloworld-liveness-readiness.yml && watch -n1 kubectl get pods`

```
administrator@ubuntu01: ~/kubernetes-course
```

File Edit View Search Terminal Tabs Help

```
administrator@ubuntu01: ~/kubernetes-course x administrator@ubuntu01: ~/kubernetes-c
```

```
Every 1.0s: kubectl get pods                                ubuntu01: Mon Jun 2
```

NAME	READY	STATUS	RESTARTS	AGE
busybox	0/1	Error	0	4d1h
helloworld-deployment-59fd54fc68-8s978	1/1	Running	0	3m34s
helloworld-deployment-59fd54fc68-fgmh6	1/1	Running	0	3m34s
helloworld-deployment-59fd54fc68-fs6zg	1/1	Running	0	3m34s
helloworld-readiness-d7cf745cb-2fqkq	0/1	ContainerCreating	0	5s
helloworld-readiness-d7cf745cb-gskhw	0/1	ContainerCreating	0	5s
helloworld-readiness-d7cf745cb-jqqqz	0/1	ContainerCreating	0	5s
nodehelloworld.example.com	1/1	Running	0	118m

พบว่าระบบจะทำงานหลังรีสตาร์ทแล้ว

## แบบฝึกหัดที่ 7.7 รู้จักกับ Pod state และ Pod lifecycle

ผู้เรียนจะรู้จักกับสถานะของ Pod state และ container state ถ้าเรียนรู้กับ Lifecycle จะกล่าวต่อไป

pod state จะขึ้นเมื่อเรียกคำสั่ง kubectl get pods จะมีฟิลด์แสดงและสถานะต่างๆ ถ้าปกติจะเป็น Running

1. เปิด Terminal ในเครื่องหลัก เข้าดูในไฟล์ lifecycle.yaml

vim pod-lifecycle/lifecycle.yaml (ถ้ายังไม่ได้ติดตั้งให้ sudo apt install vim vim-gtk3 vim-tiny neovim

vim-athena vim-nox vim-gtk)

```
replicas: 1
selector:
  matchLabels:
    app: lifecycle
template:
  metadata:
    labels:
      app: lifecycle
  spec:
    initContainers:
      - name: init
        image: busybox
        command: ['sh', '-c', 'sleep 10']
    containers:
      - name: lifecycle-container
        image: busybox
        command: ['sh', '-c', 'echo $(date +%s): Running >> /timing && echo "The app is running!" && /bin/sleep 120']
        readinessProbe:
          exec:
            command: ['sh', '-c', 'echo $(date +%s): readinessProbe >> /timing']
          initialDelaySeconds: 35
        livenessProbe:
          exec:
            command: ['sh', '-c', 'echo $(date +%s): livenessProbe >> /timing']
          initialDelaySeconds: 35
          timeoutSeconds: 30
        lifecycle:
          postStart:
            exec:
              command: ['sh', '-c', 'echo $(date +%s): postStart >> /timing && sleep 10 && echo $(date +%s) end postStart >> /timing']
          preStop:
            exec:
              command: ['sh', '-c', 'echo $(date +%s): preStop >> /timing && sleep 10']
```

ตรวจสอบค่าตารางการสั่งการที่ระบุมี initContainers, readinessProbe, livenessProbe, postStart และ

prestop

การทดสอบเปรียบเทียบ

screen -r (ถ้ายังไม่ได้ติดตั้งให้ติดตั้ง sudo apt install screen)

กดคีย์ Ctrl+A เพื่อเข้าโหมดคำสั่ง และพิมพ์ S ตัวใหญ่เพื่อสั่ง Split หน้าจอขวาง

กดคีย์ Ctrl+A และกด Tab ไปหน้าอีกสกรีน

กดคีย์ Ctrl+A และ c หรือ Ctrl+C เพื่อสร้างหน้าจอเพิ่ม

กดคีย์ Ctrl+A และ Ctrl+N เพื่อดูหน้าต่างต่อไป (ข้ามไป)

กดคีย์ Ctrl+A และ Ctrl+| เพื่อแบ่งหน้า (ข้ามไป)

2. ไปเรียกคำสั่งที่หน้าจอด้านบน

```
watch -n1 kubectl get pods
```

3. สลับไปหน้าจอ Ctrl+A และ tab เพื่อไปอีกหน้าจอแล้วไปที่

```
cd kubernetes-course/pod-lifecycle
```

```
kubectl create -f lifecycle.yaml
```

```
kubectl exec -it lifecycle-69f486767f-8qzcd -- cat /timing
```

```
vagrant@ubuntu-bionic: ~  
File Edit View Search Terminal Help  
Every 1.0s: kubectl get pods ubuntu-bionic: Mon Jun 29 14:31:47 2020  


| NAME                       | READY | STATUS  | RESTARTS | AGE   |
|----------------------------|-------|---------|----------|-------|
| lifecycle-69f486767f-8qzcd | 1/1   | Running | 0        | 104s  |
| nodehelloworld.example.com | 1/1   | Running | 0        | 3d10h |

  
0 bash  
vagrant@ubuntu-bionic:~/kubernetes-course/pod-lifecycle$ kubectl exec -it lifecycle-69f486767f-8qzcd --cat /timing  
Error: unknown flag: --cat  
See 'kubectl exec --help' for usage.  
vagrant@ubuntu-bionic:~/kubernetes-course/pod-lifecycle$ kubectl exec -it lifecycle-69f486767f-8qzcd -- cat /timing  
1593441068: Running  
1593441068: postStart  
1593441078: end postStart  
1593441109: livenessProbe  
1593441112: readinessProbe  
1593441119: livenessProbe  
1593441122: readinessProbe  
1593441129: livenessProbe  
1593441132: readinessProbe  
1593441139: livenessProbe  
1593441142: readinessProbe  
vagrant@ubuntu-bionic:~/kubernetes-course/pod-lifecycle$  
1 bash
```

4. คำสั่งเพื่อดูอีก

```
kubectl exec -it lifecycle-69f486767f-8qzcd -- tail /timing -f
```

```
vagrant@ubuntu-bionic:~/kubernetes-course/pod-lifecycle$ kubectl exec -it lifecycle-69f486767f-8qzcd -- tail /timing -f  
1593441239: livenessProbe  
1593441242: readinessProbe  
1593441249: livenessProbe  
1593441252: readinessProbe  
1593441259: livenessProbe  
1593441262: readinessProbe  
1593441269: livenessProbe  
1593441272: readinessProbe  
1593441279: livenessProbe  
1593441282: readinessProbe  
1593441289: livenessProbe  
1593441292: readinessProbe  
1593441299: livenessProbe
```



รีเฟรชทุก 10 วินาที

เมื่อดำเนินการเสร็จก็จะเปลี่ยน STATUS เป็น Running

```
Every 1.0s: kubectl get pods                                ubuntu-bio

NAME                                READY    STATUS              RESTARTS   AGE
lifecycle-69f486767f-8qzcd         0/1      CrashLoopBackOff    2           6m56s
nodehelloworld.example.com         1/1      Running              0           3d10h

vagrant@ubuntu-bionic: ~

File Edit View Search Terminal Help
Every 1.0s: kubectl get pods                                ubuntu-bi

NAME                                READY    STATUS              RESTARTS   AGE
lifecycle-69f486767f-8qzcd         1/1      Running              3           8m9s
nodehelloworld.example.com         1/1      Running              0           3d10h

0 bash
1593441242: readinessProbe
1593441249: livenessProbe
1593441252: readinessProbe
1593441259: livenessProbe
1593441262: readinessProbe
1593441269: livenessProbe
1593441272: readinessProbe
1593441279: livenessProbe
1593441282: readinessProbe
1593441289: livenessProbe
1593441292: readinessProbe
1593441299: livenessProbe
1593441302: readinessProbe
1593441309: livenessProbe
1593441312: readinessProbe
command terminated with exit code 137
vagrant@ubuntu-bionic:~/kubernetes-course/pod-lifecycle$ clear
```

## แบบฝึกหัดที่ 7.8 การใช้ Secrets

สามารถเรียกผ่านคำสั่ง หรือจะใช้ไฟล์ .yaml

1. เปิด terminal บนเครื่องหลัก

```
cat deployment/helloworld-secrets.yml
```

```
vagrant@ubuntu-bionic:~/kubernetes-course$ cat deployment/helloworld-secrets.yml
apiVersion: v1
kind: Secret
metadata:
  name: db-secrets
type: Opaque
data:
  username: cm9vdA==
  password: cGFzc3dvcmQ=
```

2. ทำการสร้าง deployment ชื่อว่า db-secrets

```
kubectl create -f deployment/helloworld-secrets.yml
```

3. เข้าไปดูไฟล์ helloworld-secrets-volumes.yml

```
cat deployment/helloworld-secrets-volumes.yml
```

```
administrator@ubuntu01:~/kubernetes-course$ cat deployment/helloworld-secrets-volumes.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: helloworld
  template:
    metadata:
      labels:
        app: helloworld
    spec:
      containers:
        - name: k8s-demo
          image: sipadocker/k8s-demo
          ports:
            - name: nodejs-port
              containerPort: 3000
          volumeMounts:
            - name: cred-volume
              mountPath: /etc/creds
          readOnly: true
      volumes:
        - name: cred-volume
          secret:
            secretName: db-secrets
```

4. สร้าง deployment กับ deployment/helloworld-secrets-volumes.yml ถ้ามี helloworld-deployment ให้ทำการลบก่อน

```
kubectl create -f deployment/helloworld-secrets-volumes.yml
```

5. ตรวจสอบ pods

kubectl get pods

```
administrator@ubuntu01:~/kubernetes-course$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
helloworld-deployment-7db64c46d6-246l4 1/1     Running   0          18s
helloworld-deployment-7db64c46d6-rdtq7 1/1     Running   0          18s
helloworld-deployment-7db64c46d6-v7n5k 1/1     Running   0          18s
helloworld-readiness-d7cf745cb-2fqkq 1/1     Running   0          27h
helloworld-readiness-d7cf745cb-gskhw 1/1     Running   0          27h
helloworld-readiness-d7cf745cb-jqqqz 1/1     Running   0          27h
no-show-applications-sample.com        1/1     Running   0          29h
administrator@ubuntu01:~/kubernetes-course$
```

รอกจนสถานะเป็น Running

6. ตรวจสอบรายละเอียด

kubectl describe pod helloworld-deployment-7db64c46d6-246l4

```
administrator@ubuntu01: ~/kubernetes-course
File Edit View Search Terminal Help

  Started:      Tue, 30 Jun 2020 21:46:37 +0700
  Ready:        True
  Restart Count: 0
  Environment:  <none>
  Mounts:
    /etc/creds from cred-volume (ro)
    /var/run/secrets/kubernetes.io/serviceaccount from default-token-7xhh9 (ro)
Conditions:
  Type           Status
  Initialized     True
  Ready           True
  ContainersReady True
  PodScheduled    True
Volumes:
  cred-volume:
    Type:          Secret (a volume populated by a Secret)
    SecretName:    db-secrets
    Optional:      false
  default-token-7xhh9:
    Type:          Secret (a volume populated by a Secret)
    SecretName:    default-token-7xhh9
    Optional:      false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                  node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason      Age   From          Message
  ----    -
  Normal  Scheduled   99s   default-scheduler  Successfully assigned default/helloworld-deployment-7db64c46d6-246l4 to minikube
  Warning  FailedMount 97s   kubelet, minikube  MountVolume.SetUp failed for volume "cred-volume" : failed to sync secret cache: timed out waiting for the condition
  Normal  Pulling     94s   kubelet, minikube  Pulling image "sipadocker/k8s-demo"
  Normal  Pulled      90s   kubelet, minikube  Successfully pulled image "sipadocker/k8s-demo"
  Normal  Created     89s   kubelet, minikube  Created container k8s-demo
  Normal  Started     89s   kubelet, minikube  Started container k8s-demo
administrator@ubuntu01:~/kubernetes-course$
```

อ่านสถานะที่ดำเนินการเริ่มตั้งแต่การเม้าท์

7. การสร้างรันบน Pod ที่ต้องการ

```
kubectl exec helloworld-deployment-7db64c46d6-246l4 -i -t -- /bin/bash
```

8. ตรวจสอบไฟล์ที่สร้าง

```
cat /etc/creds/username
```

```
cat /etc/creds/password
```

9. ตรวจสอบจุดเม้าท์

```
mount
```

ดูตำแหน่งที่เม้าท์เพื่อใช้งาน

10. ดูรายชื่อไฟล์ที่ใช้

```
ls /run/secrets/kubernetes.io/serviceaccount
```

## แบบฝึกหัดที่ 7.9 การสร้าง wordpress

ตัวอย่างการติดตั้ง wordpress

1. เปิด Terminal แล้วดูรายละเอียดไฟล์

```
ls wordpress/wordpress-se*
```

เพื่อดูรายการไฟล์ขึ้นต้นด้วย wordpress-se มีสองไฟล์

```
cat wordpress/wordpress-secrets.yml
```

อธิบายถึงรายละเอียดในไฟล์ wordpress-secrets.yml ค่ากำหนดของรหัสผ่าน

```
administrator@ubuntu01:~/kubernetes-course$ ls wordpress/wordpress-se*
wordpress/wordpress-secrets.yml  wordpress/wordpress-service.yml
administrator@ubuntu01:~/kubernetes-course$ cat wordpress/wordpress-secrets.yml
apiVersion: v1
kind: Secret
metadata:
  name: wordpress-secrets
type: Opaque
data:
  db-password: cGFzc3dvcmQ=
administrator@ubuntu01:~/kubernetes-course$
```

cat wordpress/wordpress-single-deployment-no-volumes.yml

```
administrator@ubuntu01:~/kubernetes-course$ cat wordpress/wordpress-single-deployment-no-volumes.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  template:
    metadata:
      labels:
        app: wordpress
    spec:
      containers:
        - name: wordpress
          image: wordpress:4-php7.0
          ports:
            - name: http-port
              containerPort: 80
          env:
            - name: WORDPRESS_DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: wordpress-secrets
                  key: db-password
            - name: WORDPRESS_DB_HOST
              value: 127.0.0.1
        - name: mysql
          image: mysql:5.7
          ports:
            - name: mysql-port
              containerPort: 3306
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: wordpress-secrets
                  key: db-password
administrator@ubuntu01:~/kubernetes-course$
```

อธิบายเกี่ยวกับจำนวน Replica และชื่อ app ระบุการดึง Image พร้อมดึงข้อมูล และพอร์ตที่ใช้คือ 80 ใน mysql ก็เช่นเดียวกัน

2. ตรวจสอบค่าใน Docker ว่ามีอิมเมจในสคริปต์หรือไม่  
เปิด Docker เข้าไปที่

[https://hub.docker.com/\\_/wordpress](https://hub.docker.com/_/wordpress)



เลื่อนไปดูค่าพารามิเตอร์ด้านล่าง



## How to use this image

```
$ docker run --name some-wordpress --network some-network -d wordpress
```

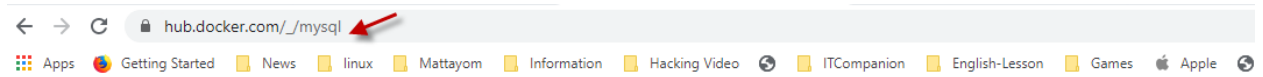
The following environment variables are also honored for configuring your WordPress instance:

- `-e WORDPRESS_DB_HOST=...`
- `-e WORDPRESS_DB_USER=...` 
- `-e WORDPRESS_DB_PASSWORD=...` 
- `-e WORDPRESS_DB_NAME=...`
- `-e WORDPRESS_TABLE_PREFIX=...`
- `-e WORDPRESS_AUTH_KEY=...`, `-e WORDPRESS_SECURE_AUTH_KEY=...`, `-e WORDPRESS_LOGGED_IN_KEY=...`, `-e WORDPRESS_NONCE_KEY=...`, `-e WORDPRESS_AUTH_SALT=...`, `-e WORDPRESS_SECURE_AUTH_SALT=...`, `-e WORDPRESS_LOGGED_IN_SALT=...`, `-e WORDPRESS_NONCE_SALT=...` (default to unique random SHA1s, but only if other environment variable configuration is provided)
- `-e WORDPRESS_DEBUG=1` (defaults to disabled, non-empty value will enable `WP_DEBUG` in `wp-config.php`)
- `-e WORDPRESS_CONFIG_EXTRA=...` (defaults to nothing, non-empty value will be embedded verbatim inside `wp-config.php` -- especially useful for applying extra configuration values this image does not provide by default such as `WP_ALLOW_MULTISITE`; see [docker-library/wordpress#142](#) for more details)

If the `WORDPRESS_DB_NAME` specified does not already exist on the given MySQL server, it will be created automatically upon startup of the `wordpress` container, provided that the `WORDPRESS_DB_USER` specified has the necessary permissions to create it.

If you'd like to be able to access the instance from the host without the container's IP, standard port mappings can be used:

เลื่อนดูด้านบนพบว่ามีค่า 5-php7.2 อาจจะต้องเปลี่ยนในสคริปต์



## Description

## Reviews

## Tags

### Quick reference

- **Maintained by:** the Docker Community and the MySQL Team
- **Where to get help:** the Docker Community Forums, the Docker Community Slack, or Stack Overflow

### Supported tags and respective Dockerfile links

- 8.0.20, 8.0, 8, latest
- 5.7.30, 5.7, 5
- 5.6.48, 5.6

พบว่า mysql ยังรองรับอยู่

ให้ผู้เรียนสรุปจากการเข้าดู cat ไฟล์นี้

### 3. การสร้าง Deployment

```
kubectl create -f wordpress/wordpress-secrets.yml
```

```
kubectl create -f wordpress/wordpress-single-deployment-no-volumes.yml
```

```
administrator@ubuntu01:~/kubernetes-course$ kubectl create -f wordpress/wordpress-secrets.yml
secret/wordpress-secrets created
administrator@ubuntu01:~/kubernetes-course$ kubectl create -f wordpress/wordpress-single-deployment-no-volumes.yml
deployment.apps/wordpress-deployment created
administrator@ubuntu01:~/kubernetes-course$
```

### 4. เข้าดู pods

```
kubectl get pods
```

```
administrator@ubuntu01:~/kubernetes-course$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
helloworld-deployment-7db64c46d6-246l4  1/1     Running   0           4d17h
helloworld-deployment-7db64c46d6-rdtq7  1/1     Running   0           4d17h
helloworld-deployment-7db64c46d6-v7n5k  1/1     Running   0           4d17h
helloworld-readiness-d7cf745cb-2fqkq    1/1     Running   0           5d21h
helloworld-readiness-d7cf745cb-gskhw    1/1     Running   0           5d21h
helloworld-readiness-d7cf745cb-jqqqz    1/1     Running   0           5d21h
nodehelloworld.example.com              1/1     Running   0           5d23h
wordpress-deployment-7d4896594c-m8bvp   2/2     Running   1           3m24s
administrator@ubuntu01:~/kubernetes-course$
```

ตรวจสอบดูเลขของ Deployment

kubectl describe pod wordpress-deployment-เลขที่พบ

```
node.kubernetes.io/unreacheable:NoExecute for 300s
Events:
  Type      Reason      Age           From          Message
  ----      -
  Normal    Scheduled   <unknown>     default-scheduler   Successfully assigned default/wordpress-deployment-7d4896594c-m8bvp to minikube
  Normal    Pulling     3m47s        kubelet, minikube   Pulling image "wordpress:4-php7.0"
  Normal    Pulled      2m40s        kubelet, minikube   Successfully pulled image "wordpress:4-php7.0"
  Normal    Pulling     2m39s        kubelet, minikube   Pulling image "mysql:5.7"
  Normal    Pulled      99s          kubelet, minikube   Successfully pulled image "mysql:5.7"
  Normal    Created     98s          kubelet, minikube   Created container mysql
  Normal    Started     98s          kubelet, minikube   Started container mysql
  Normal    Created     97s (x2 over 2m39s) kubelet, minikube   Created container wordpress
  Normal    Pulled      97s          kubelet, minikube   Container image "wordpress:4-php7.0" already present on machine
  Normal    Started     96s (x2 over 2m39s) kubelet, minikube   Started container wordpress
administrator@ubuntu01:~/kubernetes-course$
```

## 5. การกำหนดค่า Services

Cat wordpress/wordpress-service.yml

```
administrator@ubuntu01:~/kubernetes-course$ cat wordpress/wordpress-service.yml
apiVersion: v1
kind: Service
metadata:
  name: wordpress-service
spec:
  ports:
    - port: 31001
      nodePort: 31001
      targetPort: http-port
      protocol: TCP
  selector:
    app: wordpress
    type: NodePort
administrator@ubuntu01:~/kubernetes-course$
```

Kubectl create -f wordpress/wordpress-service.yml

ถ้าพบว่ามีข้อความว่าพอร์ตถูกใช้แล้วให้เปลี่ยนพอร์ตก่อน

```
administrator@ubuntu01:~/kubernetes-course$ kubectl create -f wordpress/wordpress-service.yml
service/wordpress-service created
administrator@ubuntu01:~/kubernetes-course$
```

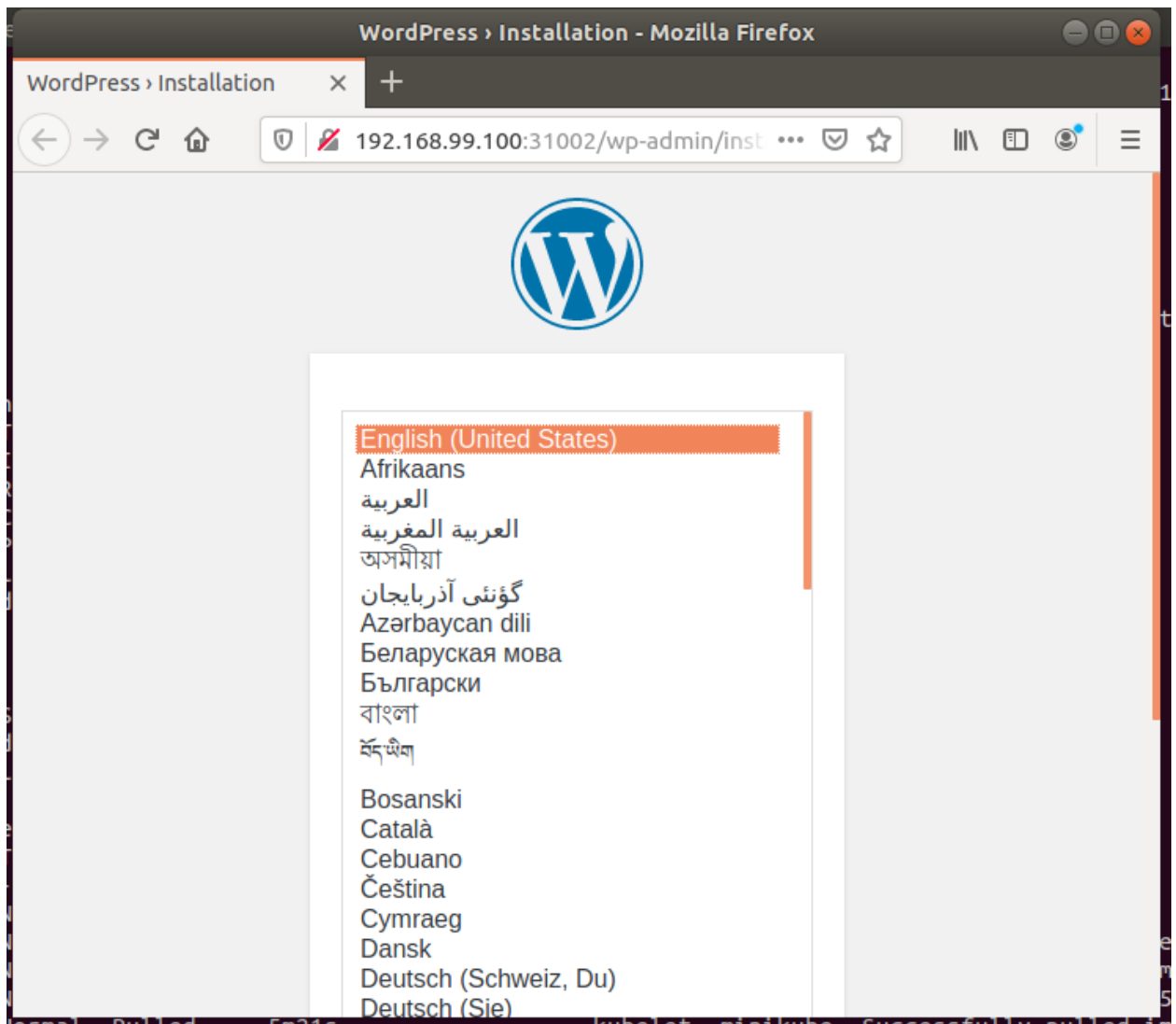
## 6. การตรวจสอบบริการ

minikube service wordpress-service --url

```
administrator@ubuntu01:~/kubernetes-course$ minikube service wordpress-service --url
http://192.168.99.100:31002
administrator@ubuntu01:~/kubernetes-course$
```



7. เปิดเบราว์เซอร์แล้วนำ url ที่ได้ไปทดสอบ



- คลิกปุ่ม Continue
- ในช่อง Site Title ใส่ค่าหัวข้อ
- ในช่อง UserName ใส่ชื่อที่ต้องการ
- ในช่อง Password ใส่รหัสผ่าน

- ในช่อง e-mail ระบุชื่อ e-mail ที่ต้องการ

## Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

### Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title

My Blog

Username


khajorn

Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password

p@ssw0rd

Very weak

 Hide

**Important:** You will need this password to log in. Please store it in a secure location.

Confirm Password

☒ Confirm use of weak password

Your Email


khajorns@gmail.com

Double-check your email address before continuing.

Search Engine Visibility

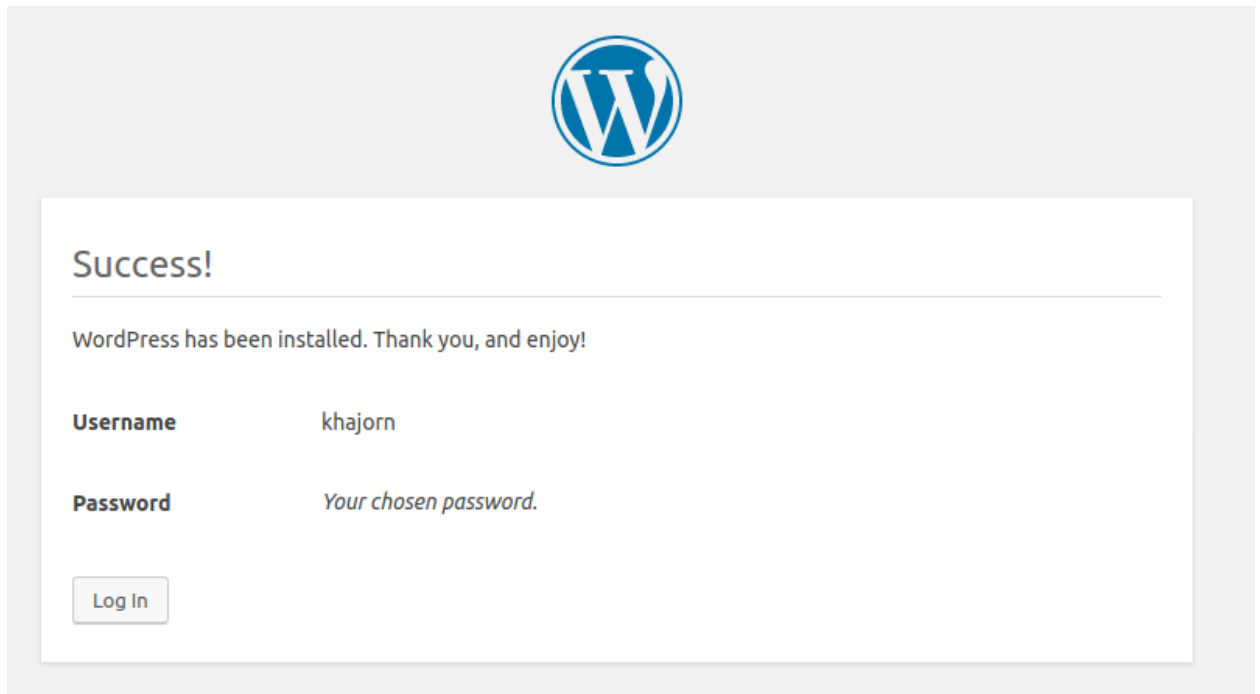
☐ Discourage search engines from indexing this site  
It is up to search engines to honor this request.

Install WordPress

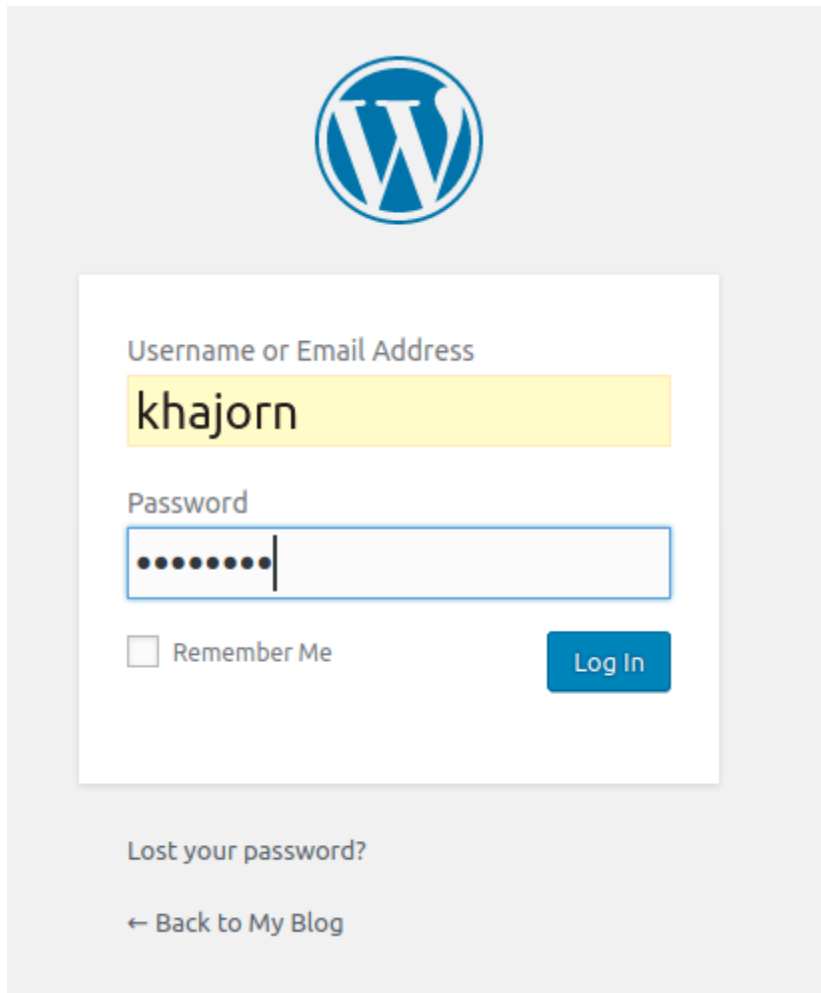


- คลิก Install WordPress
- รอโปรแกรมทำการติดตั้ง WordPress พร้อมฐานข้อมูล

- คลิก Login



5. ใส่ชื่อผู้ใช้งาน และรหัสผ่าน



คลิก Log In เพื่อเข้าไปบริหาร WordPress Dashboard

6. คลิกที่ชื่อ Title เพื่อดูรายละเอียด และการใช้งานของ Wordpress  
แนะนำ Wordpress และการใช้งานต่างๆ
7. ตรวจสอบ pods  
kubectl get pods
8. ลบ pods  
kubectl delete pod/wordpress-deployment-รหัสที่ตรวจสอบ
9. ตรวจสอบ pods อีกครั้ง

kubectl get pods

```
administrator@ubuntu01:~/kubernetes-course$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
helloworld-deployment-7db64c46d6-246l4 1/1     Running   0           4d18h
helloworld-deployment-7db64c46d6-rdtq7 1/1     Running   0           4d18h
helloworld-deployment-7db64c46d6-v7n5k 1/1     Running   0           4d18h
helloworld-readiness-d7cf745cb-2fqkq 1/1     Running   0           5d22h
helloworld-readiness-d7cf745cb-gskhw 1/1     Running   0           5d22h
helloworld-readiness-d7cf745cb-jqqqz 1/1     Running   0           5d22h
nodehelloworld.example.com            1/1     Running   0           6d
wordpress-deployment-7d4896594c-m8bvp 2/2     Running   1           85m
administrator@ubuntu01:~/kubernetes-course$ kubectl delete pod/wordpress-deployment-7d4896594c-m8bvp
pod "wordpress-deployment-7d4896594c-m8bvp" deleted
administrator@ubuntu01:~/kubernetes-course$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
helloworld-deployment-7db64c46d6-246l4 1/1     Running   0           4d18h
helloworld-deployment-7db64c46d6-rdtq7 1/1     Running   0           4d18h
helloworld-deployment-7db64c46d6-v7n5k 1/1     Running   0           4d18h
helloworld-readiness-d7cf745cb-2fqkq 1/1     Running   0           5d22h
helloworld-readiness-d7cf745cb-gskhw 1/1     Running   0           5d22h
helloworld-readiness-d7cf745cb-jqqqz 1/1     Running   0           5d22h
nodehelloworld.example.com            1/1     Running   0           6d
wordpress-deployment-7d4896594c-bfq87 2/2     Running   0           15s
administrator@ubuntu01:~/kubernetes-course$
```

10. เข้าไปตรวจสอบในเว็บใหม่

พบว่าข้อมูลไม่มีเริ่มต้นกำหนด Wordpress ใหม่เนื่องจากการติดตั้งเรามีทั้ง wordpress และ mysql อยู่ในเครื่อง  
และไม่ได้กำหนด volume ข้อมูลในการใช้งาน

เราทำความเข้าใจเกี่ยวกับการ deployment และค่ากำหนดใน yaml

## แบบฝึกหัดที่ 7.10 การบริหารงาน WebUI

ใน Kubernetes จะมีหน้าจอ Web UI ให้บริหารที่ <https://<kubernetes-master>/UI>

1. เปิด Terminal ตรวจสอบค่า

```
cd dashboard
```

```
ls
```

```
cat README.md
```

```
administrator@ubuntu01:~/kubernetes-course/dashboard$ cat README.md
# Setting up the dashboard

## Start dashboard

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

## Create user

Create sample user (if using RBAC - on by default on new installs with kops / kubeadm):
...
kubectl create -f sample-user.yaml
...

## Get login token:
...
kubectl -n kube-system get secret | grep admin-user
kubectl -n kube-system describe secret admin-user-token-<id displayed by previous command>
...

## Login to dashboard
Go to http://api.yourdomain.com:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy#!/login
Login: admin
Password: the password that is listed in ~/.kube/config (open file in editor and look for "password: ...")
Choose for login token and enter the login token from the previous step
administrator@ubuntu01:~/kubernetes-course/dashboard$
```

ค่ากำหนด RBAC ตอนนี้เป็น Default ซึ่งผู้ใช้สามารถล็อกอินเข้าไปใช้งานได้

2. ตรวจสอบในไฟล์ sample-user.yaml

```

administrator@ubuntu01:~/kubernetes-course/dashboard$ cat sample-user.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kube-system
administrator@ubuntu01:~/kubernetes-course/dashboard$

```

### 3. การเรียกสั่งรัน sample-user.yaml

```
administrator@ubuntu01:~/kubernetes-course/dashboard$ kubectl create -f sample-user.yaml
serviceaccount/admin-user created
clusterrolebinding.rbac.authorization.k8s.io/admin-user created
administrator@ubuntu01:~/kubernetes-course/dashboard$
```

```
administrator@ubuntu01:~/kubernetes-course/dashboard$ kubectl -n kube-system get secret | grep admin-user
admin-user-token-96cjlw          kubernetes.io/service-account-token  3      95s
administrator@ubuntu01:~/kubernetes-course/dashboard$
```

[illegible]

จะเห็นค่าใบรับรอง

เราสามารถที่ไปใส่ในไฟล์ ~/.kube/config

## 6. การตรวจสอบรหัสผ่าน

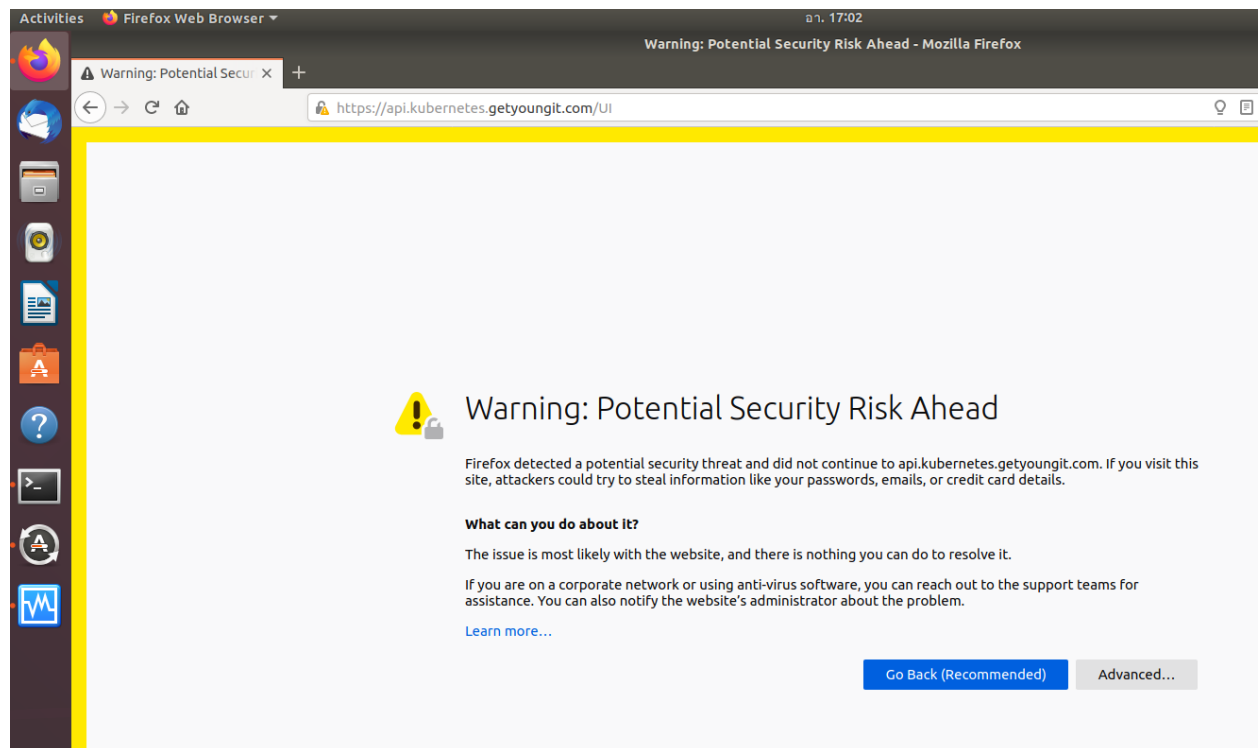
kubectl config view

```
administrator@ubuntu01:~/kubernetes-course/dashboard$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /home/administrator/.minikube/ca.crt
    server: https://192.168.99.100:8443
    name: minikube
contexts:
- context:
    cluster: minikube
    user: minikube
    name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: minikube
  user:
    client-certificate: /home/administrator/.minikube/profiles/minikube/client.crt
    client-key: /home/administrator/.minikube/profiles/minikube/client.key
administrator@ubuntu01:~/kubernetes-course/dashboard$
```

พบว่าเราสามารถที่จะเปลี่ยนมาใช้ Username และรหัสผ่านแทนได้จะกล่าวในหลักสูตรต่อไป

## 7. การเข้าไปใน Dashboard (ข้ามเพราะยังไม่ได้ระบุชื่อผู้ใช้ และรหัสผ่าน)

เข้าไปในเครื่องที่อยู่ Cloud แล้วพิมพ์ต่อท้ายด้วย /UI





- คลิกที่ Advanced
  - และคลิก Accept the Risk and Continue
  - ระบุชื่อผู้ใช้ และรหัสผ่าน
8. ถ้าเป็น minikube ให้พิมพ์คำสั่ง  
minikube dashboard -url  
เพื่อดู URL
9. พิมพ์ URL ที่ได้เพื่อเข้าสู่หน้า Dashboard