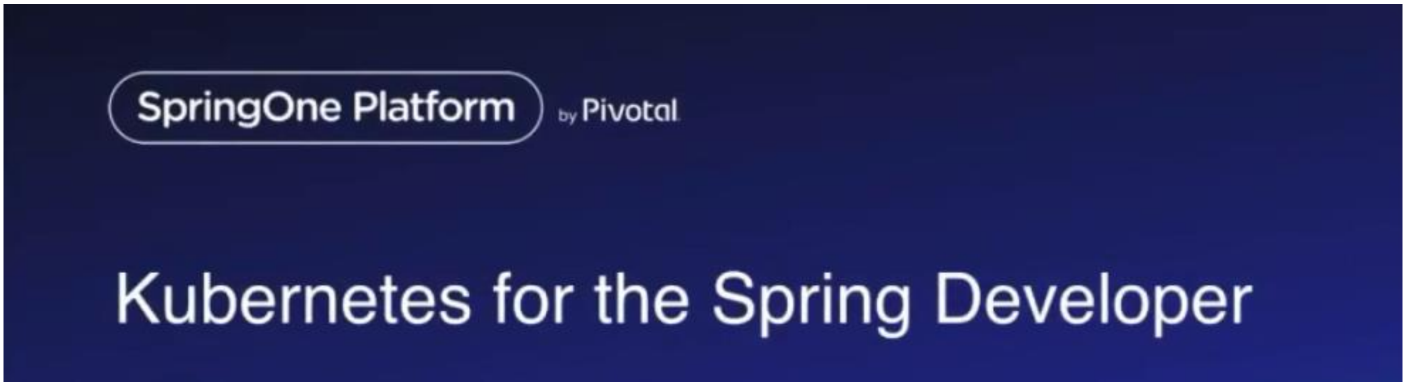
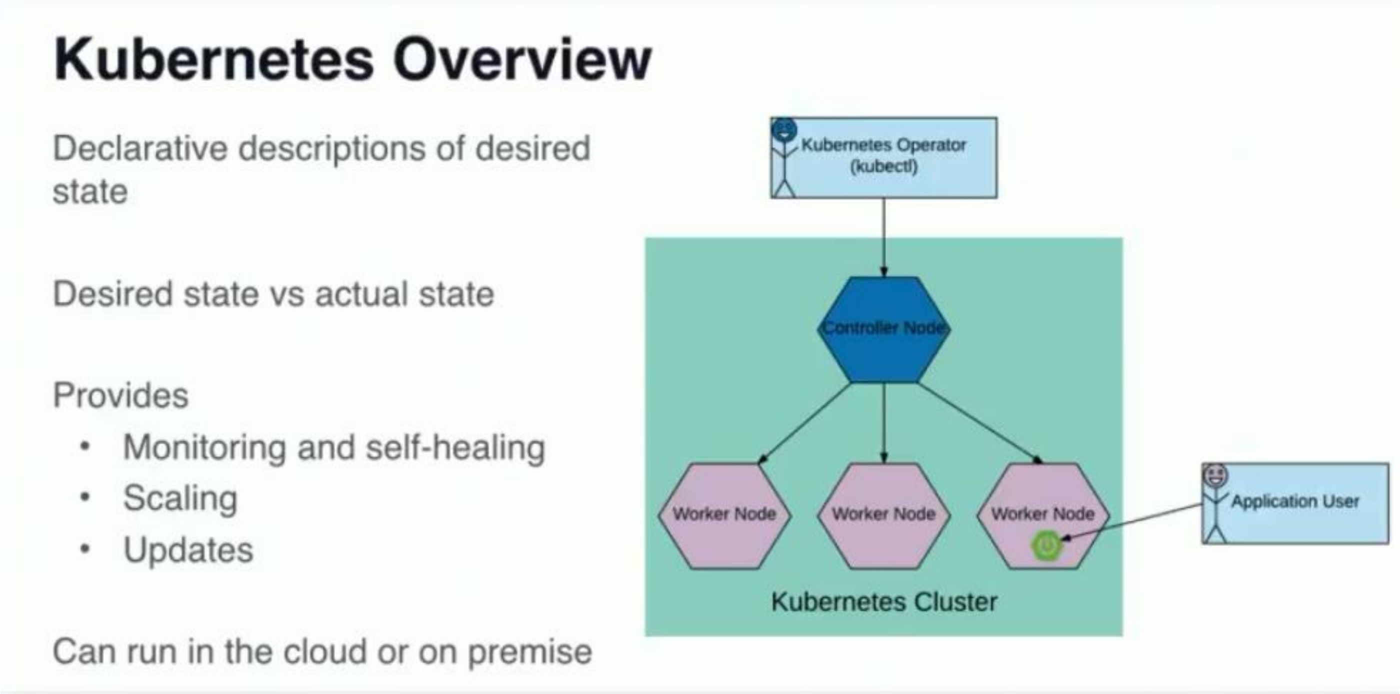


SpringOne Platform by Pivotal

Kubernetes for the Spring Developer



Kubernetes Overview

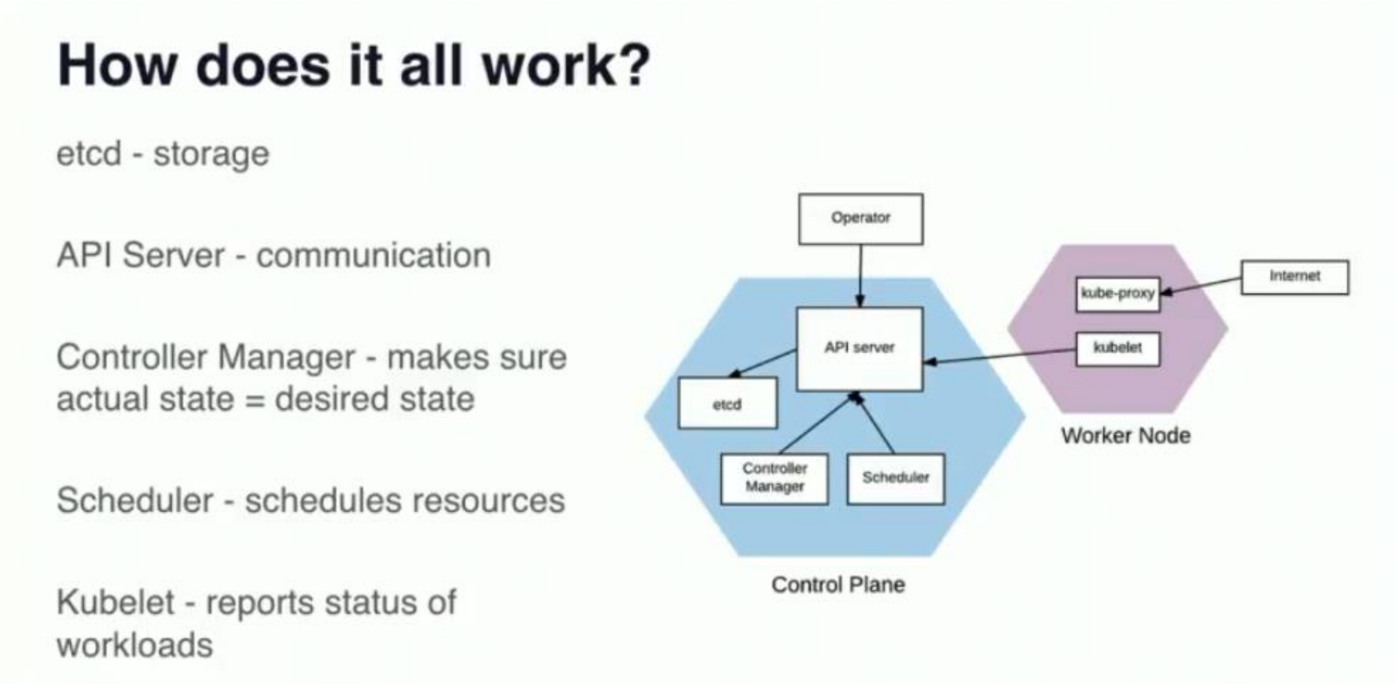
Declarative descriptions of desired state

Desired state vs actual state

Provides

- Monitoring and self-healing
- Scaling
- Updates

Can run in the cloud or on premise



How does it all work?

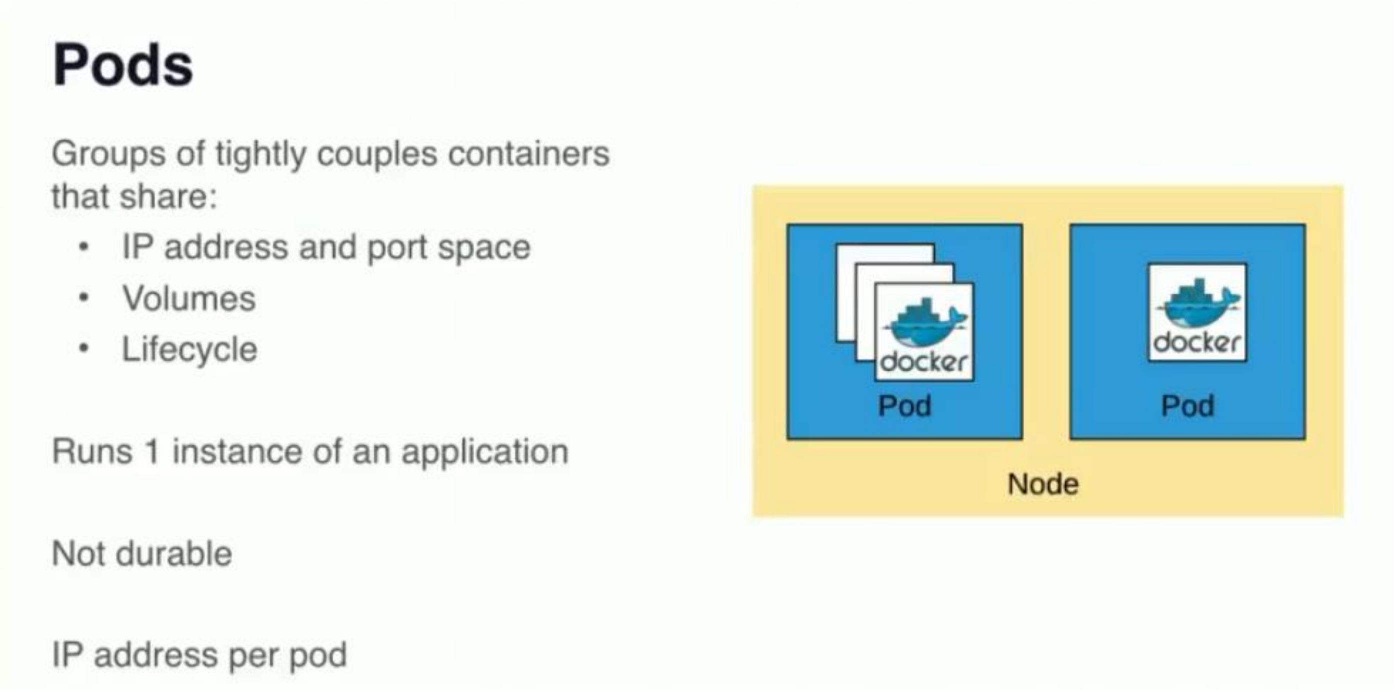
etcd - storage

API Server - communication

Controller Manager - makes sure actual state = desired state

Scheduler - schedules resources

Kubelet - reports status of workloads



Pods

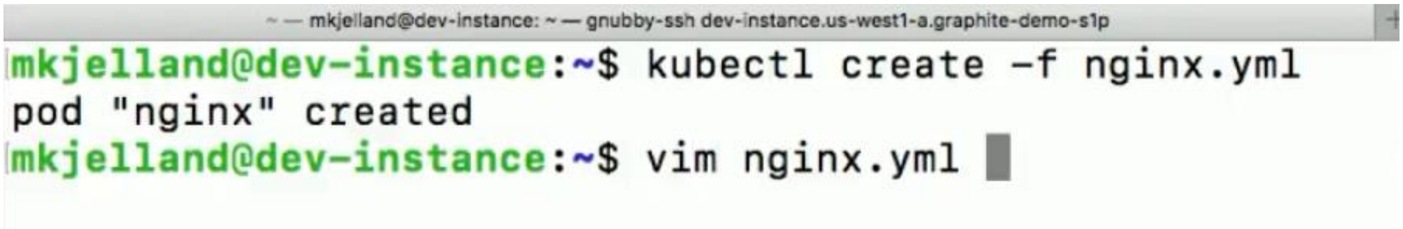
Groups of tightly couples containers that share:

- IP address and port space
- Volumes
- Lifecycle

Runs 1 instance of an application

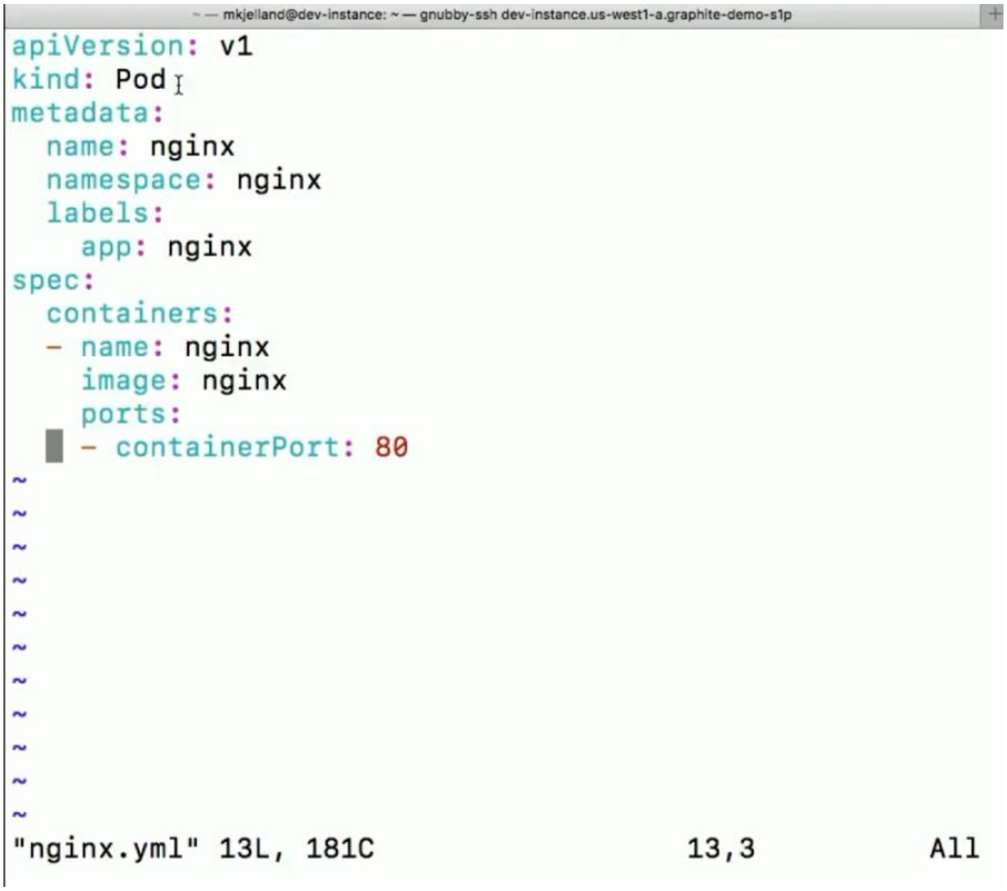
Not durable

IP address per pod



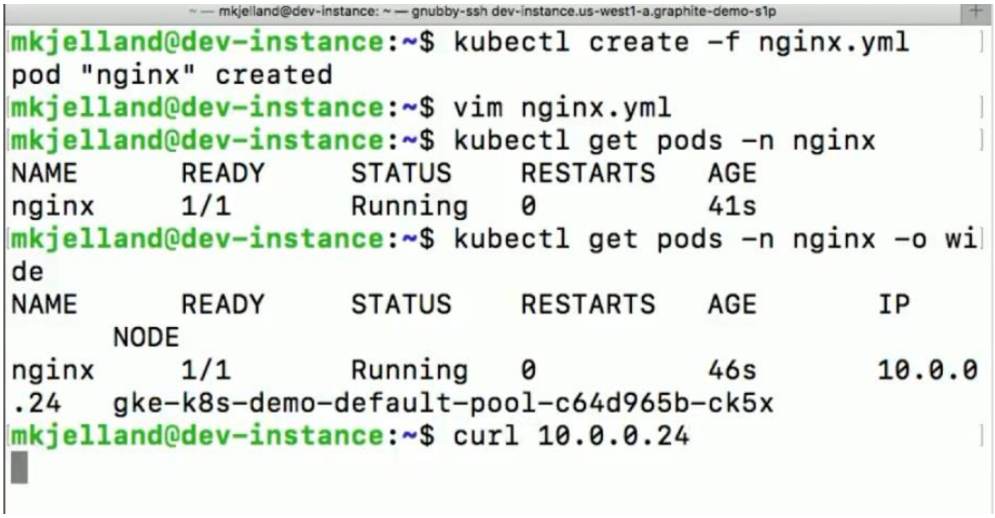
~ — mkjelland@dev-instance: ~ — gnubby-ssh dev-instance.us-west1-a.graphite-demo-s1p

```
mkjelland@dev-instance:~$ kubectl create -f nginx.yml
pod "nginx" created
mkjelland@dev-instance:~$ vim nginx.yml
```

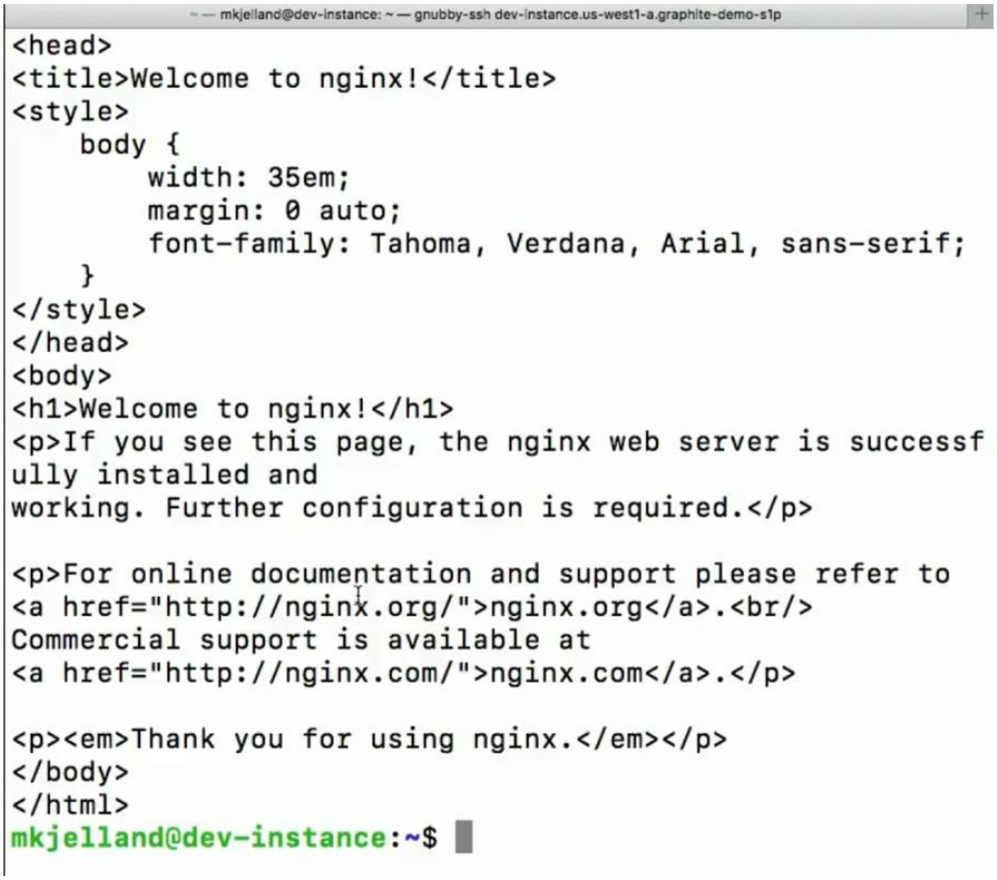


~ — mkjelland@dev-instance: ~ — gnubby-ssh dev-instance.us-west1-a.graphite-demo-s1p

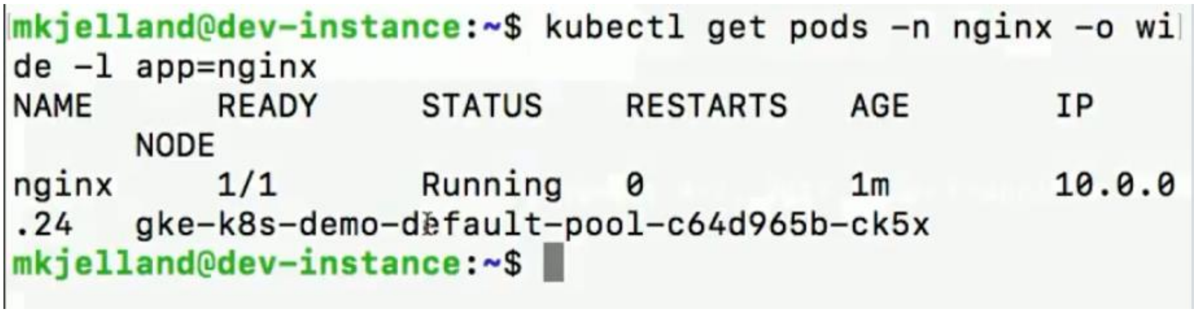
```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: nginx
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
~
~
~
~
~
~
~
~
~
~
~
"nginx.yml" 13L, 181C                         13,3           All
```


~ — mkjelland@dev-instance: ~ — gnubby-ssh dev-instance.us-west1-a.graphite-demo-s1p

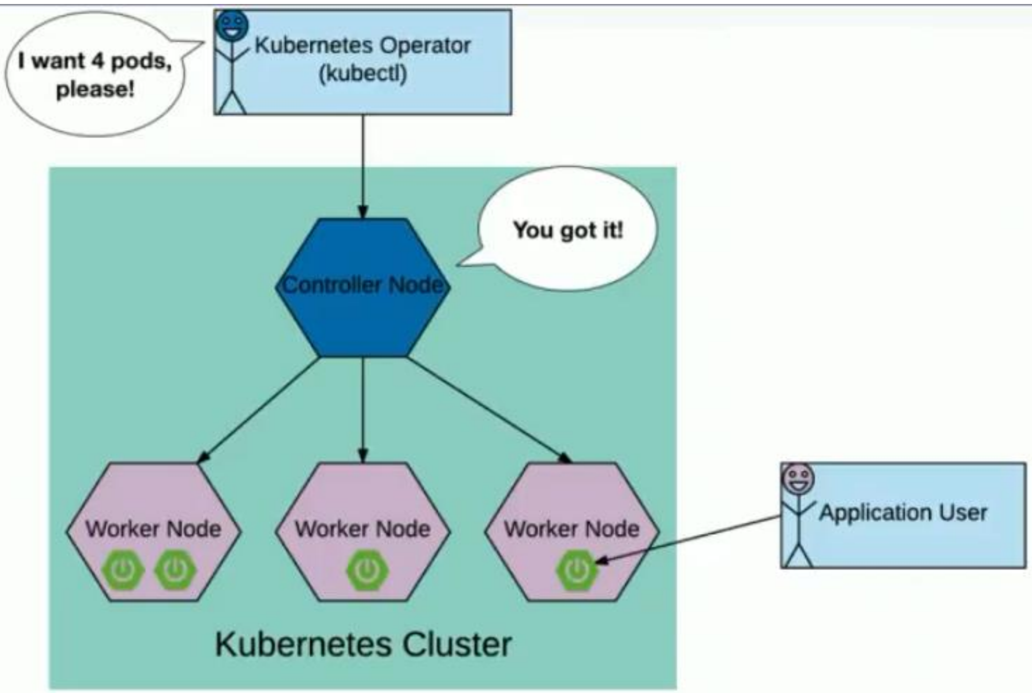
```
mkjelland@dev-instance:~$ kubectl create -f nginx.yml
pod "nginx" created
mkjelland@dev-instance:~$ vim nginx.yml
mkjelland@dev-instance:~$ kubectl get pods -n nginx
NAME      READY     STATUS    RESTARTS   AGE
nginx     1/1       Running   0          41s
mkjelland@dev-instance:~$ kubectl get pods -n nginx -o wi
de
NAME      READY     STATUS    RESTARTS   AGE       IP
      NODE
nginx     1/1       Running   0          46s       10.0.0
.24   gke-k8s-demo-default-pool-c64d965b-ck5x
mkjelland@dev-instance:~$ curl 10.0.0.24
```

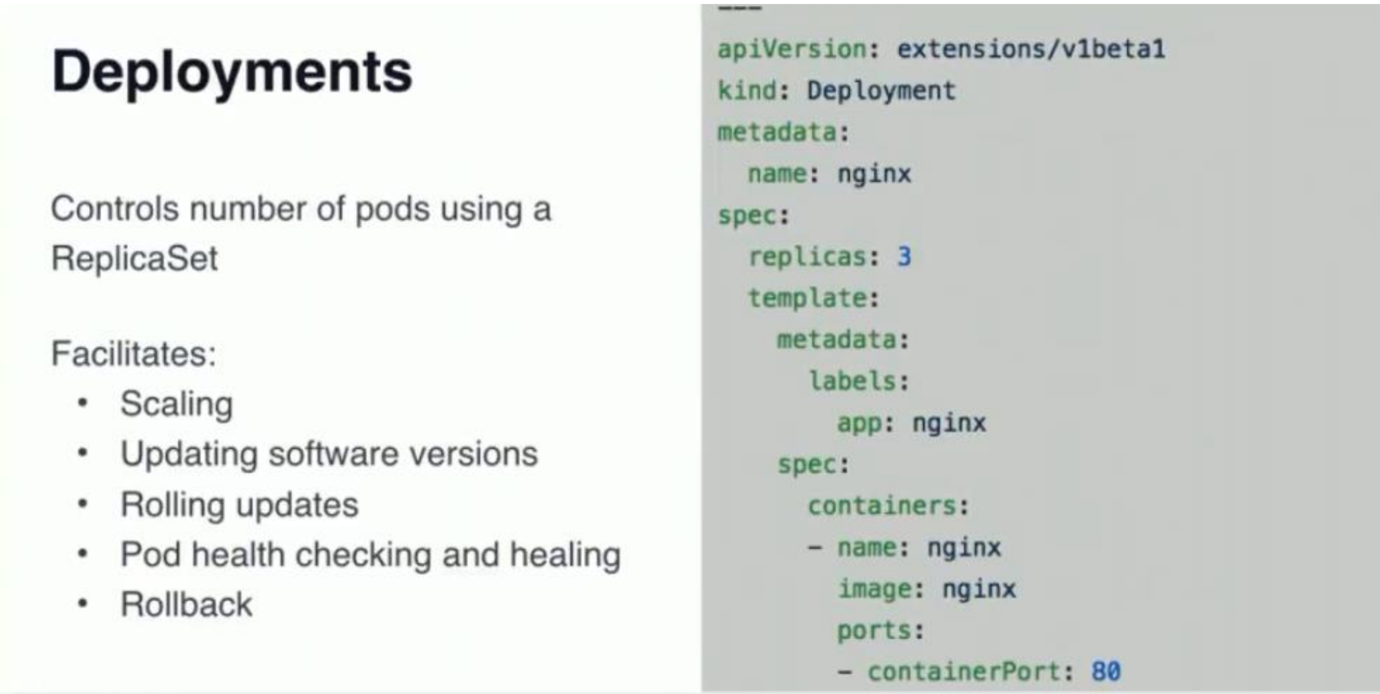

~ — mkjelland@dev-instance: ~ — gnubby-ssh dev-instance.us-west1-a.graphite-demo-s1p

```
<head>
<title>Welcome to nginx!</title>
<style>
    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 successf
ully 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>
mkjelland@dev-instance:~$
```



```
mkjelland@dev-instance:~$ kubectl get pods -n nginx -o wi
de -l app=nginx
NAME      READY     STATUS    RESTARTS   AGE       IP
      NODE
nginx     1/1       Running   0          1m        10.0.0
.24   gke-k8s-demo-default-pool-c64d965b-ck5x
mkjelland@dev-instance:~$
```





Deployments

Controls number of pods using a ReplicaSet

Facilitates:

- Scaling
- Updating software versions
- Rolling updates
- Pod health checking and healing
- Rollback

```yaml
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

Services

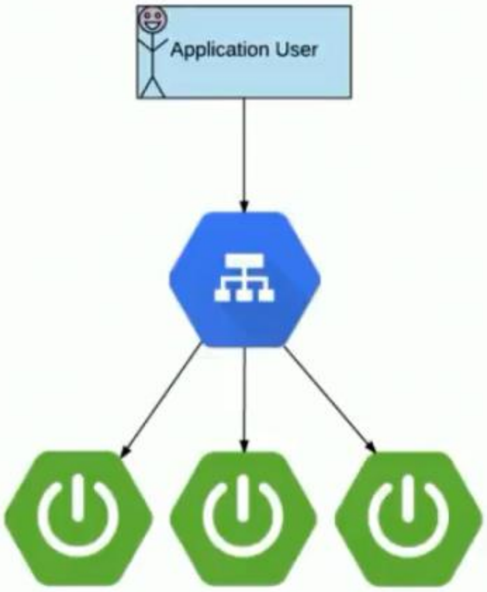
Access to dynamic set of pods

Uses labels to find pods

Load balances traffic across nodes

Types:

- ClusterIP
- NodePort
- LoadBalancer
- ExternalName



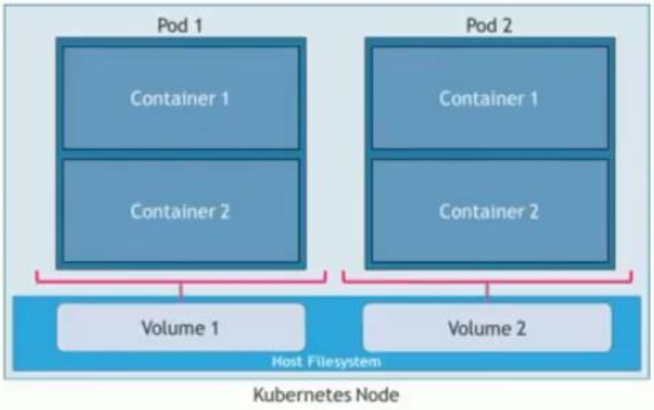
Volumes

hostPath

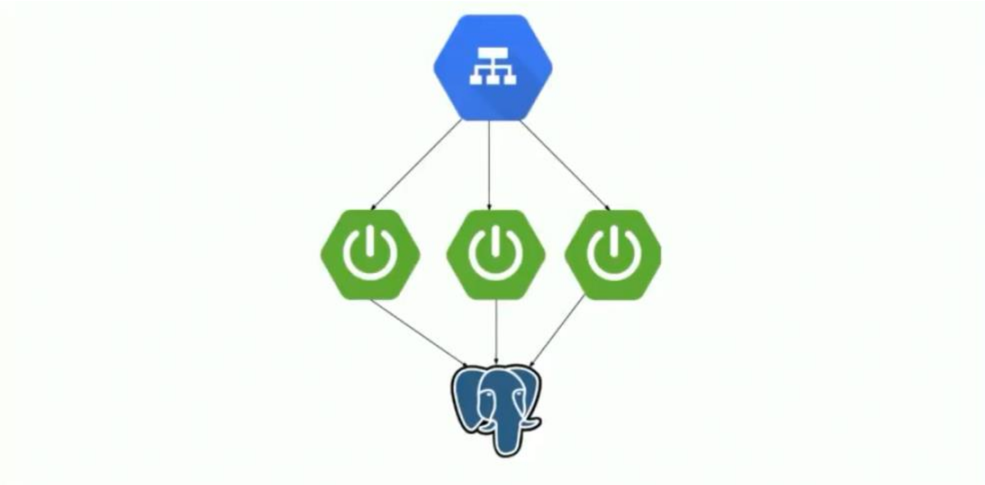
emptyDir

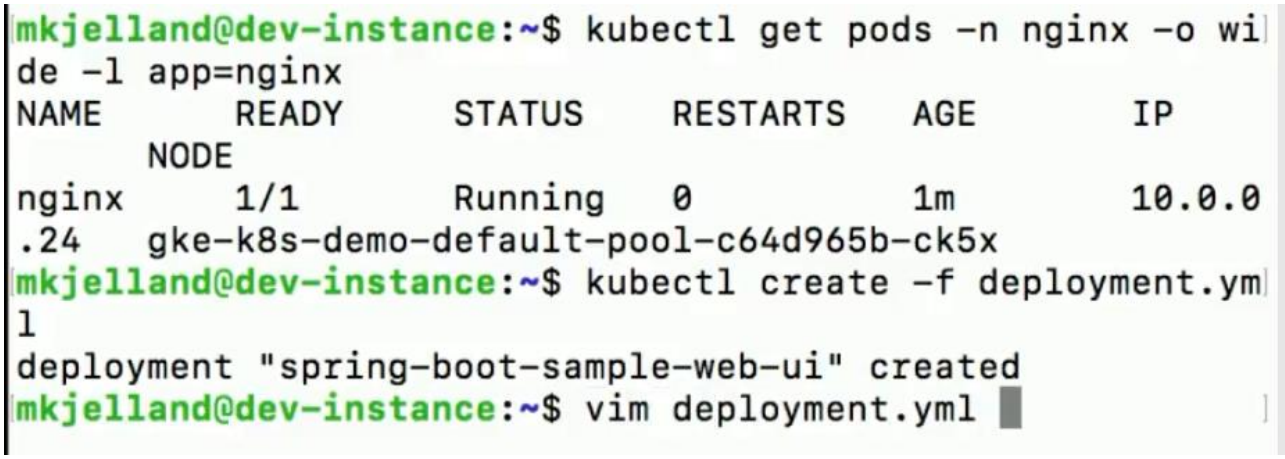
Infrastructure specific persistent disks

secrets



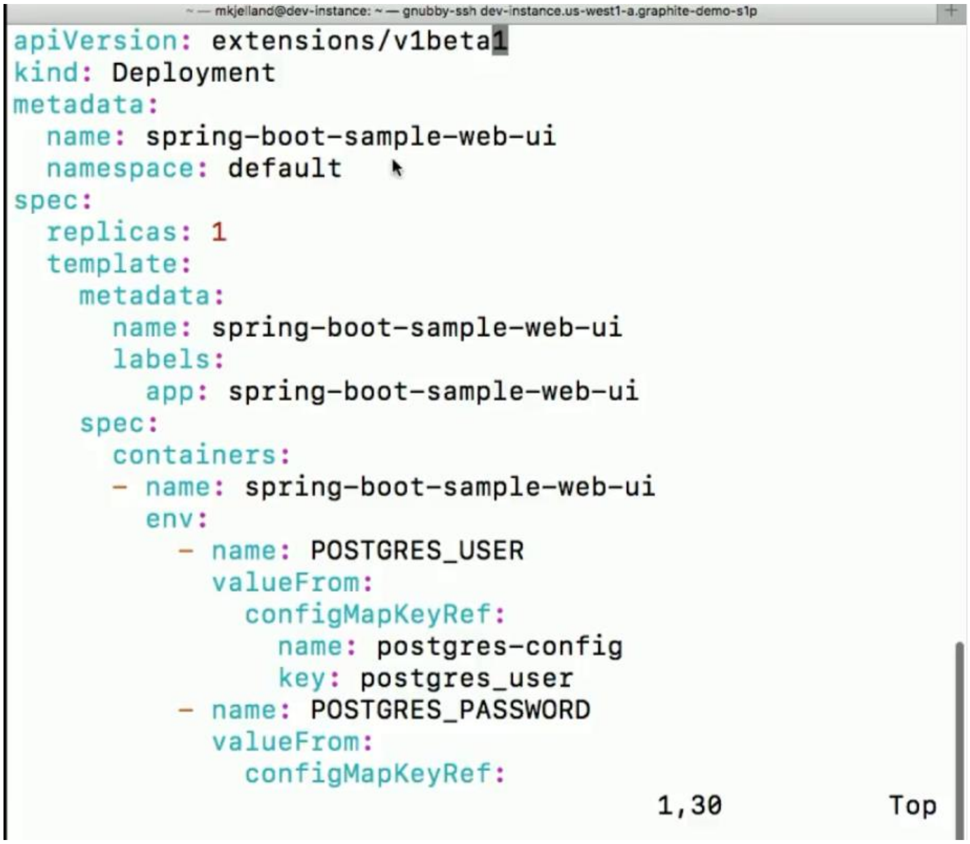
Demo





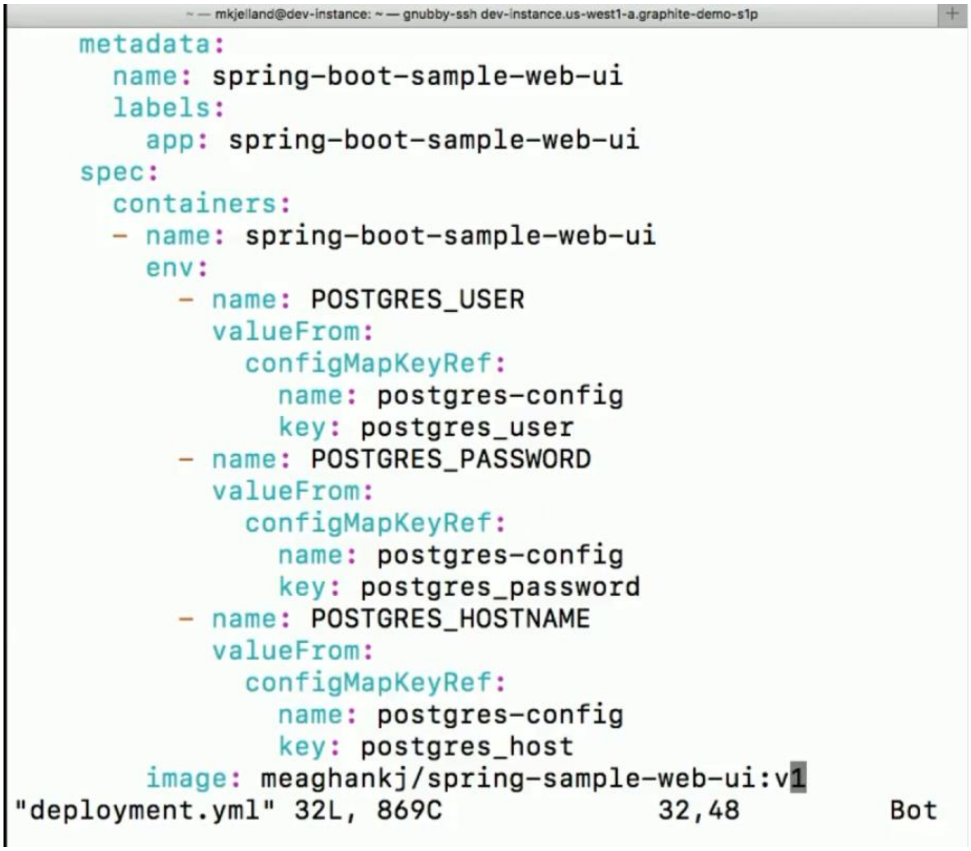
```
mkjelland@dev-instance:~$ kubectl get pods -n nginx -o wi
de -l app=nginx
NAME      READY     STATUS    RESTARTS   AGE       IP
      NODE
nginx     1/1       Running   0          1m        10.0.0
.24   gke-k8s-demo-default-pool-c64d965b-ck5x
mkjelland@dev-instance:~$ kubectl create -f deployment.ym
l
deployment "spring-boot-sample-web-ui" created
mkjelland@dev-instance:~$ vim deployment.yml
```

We create one of the pods using the ***$ kubectl create -f deployment.yml*** command

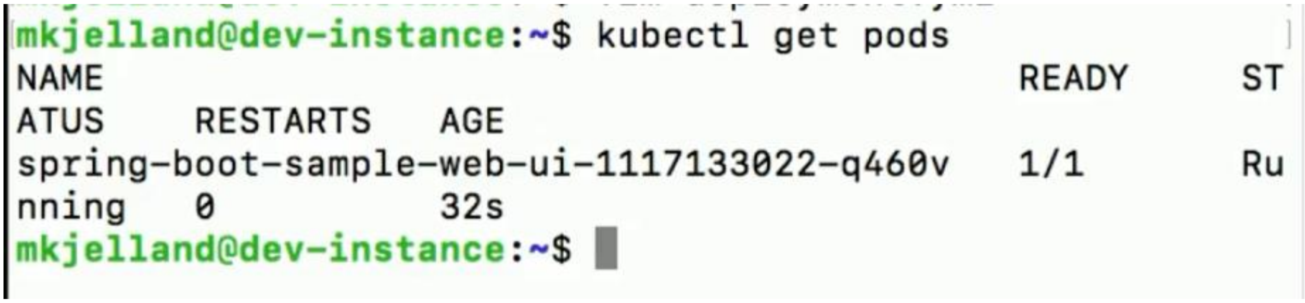


~ — mkjelland@dev-instance: ~ — gnubby-ssh dev-instance.us-west1-a.graphite-demo-s1p

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: spring-boot-sample-web-ui
  namespace: default
spec:
  replicas: 1
  template:
    metadata:
      name: spring-boot-sample-web-ui
      labels:
        app: spring-boot-sample-web-ui
    spec:
      containers:
      - name: spring-boot-sample-web-ui
        env:
          - name: POSTGRES_USER
            valueFrom:
              configMapKeyRef:
                name: postgres-config
                key: postgres_user
          - name: POSTGRES_PASSWORD
            valueFrom:
              configMapKeyRef:
                                           1,30          Top
```

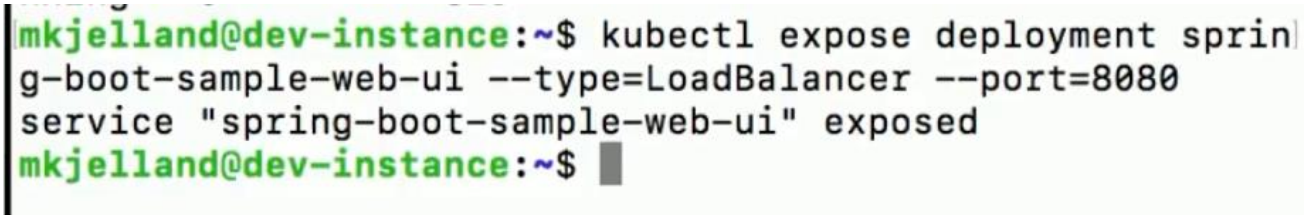

~ — mkjelland@dev-instance: ~ — gnubby-ssh dev-instance.us-west1-a.graphite-demo-s1p

```
  metadata:
   name: spring-boot-sample-web-ui
   labels:
    app: spring-boot-sample-web-ui
  spec:
   containers:
   - name: spring-boot-sample-web-ui
    env:
     - name: POSTGRES_USER
      valueFrom:
       configMapKeyRef:
        name: postgres-config
        key: postgres_user
     - name: POSTGRES_PASSWORD
      valueFrom:
       configMapKeyRef:
        name: postgres-config
        key: postgres_password
     - name: POSTGRES_HOSTNAME
      valueFrom:
       configMapKeyRef:
        name: postgres-config
        key: postgres_host
    image: meaghankj/spring-sample-web-ui:v1
"deployment.yml" 32L, 869C             32,48          Bot
```



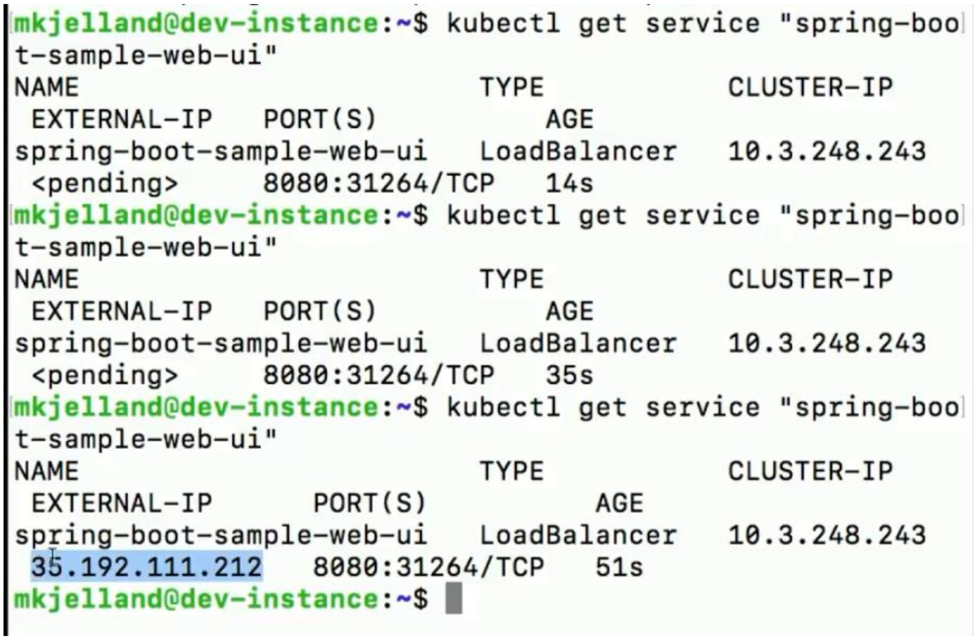
```
mkjelland@dev-instance:~$ kubectl get pods
NAME                                        READY     ST
ATUS     RESTARTS   AGE
spring-boot-sample-web-ui-1117133022-q460v   1/1       Ru
nning   0          32s
mkjelland@dev-instance:~$
```

We use the ***$ kubectl get pods*** command to see that we have 1 pod running now



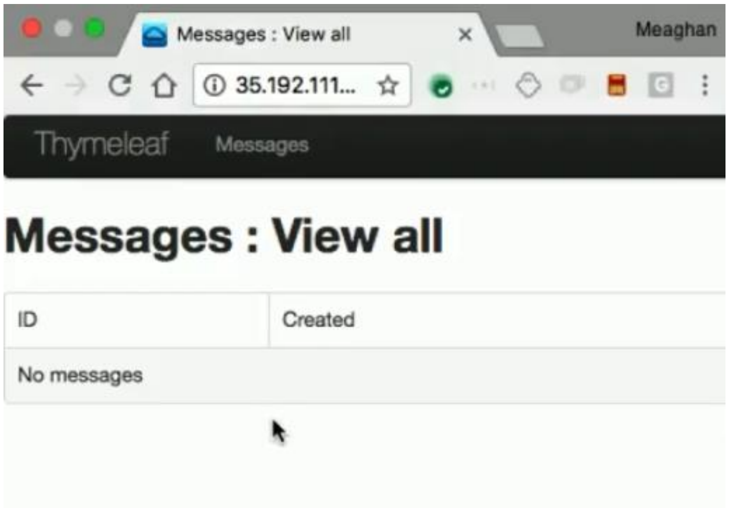
```
mkjelland@dev-instance:~$ kubectl expose deployment sprin
g-boot-sample-web-ui --type=LoadBalancer --port=8080
service "spring-boot-sample-web-ui" exposed
mkjelland@dev-instance:~$
```

We use the ***$ kubectl expose deployment spring-boot-sample-web-ui - -type=LoadBalancer - -port=8080*** command to expose that pod as a service with a type of ***LoadBalancer*** because the ***default is clusterIp***.



```
mkjelland@dev-instance:~$ kubectl get service "spring-boo
t-sample-web-ui"
NAME                        TYPE           CLUSTER-IP
 EXTERNAL-IP   PORT(S)          AGE
spring-boot-sample-web-ui   LoadBalancer   10.3.248.243
 <pending>     8080:31264/TCP   14s
mkjelland@dev-instance:~$ kubectl get service "spring-boo
t-sample-web-ui"
NAME                        TYPE           CLUSTER-IP
 EXTERNAL-IP   PORT(S)          AGE
spring-boot-sample-web-ui   LoadBalancer   10.3.248.243
 <pending>     8080:31264/TCP   35s
mkjelland@dev-instance:~$ kubectl get service "spring-boo
t-sample-web-ui"
NAME                        TYPE           CLUSTER-IP
 EXTERNAL-IP      PORT(S)          AGE
spring-boot-sample-web-ui   LoadBalancer   10.3.248.243
 35.192.111.212   8080:31264/TCP   51s
mkjelland@dev-instance:~$
```

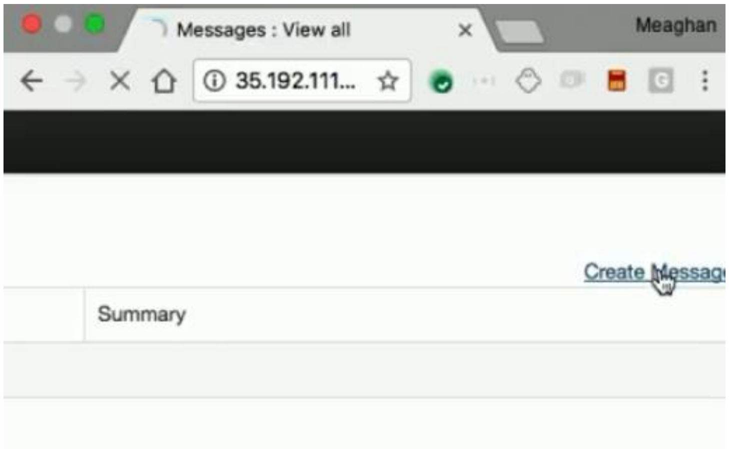
After exposing the pod deployment as a service, we then use the ***$ kubectl get service "spring-boot-sample-web-ui"*** command to check if the service has been created. We finally get an IP address that we can see the app on



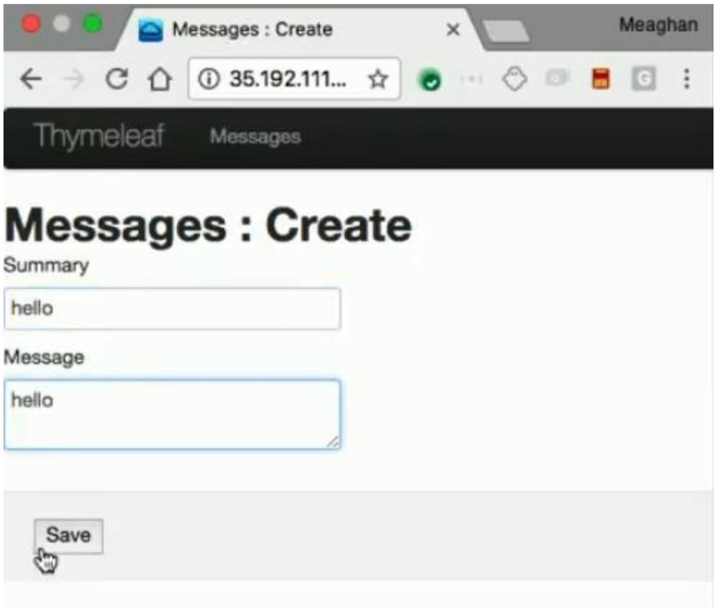
Messages : View all

ID	Created
No messages	

This is just an app that stores messages in-memory at the moment, this means that we will lose the data once we delete this pod



	Summary



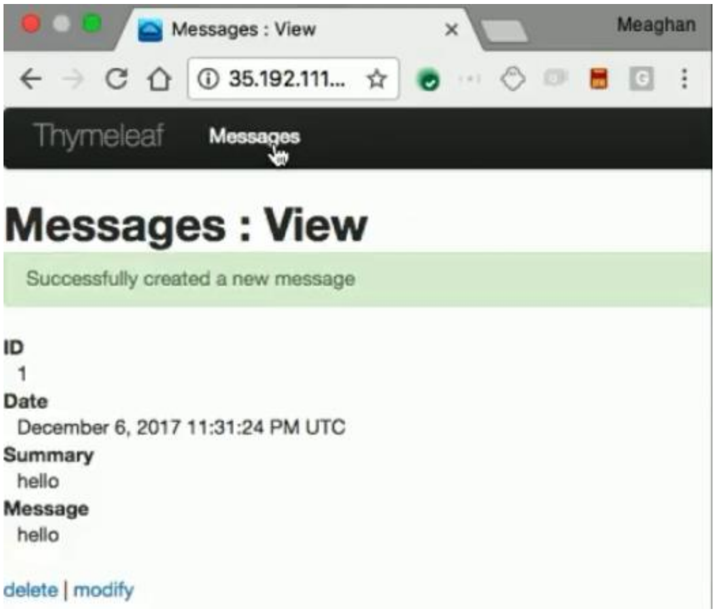
Messages : Create

Summary

hello

Message

hello



Messages : View

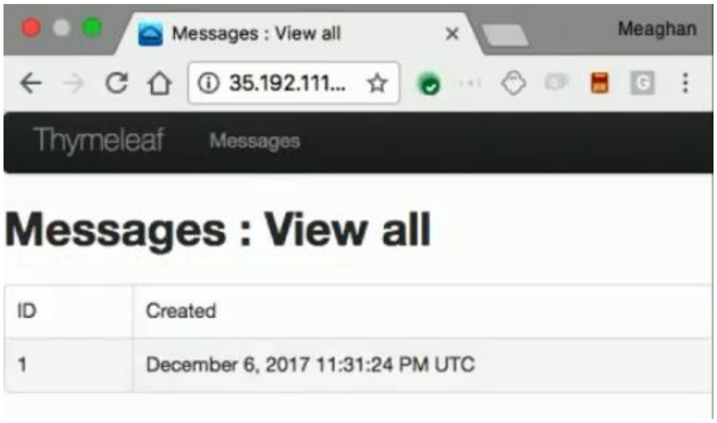
Successfully created a new message

ID
1

Date
December 6, 2017 11:31:24 PM UTC

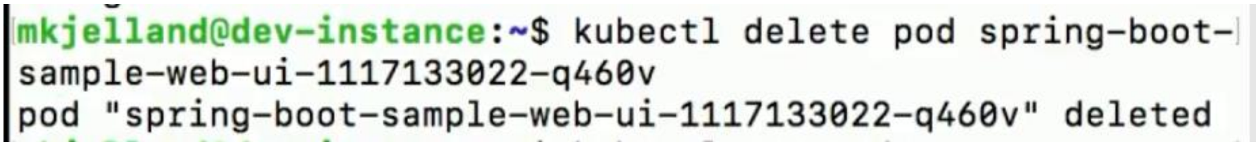
Summary
hello

Message
hello



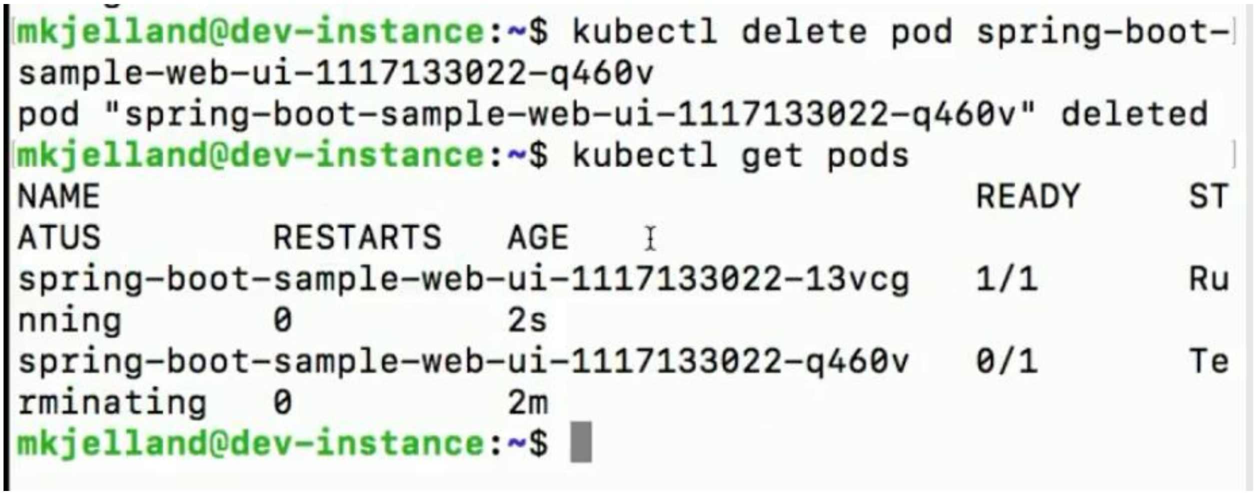
Messages : View all

ID	Created
1	December 6, 2017 11:31:24 PM UTC



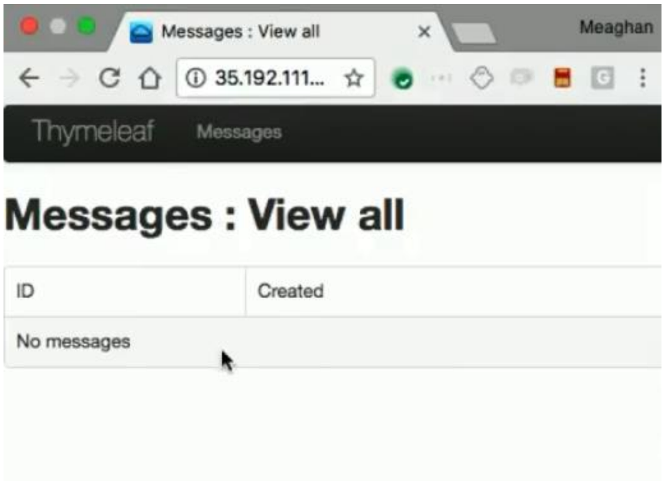
```
mkjelland@dev-instance:~$ kubectl delete pod spring-boot-
sample-web-ui-1117133022-q460v
pod "spring-boot-sample-web-ui-1117133022-q460v" deleted
```

We can now delete the pod using the ***$ kubectl delete pod spring-boot-sample-web-ui-111713302-q460v*** command, the data should now be gone with the old pod and the new replacement pod will not have our old messages data to display.



```
mkjelland@dev-instance:~$ kubectl delete pod spring-boot-
sample-web-ui-1117133022-q460v
pod "spring-boot-sample-web-ui-1117133022-q460v" deleted
mkjelland@dev-instance:~$ kubectl get pods
NAME                                          READY     ST
ATUS        RESTARTS   AGE
spring-boot-sample-web-ui-1117133022-13vcg   1/1       Ru
nning       0          2s
spring-boot-sample-web-ui-1117133022-q460v   0/1       Te
rminating   0          2m
mkjelland@dev-instance:~$
```

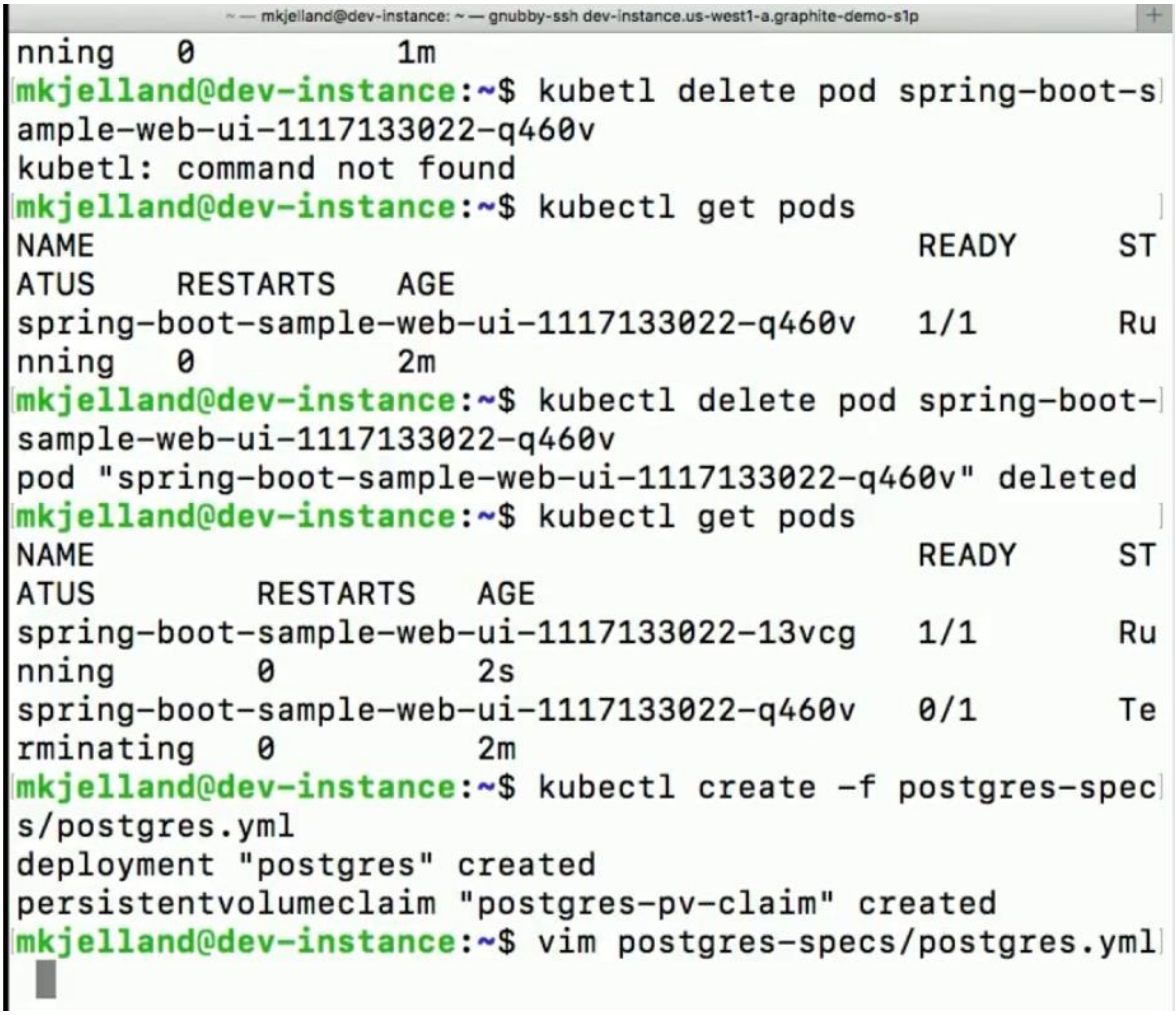
Kubernetes replaces the deleted pod immediately with a new pod as seen above



Messages : View all

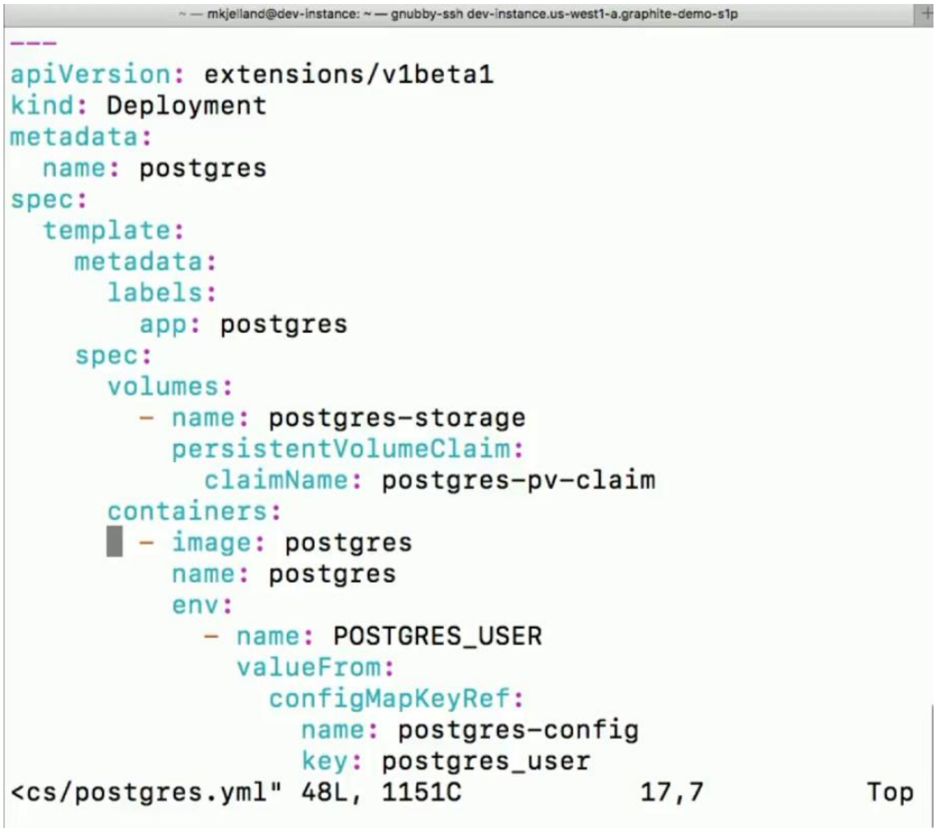
ID	Created
No messages	

When we now refresh the application, we see that we don't have any messages anymore because we did not persist the data.


~ — mkjelland@dev-instance: ~ — gnubby-ssh dev-instance.us-west1-a.graphite-demo-s1p

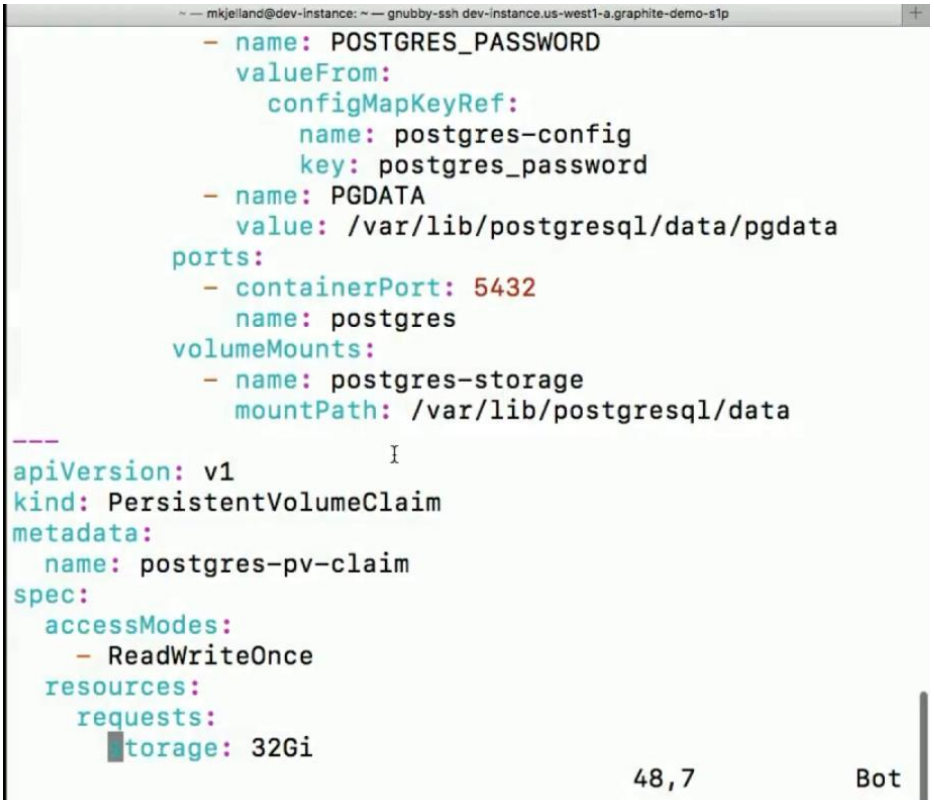
```
nning   0          1m
mkjelland@dev-instance:~$ kubetl delete pod spring-boot-s
ample-web-ui-1117133022-q460v
kubetl: command not found
mkjelland@dev-instance:~$ kubectl get pods
NAME                                         READY     ST
ATUS    RESTARTS   AGE
spring-boot-sample-web-ui-1117133022-q460v   1/1       Ru
nning   0          2m
mkjelland@dev-instance:~$ kubectl delete pod spring-boot-
sample-web-ui-1117133022-q460v
pod "spring-boot-sample-web-ui-1117133022-q460v" deleted
mkjelland@dev-instance:~$ kubectl get pods
NAME                                         READY     ST
ATUS        RESTARTS   AGE
spring-boot-sample-web-ui-1117133022-13vcg   1/1       Ru
nning       0          2s
spring-boot-sample-web-ui-1117133022-q460v   0/1       Te
rminating   0          2m
mkjelland@dev-instance:~$ kubectl create -f postgres-spec
s/postgres.yml
deployment "postgres" created
persistentvolumeclaim "postgres-pv-claim" created
mkjelland@dev-instance:~$ vim postgres-specs/postgres.yml
```

Now we can deploy a PostGres pod with a persistent disk used for keeping the data available beyond pod sessions, we use the ***$ kubectl create -f postgres-specs/postgres.yml*** command to create the PostGres pod instance


~ — mkjelland@dev-instance: ~ — gnubby-ssh dev-instance.us-west1-a.graphite-demo-s1p

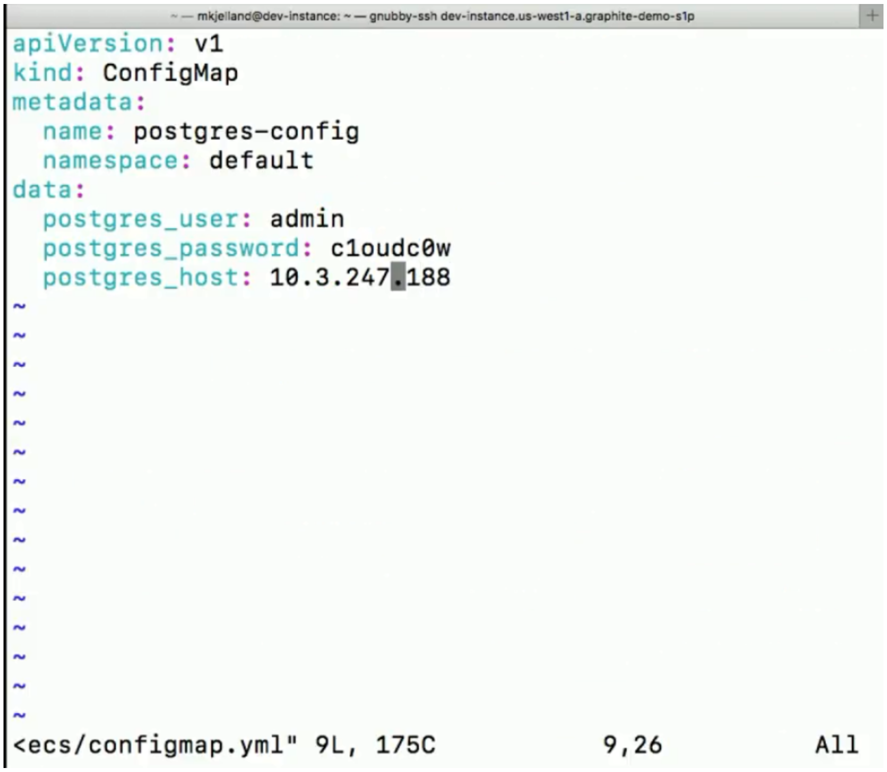
```
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: postgres
spec:
  template:
    metadata:
      labels:
        app: postgres
    spec:
      volumes:
        - name: postgres-storage
          persistentVolumeClaim:
            claimName: postgres-pv-claim
      containers:
        - image: postgres
          name: postgres
          env:
            - name: POSTGRES_USER
              valueFrom:
                configMapKeyRef:
                  name: postgres-config
                  key: postgres_user
<cs/postgres.yml" 48L, 1151C          17,7          Top
```

This deployment uses an already existing ***persistentVolumeClaim*** called ***postgres-pv-claim*** to get the volume to store its data in,

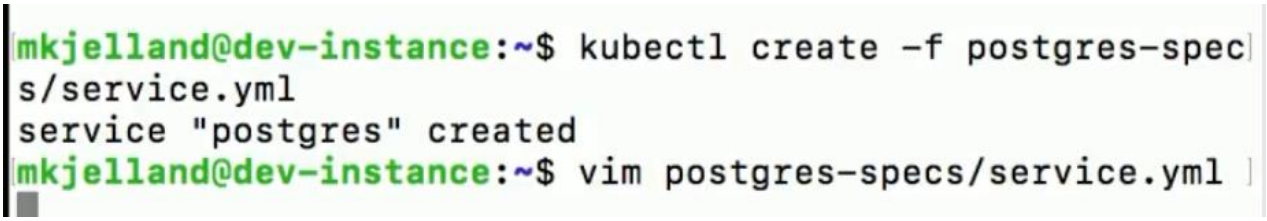

~ — mkjelland@dev-instance: ~ — gnubby-ssh dev-instance.us-west1-a.graphite-demo-s1p

```
            - name: POSTGRES_PASSWORD
              valueFrom:
                configMapKeyRef:
                  name: postgres-config
                  key: postgres_password
            - name: PGDATA
              value: /var/lib/postgresql/data/pgdata
          ports:
            - containerPort: 5432
              name: postgres
          volumeMounts:
            - name: postgres-storage
              mountPath: /var/lib/postgresql/data
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgres-pv-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 32Gi
                                             48,7          Bot
```

Then when we create the container we just mount that pod onto the ***mountPath*** directory on ***persistent disk***. Note that for PostGres, we have to tell it the ***PGDATA*** directory location so that it knows where to put the data. we also already created a ***ConfigMap*** that had the ***postgress_user*** and ***postgres_password*** in it. We then use that ***ConfigMap*** here to fetch the values to use from, we could have also used ***Secrets*** for keeping the user and password data.

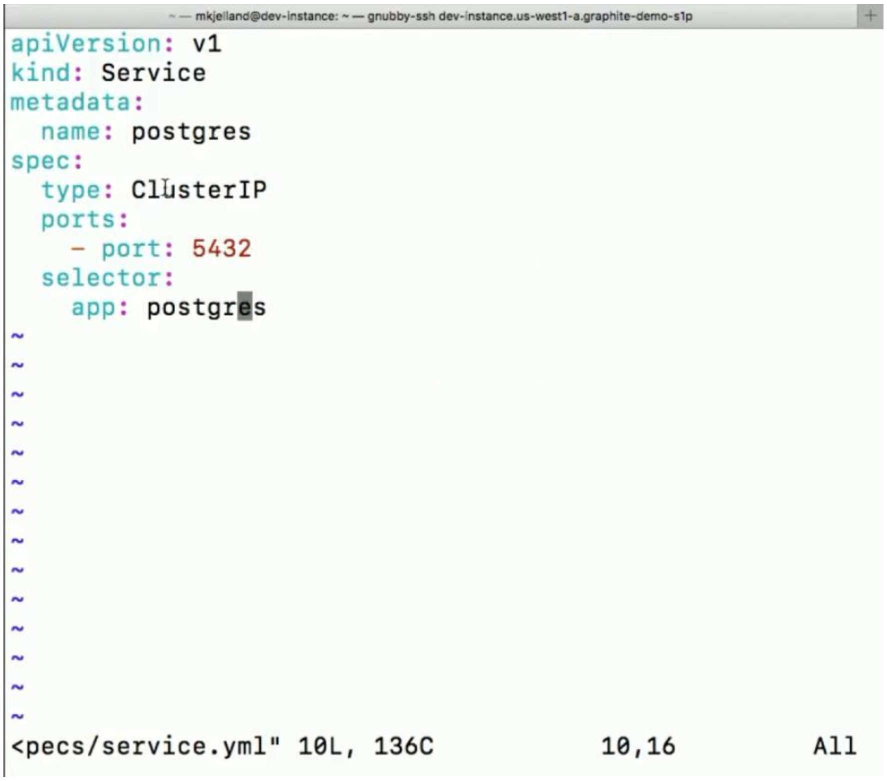


```
~ — mkjelland@dev-instance: ~ — gnubby-ssh dev-instance.us-west1-a.graphite-demo-s1p
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-config
  namespace: default
data:
  postgres_user: admin
  postgres_password: c1oudc0w
  postgres_host: 10.3.247.188
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
<ecs/configmap.yml" 9L, 175C            9,26          All
```

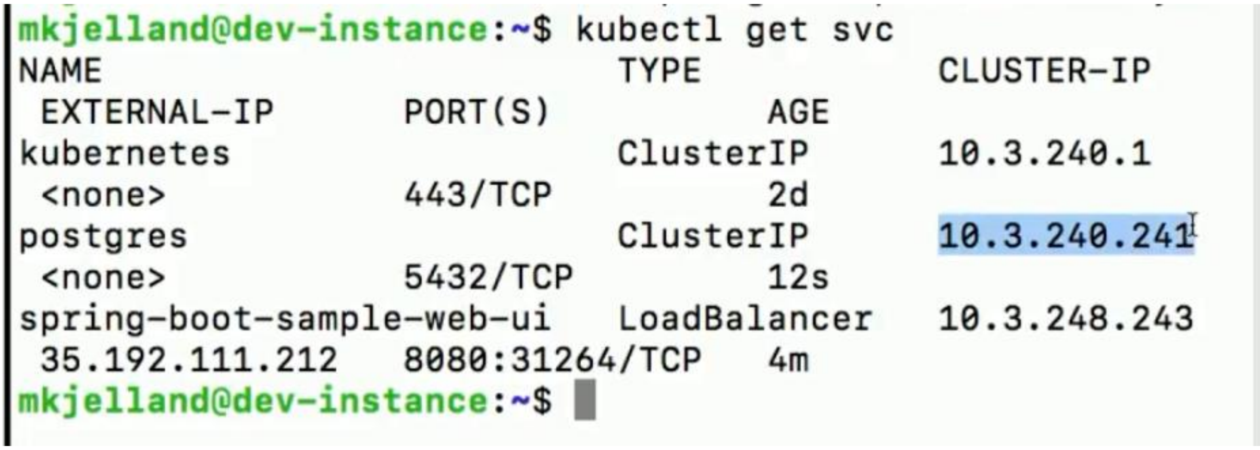


```
mkjelland@dev-instance:~$ kubectl create -f postgres-spec
s/service.yml
service "postgres" created
mkjelland@dev-instance:~$ vim postgres-specs/service.yml
```

We then create a service to be able to access the deployment using the ***$ kubectl create -f postgres-specs/service.yml*** command, this time we are using clusterIp because we don't need to access the service outside of the cluster.

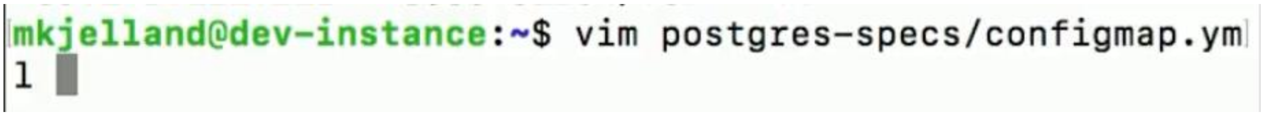


```
~ — mkjelland@dev-instance: ~ — gnubby-ssh dev-instance.us-west1-a.graphite-demo-s1p
apiVersion: v1
kind: Service
metadata:
  name: postgres
spec:
  type: ClusterIP
  ports:
    - port: 5432
  selector:
    app: postgres
~
~
~
~
~
~
~
~
~
~
~
~
~
~
<pecs/service.yml" 10L, 136C            10,16          All
```

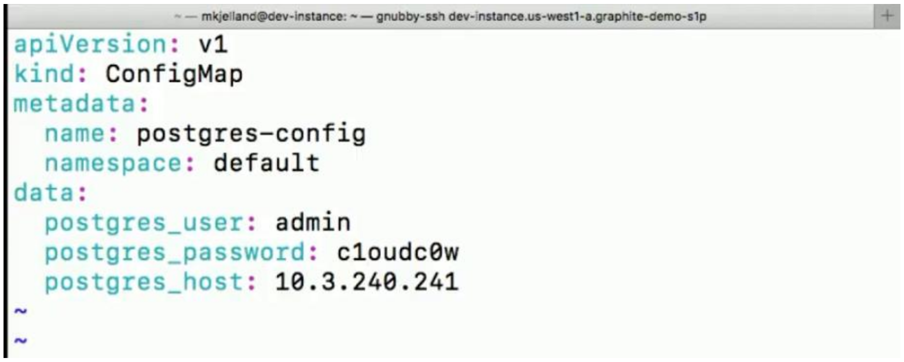


```
mkjelland@dev-instance:~$ kubectl get svc
NAME                        TYPE           CLUSTER-IP
 EXTERNAL-IP      PORT(S)          AGE
kubernetes                  ClusterIP      10.3.240.1
 <none>           443/TCP          2d
postgres                    ClusterIP      10.3.240.241
 <none>           5432/TCP         12s
spring-boot-sample-web-ui   LoadBalancer   10.3.248.243
 35.192.111.212   8080:31264/TCP   4m
mkjelland@dev-instance:~$
```

We use the ***$ kubectl get svc*** command to retrieve the clusterIp of the new service as above

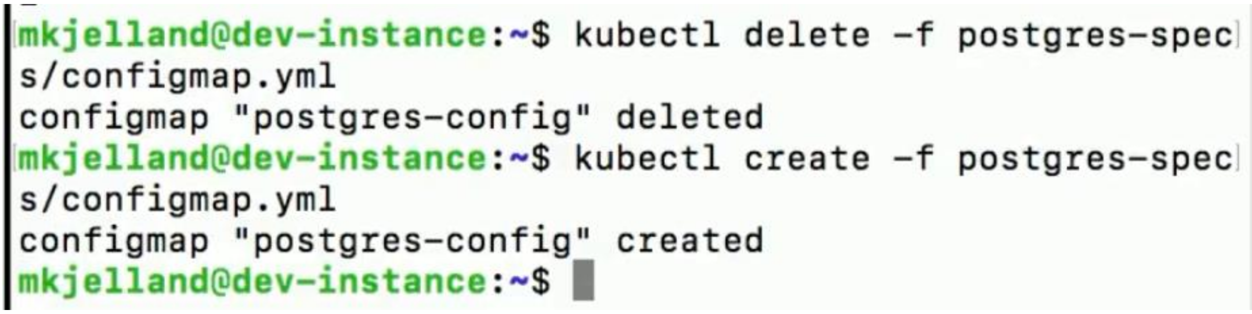


```
mkjelland@dev-instance:~$ vim postgres-specs/configmap.ym
l
```


~ — mkjelland@dev-instance: ~ — gnubby-ssh dev-instance.us-west1-a.graphite-demo-s1p

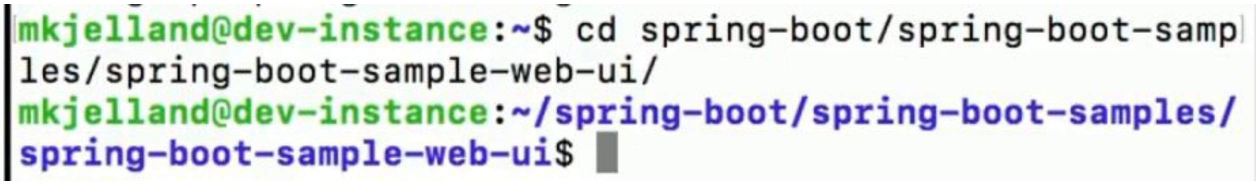
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-config
  namespace: default
data:
  postgres_user: admin
  postgres_password: c1oudc0w
  postgres_host: 10.3.240.241
~
~
```

We now need to update the ***ConfigMap*** with the new ***clusterIp*** value in the ***postgres_host*** filed above, ***this is because the web app pod we are going to be deploying next is going to need to know the clusterIp that it can reach the PostGres service on***.



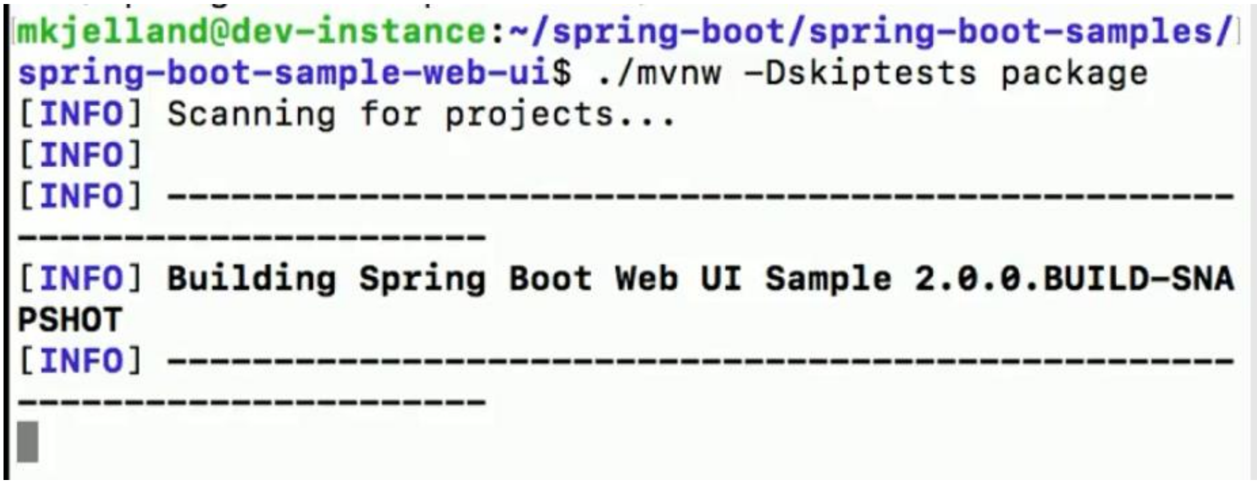
```
mkjelland@dev-instance:~$ kubectl delete -f postgres-spec
s/configmap.yml
configmap "postgres-config" deleted
mkjelland@dev-instance:~$ kubectl create -f postgres-spec
s/configmap.yml
configmap "postgres-config" created
mkjelland@dev-instance:~$
```

We then delete the current PostGress ConfigMap using the ***$ kubectl delete -f postgres-specs/configmap.yml*** command and re-create a new ConfigMap with the current settings using the ***$ kubectl create -f postgres-specs/configmap.yml*** command.



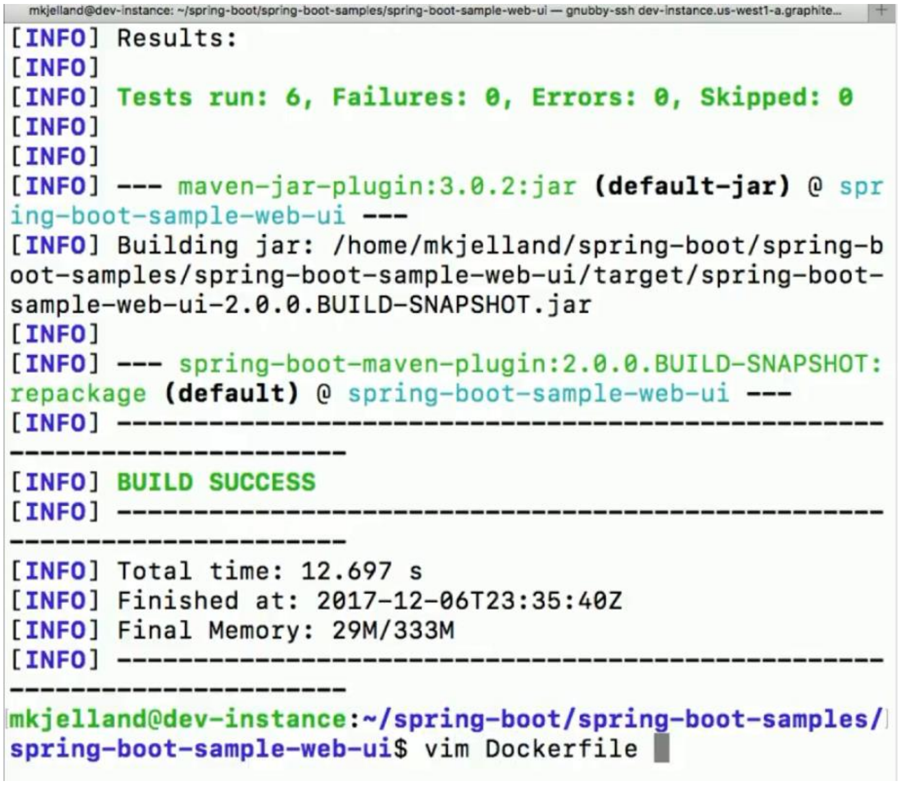
```
mkjelland@dev-instance:~$ cd spring-boot/spring-boot-samp
les/spring-boot-sample-web-ui/
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$
```

We change directory to the web app files directory, we have modified the files to now use the PostGress pod instance to persist data.

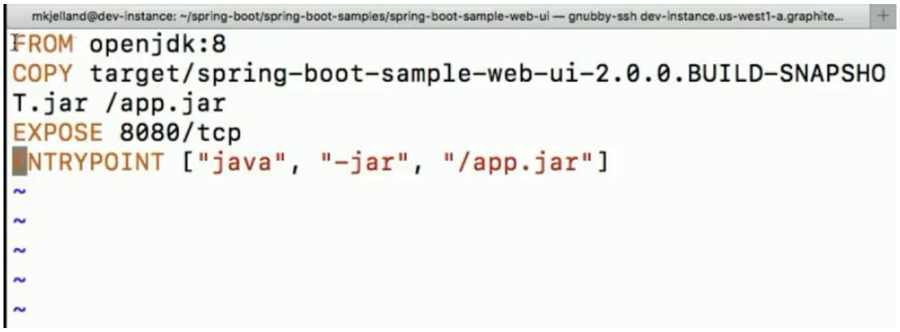


```
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ ./mvnw -Dskiptests package
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Spring Boot Web UI Sample 2.0.0.BUILD-SNAPSHOT
[INFO] ------------------------------------------------------------------------
```

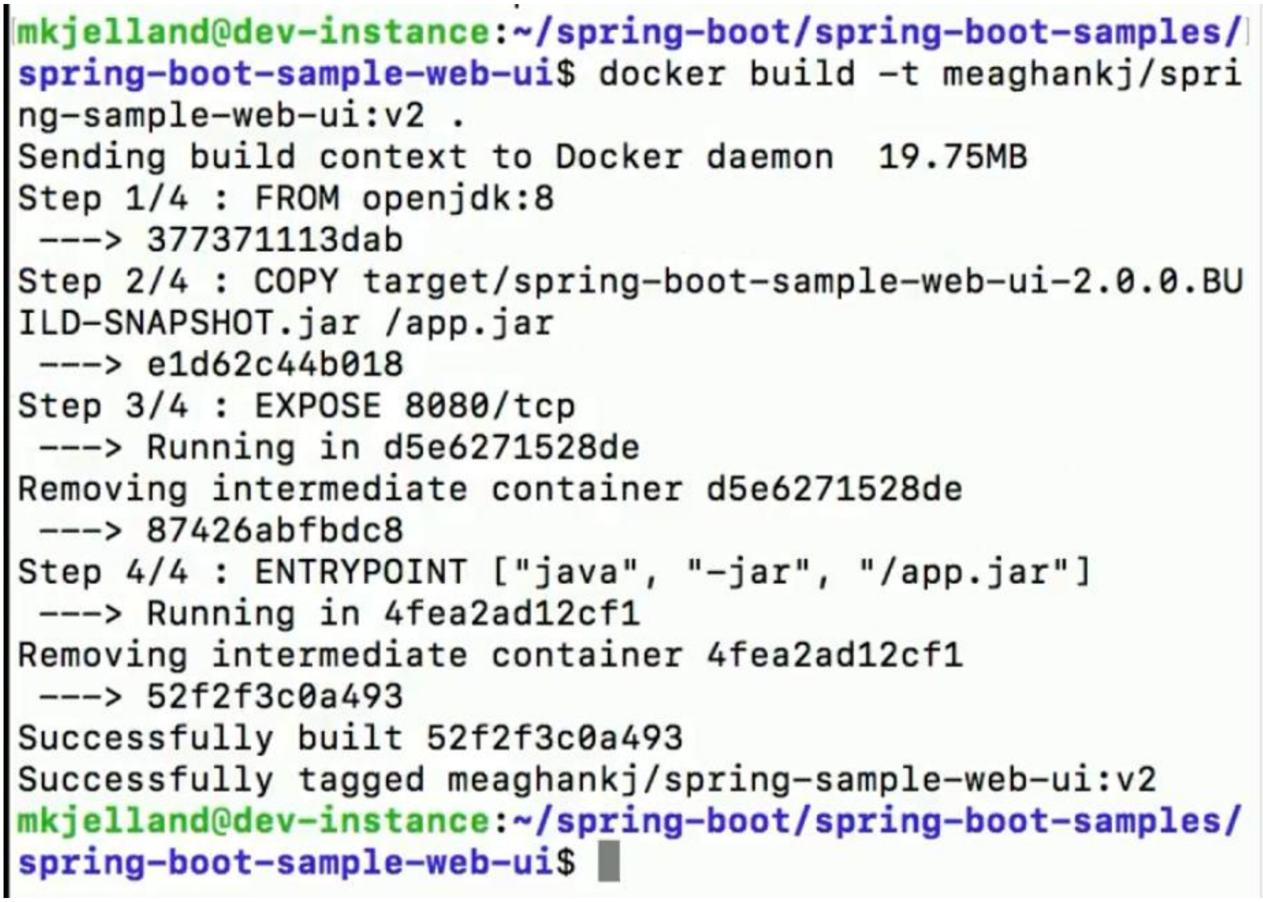
First, we build the web app using the ***$ ./mvnw -Dskiptests package*** command so that we get the JAR file, then we build the Docker image from the ***JAR file***, then push the ***Docker image*** to the ***Docker registry***, then we can reference that image in our deployment file.


mkjelland@dev-instance: ~/spring-boot/spring-boot-samples/spring-boot-sample-web-ui — gnubby-ssh dev-instance.us-west1-a.graphite...

```
[INFO] Results:
[INFO]
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ spring-boot-sample-web-ui ---
[INFO] Building jar: /home/mkjelland/spring-boot/spring-boot-samples/spring-boot-sample-web-ui/target/spring-boot-sample-web-ui-2.0.0.BUILD-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.0.0.BUILD-SNAPSHOT:repackage (default) @ spring-boot-sample-web-ui ---
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 12.697 s
[INFO] Finished at: 2017-12-06T23:35:40Z
[INFO] Final Memory: 29M/333M
[INFO] ------------------------------------------------------------------------
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ vim Dockerfile
```

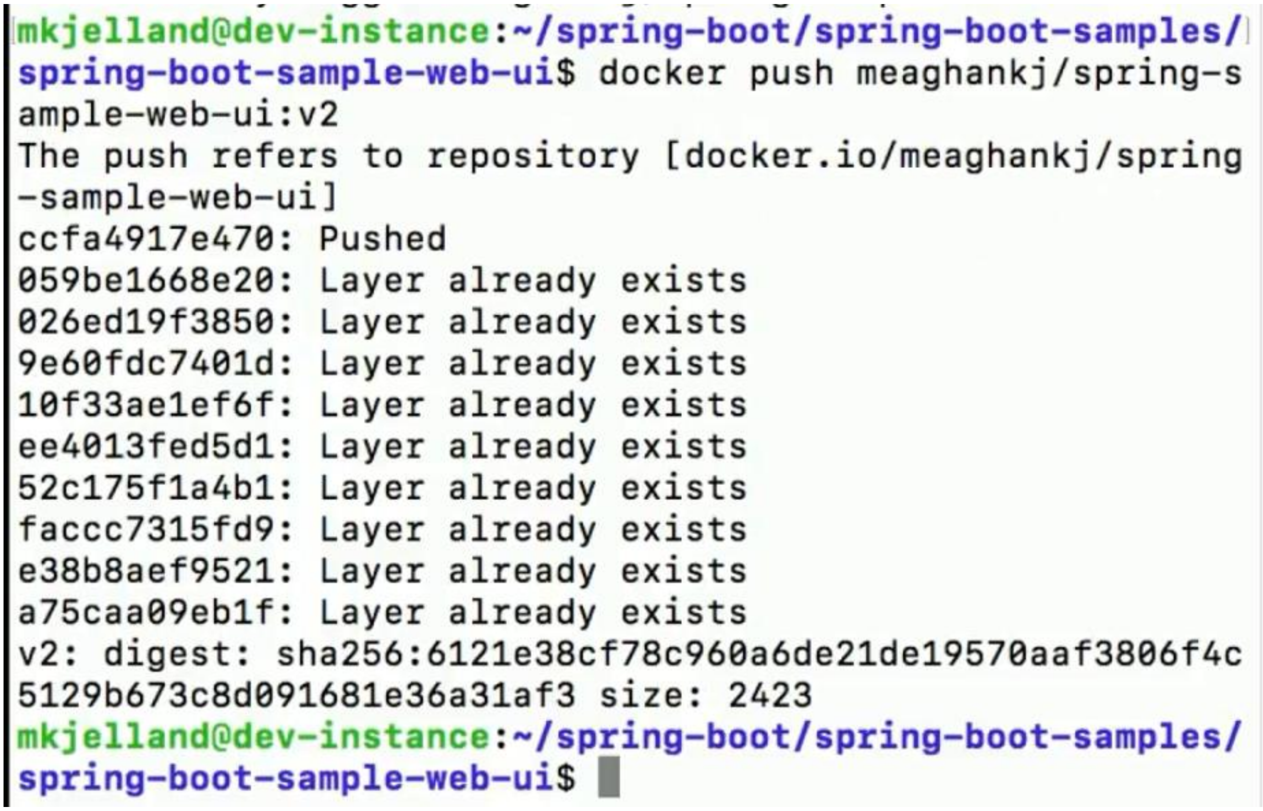

mkjelland@dev-instance: ~/spring-boot/spring-boot-samples/spring-boot-sample-web-ui — gnubby-ssh dev-instance.us-west1-a.graphite...

```
FROM openjdk:8
COPY target/spring-boot-sample-web-ui-2.0.0.BUILD-SNAPSHOT.jar /app.jar
EXPOSE 8080/tcp
ENTRYPOINT ["java", "-jar", "/app.jar"]
~
~
~
~
~
```



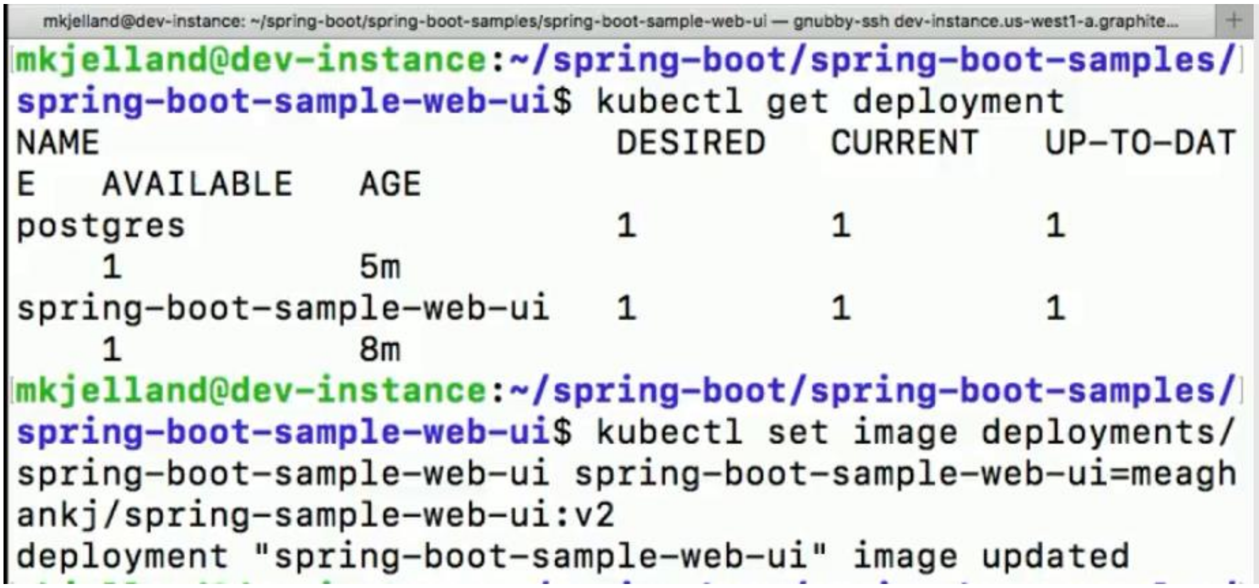
```
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ docker build -t meaghankj/spri
ng-sample-web-ui:v2 .
Sending build context to Docker daemon  19.75MB
Step 1/4 : FROM openjdk:8
 ---> 377371113dab
Step 2/4 : COPY target/spring-boot-sample-web-ui-2.0.0.BU
ILD-SNAPSHOT.jar /app.jar
 ---> e1d62c44b018
Step 3/4 : EXPOSE 8080/tcp
 ---> Running in d5e6271528de
Removing intermediate container d5e6271528de
 ---> 87426abfbdc8
Step 4/4 : ENTRYPOINT ["java", "-jar", "/app.jar"]
 ---> Running in 4fea2ad12cf1
Removing intermediate container 4fea2ad12cf1
 ---> 52f2f3c0a493
Successfully built 52f2f3c0a493
Successfully tagged meaghankj/spring-sample-web-ui:v2
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$
```

We then build the Docker image from the JAR file by running the Dockerfile using the ***$ docker build -t meaghankj/spring-sample-web-ui:v2*** command



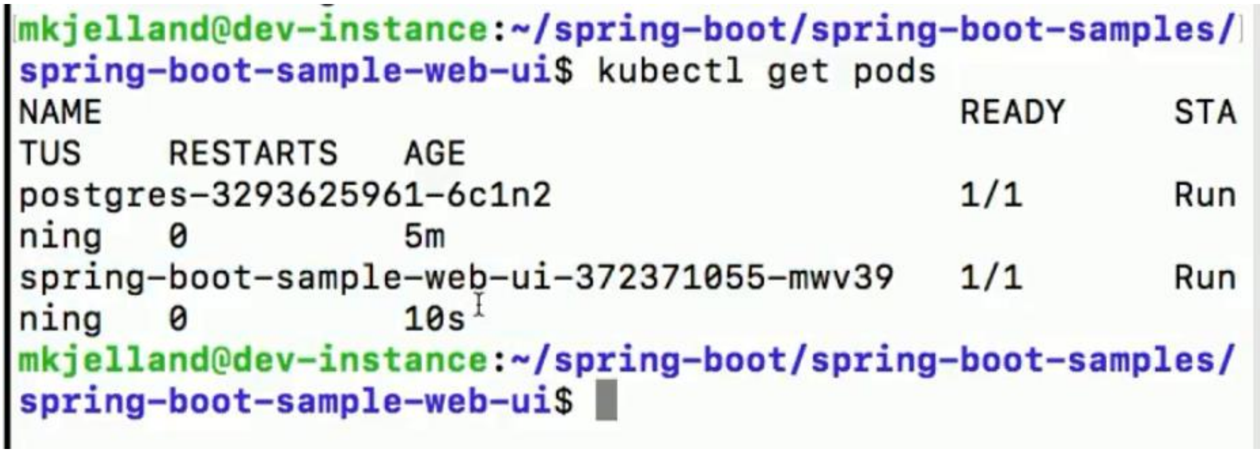
```
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ docker push meaghankj/spring-s
ample-web-ui:v2
The push refers to repository [docker.io/meaghankj/spring
-sample-web-ui]
ccfa4917e470: Pushed
059be1668e20: Layer already exists
026ed19f3850: Layer already exists
9e60fdc7401d: Layer already exists
10f33ae1ef6f: Layer already exists
ee4013fed5d1: Layer already exists
52c175f1a4b1: Layer already exists
faccc7315fd9: Layer already exists
e38b8aef9521: Layer already exists
a75caa09eb1f: Layer already exists
v2: digest: sha256:6121e38cf78c960a6de21de19570aaf3806f4c
5129b673c8d091681e36a31af3 size: 2423
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$
```

We then push the new Docker image to the Docker registry using the ***$ docker push meaghankj/spring-sample-web-ui:v2*** command


mkjelland@dev-instance: ~/spring-boot/spring-boot-samples/spring-boot-sample-web-ui — gnubby-ssh dev-instance.us-west1-a.graphite...

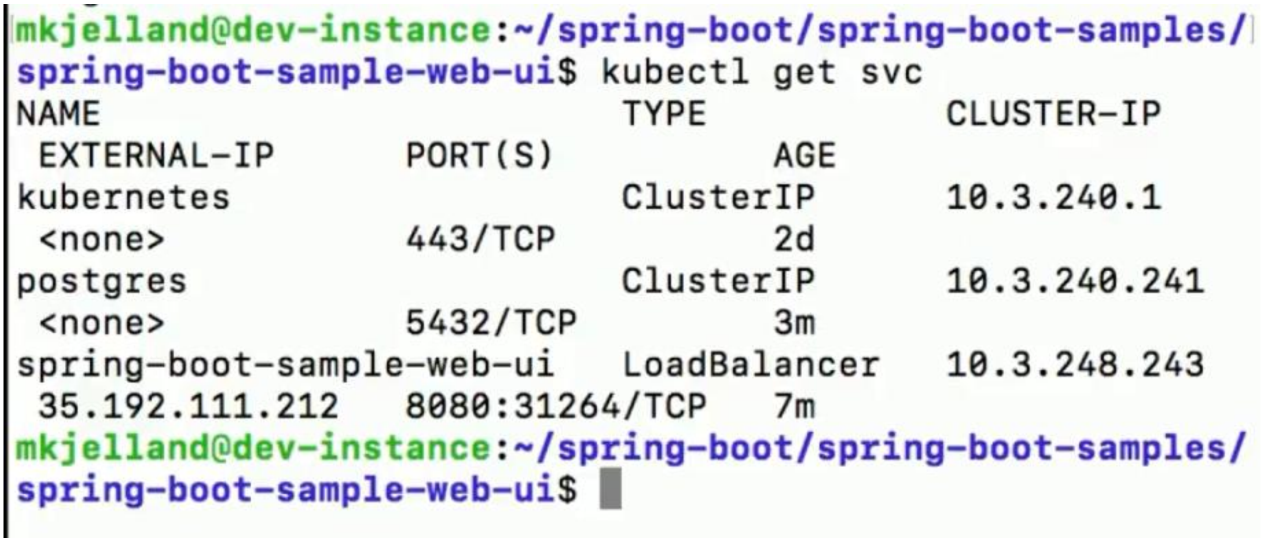
```
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ kubectl get deployment
NAME                        DESIRED   CURRENT   UP-TO-DAT
E   AVAILABLE   AGE
postgres                    1         1         1
    1           5m
spring-boot-sample-web-ui   1         1         1
    1           8m
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ kubectl set image deployments/
spring-boot-sample-web-ui spring-boot-sample-web-ui=meagh
ankj/spring-sample-web-ui:v2
deployment "spring-boot-sample-web-ui" image updated
```

We then update the current web deployment file with the new Docker image using the ***$ kubectl set image deployments/spring-boot-sample-web-ui spring-boot-sample-web-ui=meaghankj/spring-sample-web-ui:v2*** command,



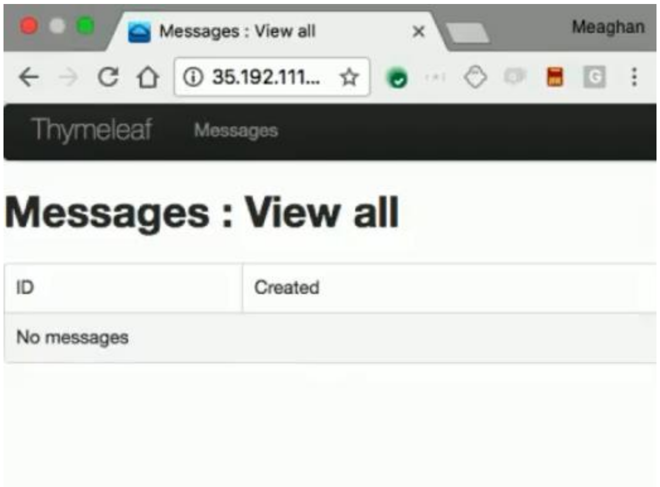
```
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ kubectl get pods
NAME                                        READY     STA
TUS     RESTARTS   AGE
postgres-3293625961-6c1n2                   1/1       Run
ning   0           5m
spring-boot-sample-web-ui-372371055-mwv39   1/1       Run
ning   0           10s
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ 
```

We use the ***$ kubectl get pods*** command to see if the new pod is created successfully and running



```
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ kubectl get svc
NAME                        TYPE           CLUSTER-IP
 EXTERNAL-IP      PORT(S)          AGE
kubernetes                  ClusterIP      10.3.240.1
 <none>           443/TCP          2d
postgres                    ClusterIP      10.3.240.241
 <none>           5432/TCP         3m
spring-boot-sample-web-ui   LoadBalancer   10.3.248.243
 35.192.111.212   8080:31264/TCP   7m
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ 
```

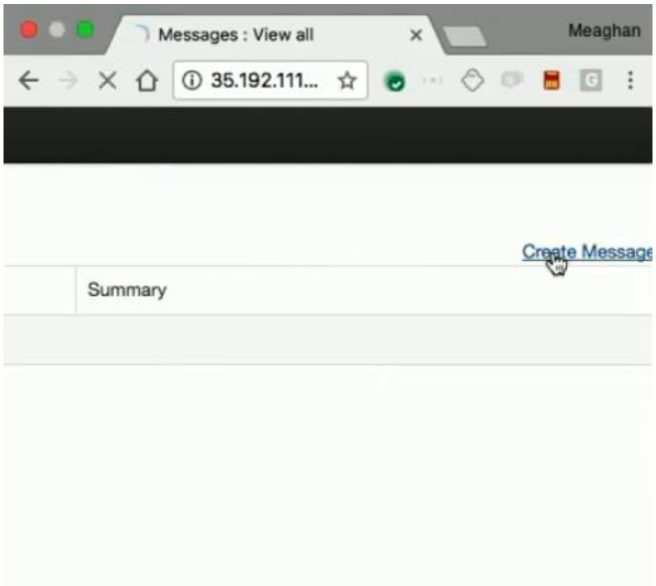
Then we can use the ***$ kubectl get svc*** command to check the service endpoint

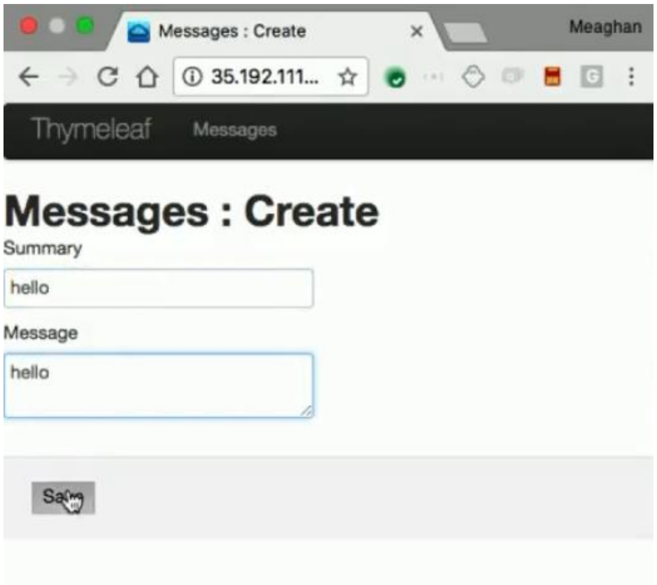


Messages : View all

ID	Created
No messages	

So, we have the exact same web app running but now using the PostGres pod to persist data





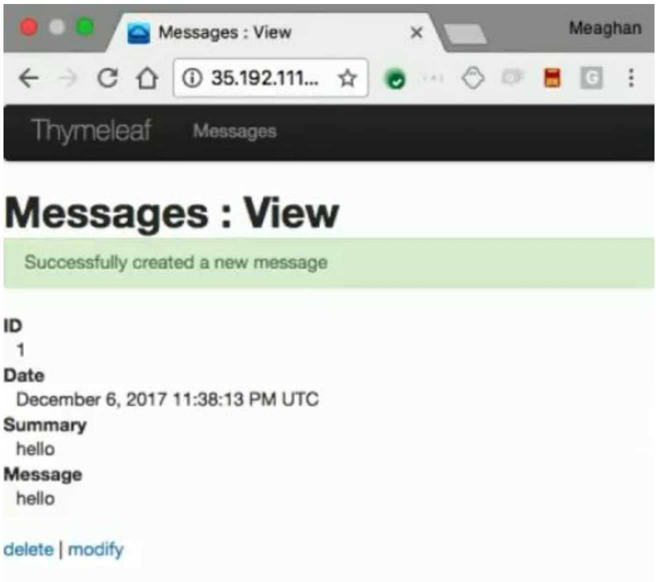
Messages : Create

Summary

hello

Message

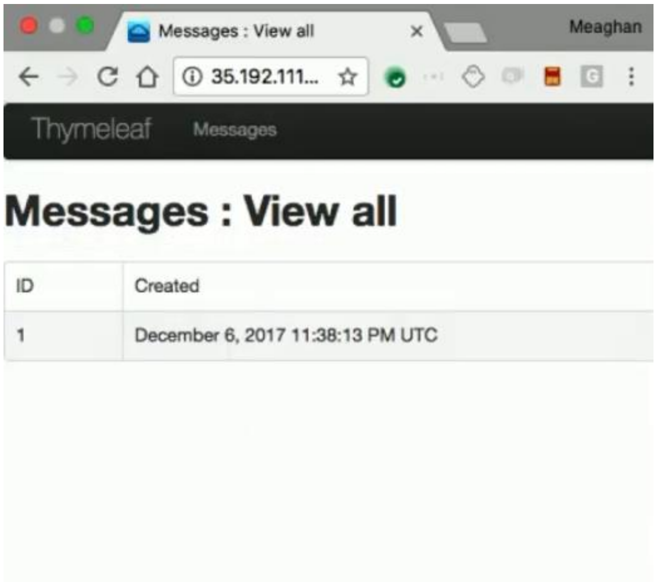
hello



Messages : View

Successfully created a new message

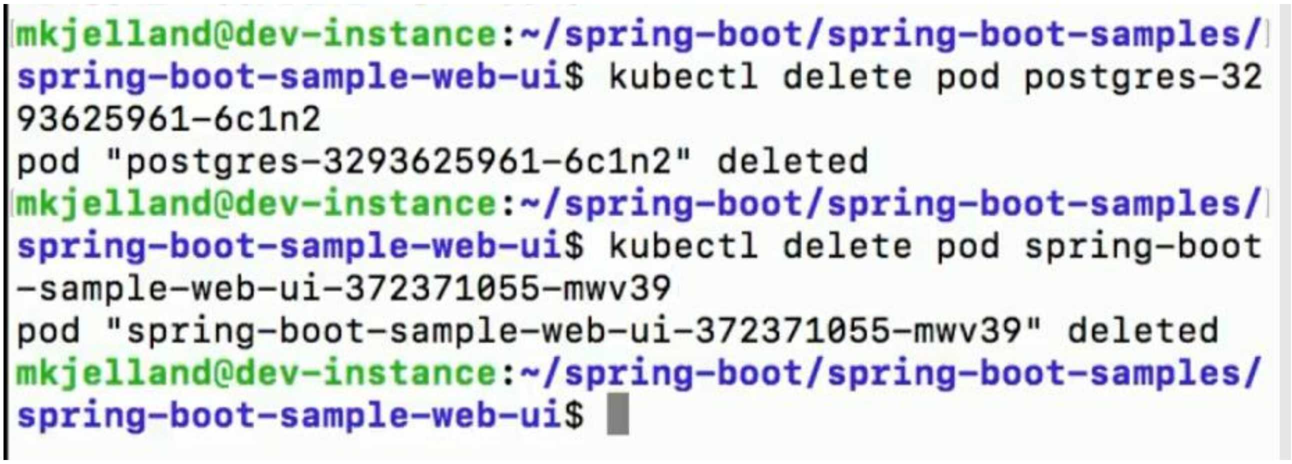
ID
1
Date
December 6, 2017 11:38:13 PM UTC
Summary
hello
Message
hello



Messages : View all

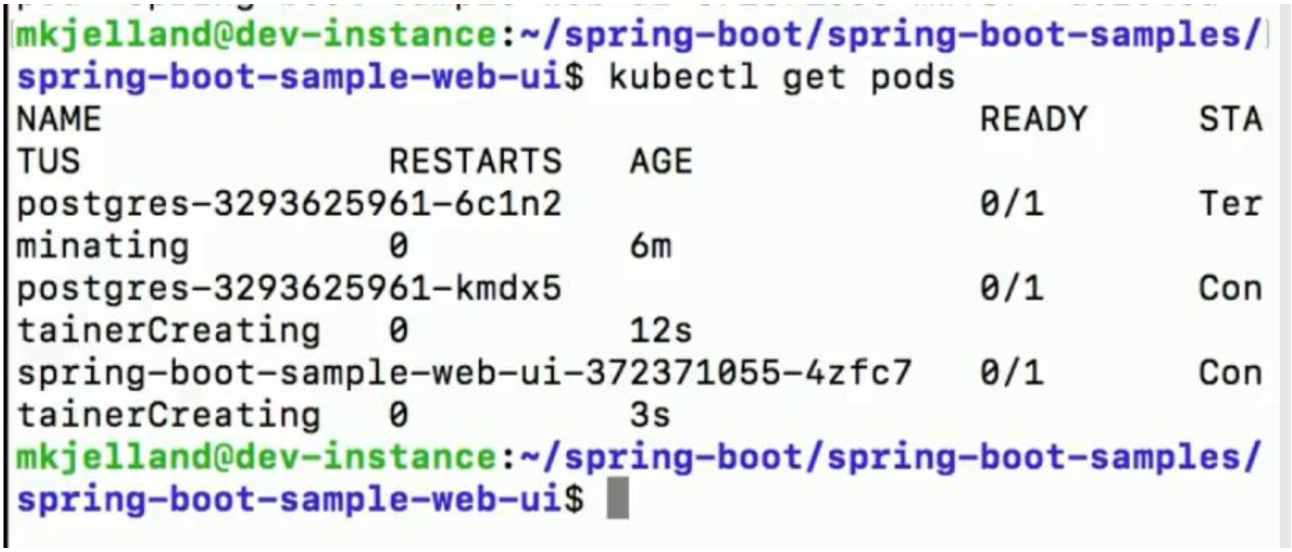
ID	Created
1	December 6, 2017 11:38:13 PM UTC

We can see that the message has been created.



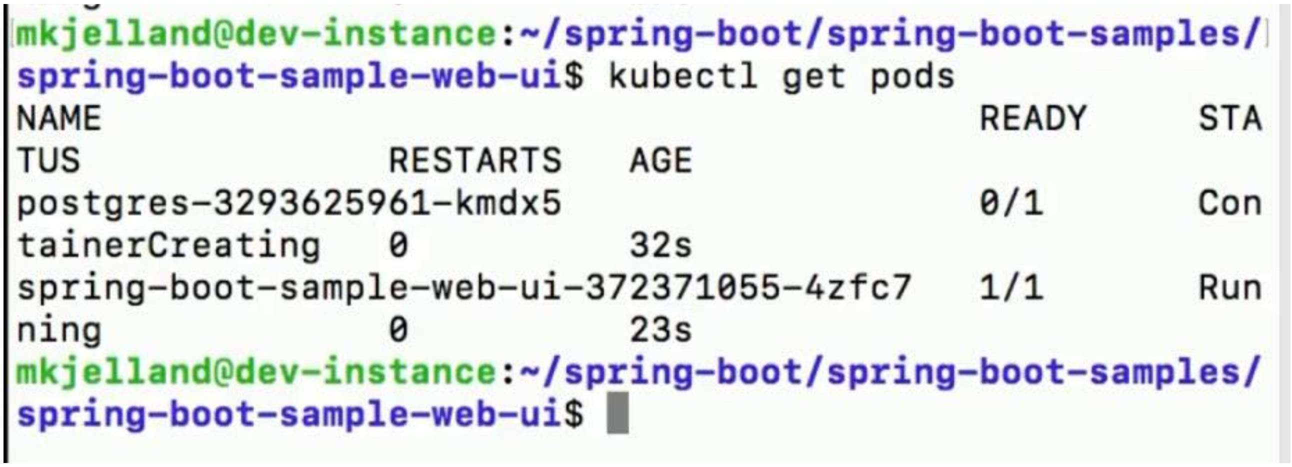
```
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ kubectl delete pod postgres-32
93625961-6c1n2
pod "postgres-3293625961-6c1n2" deleted
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ kubectl delete pod spring-boot
-sample-web-ui-372371055-mwv39
pod "spring-boot-sample-web-ui-372371055-mwv39" deleted
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ 
```

We can now delete both the PostGress pod using ***$ kubectl delete pod postgres-3297497487*** and the web app pod using ***$ kubectl delete pod spring-boot-sample-web-ui-378466546*** to check if the message was indeed persisted to disk



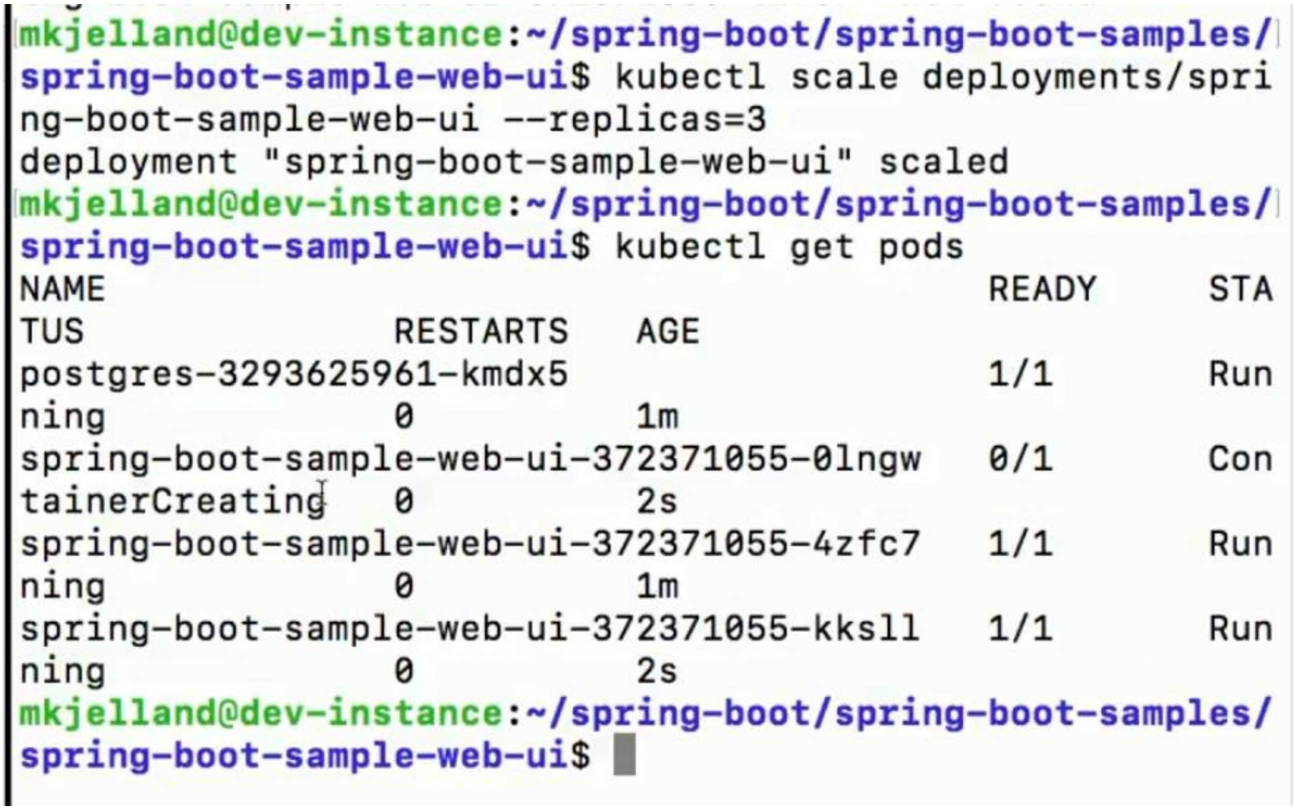
```
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ kubectl get pods
NAME                                         READY     STA
TUS                 RESTARTS   AGE
postgres-3293625961-6c1n2                    0/1       Ter
minating            0          6m
postgres-3293625961-kmdx5                    0/1       Con
tainerCreating      0          12s
spring-boot-sample-web-ui-372371055-4zfc7    0/1       Con
tainerCreating      0          3s
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ 
```

We then use the ***$ kubectl get pods*** command to see that the deleted pods are being replaced with new ones by K8s.



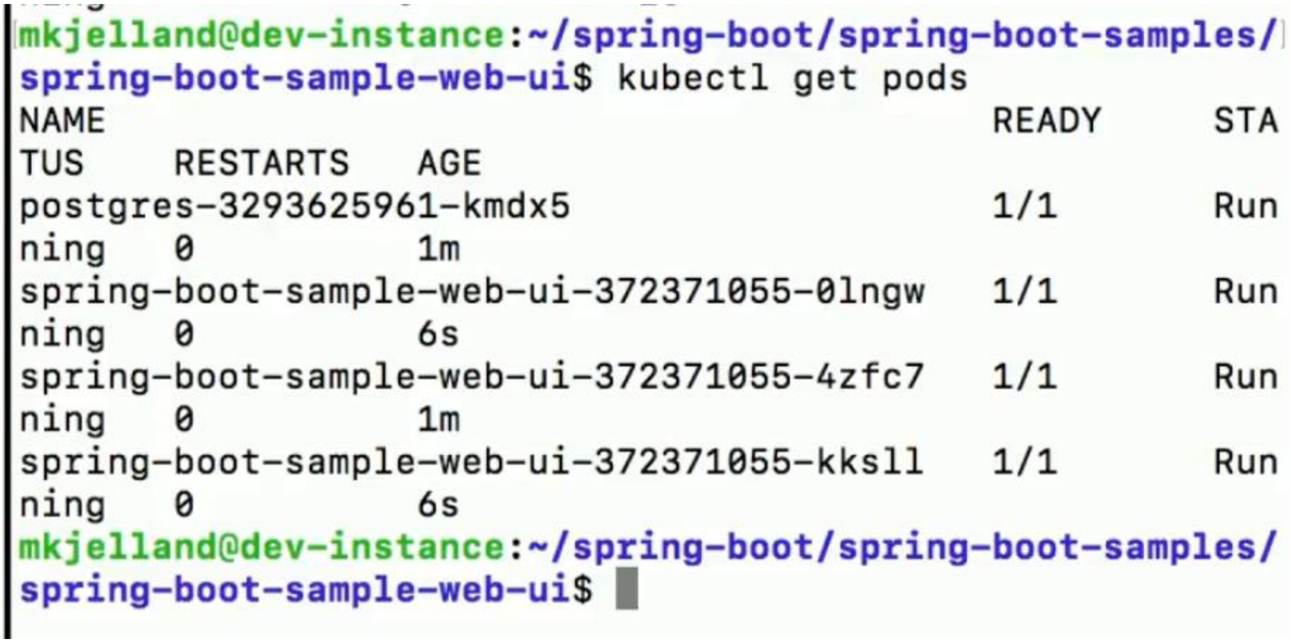
```
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ kubectl get pods
NAME                                         READY     STA
TUS                 RESTARTS   AGE
postgres-3293625961-kmdx5                    0/1       Con
tainerCreating      0          32s
spring-boot-sample-web-ui-372371055-4zfc7    1/1       Run
ning                0          23s
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ 
```

For the new PostGress pod, K8s is going to have to unattached the disk from the deleted pod and reattach the disk to the newly created pod instance. This is why creating and starting up the PostGress pod takes longer time to finish.



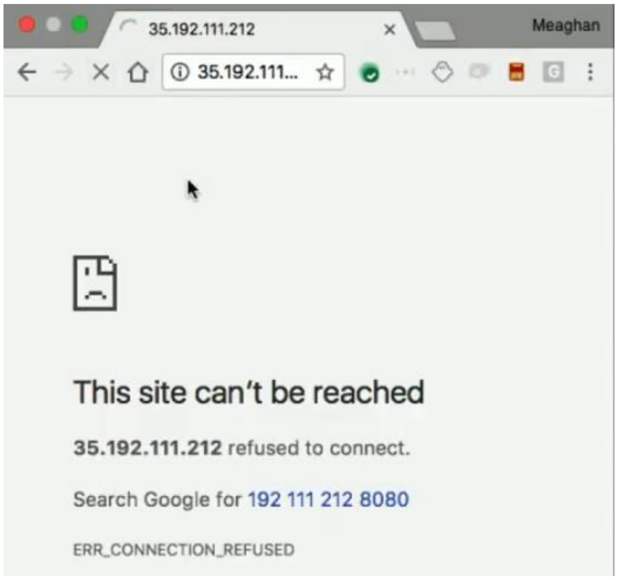
```
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ kubectl scale deployments/spri
ng-boot-sample-web-ui --replicas=3
deployment "spring-boot-sample-web-ui" scaled
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ kubectl get pods
NAME                                         READY     STA
TUS                RESTARTS   AGE
postgres-3293625961-kmdx5                    1/1       Run
ning               0          1m
spring-boot-sample-web-ui-372371055-0lngw    0/1       Con
tainerCreating     0          2s
spring-boot-sample-web-ui-372371055-4zfc7    1/1       Run
ning               0          1m
spring-boot-sample-web-ui-372371055-kksll    1/1       Run
ning               0          2s
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$
```

We can also scale up the web app pods to 3 using the ***$ kubectl scale deployments/spring-boot-sample-web-ui - -replicas=3*** command as above



```
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$ kubectl get pods
NAME                                         READY     STA
TUS     RESTARTS   AGE
postgres-3293625961-kmdx5                    1/1       Run
ning    0          1m
spring-boot-sample-web-ui-372371055-0lngw    1/1       Run
ning    0          6s
spring-boot-sample-web-ui-372371055-4zfc7    1/1       Run
ning    0          1m
spring-boot-sample-web-ui-372371055-kksll    1/1       Run
ning    0          6s
mkjelland@dev-instance:~/spring-boot/spring-boot-samples/
spring-boot-sample-web-ui$
```

Now we have all 3 web app pods running



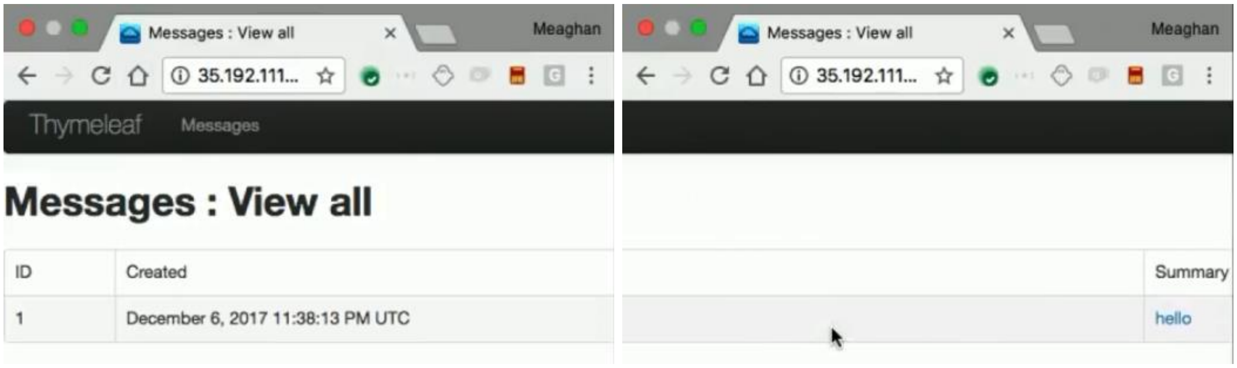
This site can't be reached

35.192.111.212 refused to connect.

Search Google for 192 111 212 8080

ERR_CONNECTION_REFUSED

We now refresh the web page to see if our data was indeed persisted to disk the previous time before deletion



Messages : View all

ID	Created	Summary
1	December 6, 2017 11:38:13 PM UTC	hello

Yes, we can see our data show up. This means that we are now persisting our data to disk using a web app, a PostGress database, and a persistent volume/disk.

Thank you!

SpringOne Platform by Pivotal

@s1p #springone