# #040 Kubernetes - Deployment Lab 2

# Introduction

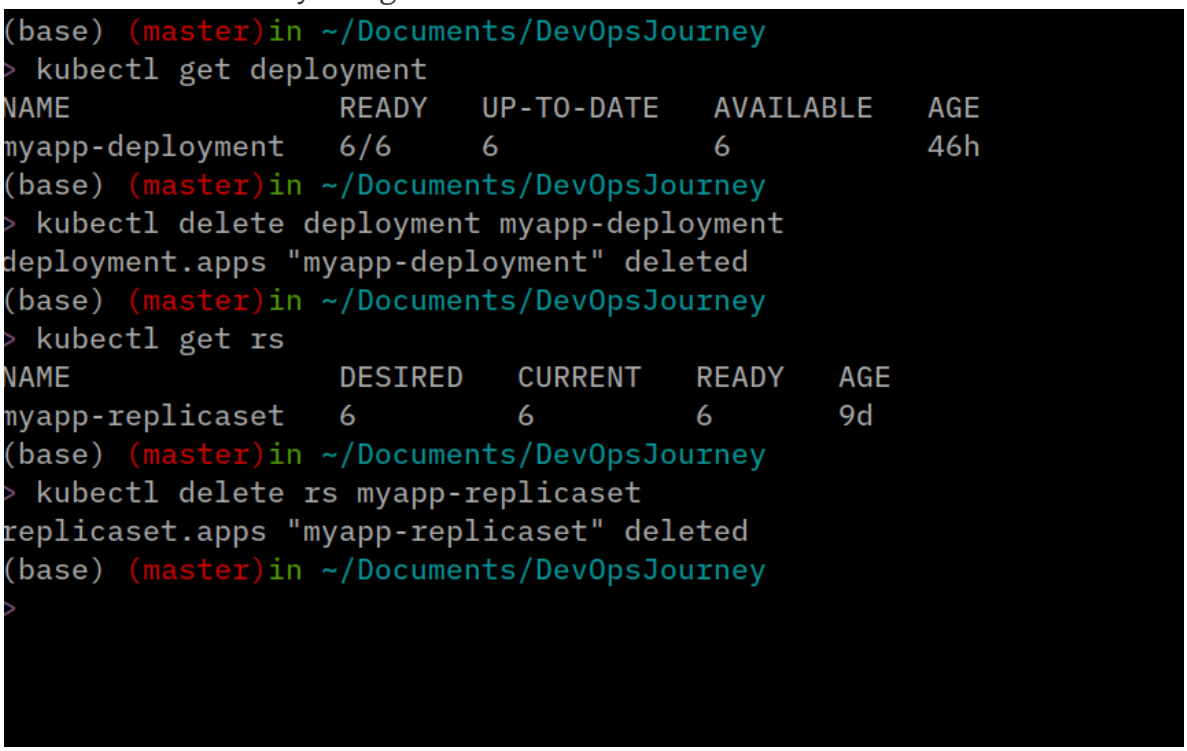

this is part 40 from the journey it's a long journey(360 day) so go please check previous parts , and if you need to walk in the journey with me please make sure to follow because I may post more than once in 1 Day but surely I will post daily at least one 😍.

And I will cover lot of tools as we move on.

---

# Prepare images for this Lab

first let's delete every thing we done so far




```
kubectl delete deployment myapp-deployment
kubectl delete rs myapp-replicaset
```

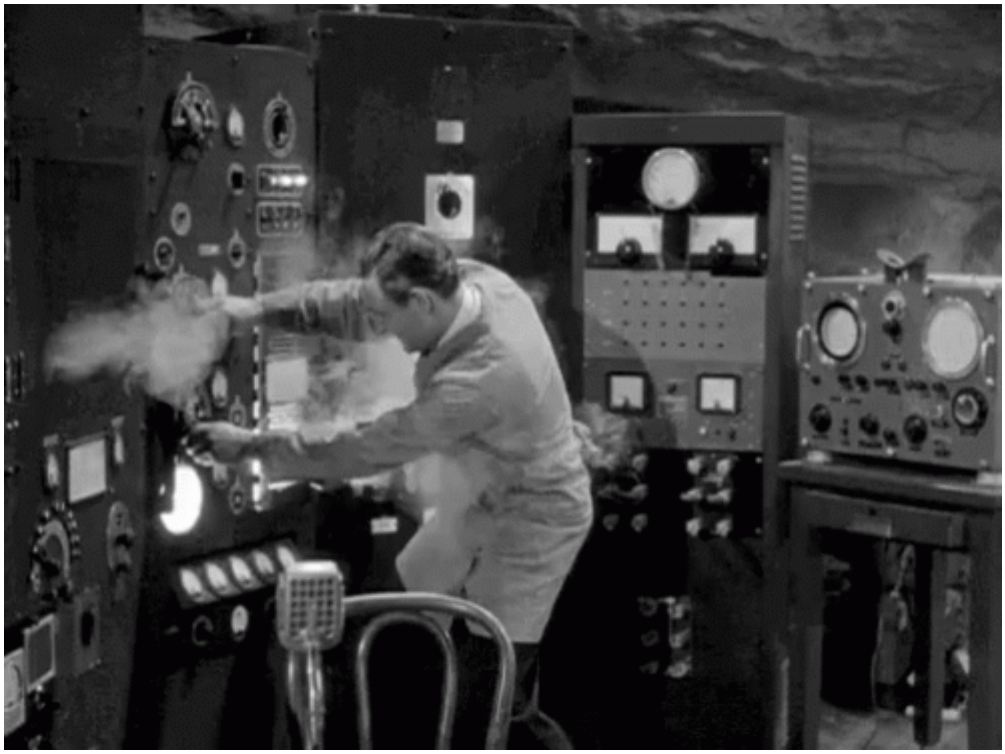
let's recreate a new container from scratch will be hosted on my dockerhub. It's same as old one but now we need to do some edits to the image. here how we can build and push to dockerhub.
I am going to build them as tags 1.0 and 2.0
1.0 is same as old one , 2.0 will have some edits on image.
So we can see how to switch between two versions.
it's not my API so it doesn't have an docker-compose file.
We can add one but meh let's just build it without docker-compose because it's doesn't have a lot to do with it.

if you already have the DevOps folder just pull it and you will get the new files.
if not go and clone the repo

---

# Lab 2

Now after we have our image with tag 1.0 and 2.0 we can start.
You don't need to build them , because I already do they are on my docker hub as public just follow the steps , Kubernetes will download them for you when you run the yml file.



first let's look what we have in the folder of today lab

```
(base) (master)in ~/Documents/DevOpsJourney/app_040
> l
v1/  v2/  app_040_v1.yml  app_040_v2.yml
```

v1 and v2 contain the source code of the simple-api , v2 have different message than v1 that will shown.
app_040_v1.yml  app_040_v2.yml are the Kubernetes configs the only difference

between 1 and 2 it's the version in image here a look at v1

```yaml
app_040_v1.yml
kind: Deployment
metadata:
  name: myapp-deployment

spec:
  template:
    metadata:
      name: first-pod-dec
      labels:
        app: myapp
        type: restapi

    spec:
      containers:
        - name: simple-api
          image: omarelkhatib/app_040:1.0

  replicas: 6
  selector:
    matchLabels:
      app: myapp
      type: restapi
```

let's create our deployment now.

```
(base) (master)in ~/Documents/DevOpsJourney/app_040
> kubectl create -f app_040_v1.yml --record
deployment.apps/myapp-deployment created
```

```
kubectl create -f app_040_v1.yml --record
```

you know the process from the older part , he going to create 6 pods and...
take a break while the pods are creating

```
(base) (master)in ~/Documents/DevOpsJourney/app_040
> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
myapp-deployment-7c74677865-44xpp   1/1     Running   0          11s
myapp-deployment-7c74677865-7pjvd   1/1     Running   0          11s
myapp-deployment-7c74677865-9bprs   1/1     Running   0          11s
myapp-deployment-7c74677865-c2qz2   1/1     Running   0          11s
myapp-deployment-7c74677865-rlgnc   1/1     Running   0          11s
myapp-deployment-7c74677865-xwbpz   1/1     Running   0          11s
(base) (master)in ~/Documents/DevOpsJourney/app_040
>
```

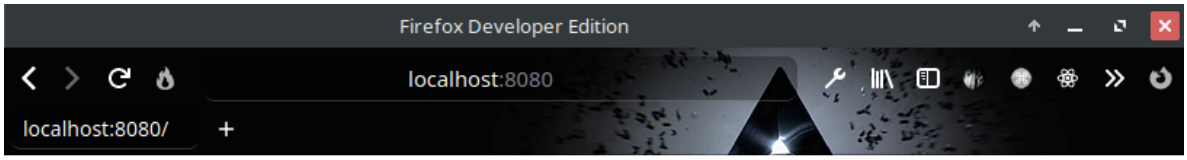
```
kubectl get pods
```

all my pods are ready , so we are good to move on

```
(base) (master)in ~/Documents/DevOpsJourney/app_040
> kubectl port-forward deployments/myapp-deployment 8080
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
```

```
kubectl port-forward deployments/myapp-deployment 8080
```

this trick will start our app , 8080 is the port we specify in the source code.




Simple API says "Hello!"

Now we can see in browser we have our v1 running.

```
^C(base) (master)in ~/Documents/DevOpsJourney/app_040
> kubectl apply -f app_040_v2.yml
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
deployment.apps/myapp-deployment configured
```

```
kubectl apply -f app_040_v2.yml
```

this is how we update the version from v1 to v2

```
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=myapp
           type=restapi
  Containers:
   simple-api:
    Image:        omarelkhatib/app_040:2.0
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
```

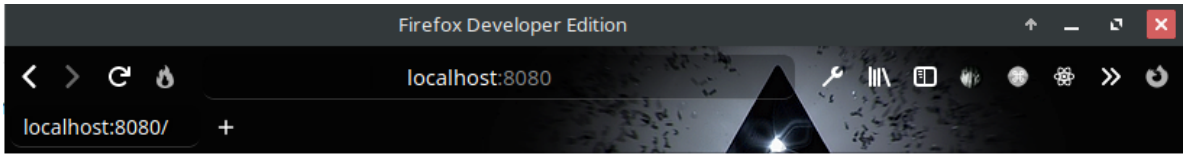
```
kubectl describe deployments myapp-deployment
```

here we can see we have the new version 2.0

```
(base) (master)in ~/Documents/DevOpsJourney/app_040
> kubectl port-forward deployments/myapp-deployment 8080
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
```

```
kubectl port-forward deployments/myapp-deployment 8080
```

now let's go to browser and refresh our page and see what happens.




Simple API says I am version 2

voila we have now the v2 live!

---

# little talk about DevOps

I edit the app in this lab , the moment between the commit and the customer see the updates is the DevOps Engineer. In advanced parts when we push our code all those stuff will be automated , it's called Contentious Integration and Contentious Delivery as shortcut CI/CD.
In our case the task will be after commit he will send the version to the docker hub then Kubernetes will handle the new image.
The automation here will be described in a script written by you (the DevOps).
Basically this is the core of the DevOps.

# Back to lab

let's say now we don't like this version we need to roll back to v1.
let's look at the history and decide what we need to rollback

```
^C(base) (master)in ~/Documents/DevOpsJourney/app_040
> kubectl rollout history deployment/myapp-deployment
deployment.apps/myapp-deployment
REVISION  CHANGE-CAUSE
1         kubectl create --filename=app_040_v1.yml --record=true
2         kubectl create --filename=app_040_v1.yml --record=true
```

```
kubectl rollout history deployment/myapp-deployment
```

I need the revision 1 , so I will go back to it using :

```
(base) (master)in ~/Documents/DevOpsJourney/app_040
> kubectl rollout undo deployment/myapp-deployment --to-revision 1
deployment.apps/myapp-deployment rolled back
```

```
kubectl rollout undo deployment/myapp-deployment --to-revision 1
```

now let's check what version it is

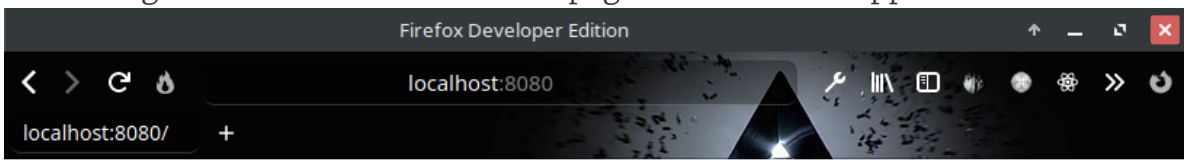
```
kubectl describe deployments myapp-deployment
```

```
Containers:
 simple-api:
  Image:        omarelkhatib/app_040:1.0
  Port:         <none>
  Host Port:    <none>
```

here we can see now the version is back to 1

```
kubectl port-forward deployments/myapp-deployment 8080
```

now let's go to browser and refresh our page and see what happens.




Simple API says "Hello!"

It's the old version again!