6. Deployments

27 February 2022 15:29

Deployments :-

→ Till now we created pods directly. normally/ideally we should not creat Pod directly but instead use the higher level to creat pods for example deployments.

Deployments

- Represent multiple replicas of a Pod
- Describe a desired state that Kubernetes needs to achieve
- Deployment Controller master component converges actual state to the desired state

We'll replace the individual pods with deployments that manage the pods for us.

Deployment is a template to creat pods.

↳ used to creat replicas ↳ are copies of pods & scaled.

Comparing pod & deployment

```
    protocol: TCP # default
    name: redis # optional when only 1 port
  selector:
    tier: data
  type: ClusterIP # default
---
apiVersion: v1
kind: Pod
metadata:
  name: data-tier
  labels:
    app: microservices
    tier: data
spec:
  containers:
  - name: redis
    image: redis:latest
    imagePullPolicy: IfNotPresent
    ports:
      - containerPort: 6379
```

```
admin@Jamess-MBP src %
```

```
    protocol: TCP # default
    name: redis # optional when only 1 port
  selector:
    tier: data
  type: ClusterIP # default
---
apiVersion: apps/v1 # apps API group
kind: Deployment
metadata:
  name: data-tier
  labels:
    app: microservices
    tier: data
spec:
  replicas: 1
  selector:
    matchLabels:
      tier: data
  template:
    metadata:
      labels:
        app: microservices
        tier: data
    spec: # Pod spec
      containers:
      - name: redis
        image: redis:latest
        imagePullPolicy: IfNotPresent
        ports:
          - containerPort: 6379
```

no of pods.

same

→ instead of v1 its apps/v1 and is deployment

→ spec → contains deployment specific settings and also a pod template with exactly same as pod

To group pods during deployments.

does not need a name for pod here because ks creat unique name for each pod.

```
admin@Jamess-MBP src % kubectl create -n deployments -f 5.2-data_tier.yaml -f 5.3-app_tier.yaml -f 5.4-support_tier.yaml
```

↳ using multiple files at a time.

↳ multiple pods at a time.

```
admin@Jamess-MBP src % kubectl get -n deployments deployments
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
app-tier       1/1     1            1           76s
data-tier      1/1     1            1           76s
support-tier   1/1     1            1           76s
admin@Jamess-MBP src %
```

```
admin@Jamess-MBP src % kubectl -n deployments get pods
NAME                           READY   STATUS    RESTARTS   AGE
app-tier-74cd4c68c9-z99h4      1/1     Running   0          2m24s
data-tier-8646dd765b-k6h98     1/1     Running   0          2m24s
support-tier-997bc57fb-c579t   2/2     Running   0          2m24s
admin@Jamess-MBP src %
```

↳ counts & pollers

we can use scale command to scale the replicas.

```
admin@Jamess-MBP src % kubectl scale -n deployments deployments support-tier --repl
cas=5
deployment.apps/support-tier scaled
admin@Jamess-MBP src %
```

```
admin@Jamess-MBP src % kubectl -n deployments get pods
NAME                           READY   STATUS    RESTARTS   AGE
app-tier-74cd4c68c9-z99h4      1/1     Running   0          4m49s
data-tier-8646dd765b-k6h98     1/1     Running   0          4m49s
support-tier-997bc57fb-42s98   2/2     Running   0          54s
support-tier-997bc57fb-c579t   2/2     Running   0          4m49s
support-tier-997bc57fb-l9lkc   2/2     Running   0          54s
support-tier-997bc57fb-lp56v   2/2     Running   0          54s
support-tier-997bc57fb-mtfrf   2/2     Running   0          54s
```

notice containers poller & counter remain same per pod because

replicas replicate pods not individual containers inside the pod.

Deployments ensure that specified number of replicated pods keep on running

(.)

Test this by deleting pods.

```
admin@Jamess-MBP src % kubectl delete -n deployments pods support-tier-997bc57fb-mtf
rf support-tier-997bc57fb-lp56v support-tier-997bc57fb-l9lkc
pod "support-tier-997bc57fb-mtfrf" deleted
pod "support-tier-997bc57fb-lp56v" deleted
pod "support-tier-997bc57fb-l9lkc" deleted
admin@Jamess-MBP src % watch -n 1 kubectl -n deployments get pods
```

watch every 1 sec update output.

↳ linux command

```
Every 1.0s: kubectl -n deployments get pods

NAME                           READY   STATUS        RESTARTS   AGE
app-tier-74cd4c68c9-h7hvf      1/1     Running       0          9m37s
data-tier-8646dd765b-wjmjd     1/1     Running       0          9m37s
support-tier-997bc57fb-7597k   2/2     Running       0          3m8s
support-tier-997bc57fb-79gsl   2/2     Running       0          7m59s
support-tier-997bc57fb-8wjnf   2/2     Terminating   0          7m59s
support-tier-997bc57fb-b2f9f   2/2     Terminating   0          3m8s
support-tier-997bc57fb-b8c4q   2/2     Running       0          19s
support-tier-997bc57fb-fggcx   2/2     Running       0          19s
support-tier-997bc57fb-hprh6   2/2     Terminating   0          3m8s
support-tier-997bc57fb-k8kfd   2/2     Running       0          19s
```

k8's automatically make sure replicas are running.

```
Every 1.0s: kubectl -n deployments get pods

NAME                           READY   STATUS    RESTARTS   AGE
app-tier-74cd4c68c9-h7hvf      1/1     Running   0          10m
data-tier-8646dd765b-wjmjd     1/1     Running   0          10m
support-tier-997bc57fb-7597k   2/2     Running   0          3m36s
support-tier-997bc57fb-79gsl   2/2     Running   0          8m27s
support-tier-997bc57fb-b8c4q   2/2     Running   0          47s
support-tier-997bc57fb-fggcx   2/2     Running   0          47s
support-tier-997bc57fb-k8kfd   2/2     Running   0          47s
```

```
admin@Jamess-MBP src % kubectl scale -n deployments deployment app-tier --replicas=5

deployment.apps/app-tier scaled
deployment.apps/app-tier scaled
admin@Jamess-MBP src % kubectl -n deployments get pods
NAME                           READY   STATUS    RESTARTS   AGE
app-tier-74cd4c68c9-kd7z4      1/1     Running   0          13s
app-tier-74cd4c68c9-qnrrh      1/1     Running   0          13s
app-tier-74cd4c68c9-rkbxh      1/1     Running   0          13s
app-tier-74cd4c68c9-thh25      1/1     Running   0          13s
app-tier-74cd4c68c9-z99h4      1/1     Running   0          9m48s
data-tier-8646dd765b-k6h98     1/1     Running   0          9m48s
support-tier-997bc57fb-2lptf   2/2     Running   0          3m4s
support-tier-997bc57fb-42s98   2/2     Running   0          5m53s
support-tier-997bc57fb-c579t   2/2     Running   0          9m48s
support-tier-997bc57fb-gpmj5   2/2     Running   0          3m3s
support-tier-997bc57fb-zfcrn   2/2     Running   0          3m3s
admin@Jamess-MBP src %
```

```
admin@Jamess-MBP src % kubectl describe -n deployments service app-tier
Name:              app-tier
Namespace:         deployments
Labels:            app=microservices
Annotations:       <none>
Selector:          tier=app
Type:              ClusterIP
IP:                10.111.185.244
Port:              <unset>  8080/TCP
TargetPort:        8080/TCP
Endpoints:         172.17.0.10:8080,172.17.0.11:8080,172.17.0.12:8080 + 2 more...
Session Affinity:  None
Events:            <none>
```

→ all 5 replicas are tracked & load balanced in the service.

CONCLUSION

- Deployments to manage Pods in each tier
- Kubernetes ensures actual state matches desired state
- `kubectl scale` to scale number of replicas
- Services seamlessly support scaling
- Scaling is best with stateless Pods

→ even there is any failure(or)manual deletion it ensures to bring back to desired state

→ we can't scale the data tier in above example.