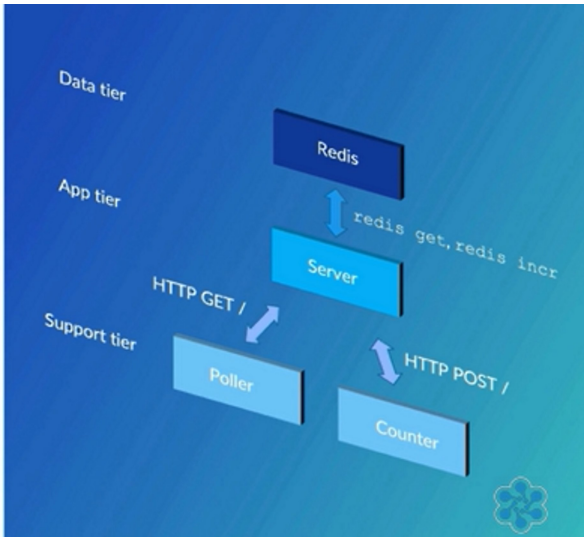
4. Multi container pods

27 February 2022 15:26

Multi-container pods

Our Application

- A counter
- 4 containers split across 3 tiers
- Application tier is a Node.js server container
- Redis data tier storing the counter
- Poller and counter in the support tier
- All containers configured using environment variables



Kubernetes Namespaces

Namespaces separate resources according to users, environments, or applications

Role-based access control (RBAC) to secure access per Namespace

Using Namespaces is a best practice

```
admin@Jamess-MBP src % cat 3.1-namespace.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: microservice
  labels:
    app: counter
admin@Jamess-MBP src % kubectl create -f 3.1-namespace.yaml
namespace/microservice created
```

→ namespace name.

```
admin@Jamess-MBP src % cat 3.2-multi_container.yaml
apiVersion: v1
kind: Pod
metadata:
  name: app
spec:
  containers:
    - name: redis
      image: redis:latest
      imagePullPolicy: IfNotPresent
      ports:
        - containerPort: 6379

    - name: server
      image: lrakai/microservices:server-v1
      ports:
        - containerPort: 8080
      env:
        - name: REDIS_URL
          value: redis://localhost:6379
```

→ Better always mention specific image than latest

→ Pully latest when pod is started.

→ we are not mentioning namespace in here because if we do that pod will become less portable.

environment variable

```
    env:
      - name: REDIS_URL
        value: redis://localhost:6379
  - name: counter
    image: lrakai/microservices:counter-v1
    env:
      - name: API_URL
        value: http://localhost:8080
```

environment variable

because containers in pod share same network stack.

```
  - name: poller
    image: lrakai/microservices:poller-v1
    env:
      - name: API_URL
        value: http://localhost:8080
```

```
admin@Jamess-MBP src % kubectl create -f 3.2-multi_container.yaml -n microservice
pod/app created
```

→ mentioning name space.

```
admin@Jamess-MBP src % kubectl get -n microservice pod app
NAME   READY   STATUS              RESTARTS   AGE
app    0/4     ContainerCreating   0          50s
admin@Jamess-MBP src %
```

describe gives more details than get

* once running we can see (stderr, stdout) logs in the containers.

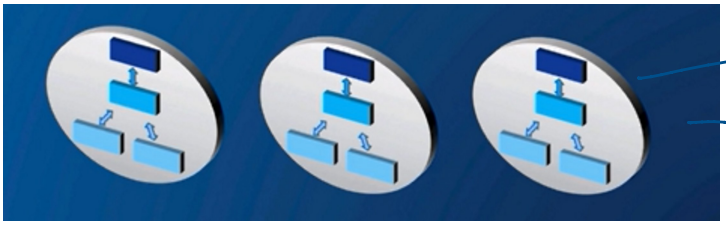
```
admin@Jamess-MBP src % kubectl logs -n microservice app counter --tail 10
Incrementing counter by 8 ...
Incrementing counter by 9 ...
Incrementing counter by 3 ...
Incrementing counter by 2 ...
Incrementing counter by 2 ...
Incrementing counter by 9 ...
Incrementing counter by 2 ...
Incrementing counter by 10 ...
Incrementing counter by 6 ...
Incrementing counter by 9 ...
```

```
admin@Jamess-MBP src % kubectl logs -n microservice app poller -f
```

→ to follow logs.

Issue with above approach.

- kubernetes → pods are building blocks
  - It can scale only by increasing number of pods not by increasing containers inside the pods.



→ This is not what we want

→ They are tightly coupled.

- we should break application to multiple pods & then scale will be ideal