#047 Kubernetes - voting app (example)

Introduction

this is part 47 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 files

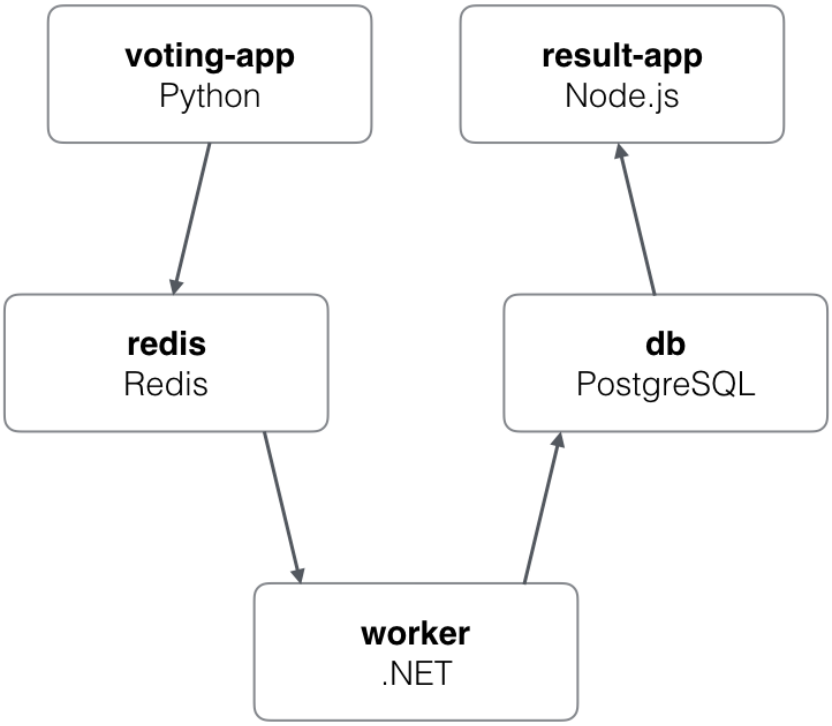
The original files can be found here
{% github dockersamples/example-voting-app no-readme %}
the files for this lab is same of the voting app from previous part (in Docker).
the files we need are located inside the k8s-specifications folder.
But since we don't know what is namespace so I edit the files and remove namespaces from them.

The edited files can be found in my github repo here
{% github OmarElKhatibCS/DevOpsJourney no-readme %}
if you already have it just pull , if not clone it.
the source code usually hold the same chapter number we are in 047 so app_047 is what we are looking for.

Understand the app

This part will be a warm up to all what we take so far by using the example of voting app.

first let's take a look at the diagram ,



the vote and result are front-end so we need to access them from outside the cluster , so we need a NodPort service for each of them in order to access it from outside the cluster.
for the redis and postgresDB we need an clusterIP service because it's still inside the cluster.
if we look at the source code of the worker we can see that he connect to each database using their services and update the votes from the redis to postgres. so technically no one is accessing the worker so we don't need a service for him.
let' take a look at the configuration files first.
my objective from reviewing the files is to remember all what we talk about.

```yaml
# db-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: db
  name: db
spec:
  replicas: 1
  selector:
    matchLabels:
      app: db
  template:
    metadata:
      labels:
        app: db
    spec:
      containers:
```

```
      - image: postgres:9.4
        name: postgres
        env:
        - name: POSTGRES_USER
          value: postgres
        - name: POSTGRES_PASSWORD
          value: postgres
        ports:
        - containerPort: 5432
          name: postgres
        volumeMounts:
        - mountPath: /var/lib/postgresql/data
          name: db-data
      volumes:
      - name: db-data
        emptyDir: {}
```

this is an deployment kind is Deployment so it can be scaled up or down , metadata are descriptive data about the deployment it self , namespace is the topic of the next part :) , spec will take the specifications about the deployment we need 1 replica of this pod selector we will use it to be able to access it , template will take the configuration of the pod without apiVersion and kind .
this is for deployment same for the other deployments.
now let's take a look at services.

```
# db-service.yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: db
  name: db
spec:
  type: ClusterIP
  ports:
  - name: "db-service"
    port: 5432
    targetPort: 5432
  selector:
    app: db
```

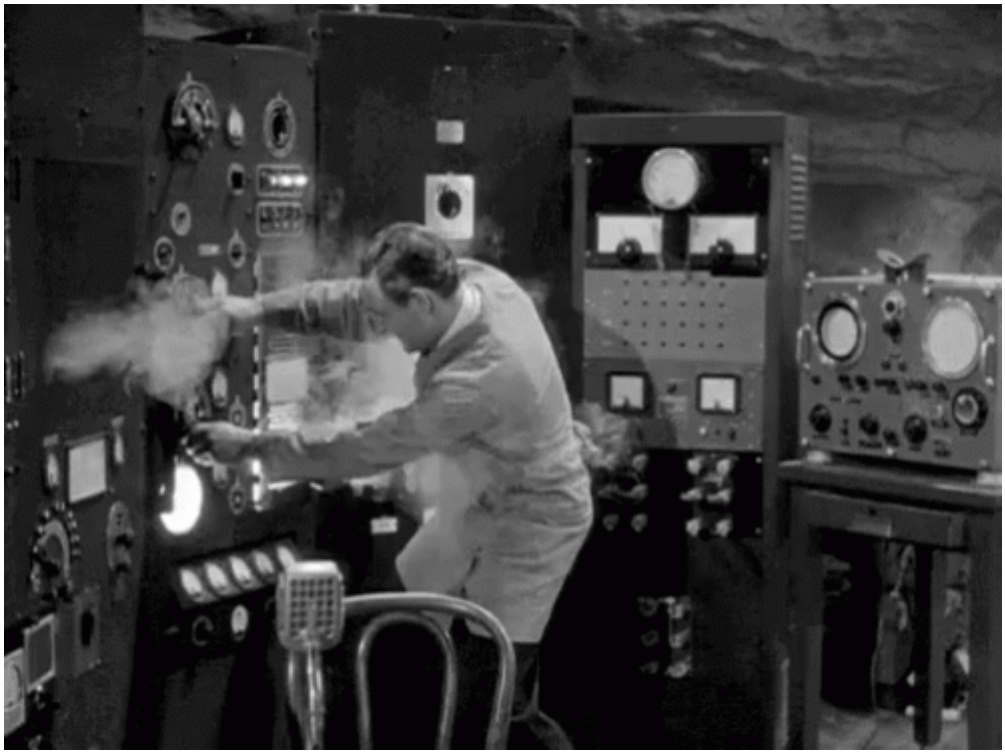
it will take type of ClusterIP(only allow access from inside cluster) and kind Service , and port of it and port of the target which in this case is the port of the postgresDB.

```
# result-service.yml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: result
  name: result
spec:
```

```
  type: NodePort
  ports:
  - name: "result-service"
    port: 5001
    targetPort: 80
    nodePort: 31001
  selector:
    app: result
```

this is a NodePort service so it will take as an extra the nodePort which is the port of the entire Node.

Lab



now let's create all of those yml files.
since they are all located in single folder we can pass the folder name and he will create them all.

```
✔ ~/Documents/DevOpsJourney [master|…1]
02:32 $ kubectl create -f app_047/
deployment.apps/db created
service/db created
deployment.apps/redis created
service/redis created
deployment.apps/result created
service/result created
deployment.apps/vote created
service/vote created
deployment.apps/worker created
✔ ~/Documents/DevOpsJourney [master|…1]
02:32 $
```

```
kubectl create -f app_047
```

```
✔ ~/Documents/DevOpsJourney/app_047 [master|…1]
01:53 $ kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
db-6789fcc76c-86bpp                 1/1     Running             0          53m
myapp-deployment-v1-754d67d968-47rg9   1/1     Running             1          47h
myapp-deployment-v1-754d67d968-ctscp   1/1     Running             1          47h
myapp-deployment-v1-754d67d968-f9mb9   1/1     Running             1          47h
myapp-deployment-v1-754d67d968-fb9tb   1/1     Running             1          47h
myapp-deployment-v1-754d67d968-gvqw5   1/1     Running             1          47h
myapp-deployment-v1-754d67d968-v6k7s   1/1     Running             1          47h
myapp-deployment-v2-55fb8ccd95-5ch6x   1/1     Running             1          47h
myapp-deployment-v2-55fb8ccd95-5wxw8   1/1     Running             1          47h
myapp-deployment-v2-55fb8ccd95-6q4qq   1/1     Running             1          47h
myapp-deployment-v2-55fb8ccd95-lhhgk   1/1     Running             1          47h
myapp-deployment-v2-55fb8ccd95-s8zcl   1/1     Running             1          47h
myapp-deployment-v2-55fb8ccd95-tmzn8   1/1     Running             1          47h
redis-554668f9bf-tkmfw              1/1     Running             0          53m
result-79bf6bc748-424ls             1/1     Running             2          52m
vote-7478984bfb-mzcf9               1/1     Running             0          52m
worker-dd46d7584-645s5              0/1     ContainerCreating   0          52m
✔ ~/Documents/DevOpsJourney/app_047 [master|…1]
01:53 $
```

```
kubectl get pods
```

my last container is still onCreating for 52 minute because I have the best connection in the world.
if we go to the app now we can interact with it but it not gonna update on results page because the worker is not ready yet. let's take a look

```
✔ ~/Documents/DevOpsJourney/app_047 [master|…1]
01:58 $ minikube ip
192.168.99.100
```

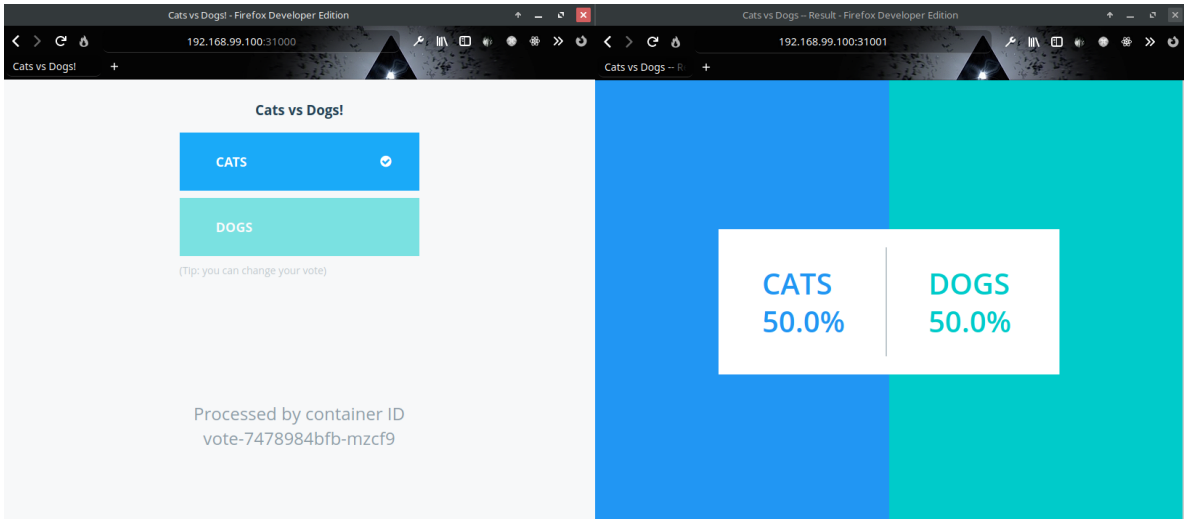
```
minikube ip
```

my ip is 192.168.99.100 and if we take a look at the configuration files

```
✔ ~/Documents/DevOpsJourney/app_047 [master|…1]
01:58 $ cat vote-service.yaml result-service.yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: vote
  name: vote
spec:
  type: NodePort
  ports:
  - name: "vote-service"
    port: 5000
    targetPort: 80
    nodePort: 31000
  selector:
    app: vote

apiVersion: v1
kind: Service
metadata:
  labels:
    app: result
  name: result
spec:
  type: NodePort
  ports:
  - name: "result-service"
    port: 5001
    targetPort: 80
    nodePort: 31001
  selector:
    app: result
✔ ~/Documents/DevOpsJourney/app_047 [master|…1]
02:00 $
```

NodePorts are 31000 and 31001 (I can get them from kubernetes also) , so we can access the app from 192.168.99.100:31000 and 192.168.99.100:31001



when I vote in the left the results are not updated yet because the worker is not ready yet.

Now when the worker finish building it's still not updating let's investigate it.

```
✔ ~/Documents/DevOpsJourney/app_047 [master|…1]
02:15 $ kubectl logs worker-dd46d7584-645s5 | head
Waiting for db
Waiting for db
Waiting for db
Waiting for db
Waiting for db
Waiting for db
Waiting for db
Waiting for db
Waiting for db
Waiting for db
✔ ~/Documents/DevOpsJourney/app_047 [master|…1]
02:15 $
```

```
kubectl logs db-6789fcc76c-hvrs6 | tail
```

(I delete the deployments and run them in investigation so names are changed)
db-6789fcc76c-hvrs6 is the name of db pod
tail is Linux command will get last few lines of the input.

we can see it's say password error and it's because the image is outdated , we can fix it by building new one and push it to docker this will take time and a lot of edits. So I am going to hard fix it.
Let's start

```
✔ ~/Documents/DevOpsJourney [master|…1]
02:43 $ kubectl exec -it db-6789fcc76c-hvrs6 -- bash
root@db-6789fcc76c-hvrs6:/#
```

```
kubectl exec db-6789fcc76c-hvrs6 -- bash
```

we are now in the root bash , let's install vim and edit the pg_hba.conf

```
02:43 $ kubectl exec -it db-6789fcc76c-hvrs6 -- bash
root@db-6789fcc76c-hvrs6:/# apt update && apt install vim
Ign:1 http://deb.debian.org/debian stretch InRelease
Get:2 http://deb.debian.org/debian stretch-updates InRelea
Get:3 http://apt.postgresql.org/pub/repos/apt stretch-pgdg
Get:4 http://security.debian.org/debian-security stretch/u
Get:5 http://deb.debian.org/debian stretch Release [118 kB
Get:6 http://apt.postgresql.org/pub/repos/apt stretch-pgdg
Get:7 http://security.debian.org/debian-security stretch/u
Get:8 http://apt.postgresql.org/pub/repos/apt stretch-pgdg
```

```
apt update && apt install vim
```

after we have the vim installed let's edit our file

```
root@db-6789fcc76c-hvrs6:/# vim /var/lib/postgresql/data/pg_hba.conf
```

```
vim /var/lib/postgresql/data/pg_hba.conf
```

```
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     postgres                                trust
#host    replication     postgres        127.0.0.1/32            trust
#host    replication     postgres        ::1/128                 trust

host all all all trust
:wq
```

go to the last line and change md5 to trust press i to enter edit mode in vim.
after editing it press ESC then :wq to save and quit.

```
                                                                /bin
root@db-6789fcc76c-hvrs6:/# su - postgres
```

now let's access the postgress shell using

```
su - postgres
```

```
root@db-6789fcc76c-hvrs6:/# su - postgres
postgres@db-6789fcc76c-hvrs6:~$ psql
psql (9.4.26)
Type "help" for help.

postgres=# SELECT pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 row)

postgres=# \q
```

let's reload our Database configurations using

```
psql
SELECT pg_reload_conf();
\q
```

and \q used to quit
then

```
postgres=# \q
postgres@db-6789fcc76c-hvrs6:~$ exit
logout
root@db-6789fcc76c-hvrs6:/# exit
exit
✔ ~/Documents/DevOpsJourney [master|✔]
02:53 $
```

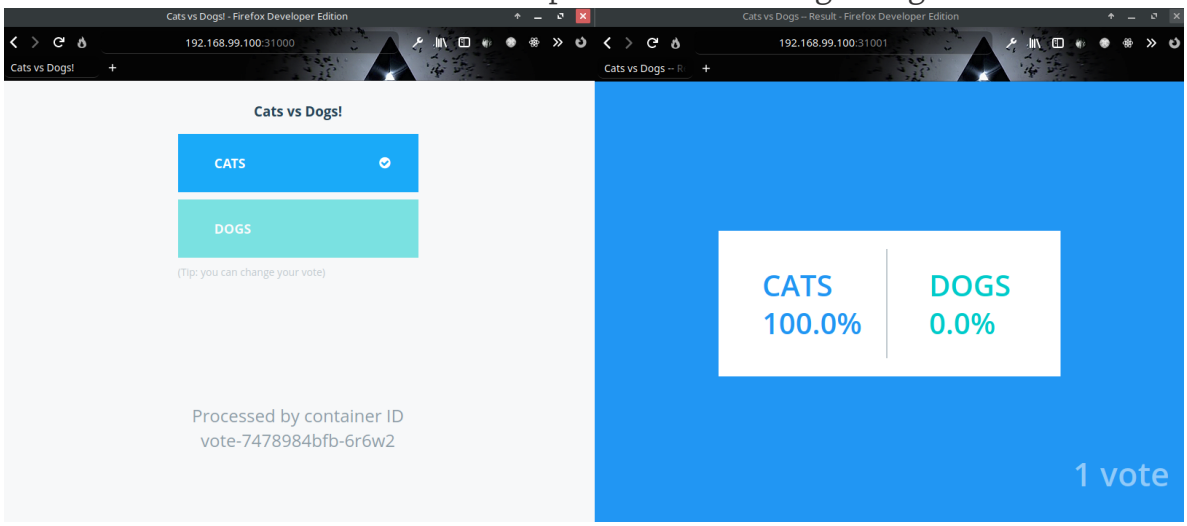
```
exit
exit
```

exit to exit first bash , and exit to exit second bash XD XD

```
DETAIL:  Connection matched pg_hba.conf line 95: "host all all all md5"
LOG:  received SIGHUP, reloading configuration files
ERROR:  relation "votes" does not exist at character 38
STATEMENT:  SELECT vote, COUNT(id) AS count FROM votes GROUP BY vote
ERROR:  duplicate key value violates unique constraint "votes_id_key"
DETAIL:  Key (id)=(841879a71f27710f) already exists.
STATEMENT:  INSERT INTO votes (id, vote) VALUES ($1, $2)
ERROR:  duplicate key value violates unique constraint "votes_id_key"
DETAIL:  Key (id)=(841879a71f27710f) already exists.
STATEMENT:  INSERT INTO votes (id, vote) VALUES ($1, $2)
ERROR:  duplicate key value violates unique constraint "votes_id_key"
DETAIL:  Key (id)=(841879a71f27710f) already exists.
STATEMENT:  INSERT INTO votes (id, vote) VALUES ($1, $2)
✔ ~/Documents/DevOpsJourney [master|✔]
02:55 $
```

```
kubectl logs db-6789fcc76c-hvrs6
```

if we scroll down we can see now it's updated and working let's go to the browser.



now it's updated congrats , we have full app deployed :D
We see also how we can troubleshoot our app , amazing isn't it ?


WE MAKE IT SO HARD
FOR OURSELVES