Kubernetes Labs - 2

- *Doc created by → Lucas Brito Rodrigues dos Santos*
- **Está é a segunda parte do projeto, para acompanhar o anterior valide o repositorio abaixo:**

  https://github.com/Luck1080p/DevOps-Studies/tree/main/Kubernetes

- *Está doc foi baseada nas seguintesa fontes:*

  Curso Online Kubernetes: Deployments, Volumes e Escalabilidade | Alura

  O Kubernetes é o principal gerenciador de cluster para aplicações containerizadas. Aprenda nesse curso como utilizar o Kubernetes no Linux com o Minikube, no Windows com o Docker Desktop e entenda o funcionamento em Cloud Providers.

  https://cursos.alura.com.br/course/kubernetes-deployments-volumes-escalabilidade

  Production-Grade Container Orchestration

  Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15

  https://kubernetes.io/

  kubern

- *Busco demonstrar em laboratórios práticos, realizados com base nessas fontes as funções e a utilização do Kubernets*

> **Summary**
>
> - Conhecendo ReplicaSets e Deployments
> - Persistindo dados com o kubernetes
> - Kubernetes on the Cloud

> - Statefulsetes
> - Checando status com probes
> - Como escalar com o Horizontal Pod Autoscales

▼ **Conhecendo ReplicaSets e Deployments**

▼ **Revisão do projeto anterior**

- Projeto anterior completo:

```
https://github.com/Luck1080p/DevOps-Studies/blob/main/Kubernetes/alura-kubernetes/Kubernetes-labs-1.pdf
```

- No caso esta é a segunda parte dos laboratarios realizados anteriormente, valide o mesmo através do link abaixo :

```
https://github.com/Luck1080p/DevOps-Studies/blob/main/Kubernetes/alura-kubernetes/Kubernetes-labs-1.pdf
```

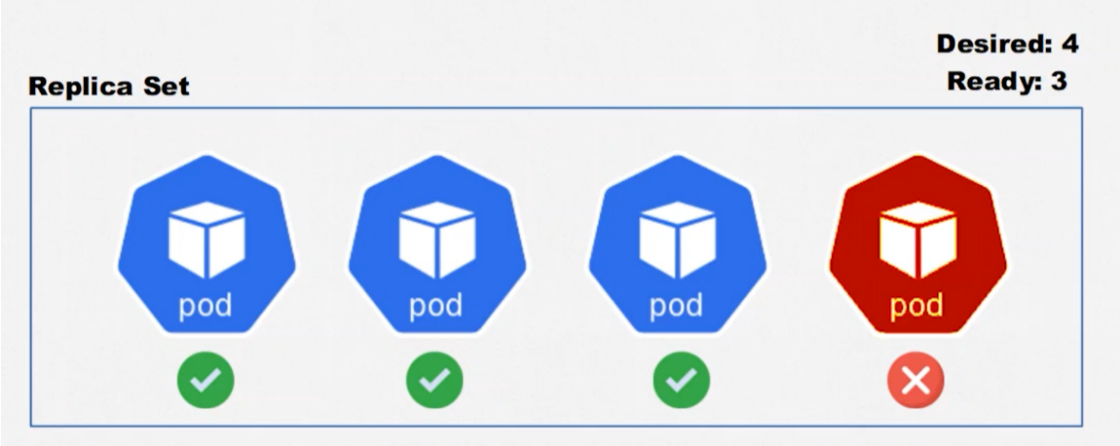
- Link do projeto anterior

```
https://github.com/alura-cursos/1846-kubernetes/archive/Aula6.zip
```

- Utilizaremos o projeto anterior para dar prosseguimento a este

▼ **Conhecendo ReplicaSets**

- Conforme vimos, o kubernetes permite a recriação dos pods caso os mesmos falhem
- Os pods por serem efemeros, caso falhem, o mesmo não irá mais voltar, porém podemos usar recursos que permitem em caso de falha voltar
- Neste caso utilizaremos o replica set, que ira recriar um pod novamente caso falhem
- O recurso replica set recria o(s) pod(s) caso os mesmo falhem



- Vamos criar um replicaset
  - Vamos apagar primeiramente o pod do portal-noticia

```
lk47@Latitude-E6410:~$ kubectl get pods
NAME                READY   STATUS    RESTARTS        AGE
db-noticias         1/1     Running   3 (7d17h ago)   22d
portal-noticias     1/1     Running   2 (7d17h ago)   22d
sistema-noticias    1/1     Running   3 (7d17h ago)   22d
lk47@Latitude-E6410:~$ kubectl delete pod portal-noticias
pod "portal-noticias" deleted
lk47@Latitude-E6410:~$ kubectl get pods
NAME                READY   STATUS    RESTARTS        AGE
db-noticias         1/1     Running   3 (7d17h ago)   22d
sistema-noticias    1/1     Running   3 (7d17h ago)   22d
lk47@Latitude-E6410:~$
```

- No caso vamos criar um YAML portal-noticia-replicaset
  - Vamos criar uma pasta em nosso projeto para podermos incluir os YAMLS de replicaset

```
lk47@Latitude-E6410:~/Desktop/Projects$ ls
pods  projeto-noticias  replicaset  svcs
lk47@Latitude-E6410:~/Desktop/Projects$ cd replicaset/
lk47@Latitude-E6410:~/Desktop/Projects/replicaset$
```

- Vamos criar o codigo da seguinte forma

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: portal-noticias-replicaset
spec:
  template:
    metadata:
      name: portal-noticias
      labels:
        app: portal-noticias
    spec:
      containers:
        - name: portal-noticias-container
          image: aluracursos/portal-noticias:1
```

```
        ports:
          - containerPort: 80
        envFrom:
          - configMapRef:
              name: portal-configmap
  replicas: 3
  selector:
    matchLabels:
      app: portal-noticias
```

```
lk47@Latitude-E6410:~/Desktop/Projects/replicaset$ nano replicaset.yaml
lk47@Latitude-E6410:~/Desktop/Projects/replicaset$ cat replicaset.yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: portal-noticias-replicaset
spec:
  template:
    metadata:
      name: portal-noticias
      labels:
        app: portal-noticias
    spec:
      containers:
        - name: portal-noticias-container
          image: aluracursos/portal-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: portal-configmap
  replicas: 3
  selector:
    matchLabels:
      app: portal-noticias
```

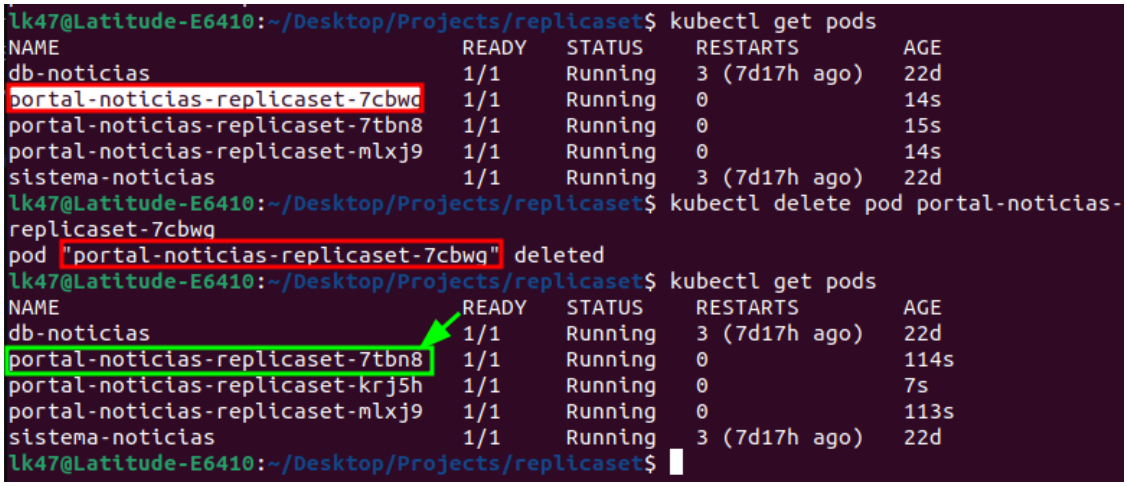
- A declaração é quase semelhante, porém especificamos um tamplate antes onde definimos todas as informações do pod
- E podemos definir a quantidade de replicas
- Usamos também um matchlabels para que o replicaset identifique os pods

- Feito isso vamos subir o replicaset

```
lk47@Latitude-E6410:~/Desktop/Projects/replicaset$ kubectl apply -f replicaset.yaml
replicaset.apps/portal-noticias-replicaset created
lk47@Latitude-E6410:~/Desktop/Projects/replicaset$ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
portal-noticias-replicaset    3         3         3       11s
lk47@Latitude-E6410:~/Desktop/Projects/replicaset$ kubectl get pods
NAME                                READY   STATUS    RESTARTS        AGE
db-noticias                         1/1     Running   3 (7d17h ago)   22d
portal-noticias-replicaset-7cbwg    1/1     Running   0               14s
portal-noticias-replicaset-7tbn8    1/1     Running   0               15s
portal-noticias-replicaset-mlxj9    1/1     Running   0               14s
sistema-noticias                    1/1     Running   3 (7d17h ago)   22d
lk47@Latitude-E6410:~/Desktop/Projects/replicaset$
```

    - Agora podemos identificar que os pods subiram, e também podemos ver que os pods possuem identificadores → 7cbwg, 7tbn8, mlxj9

- Se excluirmos um pod podemos ver que outro pod irá subir por conta do replicaset, porém com um identificador diferente



- Podemos usar o comando `kubectl get rs` que ira nos mostrar os replicasets no k8s

▼ **Conhecendo Deployments**

- Os deployments tem as mesmas funcionalidades de um replicaset
- Quando definimos um deployment estamos automaticamente definindo um replicaset
- No caso o deployment é como se estivesse uma camada acima de um replicaset
- Vamos criar um Deployment de nginx da seguinte forma

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      name: nginx-pod
      labels:
        app: nginx-pod
    spec:
      containers:
        - name: nginx-container
          image: nginx:stable
          ports:
            - containerPort: 80
  selector:
    matchLabels:
      app: nginx-pod
```

```
lk47@Latitude-E6410:~/Desktop/Projects$ ls
pods  projeto-noticias  replicaset  svcs
lk47@Latitude-E6410:~/Desktop/Projects$ mkdir deployment
lk47@Latitude-E6410:~/Desktop/Projects$ cd deployment/
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ nano nginx-deployment.yaml
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ cat nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      name: nginx-pod
      labels:
        app: nginx-pod
    spec:
      containers:
        - name: nginx-container
          image: nginx:stable
          ports:
            - containerPort: 80
  selector:
    matchLabels:
      app: nginx-pod
```

- Vamos aplicar o nosso deployment

```
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl apply -f nginx-deployment.y
aml
deployment.apps/nginx-deployment created
```

- Portanto vamos validar os ambientes
  - `kubectl get pods`
  - `kubectl get rs`
  - `kubectl get deployments`

```
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl get pods
NAME                                READY   STATUS    RESTARTS        AGE
db-noticias                         1/1     Running   3 (7d17h ago)   22d
nginx-deployment-886468cd7-7kcdb    1/1     Running   0               38s
nginx-deployment-886468cd7-7vxj9    1/1     Running   0               38s
nginx-deployment-886468cd7-sl6xc    1/1     Running   0               38s
portal-noticias-replicaset-7tbn8    1/1     Running   0               7m34s
portal-noticias-replicaset-krj5h    1/1     Running   0               5m47s
portal-noticias-replicaset-mlxj9    1/1     Running   0               7m33s
sistema-noticias                    1/1     Running   3 (7d17h ago)   22d
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-886468cd7    3         3         3       47s
portal-noticias-replicaset    3         3         3       7m43s
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           56s
lk47@Latitude-E6410:~/Desktop/Projects/deployment$
```

- A grande diferença de um deployment é o controle de versionamento de codigos
- Se usarmos o comando `kubectl rollout history deployment nginx-deployment` podemos ver revisão e a causa de mudança

```
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl rollout history deployment
nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
```

- Para está tarefa vamos deixar o pod do nginx na versão latest

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      name: nginx-pod
      labels:
        app: nginx-pod
    spec:
      containers:
        - name: nginx-container
          image: nginx:latest
          ports:
            - containerPort: 80
  selector:
    matchLabels:
      app: nginx-pod
```

- Vamos usar o comando para subir os pods novamente, `kubectl apply -f .\nginx-deployment.yaml --record`

```
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl apply -f nginx-deployment.y
aml --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/nginx-deployment configured
```

- Se rodarmos o comando de rollout novamente, podemos visualizar a segunda altaração

```
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl rollout history deployment
nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         kubectl apply --filename=nginx-deployment.yaml --record=true
```

- Podemos realizar o comando, para marcar uma anotação em change-cause `kubectl annotate deployment nginx-deployment kubernetes.io/change-cause= "Definindo a imagem com versão latest" --overwrite`
  - Executando novamente o comando rollout history podemos ver as anotações das versões na qual definimos

```
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl rollout history deployment nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         kubectl apply --filename=nginx-deployment.yaml --record=true

lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl annotate deployment nginx-deployment kubernetes.io/chang
e-cause="Definindo a imagem com versão latest" --overwrite
deployment.apps/nginx-deployment annotate
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl rollout history deployment nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         Definindo a imagem com versão latest

lk47@Latitude-E6410:~/Desktop/Projects/deployment$
```

- Vamos voltar o nginx para versão 1 e vamos subir novamente o deployment

```
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ cat nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      name: nginx-pod
      labels:
        app: nginx-pod
    spec:
      containers:
        - name: nginx-container
          image: nginx:1
          ports:
            - containerPort: 80
  selector:
    matchLabels:
      app: nginx-pod
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl apply -f nginx-deployment.yaml --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/nginx-deployment unchanged
```

- Vamos definir um annotate novamente

```
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl rollout history deployment nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         Definindo a imagem com versão latest
3         kubectl apply --filename=nginx-deployment.yaml --record=true

lk47@Latitude-E6410:~/Desktop/Projects/deployment$ ^C
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl annotate deployment nginx-deployment kubernetes.io/chang
e-cause="Definindo a imagem com versão 1" --overwrite
deployment.apps/nginx-deployment annotate
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl rollout history deployment nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         Definindo a imagem com versão latest
3         Definindo a imagem com versão 1
```

- Vamos usar o `kubectl describe pod` em qualquer um do nginx e podemos ver que está na versão 1

```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  16m   default-scheduler  Successfully assigned default/nginx-deployment-dd4f85dc5-5s9cj to min
ikube
  Normal  Pulled     16m   kubelet            Container image "nginx:1" already present on machine
  Normal  Created    16m   kubelet            Created container nginx-container
  Normal  Started    16m   kubelet            Started container nginx-container
```

- Vamos voltar para a versão latest usando o comando `kubectl rollout undo deployment nginx-deployment --to-revision=2` fazendo isso podemos voltar para a versão anterior

```
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl rollout undo deployment nginx-deployment --to-revision=2
deployment.apps/nginx-deployment rolled back
```

  - Validando algum dos pods novamente

```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  88s   default-scheduler  Successfully assigned default/nginx-deployment-655dd59ddc-kllbg to mi
nikube
  Normal  Pulled     83s   kubelet            Container image "nginx:latest" already present on machine
  Normal  Created    83s   kubelet            Created container nginx-container
  Normal  Started    82s   kubelet            Started container nginx-container
lk47@Latitude-E6410:~/Desktop/Projects/deployment$
```

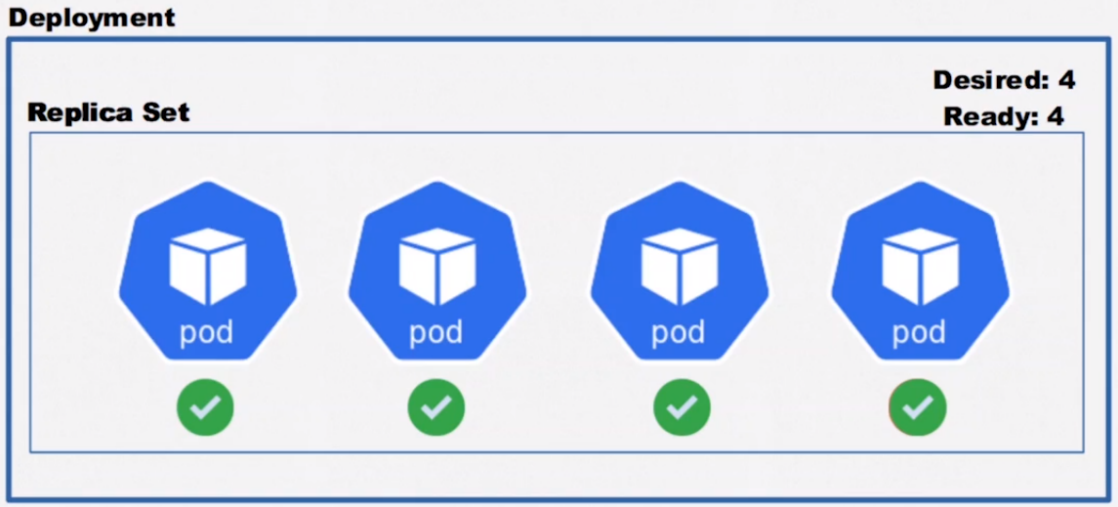
  - Podemos ver que retomou para a versão latest que no caso seria a 2 de nosso rollout history

```
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl rollout history deployment nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
3         Definindo a imagem com versão 1
4         Definindo a imagem com versão latest

lk47@Latitude-E6410:~/Desktop/Projects/deployment$
```

No caso estava no 2, porém como foi aplicado novamente ela subiu para o 4.

- Dessa forma o deployment irá gerenciar todo o versionamento, portanto está é forma mais atual para subirmos pods no k8s

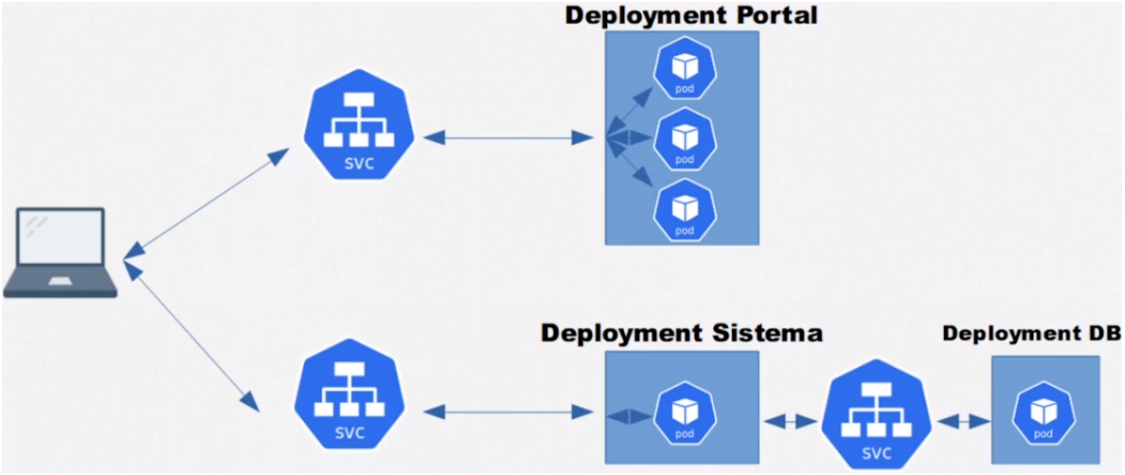


▼ **ReplicaSets vs Deployments**

- *Qual diferenças entre* `ReplicaSets` *e* `Deployments`*?*
  - **Quando criados, Deployments auxiliam com controle de versionamento e criam um ReplicaSet automaticamente.**

▼ **Aplicando Deployments ao projeto**

- Vamos aplicar os deployments para o projeto, serão 3 pods para o portal e 1 para sistema e DB



- vamos remover o deployment do nginx e do portal que criamos

```
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl get deploy
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           58m
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl delete deploy nginx-deployment
deployment.apps "nginx-deployment" deleted
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
portal-noticias-replicaset    3         3         3       65m
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl delete rs portal-noticias-replicaset
replicaset.apps "portal-noticias-replicaset" deleted
lk47@Latitude-E6410:~/Desktop/Projects/deployment$ kubectl get pods
NAME               READY   STATUS    RESTARTS        AGE
db-noticias        1/1     Running   3 (7d18h ago)   22d
sistema-noticias   1/1     Running   3 (7d18h ago)   22d
lk47@Latitude-E6410:~/Desktop/Projects/deployment$
```

- Vamos usar o conteúdo do replicaset para criar o deployment do portal-noticias

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: portal-noticias-deployment
spec:
  template:
    metadata:
      name: portal-noticias
      labels:
        app: portal-noticias
    spec:
      containers:
        - name: portal-noticias-container
          image: aluracursos/portal-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: portal-configmap
  replicas: 3
  selector:
    matchLabels:
      app: portal-noticias
```

```
lk47@Latitude-E6410:~/Desktop/Projects$ cd projeto-noticias/
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias$ ls
configmap  pods  svcs
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias$ mkdir deployments
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias$ cd deployments/
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ nano portal-noticias-deploy.yaml
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ cat portal-noticias-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: portal-noticias-deployment
spec:
  template:
    metadata:
      name: portal-noticias
      labels:
        app: portal-noticias
    spec:
      containers:
        - name: portal-noticias-container
          image: aluracursos/portal-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: portal-configmap
  replicas: 3
  selector:
    matchLabels:
      app: portal-noticias
```

- Só alteramos o replicaset para deployment
- Vamos usar o annotate para definir o change cause

```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl apply -f portal-noticias-deploy.yaml --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/portal-noticias-deployment created
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl annotate deployment portal-noticias-deployment kubernet
es.io/change-cause="portal versão 1" --overwrite
deployment.apps/portal-noticias-deployment annotate
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl rollout history deployment portal-noticias-deployment
deployment.apps/portal-noticias-deployment
REVISION  CHANGE-CAUSE
1         portal versão 1
```

- Agora vamos realizar o deployment do sistema-noticias

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sistema-noticias-deployment
spec:
  replicas: 1
  template:
    metadata:
      name: sistema-noticias
      labels:
        app: sistema-noticias
    spec:
      containers:
        - name: sistema-noticias-container
          image: aluracursos/sistema-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: sistema-configmap
  selector:
    matchLabels:
      app: sistema-noticias
```

```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ nano sistema-noticias-deploy.yaml
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ cat sistema-noticias-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sistema-noticias-deployment
spec:
  replicas: 1
  template:
    metadata:
      name: sistema-noticias
      labels:
        app: sistema-noticias
    spec:
      containers:
        - name: sistema-noticias-container
          image: aluracursos/sistema-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: sistema-configmap
  selector:
    matchLabels:
      app: sistema-noticias
```

- Feito isso vamos apagar o pod anterior de sistema noticias e vamos subir o deployment
  - Vamos colocar o annotate para definir o change cause

```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl get pods
NAME                                        READY   STATUS    RESTARTS        AGE
db-noticias                                 1/1     Running   3 (7d18h ago)   22d
portal-noticias-deployment-5dcbbf85f4-d2rb2   1/1     Running   0               3m41s
portal-noticias-deployment-5dcbbf85f4-j677f   1/1     Running   0               3m41s
portal-noticias-deployment-5dcbbf85f4-qq4p8   1/1     Running   0               3m41s
sistema-noticias                            1/1     Running   3 (7d18h ago)   22d
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl delete pod sistema-noticias
pod "sistema-noticias" deleted
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl apply -f sistema-noticias-deploy.yaml --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/sistema-noticias-deployment created
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl annotate deployment sistema-noticias-deployment kuberne
tes.io/change-cause="sistema versão 1" --overwrite
deployment.apps/sistema-noticias-deployment annotate
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl rollout history deployment portal-noticias-deployment
deployment.apps/portal-noticias-deployment
REVISION  CHANGE-CAUSE
1         portal versão 1

lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl rollout history deployment sistema-noticias-deployment
deployment.apps/sistema-noticias-deployment
REVISION  CHANGE-CAUSE
1         sistema versão 1

lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$
```

- Agora vamos realizar o yaml do banco de dados

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: db-noticias-deployment
spec:
  replicas: 1
  template:
    metadata:
      name: db-noticias
      labels:
        app: db-noticias
    spec:
      containers:
        - name: db-noticias-container
          image: aluracursos/mysql-db:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
```

```
              name: db-configmap
  selector:
    matchLabels:
      app: db-noticias
```

```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ nano db-noticias-deploy.yaml
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ cat db-noticias-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: db-noticias-deployment
spec:
  replicas: 1
  template:
    metadata:
      name: db-noticias
      labels:
        app: db-noticias
    spec:
      containers:
        - name: db-noticias-container
          image: aluracursos/mysql-db:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: db-configmap
  selector:
    matchLabels:
      app: db-noticias
```

- Vamos apagar o pod antigo do pod do banco e vamos subir o deployment
    - Vamos pontuar o annotate para o change cause

```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl get pods
NAME                                        READY   STATUS    RESTARTS        AGE
db-noticias                                 1/1     Running   3 (7d18h ago)   22d
portal-noticias-deployment-5dcbbf85f4-d2rb2   1/1     Running   0               7m9s
portal-noticias-deployment-5dcbbf85f4-j677f   1/1     Running   0               7m9s
portal-noticias-deployment-5dcbbf85f4-qq4p8   1/1     Running   0               7m9s
sistema-noticias-deployment-67c4474df4-6hzjb   1/1     Running   0               3m1s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl delete pod db-noticias
pod "db-noticias" deleted
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl apply -f db-noticias-deploy.yaml --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/db-noticias-deployment created
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl annotate deployment db-noticias-deployment kubernetes.i
o/change-cause="db versão 1" --overwrite
deployment.apps/db-noticias-deployment annotate
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl rollout history deployment db-noticias-deployment
deployment.apps/db-noticias-deployment
REVISION  CHANGE-CAUSE
1         db versão 1
```

- Dessa forma definimos o deployment e temos a recriação automatizada dos pods, e o controle de versionamento dos pods

```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl get pods
NAME                                        READY   STATUS    RESTARTS   AGE
db-noticias-deployment-5c87d7d876-xj7x2     1/1     Running   0          71s
portal-noticias-deployment-5dcbbf85f4-d2rb2 1/1     Running   0          9m13s
portal-noticias-deployment-5dcbbf85f4-j677f 1/1     Running   0          9m13s
portal-noticias-deployment-5dcbbf85f4-qq4p8 1/1     Running   0          9m13s
sistema-noticias-deployment-67c4474df4-6hzjb 1/1    Running   0          5m5s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl get deploy
NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
db-noticias-deployment      1/1     1            1           83s
portal-noticias-deployment  3/3     3            3           9m25s
sistema-noticias-deployment 1/1     1            1           5m17s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/deployments$ kubectl rollout history deployment
deployment.apps/db-noticias-deployment
REVISION  CHANGE-CAUSE
1         db versão 1

deployment.apps/portal-noticias-deployment
REVISION  CHANGE-CAUSE
1         portal versão 1

deployment.apps/sistema-noticias-deployment
REVISION  CHANGE-CAUSE
1         sistema versão 1
```

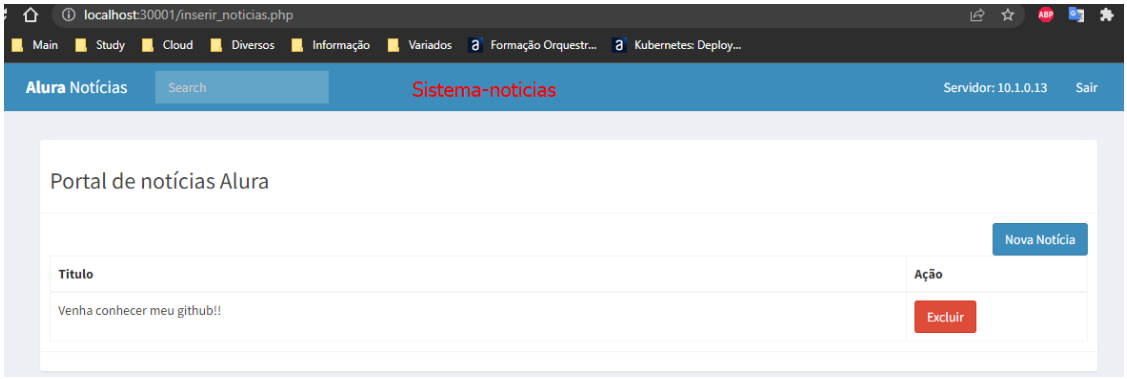
▼ Praticando Deployments

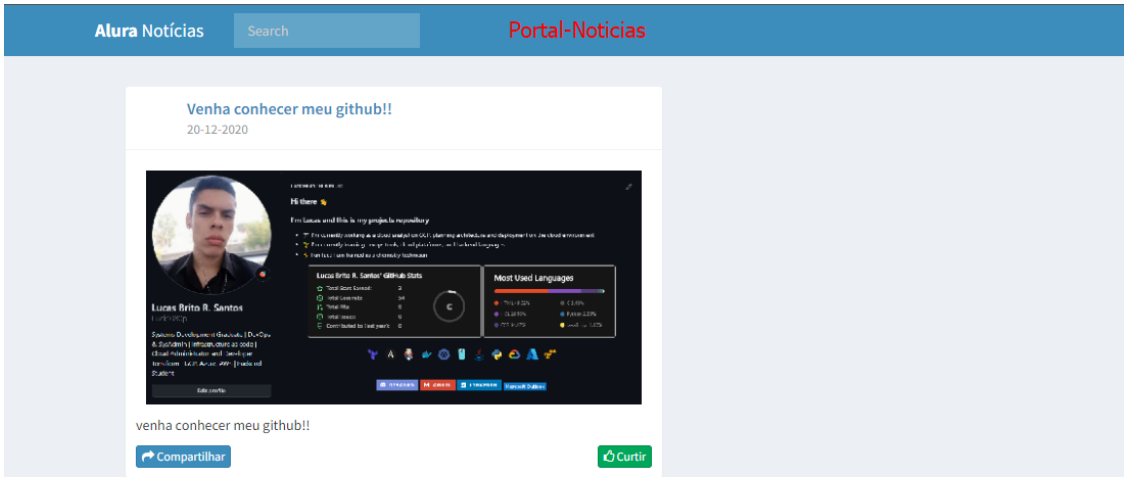
- *Qual comando podemos utilizar para voltar um deployment para uma revisão específica?*
  - `kubectl rollout undo deployment <nome do deployment> --to-revision=<versão a ser retornada>`

▼ Persistindo dados com o kubernetes

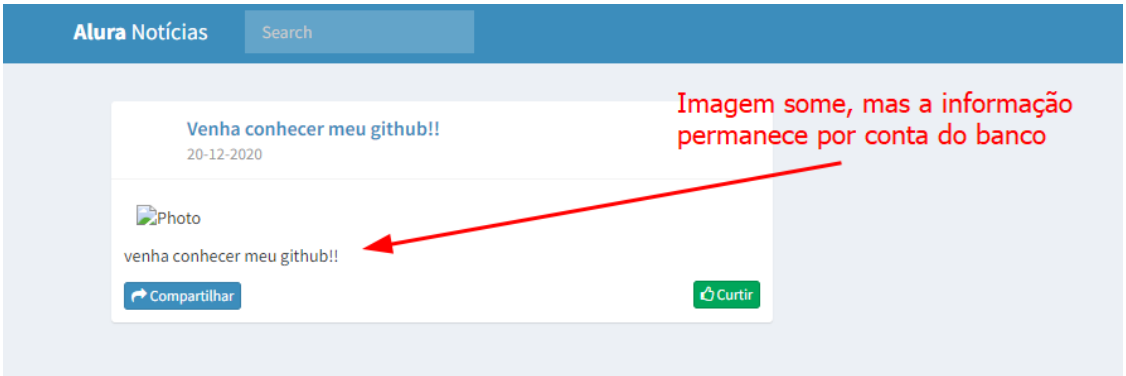
▼ Persistindo dados com volumes

- Os pods até então não possuem uma persistência de dados
- Se colocarmos uma noticia em nosso sistema de noticias e logo depois realizarmos um redeploy do pod portal, sistema ou banco, a noticia será perdida completamente



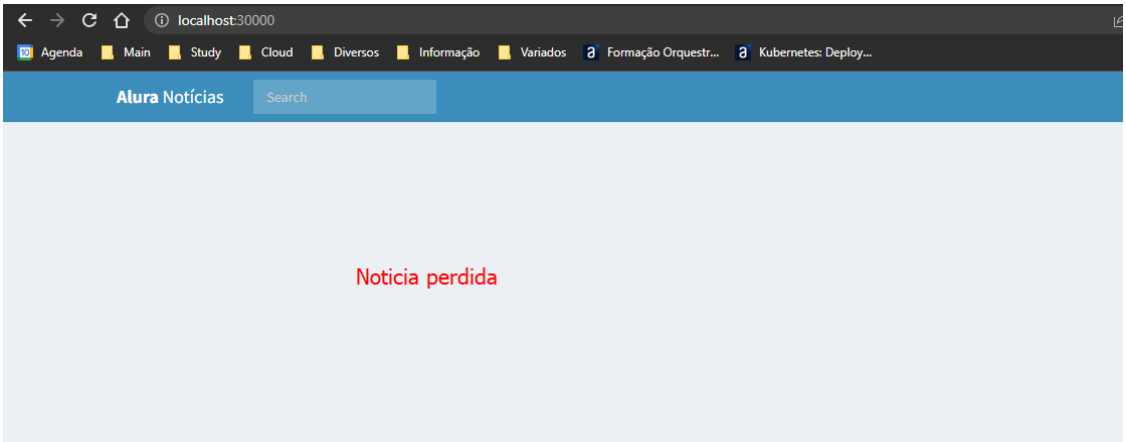



```
PS C:\Users\Lucas Brito> kubectl get pods
NAME                                        READY   STATUS    RESTARTS   AGE
db-noticias-deployment-7f95999c74-29hrc     1/1     Running   0          22m
portal-noticias-deployment-7ccc7b7c64-88958 1/1     Running   0          20m
portal-noticias-deployment-7ccc7b7c64-b7m9d 1/1     Running   0          20m
portal-noticias-deployment-7ccc7b7c64-mrbmf 1/1     Running   0          20m
sistema-noticias-deployment-664bdc4dcb-9bvjg 1/1    Running   0          20m
PS C:\Users\Lucas Brito> kubectl delete pod sistema-noticias-deployment-664bdc4dcb-9bvjg
pod "sistema-noticias-deployment-664bdc4dcb-9bvjg" deleted
PS C:\Users\Lucas Brito> kubectl get pods
NAME                                        READY   STATUS    RESTARTS   AGE
db-noticias-deployment-7f95999c74-29hrc     1/1     Running   0          22m
portal-noticias-deployment-7ccc7b7c64-88958 1/1     Running   0          21m
portal-noticias-deployment-7ccc7b7c64-b7m9d 1/1     Running   0          21m
portal-noticias-deployment-7ccc7b7c64-mrbmf 1/1     Running   0          21m
sistema-noticias-deployment-664bdc4dcb-2nxng 1/1    Running   0          6s
PS C:\Users\Lucas Brito>
```

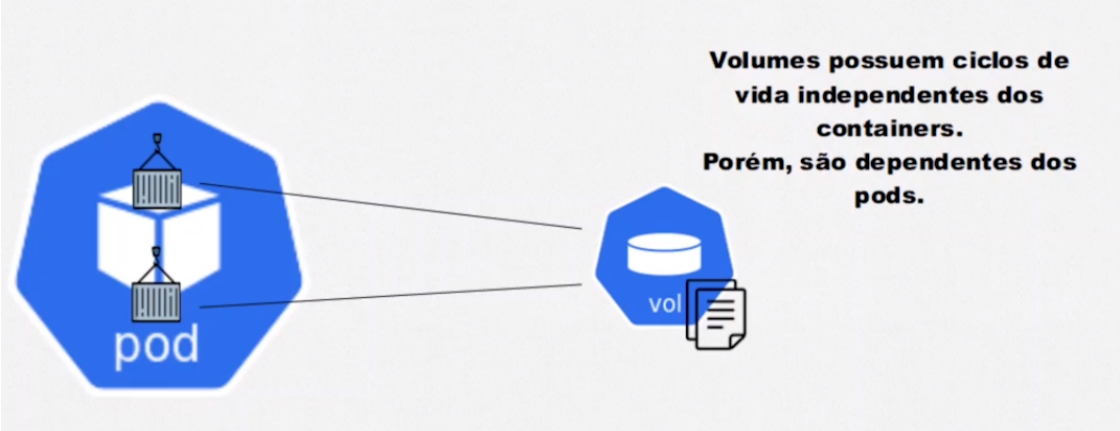


- Banco recriado

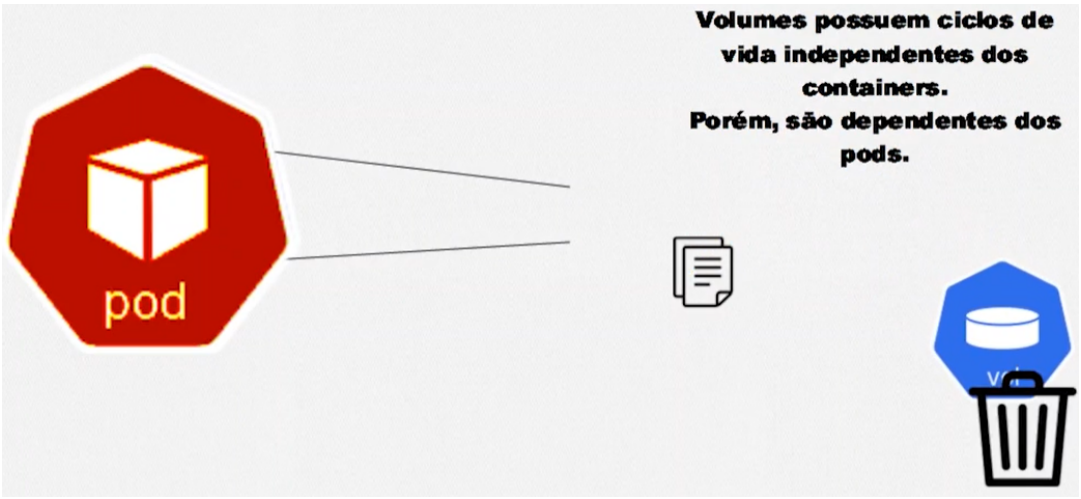
```
PS C:\Users\Lucas Brito> kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
db-noticias-deployment-7f95999c74-29hrc   1/1     Running   0          22m
portal-noticias-deployment-7ccc7b7c64-88958   1/1     Running   0          21m
portal-noticias-deployment-7ccc7b7c64-b7m9d   1/1     Running   0          21m
portal-noticias-deployment-7ccc7b7c64-mrbmf   1/1     Running   0          21m
sistema-noticias-deployment-664bdc4dcb-2nxng   1/1     Running   0          6s
PS C:\Users\Lucas Brito> kubectl delete pod db-noticias-deployment-7f95999c74-29hrc
pod "db-noticias-deployment-7f95999c74-29hrc" deleted
PS C:\Users\Lucas Brito> kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
db-noticias-deployment-7f95999c74-pb7q5   1/1     Running   0          6s
portal-noticias-deployment-7ccc7b7c64-88958   1/1     Running   0          23m
portal-noticias-deployment-7ccc7b7c64-b7m9d   1/1     Running   0          23m
portal-noticias-deployment-7ccc7b7c64-mrbmf   1/1     Running   0          23m
sistema-noticias-deployment-664bdc4dcb-2nxng   1/1     Running   0          2m39s
PS C:\Users\Lucas Brito>
```



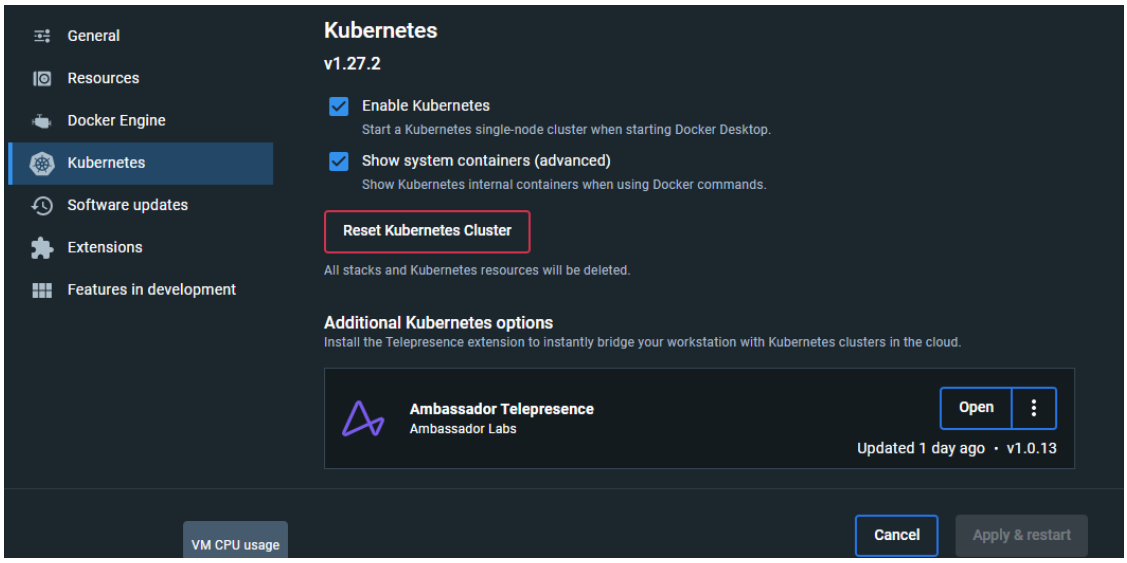
- Para isso precisamos entender a persistencia de dados primeiramente no ambito dos containers dentro dos pods para em seguida validar sobre a persistencia dentro dos pods
- No docker conseguimos compartilhar dados entre containers, no k8s precisamos entender o compartilhamento entre containers do mesmo pod
- Os principais recursos do kubernetes são : **Volumes**, **Persistent Volumes**, **Persistent Volumes Clain**, **Storage Classes**
- Começando pelos **volumes** que se assemelha ao volumes do docker, no caso os volumes são idependentes do ciclo de vida dos containers porém é dependente dos pods



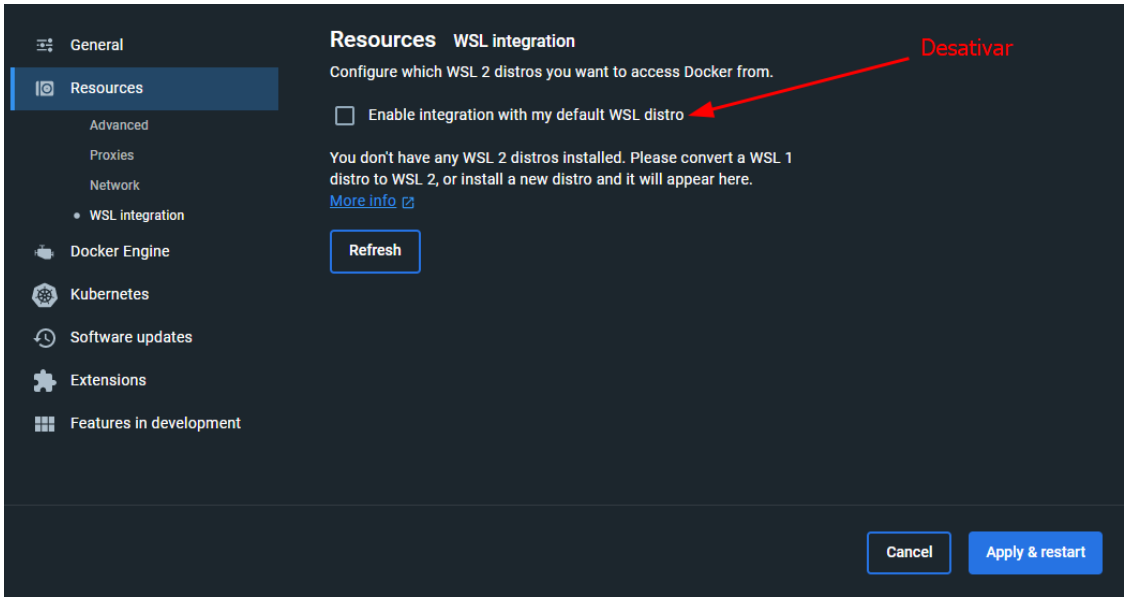
- Se o container falhar dentro do pod, o arquivo salvo no volume permanece porém se o pod falhar, o volume se perde



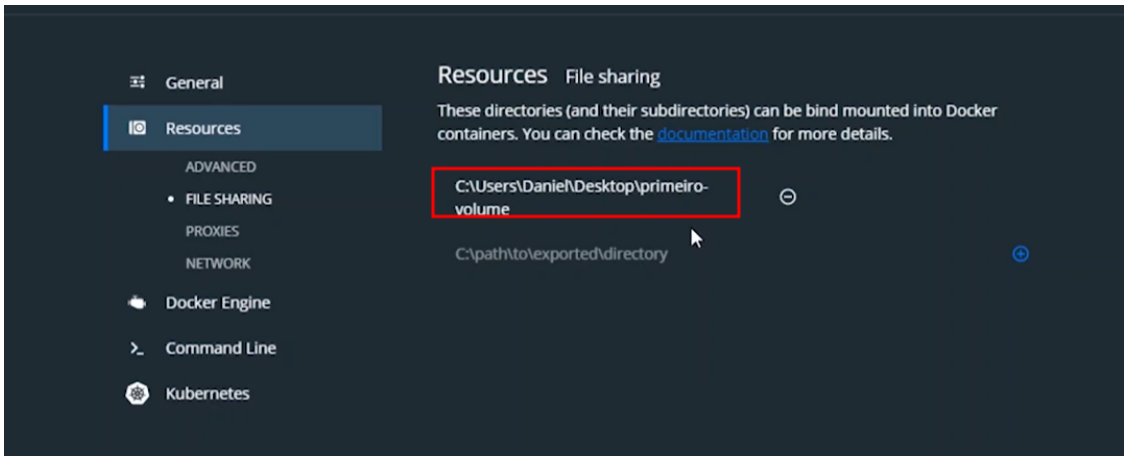
- O kubernetes possui diversos tipos de volumes, no caso vamos utilizar o **hostPath** que irá montar um diretorio dentro do nosso host, dessa forma os arquivos se manteram em nosso SO ao invés do kubernetes em sí
- No windows o principal diferencial é que há o docker desktop, por ele é possível utilizar o kubernetes



- Vamos desativar o WSL-2 caso esteja ativado, e ativar o kubernetes no docker, assim conseguiremos gerenciar o pod



- Precisaremos também setar uma opção no docker desktop, para setar nossa pasta onde será salvo os arquivos



- No linux será a mesma estrutura a diferença é que na estrutura dos arquivos que utilizamos é do minikube e não do SO pois estamos utilizando o minikube para virtualizar o ambiente kubernetes
- Dessa forma vamos criar dentro do minikube a pasta
  - Vamos acessar o minikube através do minikube ssh
  - Vamos criar a pasta primeiro-volume dentro do diretorio /home

- Vamos criar um pod com dois containers nginx latest e stable

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-volume
spec:
  containers:
    - name: container-1
      image: nginx:latest
      volumeMounts:
        - mountPath: /volume-dentro-do-container
          name: primeiro-volume
    - name: container-2
      image: nginx:stable
      volumeMounts:
        - mountPath: /volume-dentro-do-container
          name: primeiro-volume
  volumes:
    - name: primeiro-volume
      hostPath:
```

```
        path: /home/primeiro-volume
        type: Directory
```

```
lk47@Latitude-E6410:~/Desktop/Projects/pods$ cat pod-volume.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-volume
spec:
  containers:
    - name: container-1
      image: nginx:latest
      volumeMounts:
        - mountPath: /volume-dentro-do-container
          name: primeiro-volume
    - name: container-2
      image: nginx:stable
      volumeMounts:
        - mountPath: /volume-dentro-do-container
          name: primeiro-volume
  volumes:
    - name: primeiro-volume
      hostPath:
        path: /home/primeiro-volume
        type: Directory
```

- Vamos criar o pod e vamos dar um describe

```
lk47@Latitude-E6410:~/Desktop/Projects/pods$ kubectl apply -f pod-volume.yaml
pod/pod-volume created
lk47@Latitude-E6410:~/Desktop/Projects/pods$ kubectl get pods
NAME                                          READY   STATUS              RESTARTS      AGE
db-noticias-deployment-5c87d7d876-xj7x2       1/1     Running             1 (30m ago)   33h
pod-volume                                    0/2     ContainerCreating   0             4s
portal-noticias-deployment-5dcbbf85f4-d2rb2   1/1     Running             1 (30m ago)   33h
portal-noticias-deployment-5dcbbf85f4-j677f   1/1     Running             1 (30m ago)   33h
portal-noticias-deployment-5dcbbf85f4-qq4p8   1/1     Running             1 (30m ago)   33h
sistema-noticias-deployment-67c4474df4-6hzjb  1/1     Running             1 (30m ago)   33h
lk47@Latitude-E6410:~/Desktop/Projects/pods$ kubectl get pods
NAME                                          READY   STATUS    RESTARTS      AGE
db-noticias-deployment-5c87d7d876-xj7x2       1/1     Running   1 (30m ago)   33h
pod-volume                                    2/2     Running   0             5s
portal-noticias-deployment-5dcbbf85f4-d2rb2   1/1     Running   1 (30m ago)   33h
portal-noticias-deployment-5dcbbf85f4-j677f   1/1     Running   1 (30m ago)   33h
portal-noticias-deployment-5dcbbf85f4-qq4p8   1/1     Running   1 (30m ago)   33h
sistema-noticias-deployment-67c4474df4-6hzjb  1/1     Running   1 (30m ago)   33h
```

```
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-bkp92 (ro)
      /volume-dentro-do-container from primeiro-volume (rw)
Conditions:
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  primeiro-volume:
    Type:          HostPath (bare host directory volume)
    Path:          /home/primeiro-volume
    HostPathType:  Directory
  kube-api-access-bkp92:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
```

- Vamos acessar um dos container e vamos validar se está sendo possível haver persistencia de dados

```
lk47@Latitude-E6410:~/Desktop/Projects/pods$ kubectl exec -it pod-volume --container container-1 -- bash
root@pod-volume:/# ls
bin   dev                  docker-entrypoint.sh  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint.d  etc                   lib   lib64  media   opt  root  sbin  sys  usr  volume-dentro-do-container
root@pod-volume:/# cd volume-dentro-do-container/
root@pod-volume:/volume-dentro-do-container# touch teste.txt
root@pod-volume:/volume-dentro-do-container# ls
teste.txt
root@pod-volume:/volume-dentro-do-container#
```

  - Vamos acessar outro para ver se o arquivo permanece

```
lk47@Latitude-E6410:~/Desktop/Projects/pods$ kubectl exec -it pod-volume --container container-2 -- bash
root@pod-volume:/# ls
bin   dev                  docker-entrypoint.sh  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint.d  etc                   lib   lib64  media   opt  root  sbin  sys  usr  volume-dentro-do-container
root@pod-volume:/# cd volume-dentro-do-container/
root@pod-volume:/volume-dentro-do-container# ls
teste.txt
root@pod-volume:/volume-dentro-do-container#
```

  - E agora vamos apagar e vamos até a pasta no minikube para ver se os dados ainda estão lá

```
lk47@Latitude-E6410:~/Desktop/Projects/pods$ kubectl delete pod pod-volume
pod "pod-volume" deleted
lk47@Latitude-E6410:~/Desktop/Projects/pods$ minikube ssh
                         _             _
            _         _ ( )           ( )
  ___ ___  (_)  ___  (_)| |/')  _   _ | |_      __
/' _ ` _ `\| |/' _ `\| || , <  ( ) ( )| '_`\  /'__`\
| ( ) ( ) || || ( ) || || |\`\ | (_) || |_) )(  ___/
(_) (_) (_)(_)(_) (_)(_)(_) (_)`\___/'(_,__/'`\____)

$ cd /home/primeiro-volume
$ ls
teste.txt
$
```

- Está é a forma correta quando utilizamos linux, porém podemos automatizar as coisas , vamos apagar o pod atual e alterar novamente o yaml
- Vamos alterar o nome da pasta primeiro-volume no yaml para segundo volume da seguinte forma, e vamos adicionar em type: DirectoryOrCreate

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-volume
spec:
  containers:
    - name: container-1
      image: nginx:latest
      volumeMounts:
        - mountPath: /volume-dentro-do-container
          name: segundo-volume
    - name: container-2
      image: nginx:stable
      volumeMounts:
        - mountPath: /volume-dentro-do-container
          name: segundo-volume
  volumes:
```

```
    - name: segundo-volume
      hostPath:
        path: /home/segundo-volume
        type: DirectoryOrCreate
```

```
lk47@Latitude-E6410:~/Desktop/Projects/pods$ cat pod-volume.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-volume
spec:
  containers:
    - name: container-1
      image: nginx:latest
      volumeMounts:
        - mountPath: /volume-dentro-do-container
          name: segundo-volume
    - name: container-2
      image: nginx:stable
      volumeMounts:
        - mountPath: /volume-dentro-do-container
          name: segundo-volume
  volumes:
    - name: segundo-volume
      hostPath:
        path: /home/segundo-volume
        type: DirectoryOrCreate
```

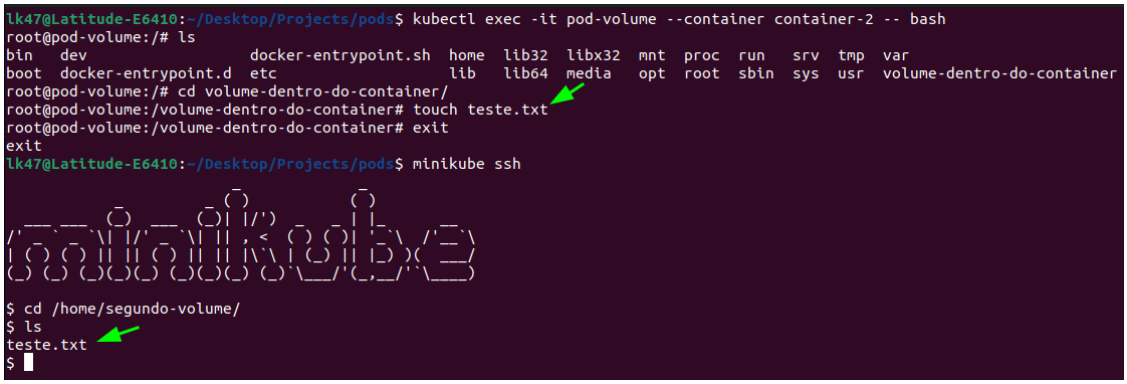
```
lk47@Latitude-E6410:~/Desktop/Projects/pods$ kubectl apply -f pod-volume.yaml
pod/pod-volume created
lk47@Latitude-E6410:~/Desktop/Projects/pods$ kubectl get pods
NAME                                        READY   STATUS              RESTARTS      AGE
db-noticias-deployment-5c87d7d876-xj7x2     1/1     Running             2 (19h ago)   2d4h
pod-volume                                  0/2     ContainerCreating   0             4s
portal-noticias-deployment-5dcbbf85f4-d2rb2 1/1     Running             2 (19h ago)   2d4h
portal-noticias-deployment-5dcbbf85f4-j677f 1/1     Running             2 (19h ago)   2d4h
portal-noticias-deployment-5dcbbf85f4-qq4p8 1/1     Running             2 (19h ago)   2d4h
sistema-noticias-deployment-67c4474df4-6hzjb 1/1    Running             2 (19h ago)   2d4h
lk47@Latitude-E6410:~/Desktop/Projects/pods$ kubectl get pods
NAME                                        READY   STATUS    RESTARTS      AGE
db-noticias-deployment-5c87d7d876-xj7x2     1/1     Running   2 (19h ago)   2d4h
pod-volume                                  2/2     Running   0             8s
portal-noticias-deployment-5dcbbf85f4-d2rb2 1/1     Running   2 (19h ago)   2d4h
portal-noticias-deployment-5dcbbf85f4-j677f 1/1     Running   2 (19h ago)   2d4h
portal-noticias-deployment-5dcbbf85f4-qq4p8 1/1     Running   2 (19h ago)   2d4h
sistema-noticias-deployment-67c4474df4-6hzjb 1/1    Running   2 (19h ago)   2d4h
```

- Dessa forma caso o diretorio não exista o mesmo será criado
- Acessando o minikube podemos visualizar a segunda pasta no /home

```
lk47@Latitude-E6410:~/Desktop/Projects/pods$ minikube ssh
                         _             _
                        _ (_)           (_)
  ___ ___  (_)  ___   (_)| |/')  _   _ | |_      __
/' _ ` _ `\| |/' _ `\ | || , <  ( ) ( )| '_`\  /'__`\
| ( ) ( ) || || ( ) || || |\`\ | (_) || |_) )(  ___/
(_) (_) (_)(_)(_) (_)(_)(_) (_)`\___/'(_,__/''\____)

$ cd /home/
$ ls
docker  primeiro-volume  segundo-volume
$
```

- Feito isso vamos validar acessando os containers e conferir se os dados estão persistindo



▼ Volumes na prática

- *Quais são as principais características de um Volume?*
  - **Volumes possuem ciclo de vida dependentes de Pods e independentes de containers.**

▼ Validando a definição de volumes

```
apiVersion: v1
kind: Pod
metadata:
  name: um-pod-qualquer
spec:
  containers:
    - name: nginx-container
      image: nginx:latest
      volumeMounts:
        - mountPath: /pasta-de-arquivos
          name: volume-pod
  volumes:
    - name: volume-pod
      hostPath:
        path: /C/Users/Daniel/Desktop/uma-pasta-no-host
        type: Directory
```

Qual será o resultado produzido pelo arquivo?

- **Caso a pasta `/C/Users/Daniel/Desktop/uma-pasta-no-host` exista no host, um volume chamado `volume-pod` será criado e montado na pasta `pasta-de-arquivos` dentro do container do Pod.**

▼ Kubernetes on the CLOUD

Neste exemplo utilizamos o Google Cloud Plataform (GCP) para criação do nosso cluster kubernetes, e explicação de alguns assuntos no qual fazem mais sentido a explicação em um ambiente cloud que tende a ser mais comum a utilização dessa infraestrutura.

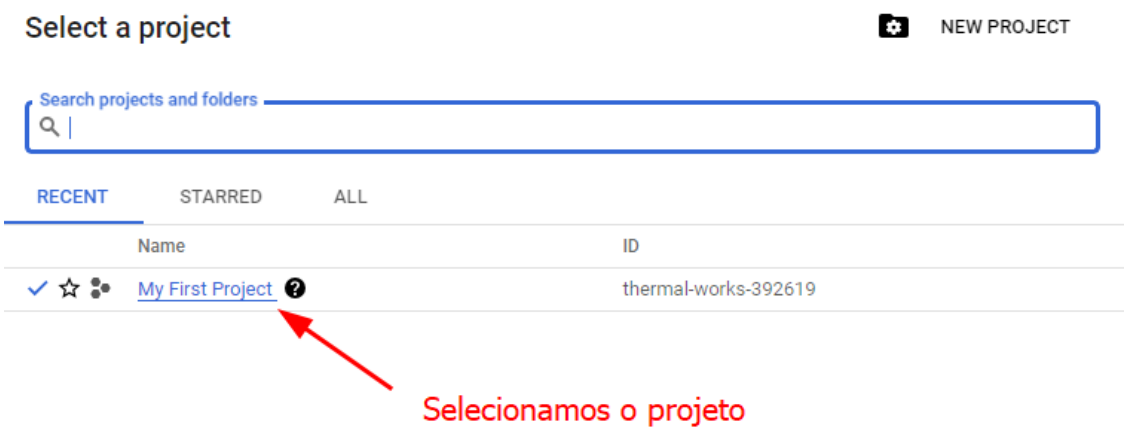
▼ GKE

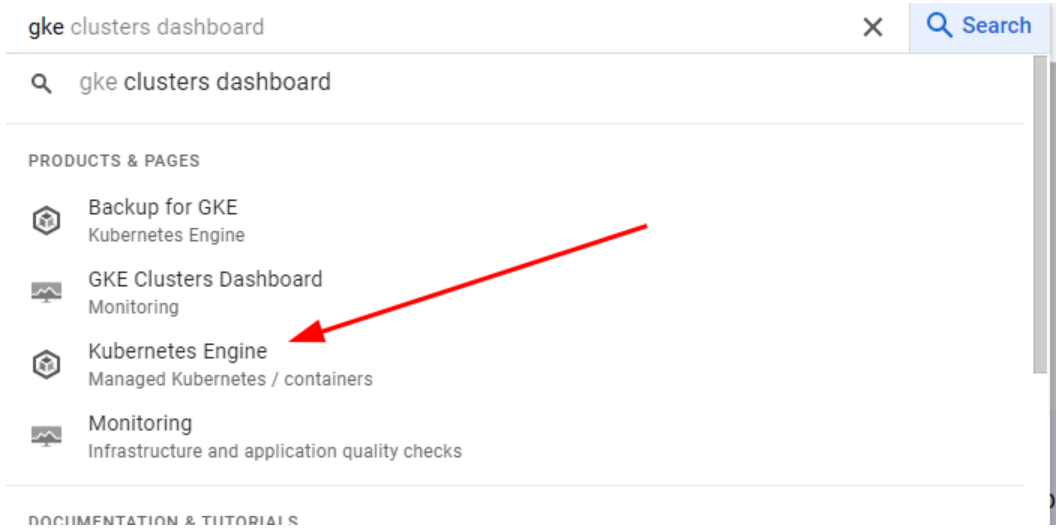
- O GCP é uma plataforma de computação em nuvem que oferece diversos serviços e recursos para empresas.
- É importante saber que, ao utilizar a Google Cloud, podem surgir cobranças que podem ser significativas, dependendo da intensidade de uso.
- O GCP possui um programa de gratuidade, liberando creditos para utilização gratuita

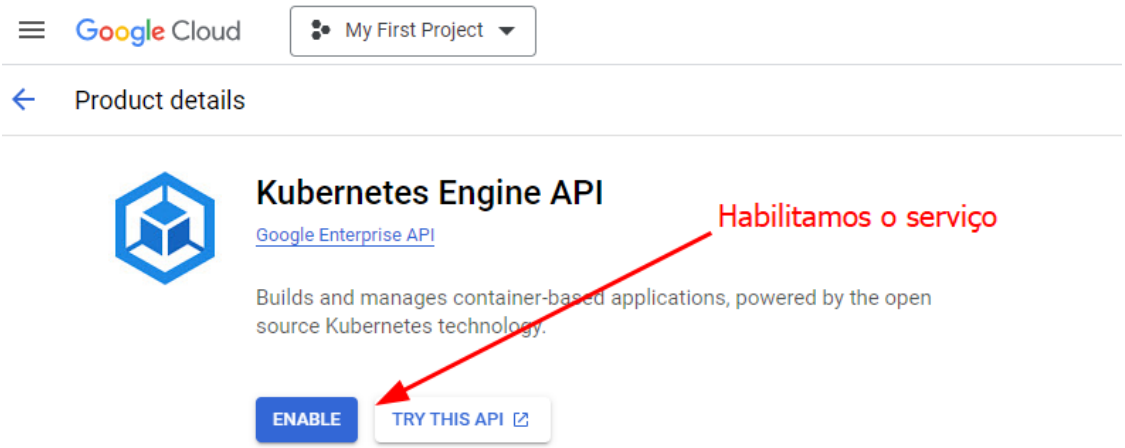
Programa gratuito do Google Cloud

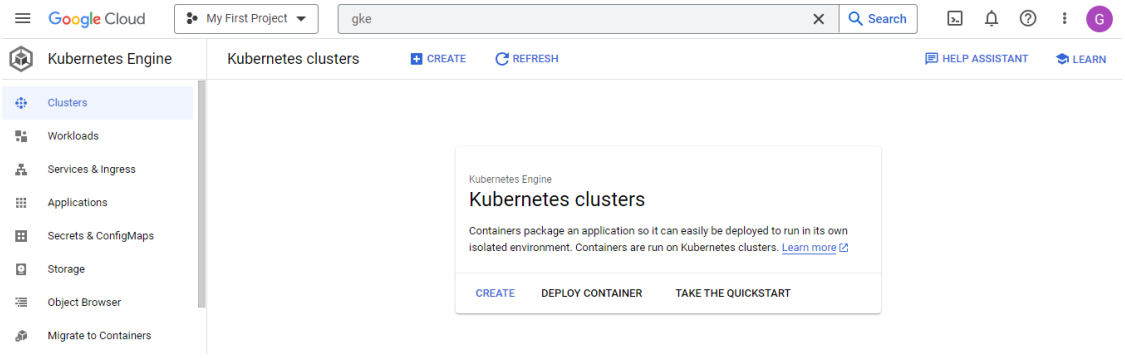
https://cloud.google.com/free/docs/free-cloud-features?hl=pt-br#kubernetes-engine

- No caso vamos criar um cluster em um ambiente em cloud
- Apenas para ciência, quaisquer recurso criado em um ambiente cloud poderá haver custos, portanto certifique-se de estar usando tiers gratuitos, ou esteja ciente dos custos, o foco do projeto não é cloud, mas para demonstração das funcionalidades do kubernetes utilizaremos este ambiente cloud para demonstração
- No GCP, estando em um projeto, vamos pesquisar por GKE (Google Kubernetes Engine), e vamos seguir com a criação

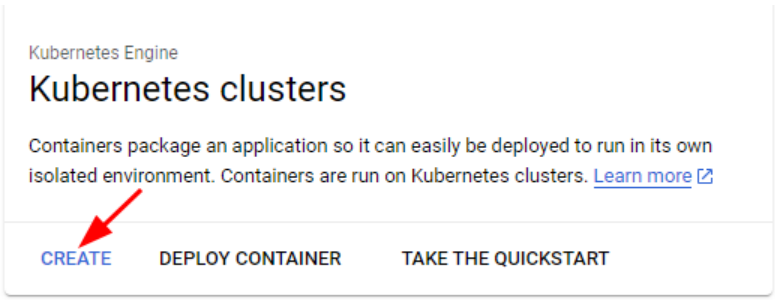


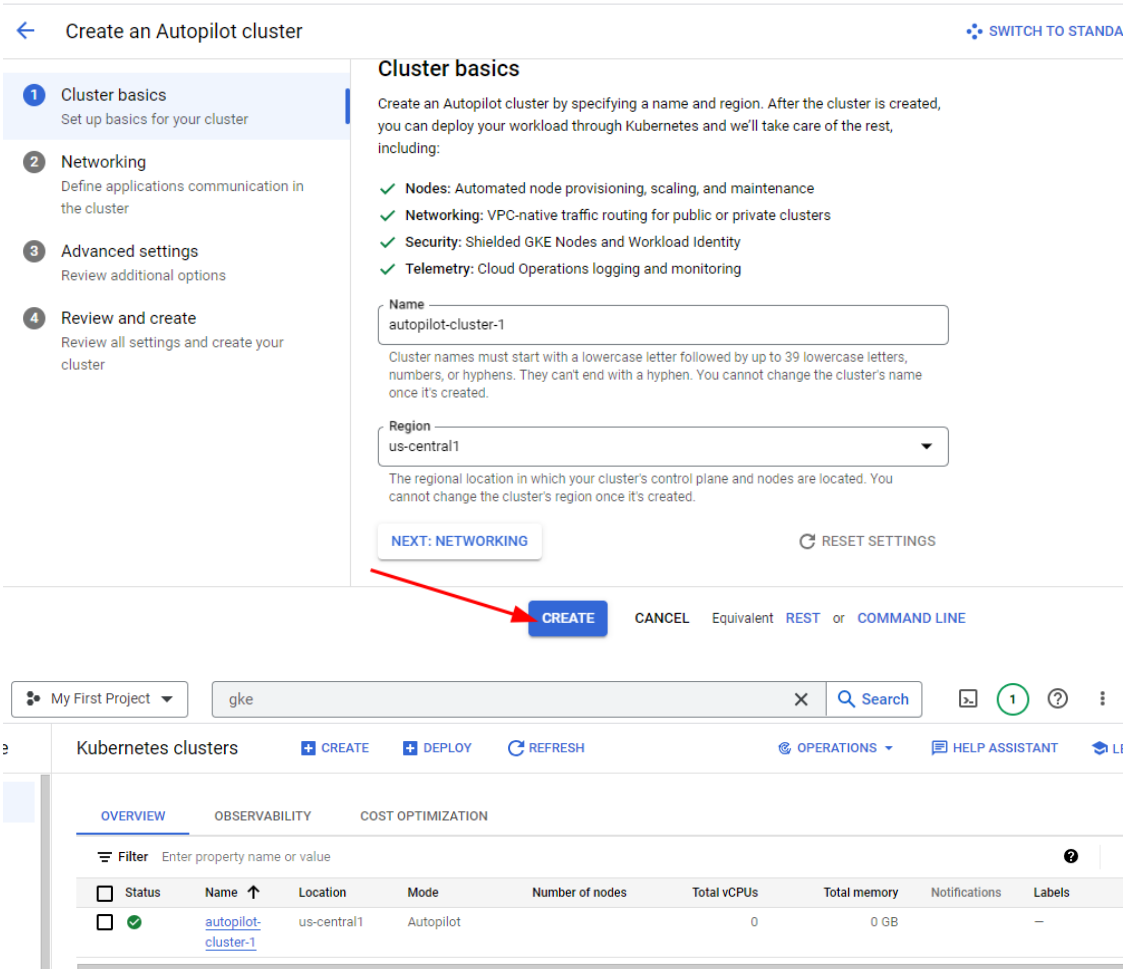




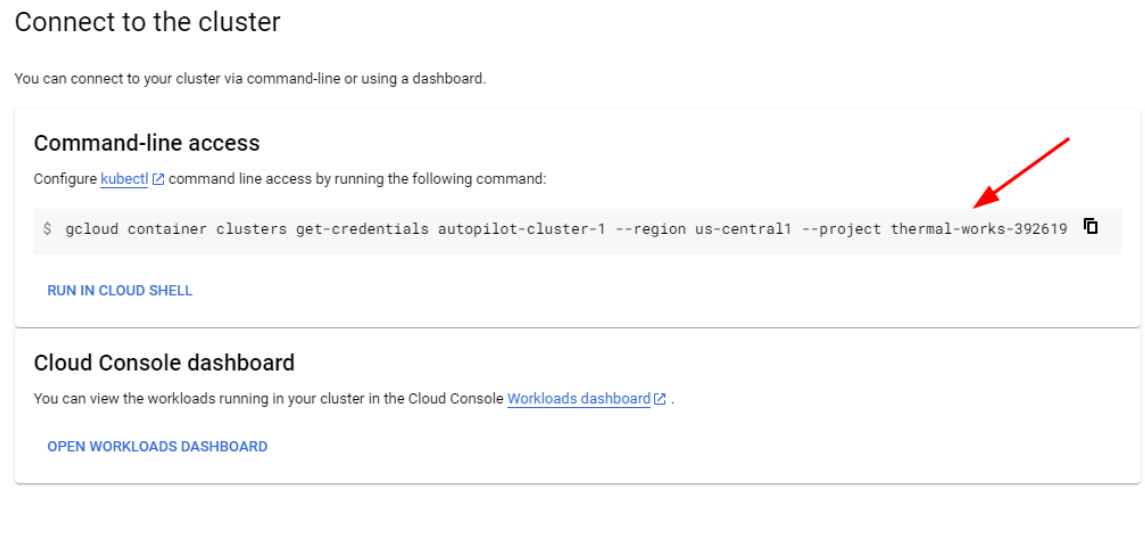


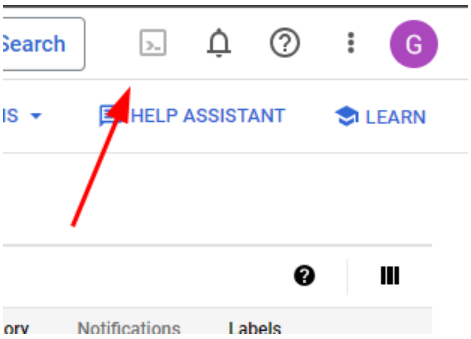
- Podemos criar um cluster menos robusto, no caso vamos criar um cluster padrão conforme orientações do console

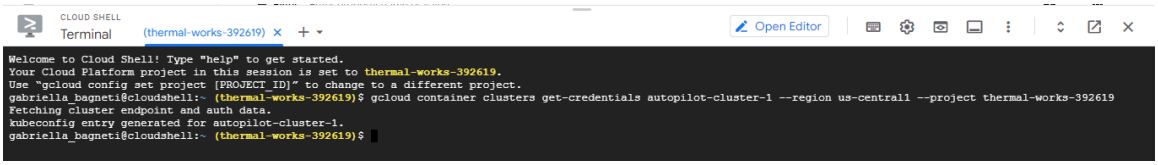




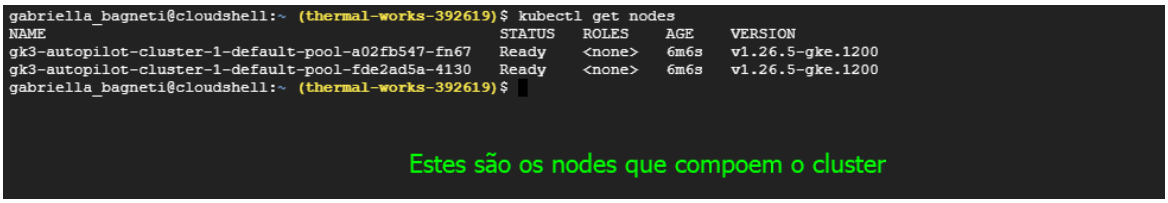
- Dessa forma podemos efetuar um acesso ao cluster através do proprio shell do google







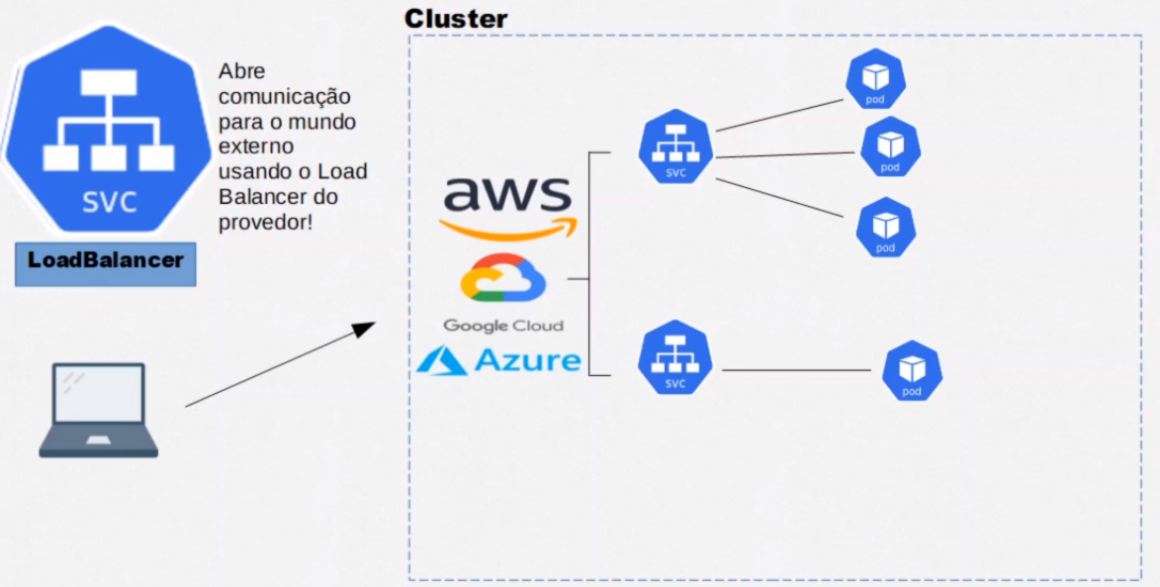
- Vamos validar a quantidade nós



- Dessa forma conseguimos ter todas as funcionalidades do modelo on premises, porém por estar em um ambiente cloud, a escalabilidade horizontal e vertical de recursos, tendem a ser superior

▼ **Criando o Load Balancer**

- O recurso loadbalancer no K8s nada mais é do que um cluster IP que permite a comunicação entre uma maquina do mundo externo e os nossos pods, porém o mesmo é integrado automaticamente a um load balancer da AWS, Azure ou GCP



- Vamos criar um pod padrão com nginx no GKE

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-1
  labels:
    app: primeiro-pod
spec:
  containers:
    - name: container-pod-1
      image: nginx:latest
      ports:
        - containerPort: 80
```

  - No caso vamos criar a declaração através da propria cloud

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ cat > pod-nginx.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-1
  labels:
    app: primeiro-pod
spec:
  containers:
    - name: container-pod-1
      image: nginx:latest
      ports:
        - containerPort: 80
^C
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ ls
pod-nginx.yaml  README-cloudshell.txt
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$
```

  - Agora vamos subir o pod

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
pod-1   1/1     Running   0          4m28s
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$
```

- Após subirmos o pod vamos criar o loadbalancer:

```
apiVersion: v1
kind: Service
metadata:
  name: svc-pod-1-loadbalancer
spec:
  type: LoadBalancer
  ports:
    - port: 80
      nodePort: 30000
  selector:
    app: primeiro-pod
```

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ cat > svc-lb.yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-pod-1-loadbalancer
spec:
  type: LoadBalancer
  ports:
    - port: 80
      nodePort: 30000
  selector:
    app: primeiro-pod
```

- Vamos subir está definição

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl apply -f svc-lb.yaml
service/svc-pod-1-loadbalancer created
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl get svc
NAME                     TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
kubernetes               ClusterIP      10.8.128.1    <none>        443/TCP        26m
svc-pod-1-loadbalancer   LoadBalancer   10.8.130.64   <pending>     80:30000/TCP   19s
```

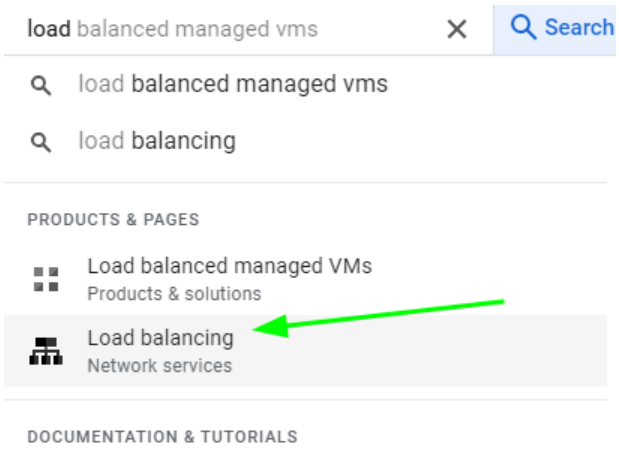
- Podemos visualizar que o LoadBalancer está sendo criado

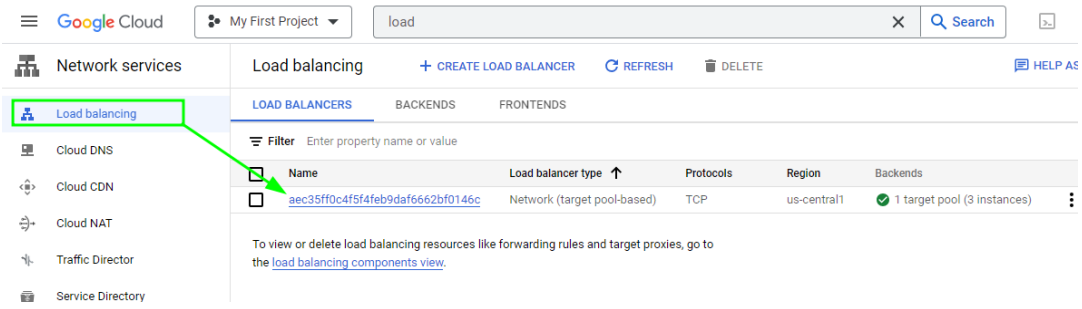
```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl get svc
NAME                     TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
kubernetes               ClusterIP      10.8.128.1    <none>        443/TCP        27m
svc-pod-1-loadbalancer   LoadBalancer   10.8.130.64   34.66.37.1    80:30000/TCP   72s
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$
```

- Após ser criado podemos podemos visualizar que gerou um IP externo

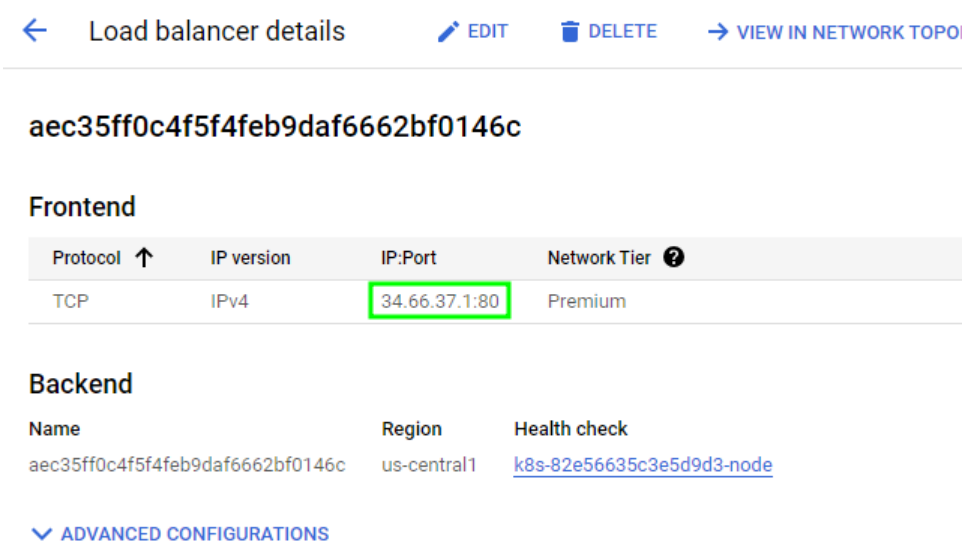
```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl get svc
NAME                     TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
kubernetes               ClusterIP      10.8.128.1    <none>        443/TCP        27m
svc-pod-1-loadbalancer   LoadBalancer   10.8.130.64   34.66.37.1    80:30000/TCP   72s
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$
```

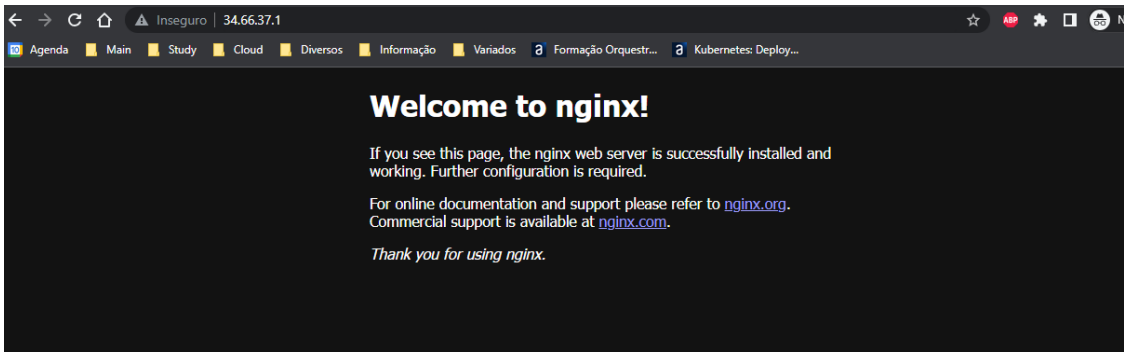
  - E podemos visualizar em load Balancer que um recurso foi criado no GCP





- Assim podemos acessar através do IP externo

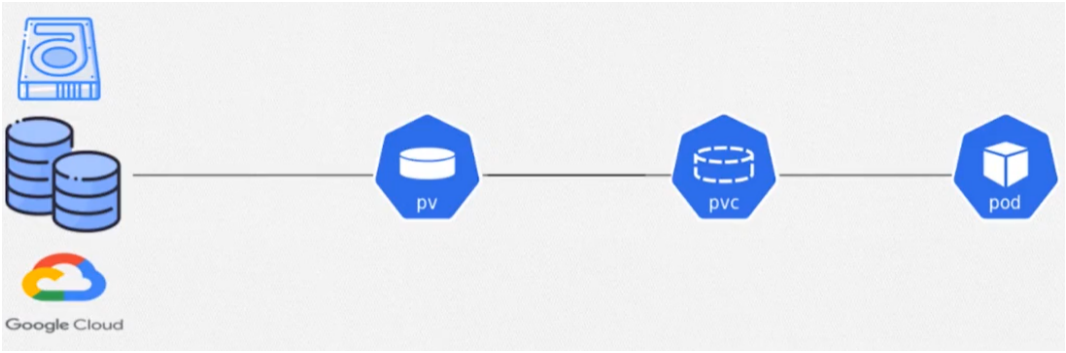




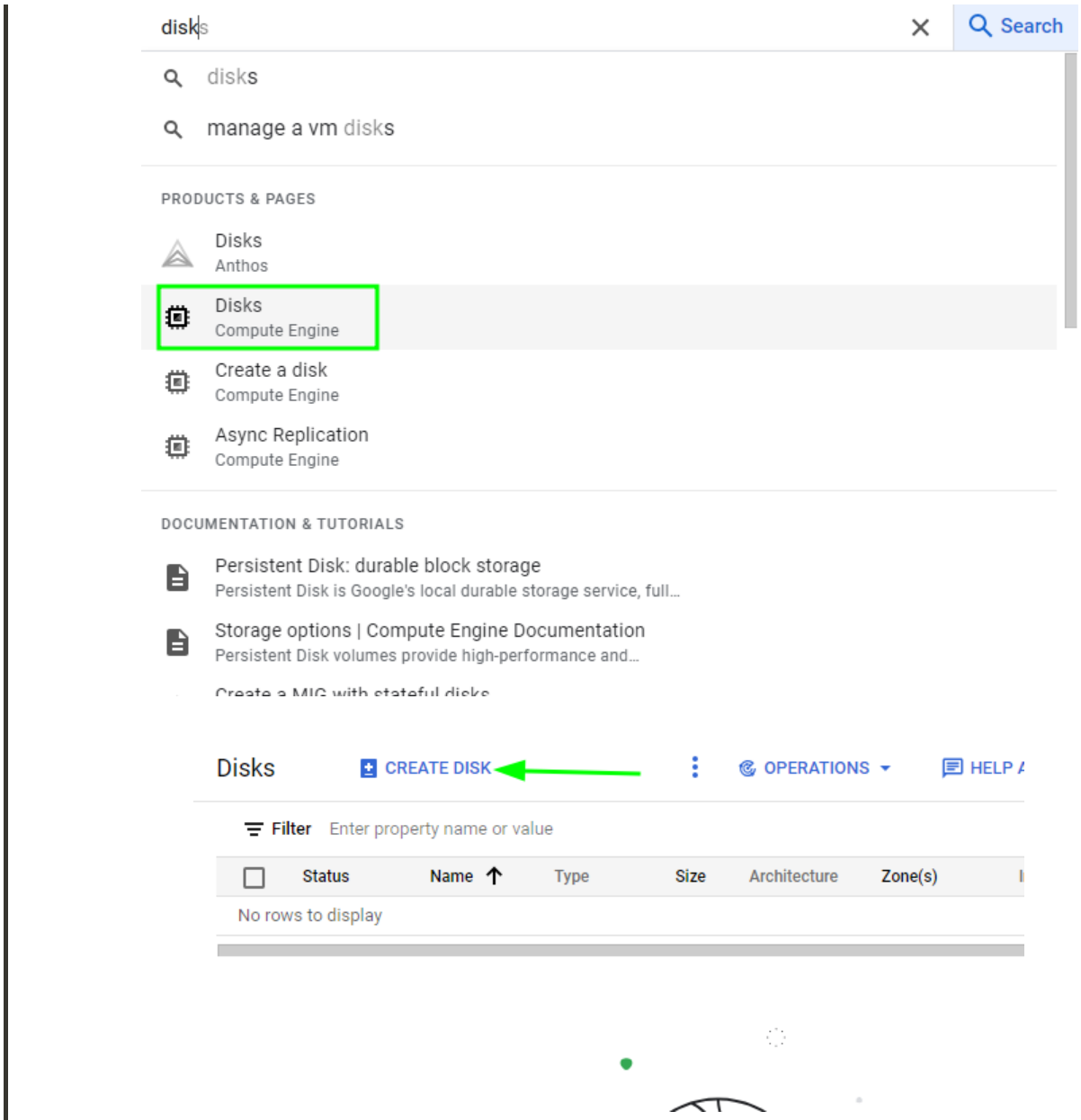
Persistencia de dados em Cloud

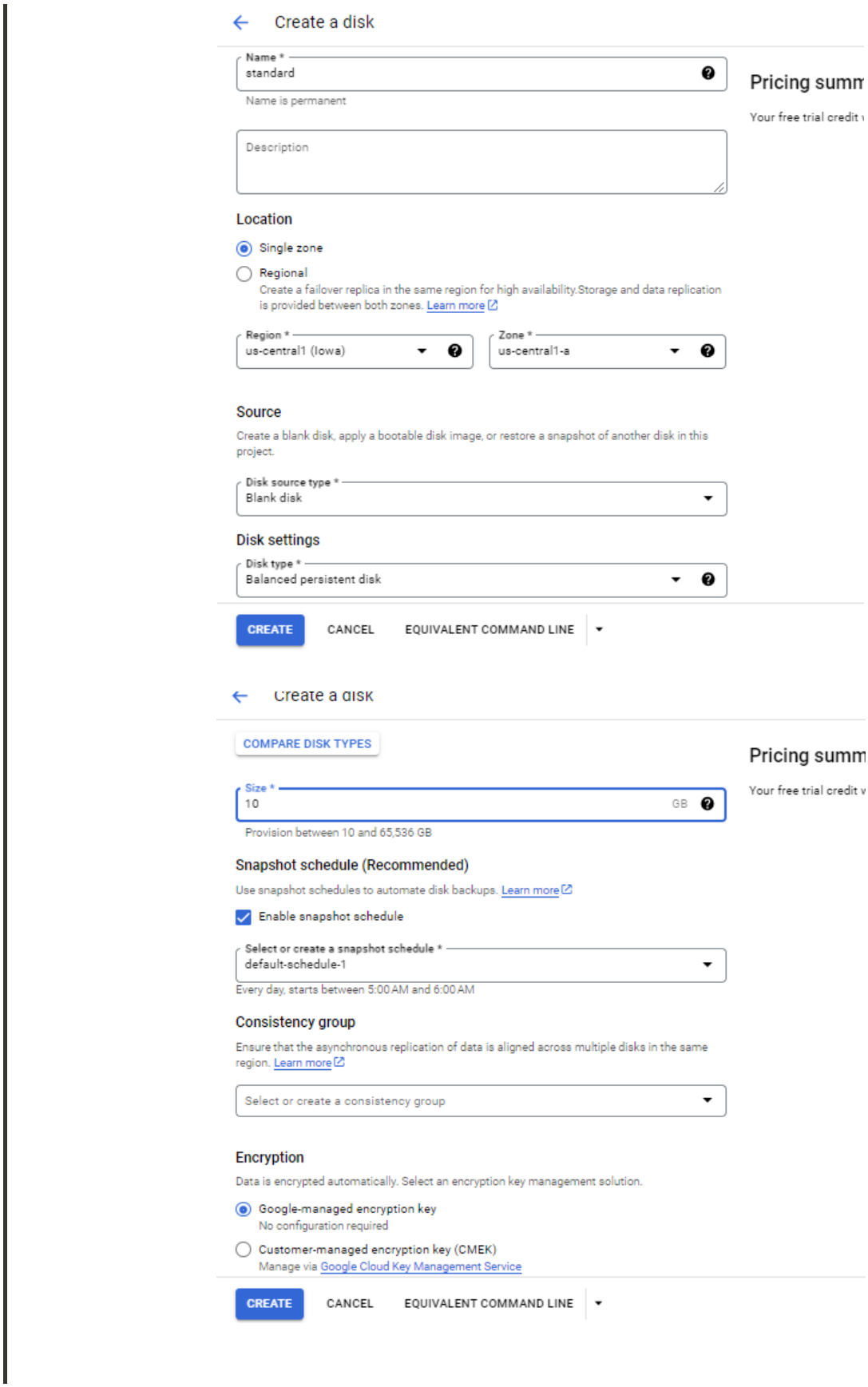
▼ Persistência com PersistentVolumes

- Vamos entender o funcionamento de um PV e PVC em cloud provider

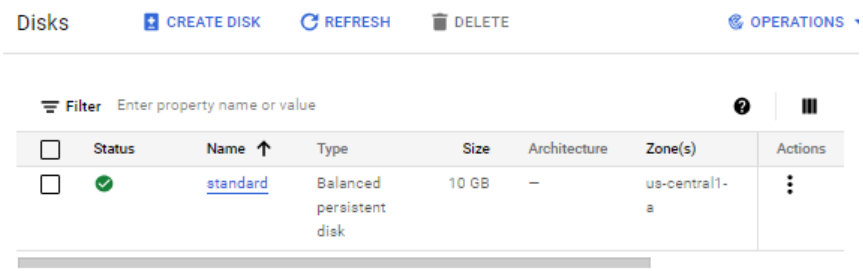


- Vamos criar um disco no GCP, e apartir dele vamos criar um PV e PVC


disks
Search
disks
manage a vm disks
PRODUCTS & PAGES
Disks
Anthos
Disks
Compute Engine
Create a disk
Compute Engine
Async Replication
Compute Engine
DOCUMENTATION & TUTORIALS
Persistent Disk: durable block storage
Persistent Disk is Google's local durable storage service, full...
Storage options | Compute Engine Documentation
Persistent Disk volumes provide high-performance and...
Disks
CREATE DISK
OPERATIONS
HELP
Filter
Enter property name or value
Status
Name
Type
Size
Architecture
Zone(s)
No rows to display


Create a disk
Name *
standard
Name is permanent
Description
Location
Single zone
Regional
Create a failover replica in the same region for high availability.Storage and data replication is provided between both zones. Learn more
Region *
us-central1 (Iowa)
Zone *
us-central1-a
Source
Create a blank disk, apply a bootable disk image, or restore a snapshot of another disk in this project.
Disk source type *
Blank disk
Disk settings
Disk type *
Balanced persistent disk
CREATE
CANCEL
EQUIVALENT COMMAND LINE
Pricing summ
Your free trial credit
Create a disk
COMPARE DISK TYPES
Size *
10
GB
Provision between 10 and 65,536 GB
Snapshot schedule (Recommended)
Use snapshot schedules to automate disk backups. Learn more
Enable snapshot schedule
Select or create a snapshot schedule *
default-schedule-1
Every day, starts between 5:00 AM and 6:00 AM
Consistency group
Ensure that the asynchronous replication of data is aligned across multiple disks in the same region. Learn more
Select or create a consistency group
Encryption
Data is encrypted automatically. Select an encryption key management solution.
Google-managed encryption key
No configuration required
Customer-managed encryption key (CMEK)
Manage via Google Cloud Key Management Service
CREATE
CANCEL
EQUIVALENT COMMAND LINE
Pricing summ
Your free trial credit

- Será um disco padrão de 10gb na mesma zona de nosso cluster



- Feito isso vamos criar um PV no nosso cluster denominado pv.yaml
  - Utilizamos as documentações como base:

Volumes

Os arquivos em disco em um contêiner são efêmeros, o que apresenta alguns problemas para aplicações não triviais quando executadas em contêineres. Um problema é a perda de

https://kubernetes.io/pt-br/docs/concepts/storage/volumes/

Persistent Volumes

This document describes persistent volumes in Kubernetes. Familiarity with volumes is suggested. Introduction Managing storage is a distinct problem from managing compute instances.

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

Storage options | Compute Engine Documentation | Google Cloud

https://cloud.google.com/compute/docs/disks

```
apiVersion: v1
kind: PersistentVolume
metadata:
    name: pv-1
spec:
    capacity:
        storage: 10Gi
    accessModes:
        - ReadWriteOnce
    gcePersistentDisk:
        pdName: standard
    storageClassName: standard
```

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ cat pv-disk.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
    name: pv-1
spec:
    capacity:
        storage: 10Gi
    accessModes:
        - ReadWriteOnce
    gcePersistentDisk:
        pdName: standard
    storageClassName: standard
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$
```

- Feito isso vamos subir o recurso, e em seguida validar o mesmo

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl apply -f pv-disk.yaml
persistentvolume/pv-1 created
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
pv-1   10Gi       RWO            Retain           Available           standard               34s
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$
```

- O **retain** na coluna **RECLAIM** indica que embora não haja nenhum claim o pv irá permanecer, no caso em claim não há nada
- Dessa forma precisaremos criar um pvc.yaml
    - o PVC precisa ter o mesmo modo de acesso, a mesma capacidade, e a mesma tier

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
    name: pvc-1
spec:
    accessModes:
        - ReadWriteOnce
    resources:
        requests:
            storage: 10Gi
    storageClassName: standard
```

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ cat>pvc.yam
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
    name: pvc-1
spec:
    accessModes:
        - ReadWriteOnce
    resources:
        requests:
            storage: 10Gi
    storageClassName: standard

^C
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$
```

- Feito isso subimos o PVC

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl apply -f  pvc.yaml
persistentvolumeclaim/pvc-1 created
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl get pvc
NAME    STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
pvc-1   Bound    pv-1     10Gi       RWO            standard       7s
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$
```

- Vamos validar o pvc e podemos verificque que o mesmo está linkado com o volume pv-1

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM           STORAGECLASS   REASON   AGE
pv-1   10Gi       RWO            Retain           Bound    default/pvc-1   standard                5m23s
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$
```

- Feito isso vamos criar um pod para poder ter esse acesso

```
apiVersion: v1
kind: Pod
metadata:
    name: pod-pv
spec:
    containers:
        - name: nginx-container
          image: nginx:latest
          volumeMounts:
              - mountPath: /volume-dentro-do-container
                name: primeiro-pv
    volumes:
        - name: primeiro-pv
          persistentVolumeClaim:
              claimName: pvc-1
```

  - Perceba que é quase a mesma ideia do hostPath a diferença é que para chegar em nosso PV precisa passar pelo PVC portanto o mesmo deve estar declaradado no YAML

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ cat>pod-pvc.yaml
apiVersion: v1
kind: Pod
metadata:
    name: pod-pv
spec:
    containers:
        - name: nginx-container
          image: nginx:latest
          volumeMounts:
              - mountPath: /volume-dentro-do-container
                name: primeiro-pv
    volumes:
        - name: primeiro-pv
          persistentVolumeClaim:
              claimName: pvc-1
^C
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$
```

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
pod-1    1/1     Running   0          40m
pod-pv   1/1     Running   0          4m42s
```

- Feito isso vamos aplicar o pod, vamos validar o describe e podemos ver que está attached com o PVC

```
      memory:             2Gi
    Requests:
      cpu:                500m
      ephemeral-storage:  1Gi
      memory:             2Gi
    Environment:          <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-xztb5 (ro)
      /volume-dentro-do-container from primeiro-pv (rw)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  primeiro-pv:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:  pvc-1
    ReadOnly:   false
  kube-api-access-xztb5:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
```

- Vamos acessar o container, criar um arquivo, e vamos recriar o pod

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
pod-1    1/1     Running   0          40m
pod-pv   1/1     Running   0          4m42s
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl exec -it pod-pv -- bash
root@pod-pv:/# ls
bin   dev                  docker-entrypoint.sh  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint.d  etc                   lib   lib64  media   opt  root  sbin  sys  usr  volume-dentro-do-container
root@pod-pv:/# cd volume-dentro-do-container/
root@pod-pv:/volume-dentro-do-container# touch txt
root@pod-pv:/volume-dentro-do-container# ls
lost+found  txt
```

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl delete pod pod-pv
pod "pod-pv" deleted
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl apply -f pod-pvc.yaml
Warning: autopilot-default-resources-mutator:Autopilot updated Pod default/pod-pv: defaulted unsp
utopilot-defaults)
pod/pod-pv created
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl exec -it pod-pv -- bash
root@pod-pv:/# cd volume-dentro-do-container/
root@pod-pv:/volume-dentro-do-container# ls
lost+found  txt
```

- Feito isso temos agora um sistema que permite a persistencia dos dados independente da existencia dos pods
- **PersistentVolumes possuem ciclos de vida independentes de Pods.**
- **É necessário um PersistentVolumeClaim para acessar um PersistentVolume.**

▼ **Mais informações sobre PersistentVolumes**

Persistent Volumes

This document describes persistent volumes in Kubernetes. Familiarity with volumes is suggested. Introduction Managing storage is a distinct problem from managing compute instances.

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

▼ **Utilizando Storage Classes**

- O storage class permite criar volumes e discos dinamicamente quando fizer um bind com um pod

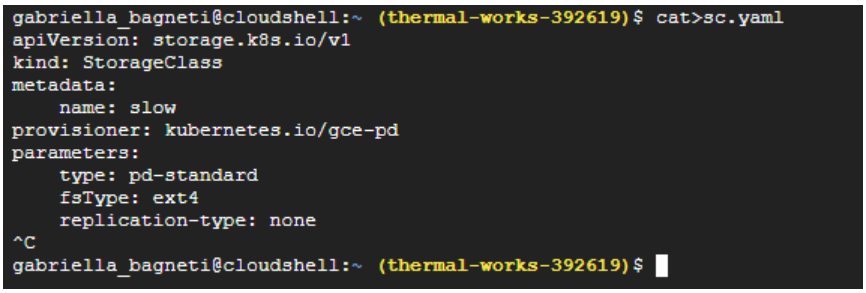
Storage Classes

This document describes the concept of a StorageClass in Kubernetes. Familiarity with volumes and persistent volumes is suggested. Introduction A StorageClass provides a way for

https://kubernetes.io/docs/concepts/storage/storage-classes/

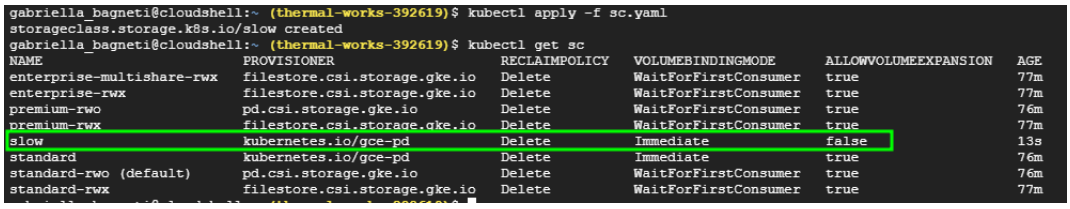
- Vamos usar a base para criar nosso arquivo storage class

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
    name: slow
provisioner: kubernetes.io/gce-pd
parameters:
    type: pd-standard
    fsType: ext4
    replication-type: none
```



  - Esses padrões foram criados de acordo com a documentação do referente ao GCP, dependendo da cloud alguns parametros podem mudar

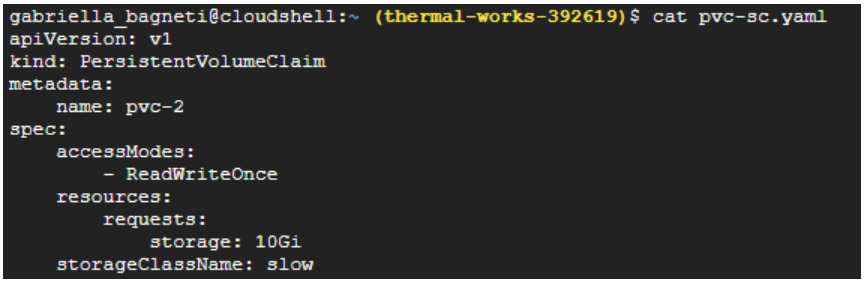
- Feito isso vamos criar o storage class em nosso ambiente kubernetes e cloud



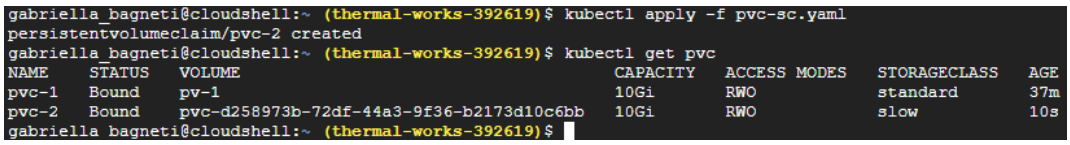
  - Alguns SCs são criados automaticamente pelo proprio GKE, isso ocorre por conta do node
  - O Standard que criamos durante a explicação dos pvs também aparece pois apresenta o que compoem um sc é pv + pvc

- Vamos criar um pvc-sc.yaml

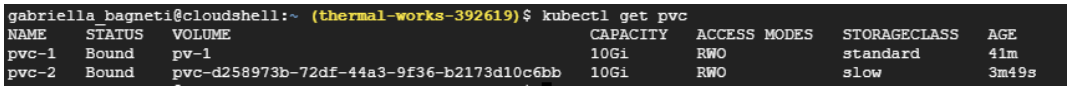
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
    name: pvc-2
spec:
    accessModes:
        - ReadWriteOnce
    resources:
        requests:
            storage: 10Gi
    storageClassName: slow
```

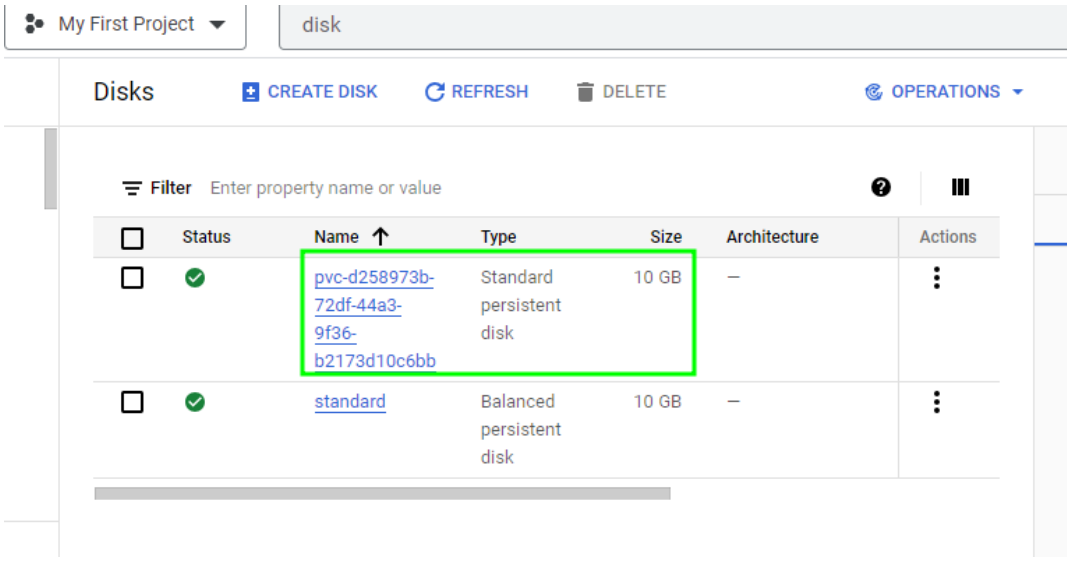


  - Em storage class name utilimamos o name do metadata do sc, no caso slow
  - Feito isso vamos aplicar nosso pvc



- Vamos validar nosso pvc e pv que foi criar automaticamente





- Dessa forma criamos dinamicamente o pv e o disco na cloud

- Vamos criar um novo pod (pod-sc)

```
apiVersion: v1
kind: Pod
metadata:
    name: pod-sc
spec:
    containers:
        - name: nginx-container
          image: nginx:latest
          volumeMounts:
              - mountPath: /volume-dentro-do-container
                name: primeiro-pv
    volumes:
        - name: primeiro-pv
          persistentVolumeClaim:
              claimName: pvc-2
```

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ cat>pv-sc.yaml
apiVersion: v1
kind: Pod
metadata:
    name: pod-sc
spec:
    containers:
        - name: nginx-container
          image: nginx:latest
          volumeMounts:
              - mountPath: /volume-dentro-do-container
                name: primeiro-pv
    volumes:
        - name: primeiro-pv
          persistentVolumeClaim:
              claimName: pvc-2
```

    - Especificando o pvc-2 em claimName
- Feito isso vamos aplicar novamente

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl apply -f pv-sc.yaml
Warning: autopilot-default-resources-mutator:Autopilot updated Pod default/pod-sc:
utopilot-defaults)
pod/pod-sc created
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl get pods
NAME     READY   STATUS              RESTARTS   AGE
pod-1    1/1     Running             0          73m
pod-pv   1/1     Running             0          30m
pod-sc   0/1     ContainerCreating   0          7s
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
pod-1    1/1     Running   0          73m
pod-pv   1/1     Running   0          30m
pod-sc   1/1     Running   0          18s
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$
```

- Dessa forma podemos acessar o pod, e realizar a criação do arquivo

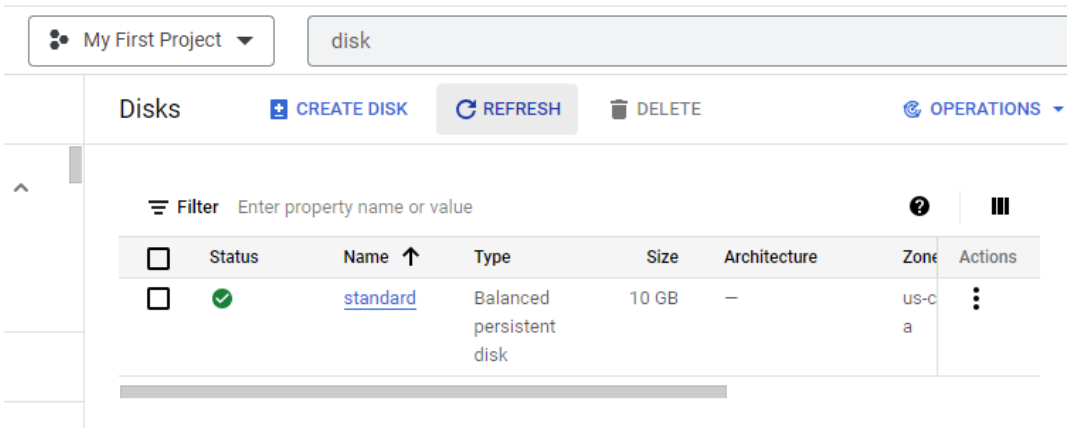
```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
pod-1    1/1     Running   0          75m
pod-pv   1/1     Running   0          32m
pod-sc   1/1     Running   0          2m16s
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl exec -it pod-sc -- bash
root@pod-sc:/# cd volume-dentro-do-container/
root@pod-sc:/volume-dentro-do-container# touch teste.txt
```

- Vamos recriar o pod, e acessar novamente, e podemos ver que os dados persistem

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl apply -f pv-sc.yaml
Warning: autopilot-default-resources-mutator:Autopilot updated Pod default/pod-sc: defau
utopilot-defaults)
pod/pod-sc created
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl exec -it pod-sc -- bash
root@pod-sc:/# cd volume-dentro-do-container/
root@pod-sc:/volume-dentro-do-container# ls
lost+found  teste.txt
root@pod-sc:/volume-dentro-do-container#
```

- Vamos deletar todos os pvc e scs e os pv serão excluidos dinamicamente, ou seja o disco criado será removido

```
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl delete pvc --all
persistentvolumeclaim "pvc-1" deleted
persistentvolumeclaim "pvc-2" deleted
^Cgabriella_bagneti@cloudshell:~ (thermal-works-392619)$ kubectl delete sc slow
storageclass.storage.k8s.io "slow" deleted
gabriella_bagneti@cloudshell:~ (thermal-works-392619)$
```



- ***Storage Classes fornecem dinamismo para criação de PersistentVolumes conforme demanda.***

▼ **Statefulsets**

▼ **Entendendo PV e PVC**

- Existem dois conceitos principais no Kubernetes relacionados à persistência de dados: PV (Persistent Volume) e PVC (Persistent Volume Claim). Vamos dar uma breve descrição de cada um deles:
  - **Persistent Volume (PV):**
    Um Persistent Volume é um recurso abstrato no Kubernetes que representa uma unidade de armazenamento físico, como um disco rígido ou um volume de rede. Ele fornece uma maneira de provisionar e consumir armazenamento persistente em um cluster Kubernetes. O PV é criado separadamente do pod que o utiliza e existe independentemente do pod. Isso permite que os PVs sejam reutilizados por vários pods durante o ciclo de vida de um cluster Kubernetes.
  - **Persistent Volume Claim (PVC):**
    Um Persistent Volume Claim é uma solicitação de armazenamento feita por um pod. É um objeto que descreve o tipo de armazenamento necessário, como capacidade, modo de acesso (leitura/escrita), tipo de armazenamento, etc. Um PVC é vinculado a um PV correspondente com base em critérios como capacidade e modo de acesso. Uma vez que um PVC é vinculado a um PV, o armazenamento persistente associado ao PV é montado no pod que fez a solicitação, permitindo assim que o pod acesse e use o armazenamento persistente conforme necessário.
- Em resumo, o PV é um recurso que representa o armazenamento físico disponível em um cluster Kubernetes, enquanto o PVC é uma solicitação feita por um pod para usar um determinado tipo de armazenamento persistente. O PVC é vinculado a um PV correspondente, permitindo que o pod acesse e utilize o armazenamento persistente fornecido pelo PV. Esses conceitos são essenciais para a persistência de dados em aplicativos implantados no Kubernetes, permitindo que os dados persistam mesmo que os pods sejam reiniciados ou movidos para nós diferentes dentro do cluster.

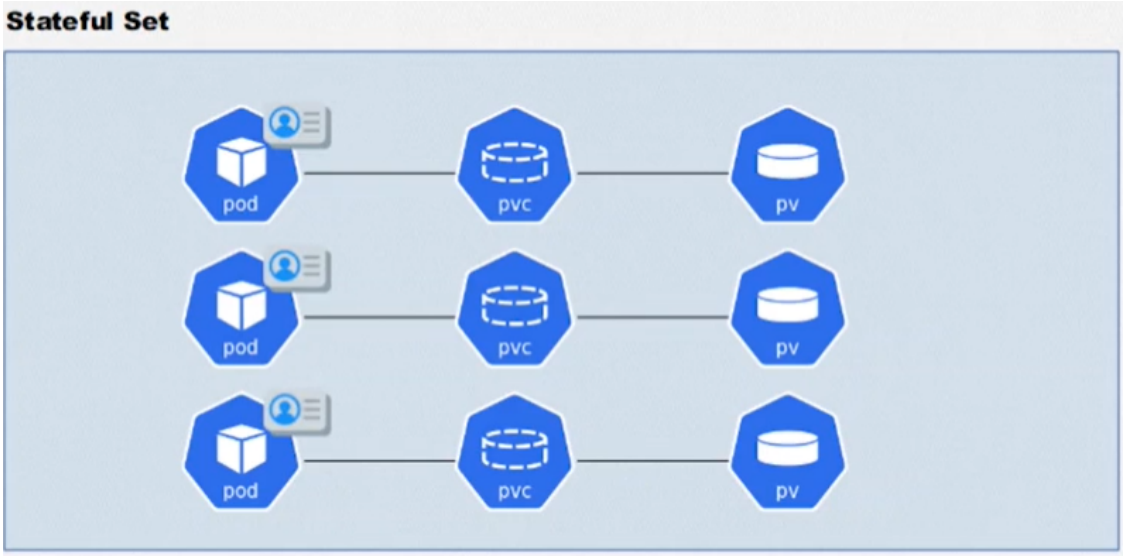
▼ **Entendendo Storage Classes**

- Storage Classes são objetos no Kubernetes que fornecem uma abstração para o provisionamento dinâmico de volumes de armazenamento. Eles permitem que os administradores de cluster definam diferentes classes de armazenamento com características específicas, como tipo de armazenamento (por exemplo, disco rígido, SSD), desempenho, replicação e outras propriedades.

- As Storage Classes são usadas em conjunto com os Persistent Volume Claims (PVCs) para solicitar armazenamento persistente em um cluster Kubernetes. Quando um PVC é criado e vinculado a uma Storage Class, o Kubernetes usa essa classe para provisionar um Persistent Volume (PV) dinamicamente, em vez de ter que criar manualmente um PV separado.
- Cada Storage Class possui um provisionador específico associado a ele. O provisionador é responsável por criar o volume de armazenamento real (como criar um disco rígido em um provedor de nuvem) e vinculá-lo ao PVC solicitado. O provisionador determina como o armazenamento será provisionado e configurado, com base nas propriedades definidas na Storage Class.
- As Storage Classes fornecem flexibilidade e escalabilidade para o armazenamento persistente no Kubernetes. Os desenvolvedores e administradores podem definir várias classes de armazenamento com diferentes características e níveis de desempenho para atender às necessidades específicas de suas aplicações. Isso permite a alocação sob demanda de volumes de armazenamento e simplifica o processo de solicitação e gerenciamento de armazenamento persistente em um cluster Kubernetes.

▼ **Conhecendo Statefulsets**

- O Stateful Set se assemelha o Deployment porém mantem o pod e arquivo no mesmo estado em conjunto com o pv e pvc
- Basicamente quando criamos um pod em um stateful set criamos um pvc e pv para cada pod, e cada pod terá uma indentificação ordinal, e caso o pod falhe, o mesmo terá a mesma identificação ao ser recriado



- Vamos criar um stateful set para o sistemas-noticias

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sistema-noticias-statefulset
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: sistema-noticias
```

```
    spec:
      containers:
        - name: sistema-noticias-container
          image: aluracursos/sistema-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: sistema-configmap
  selector:
    matchLabels:
      app: sistema-noticias
  serviceName: svc-sistema-noticias
```

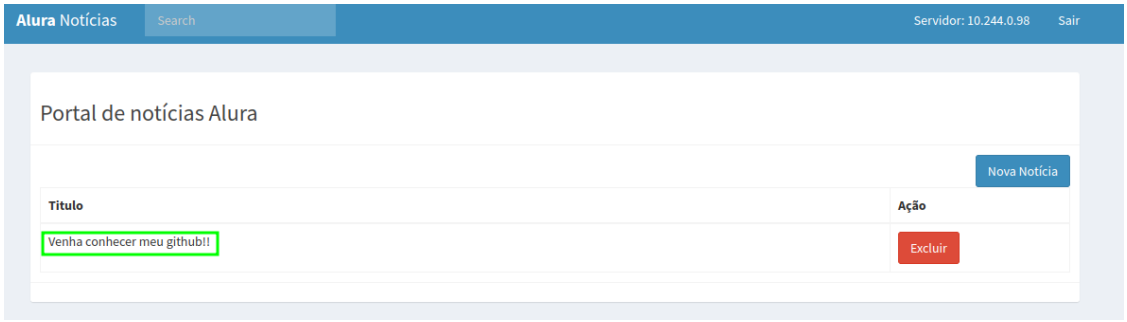
```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ nano sistema-noticias-sts.yaml
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ cat sistema-noticias-sts.yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sistema-noticias-statefulset
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: sistema-noticias
    spec:
      containers:
        - name: sistema-noticias-container
          image: aluracursos/sistema-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: sistema-configmap
  selector:
    matchLabels:
      app: sistema-noticias
  serviceName: svc-sistema-noticias
```

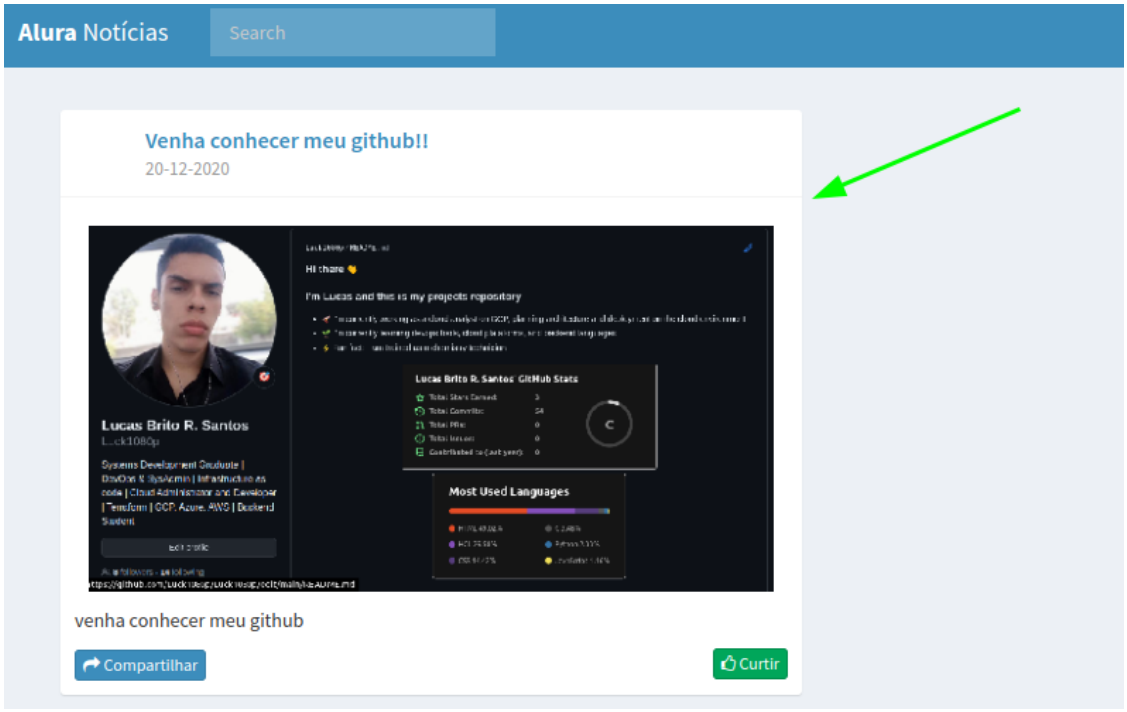
- Alteramos o kind, e name, e também adicionamos o ServiceName, que deve estar contido na declaração

- Vamos aplicar o objeto e vamos remover o deployment o sistema-noticias

```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl apply -f sistema-noticias-sts.yaml
statefulset.apps/sistema-noticias-statefulset created
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl get statefulset
NAME                          READY   AGE
sistema-noticias-statefulset   1/1     7s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl get pods
NAME                                        READY   STATUS    RESTARTS      AGE
db-noticias-deployment-5c87d7d876-xj7x2     1/1     Running   3 (11m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-d2rb2 1/1     Running   3 (11m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-j677f 1/1     Running   3 (11m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-qq4p8 1/1     Running   3 (11m ago)   2d22h
sistema-noticias-deployment-67c4474df4-hjhqx 1/1    Running   0             117s
sistema-noticias-statefulset-0              1/1     Running   0             12s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl get deploy
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
db-noticias-deployment       1/1     1            1           2d22h
portal-noticias-deployment   3/3     3            3           2d22h
sistema-noticias-deployment  1/1     1            1           2m17s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl delete deploy sistema-noticias-deployme
nt
deployment.apps "sistema-noticias-deployment" deleted
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$
```

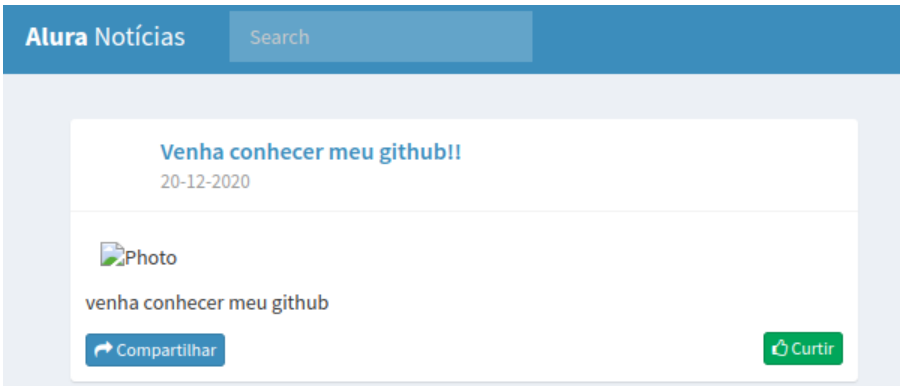
- Feito isso vamos acessar a aplicação, e vamos add uma noticia

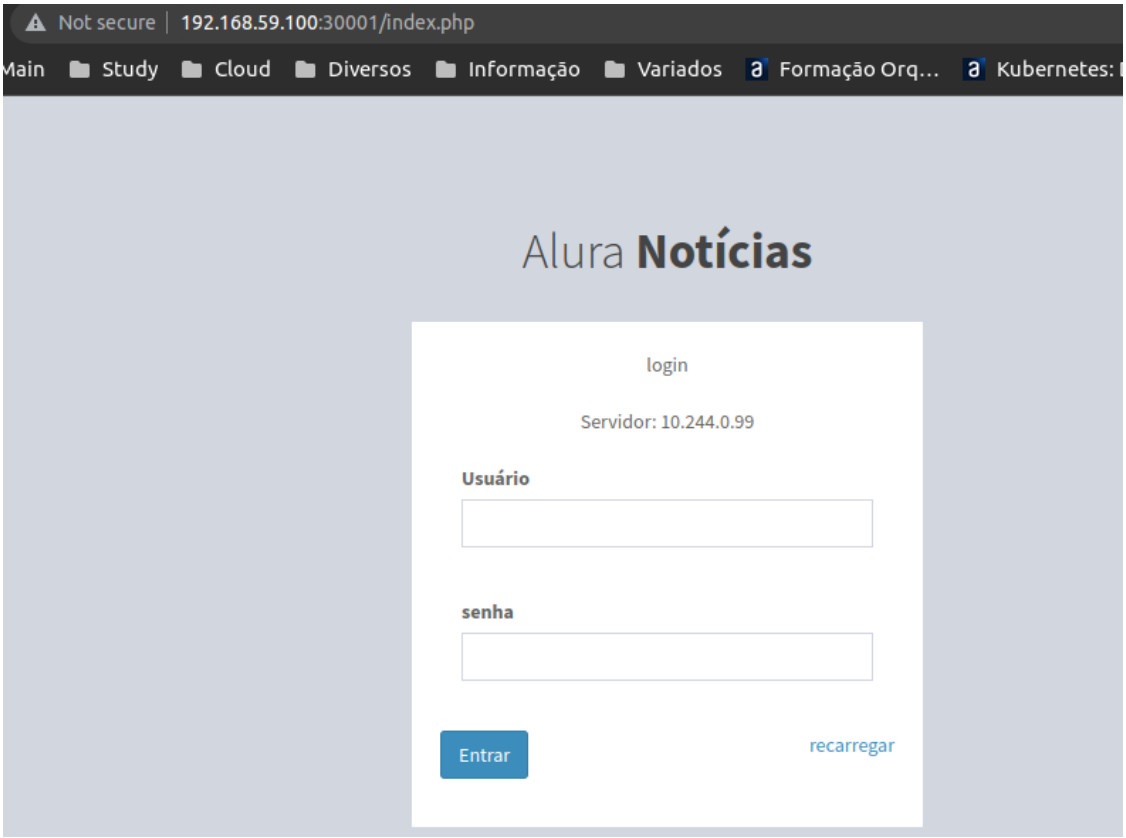




- Vamos remover o pod do statefulset e vamos recarregar o ambiente

```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl get pods
NAME                                        READY   STATUS    RESTARTS      AGE
db-noticias-deployment-5c87d7d876-xj7x2     1/1     Running   3 (19m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-d2rb2 1/1     Running   3 (19m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-j677f 1/1     Running   3 (19m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-qq4p8 1/1     Running   3 (19m ago)   2d22h
sistema-noticias-statefulset-0              1/1     Running   0             8m1s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl delete pod sistema-noticias-statefulset
-0
pod "sistema-noticias-statefulset-0" deleted
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl get pods
NAME                                        READY   STATUS              RESTARTS      AGE
db-noticias-deployment-5c87d7d876-xj7x2     1/1     Running             3 (19m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-d2rb2 1/1     Running             3 (19m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-j677f 1/1     Running             3 (19m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-qq4p8 1/1     Running             3 (19m ago)   2d22h
sistema-noticias-statefulset-0              0/1     ContainerCreating   0             2s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl get pods
NAME                                        READY   STATUS    RESTARTS      AGE
db-noticias-deployment-5c87d7d876-xj7x2     1/1     Running   3 (19m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-d2rb2 1/1     Running   3 (19m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-j677f 1/1     Running   3 (19m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-qq4p8 1/1     Running   3 (19m ago)   2d22h
sistema-noticias-statefulset-0              1/1     Running   0             12s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$
```





- Notamos que a imagem sumiu e a nossa sessão que estavamo conectados caiu também, no caso não temos um **pv** e **pvc**, ambos devem estar integrados na declaração do statefulset para isso utilizaremos o **Storage Class**
- **StatefulSets podem ser usados quando estados devem ser persistidos.**
- **StatefulSets usam PersistentVolumes e PersistentVolumeClaims para persistência de dados.**

▼ **Utilizando o Statefulsets**

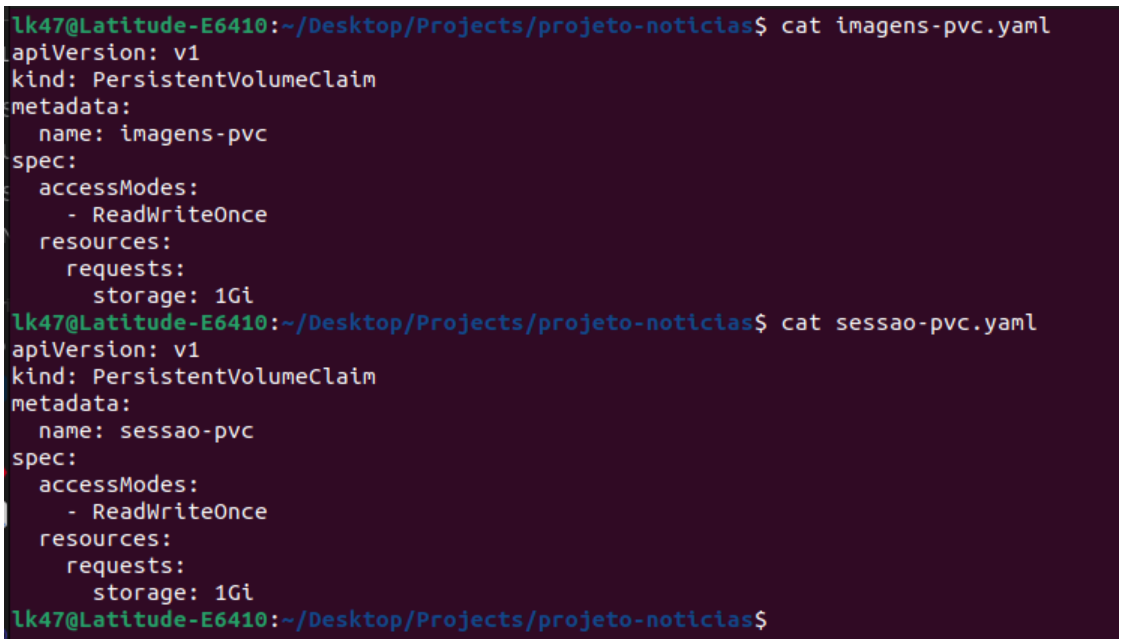
- Precisamos que seja persistido tanto as imagens quanto as sessões
- Vamos criar um pvc tanto para as imagens quanto para a sessão
- Dessa forma vamos criar uma pvc para imagens e para sessão

- imagens-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: imagens-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

- sessao-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: sessao-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```



- Dessa forma vamos declarar no nosso pod

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sistema-noticias-statefulset
spec:
  replicas: 1
  template:
    metadata:
```

```
      labels:
        app: sistema-noticias
    spec:
      containers:
        - name: sistema-noticias-container
          image: aluracursos/sistema-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: sistema-configmap
          volumeMounts:
            - name: imagens
              mountPath: /var/www/html/upload
            - name: tmp
              mountPath: /tmp
      volumes:
        - name: imagens
          persistentVolumeClaim:
            claimName: imagens-pvc
        - name: tmp
          persistentVolumeClaim:
            claimName: sessao-pvc
  selector:
    matchLabels:
      app: sistema-noticias
  serviceName: svc-sistema-noticias
```

```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ cat sistema-noticias-sts.yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sistema-noticias-statefulset
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: sistema-noticias
    spec:
      containers:
        - name: sistema-noticias-container
          image: aluracursos/sistema-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: sistema-configmap
          volumeMounts:
            - name: imagens
              mountPath: /var/www/html/upload
            - name: tmp
              mountPath: /tmp
      volumes:
        - name: imagens
          persistentVolumeClaim:
            claimName: imagens-pvc
        - name: tmp
          persistentVolumeClaim:
            claimName: sessao-pvc
  selector:
    matchLabels:
      app: sistema-noticias
  serviceName: svc-sistema-noticias
```

    - Declaremos o volumeMounts para sessão e imagem
    - Declaramos os dois pvc no YAML
- Vamos aplicar os pvcs
    - Podemos ver que dando um get nos pvcs temos um pv já alocado com os pvcs, na coluna volume

- Se dermos um get sc podemos visualizar que há um hostpath (default) com reclaimpolicy DELETE (se não usar não será criado)

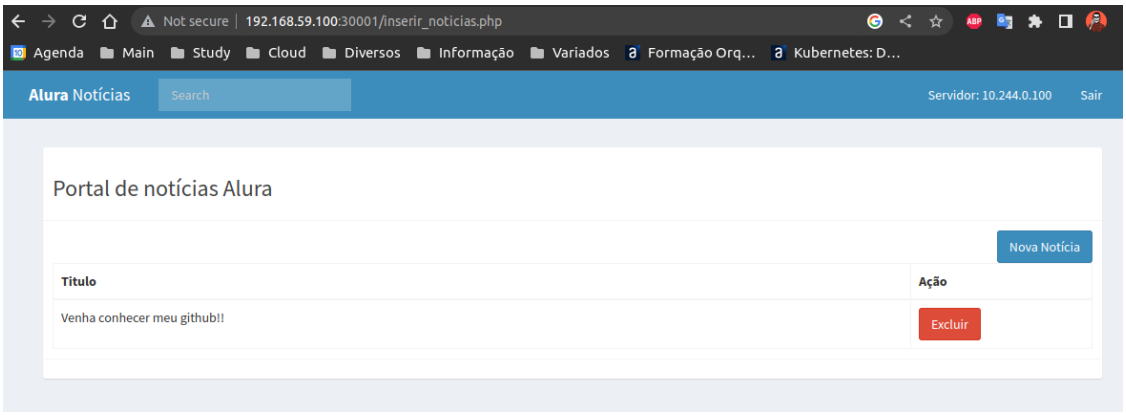
```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias$ kubectl apply -f imagens-pvc.yaml
persistentvolumeclaim/imagens-pvc created
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias$ kubectl apply -f sessao-pvc.yaml
persistentvolumeclaim/sessao-pvc created
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias$ kubectl get pvc
NAME          STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
imagens-pvc   Bound    pvc-ccb077ce-70d8-4da6-a63f-ae5a118bc771   1Gi        RWO            standard       11s
sessao-pvc    Bound    pvc-d91c66bc-62be-4bfd-9299-e38598e4e230   1Gi        RWO            standard       5s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias$
```

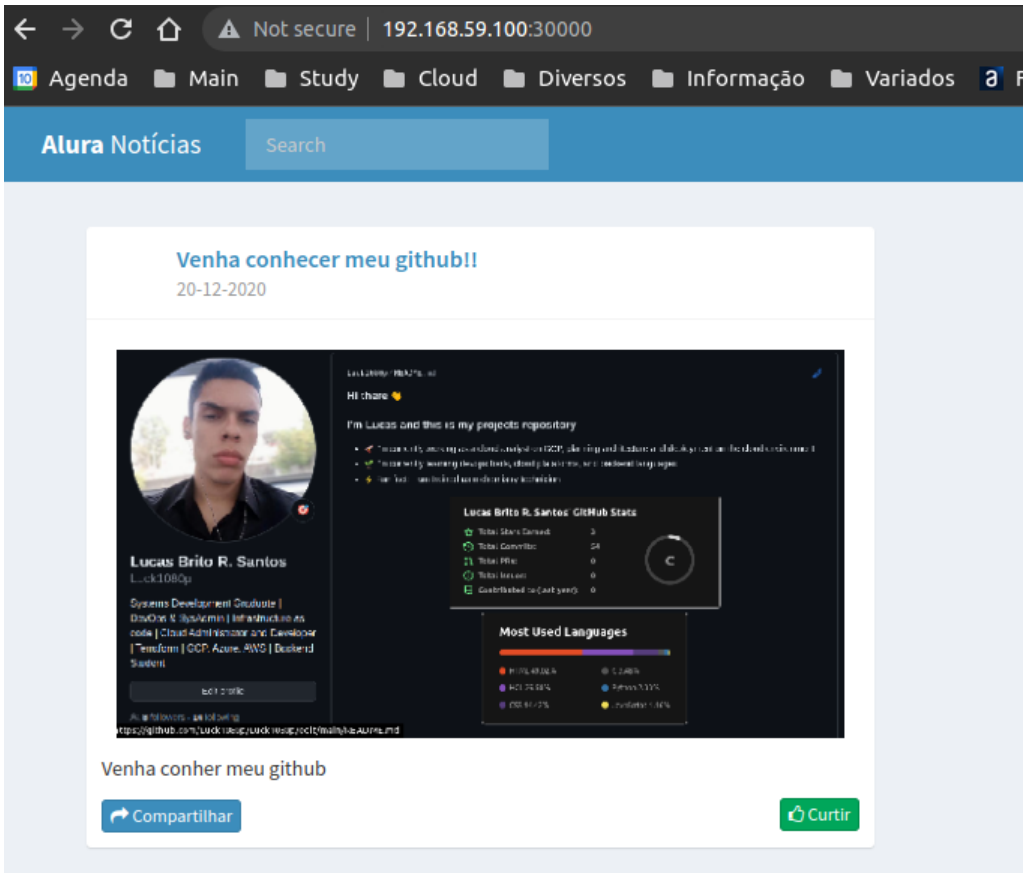
```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias$ kubectl get sc
NAME                 PROVISIONER                RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
standard (default)   k8s.io/minikube-hostpath   Delete          Immediate           false                  27d
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias$
```

- Dessa forma vamos reaplicar o sistema-noticias-statefulset

```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl apply -f sistema-noticias-sts.yaml
statefulset.apps/sistema-noticias-statefulset created
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl get pods
NAME                                      READY   STATUS    RESTARTS      AGE
db-noticias-deployment-5c87d7d876-xj7x2   1/1     Running   3 (35m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-d2rb2   1/1     Running   3 (35m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-j677f   1/1     Running   3 (35m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-qq4p8   1/1     Running   3 (35m ago)   2d22h
sistema-noticias-statefulset-0            1/1     Running   0             9s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl get statefulset
NAME                        READY   AGE
sistema-noticias-statefulset   1/1     18s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$
```

- Vamos cadastrar novamente a noticias

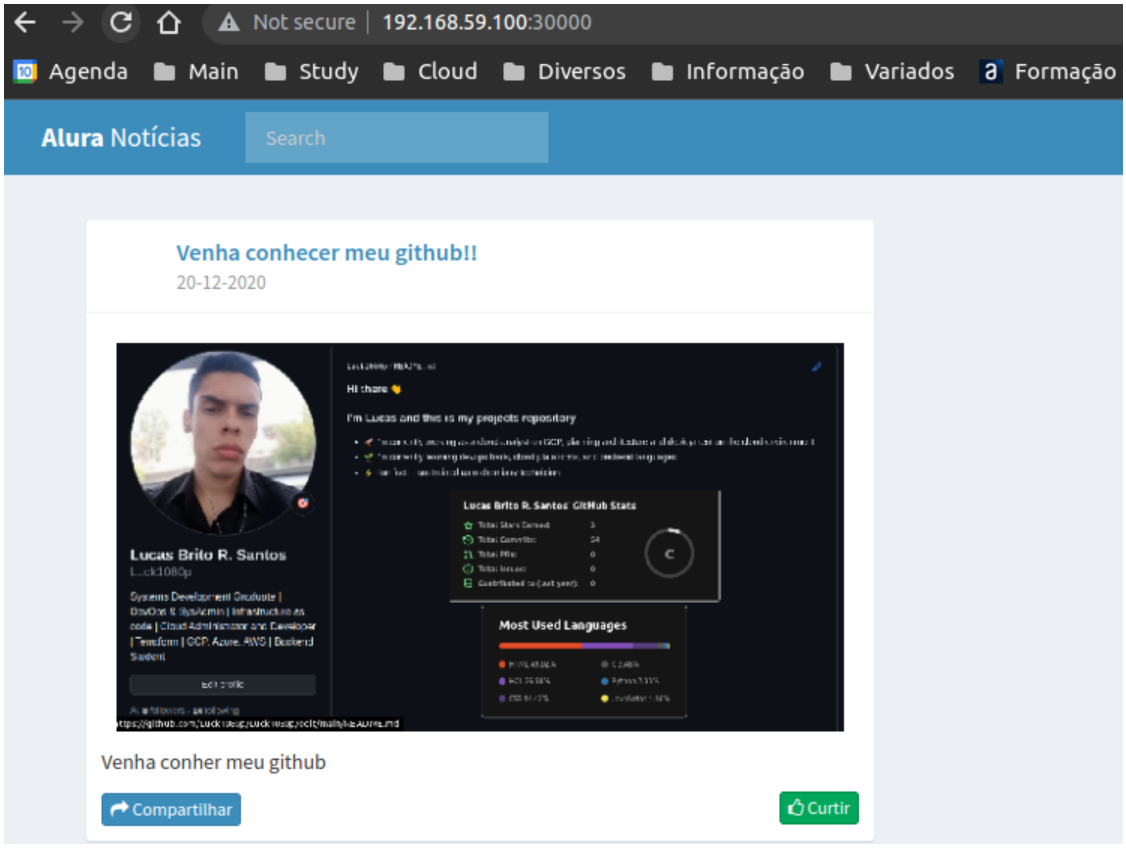


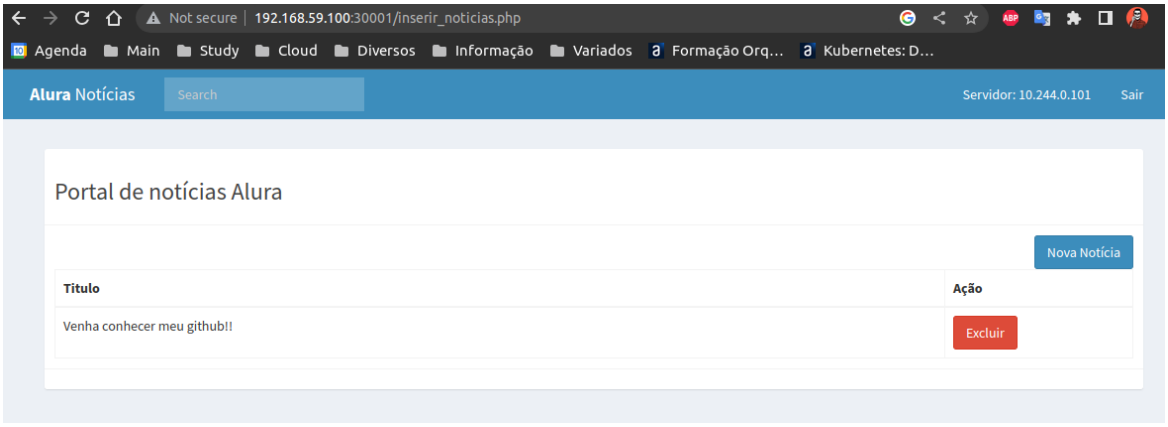


- Vamos agora apagar o pod e o mesmo será recriado

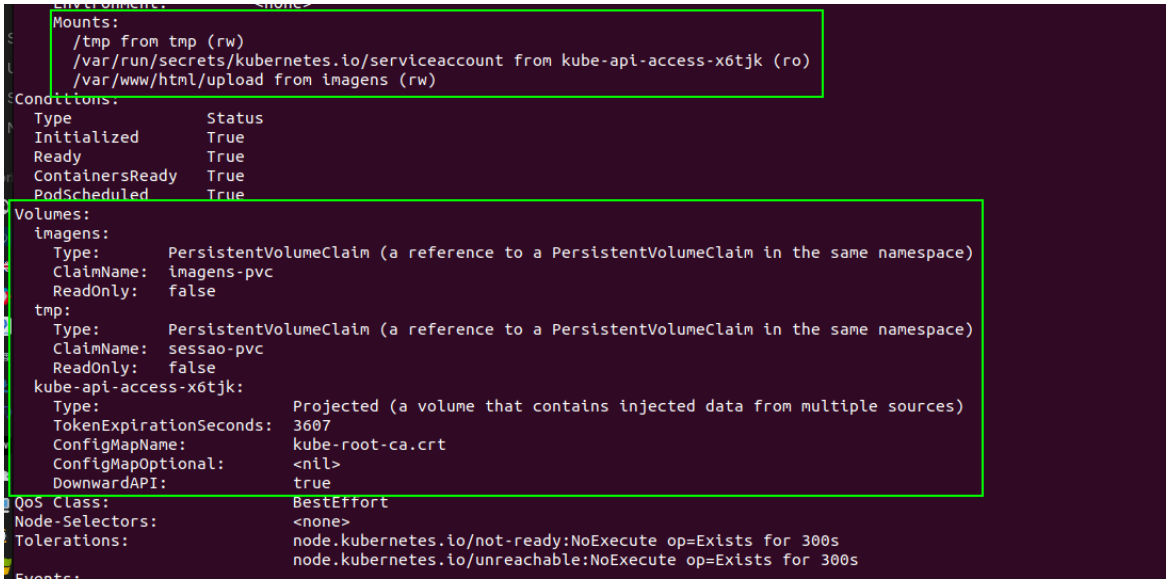
```
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl delete pod sistema-noticias-statefulset-0
pod "sistema-noticias-statefulset-0" deleted
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS      AGE
db-noticias-deployment-5c87d7d876-xj7x2       1/1     Running   3 (38m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-d2rb2   1/1     Running   3 (38m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-j677f   1/1     Running   3 (38m ago)   2d22h
portal-noticias-deployment-5dcbbf85f4-qq4p8   1/1     Running   3 (38m ago)   2d22h
sistema-noticias-statefulset-0                1/1     Running   0             7s
lk47@Latitude-E6410:~/Desktop/Projects/projeto-noticias/statefulset$
```

- Após recriar vamos validar os sistemas dando f5 e as noticias permanecem assim como as sessões





- Validando o describe podemos validar todos os eventos e bind dos volumes



▼ **Analisando definições**

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: um-statefulset-qualquer
spec:
  #Restante omitido...
    spec:
      containers:
        - name: nginx-container
          image: nginx:latest
          ports:
            - containerPort: 80
          volumeMounts:
            - name: dados-persistidos
              mountPath: /data-info
      volumes:
        - name: dados-persistidos
          persistentVolumeClaim:
            claimName: dados-pvc
  #Restante omitido....
```

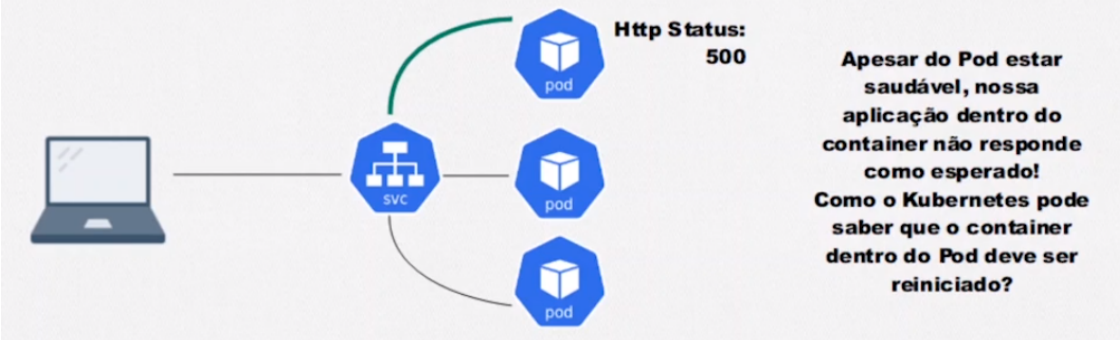
O que podemos afirmar sobre este arquivo de definição?

- **Para que funcione da maneira esperada, devemos ter um PersistentVolumeClaim chamado `dados-pvc` criado.**

▼ **Checando status com probes**

▼ **Conhecendo probes**

- Temos um serviço que fará um balanceamento de carga de 3 pods
- O status code http quando bem sucedido tende a ser 200
- Em algum momento pode haver o erro 500, e isso quando ocorre se deve quando a aplicação não funciona conforme o esperado
- Nesses casos o Kubernetes entende que o pod está em funcionamento mas não se a aplicação em sí está em funcionamento e não sabe se deve reiniciar



- Validamos em no source code através do browser em REDE, o status code que pode ser alterado
- Para isso utilizamos a prova de vida que no caso são o Liveness e Readiness probes
- No caso não precisamos declarar um arquivo para isso, podemos fazer dentro da propria declaração do pod

▼ **Sobes probes**

- *Qual é a principal utilização de liveness probes?*
  - **Tornar visível ao Kubernetes que uma aplicação não está se comportando da maneira esperada.**

▼ **Utilizando Liveness probes**

- Declaramos da seguinte forma, no caso vamos utilizar o deployment do portal-noticias

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: portal-noticias-deployment
spec:
  template:
    metadata:
      name: portal-noticias
      labels:
        app: portal-noticias
    spec:
      containers:
        - name: portal-noticias-container
          image: aluracursos/portal-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: portal-configmap
          livenessProbe:
            httpGet:
              path: /
              port: 80
            periodSeconds: 10
            failureThreshold: 3
            initialDelaySeconds: 20
  replicas: 3
  selector:
    matchLabels:
      app: portal-noticias
```

```
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias/deployments$ cat portal-noticias-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: portal-noticias-deployment
spec:
  template:
    metadata:
      name: portal-noticias
      labels:
        app: portal-noticias
    spec:
      containers:
        - name: portal-noticias-container
          image: aluracursos/portal-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: portal-configmap
          livenessProbe:
            httpGet:
              path: /
              port: 80
            periodSeconds: 10
            failureThreshold: 3
            initialDelaySeconds: 20
  replicas: 3
  selector:
    matchLabels:
      app: portal-noticias
```

- No caso definimos o liveness probe dentro da declaração do deployment portal noticias
- Especificamos o caminho, porta do container, o periodo de teste, maximo de falhas antes do reinicio, e informamos a partir de qual momento ele irá executar o teste
- No caso ele irá definir como saudavel quando o status code for igual ou maior a 200 e menor do que 400

- Feito isso executamos um redeploy

```
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias/deployments$ kubectl apply -f portal-noticias-deploy.yaml
deployment.apps/portal-noticias-deployment configured
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias/deployments$ kubectl get deploy
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
db-noticias-deployment       1/1     1            1           50m
portal-noticias-deployment   3/3     3            3           5m51s
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias/deployments$ kubectl get pods
NAME                                         READY   STATUS    RESTARTS      AGE
db-noticias-deployment-5c87d7d876-4xws6      1/1     Running   1 (30m ago)   50m
portal-noticias-deployment-c64796dd-5fdk7    1/1     Running   0             86s
portal-noticias-deployment-c64796dd-7l4dh    1/1     Running   0             79s
portal-noticias-deployment-c64796dd-wblrk    1/1     Running   0             82s
sistema-noticias-statefulset-0               1/1     Running   1 (30m ago)   49m
```

- Dessa forma o status code estará ok e caso mude o pod será reiniciado
- Vamos fazer o mesmo para o sistema-noticias

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sistema-noticias-statefulset
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: sistema-noticias
      name: sistema-noticias
    spec:
```

```
    containers:
      - name: sistema-noticias-container
        image: aluracursos/sistema-noticias:1
        ports:
          - containerPort: 80
        envFrom:
          - configMapRef:
              name: sistema-configmap
        volumeMounts:
          - name: imagens
            mountPath: /var/www/html/uploads
          - name: sessao
            mountPath: /tmp
        livenessProbe:
          httpGet:
            path: /
            port: 80
          periodSeconds: 10
          failureThreshold: 3
          initialDelaySeconds: 20
    volumes:
      - name: imagens
        persistentVolumeClaim:
          claimName: imagens-pvc
      - name: sessao
        persistentVolumeClaim:
          claimName: sessao-pvc
selector:
  matchLabels:
    app: sistema-noticias
```

```
metadata:
  name: sistema-noticias-statefulset
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: sistema-noticias
      name: sistema-noticias
    spec:
      containers:
        - name: sistema-noticias-container
          image: aluracursos/sistema-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: sistema-configmap
          volumeMounts:
            - name: imagens
              mountPath: /var/www/html/uploads
            - name: sessao
              mountPath: /tmp
          livenessProbe:
            httpGet:
              path: /
              port: 80
            periodSeconds: 10
            failureThreshold: 3
            initialDelaySeconds: 20
      volumes:
        - name: imagens
```

- Feito isso só executamos o redeploy

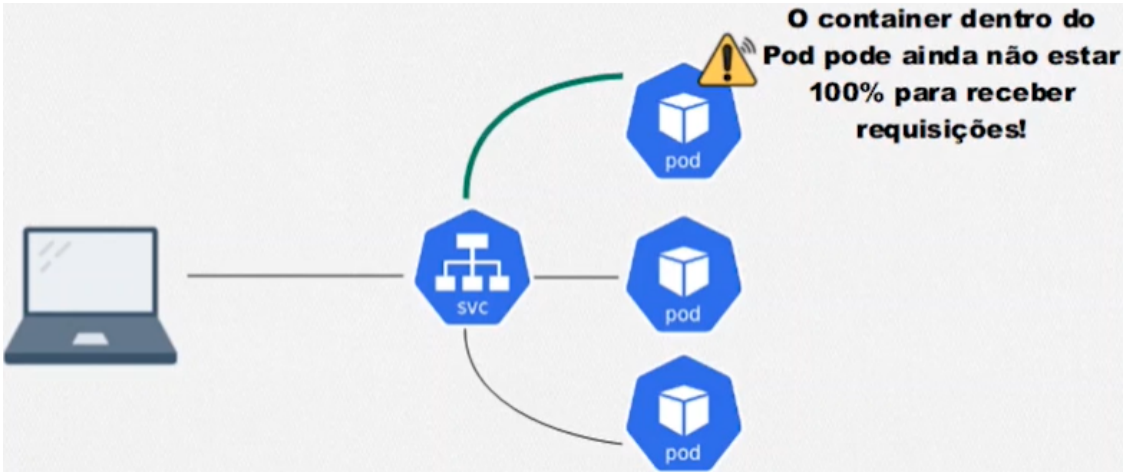
```
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl get statefulset
NAME                          READY   AGE
sistema-noticias-statefulset  1/1     51m
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl delete statefulset sistema-noticias-statefulset
statefulset.apps "sistema-noticias-statefulset" deleted
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl apply -f sistema-noticias-sts.yaml
statefulset.apps/sistema-noticias-statefulset created
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias/statefulset$ kubectl get statefulset
NAME                          READY   AGE
sistema-noticias-statefulset  1/1     84s
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias/statefulset$
```

▼ **Critérios do liveness probes**

- *Como um liveness probe entende que a aplicação não está respondendo?*
  - **Ele indicará falha caso o código de retorno seja menor que 200 ou maior/igual a 400.**

▼ **Utilizando Readiness probes**

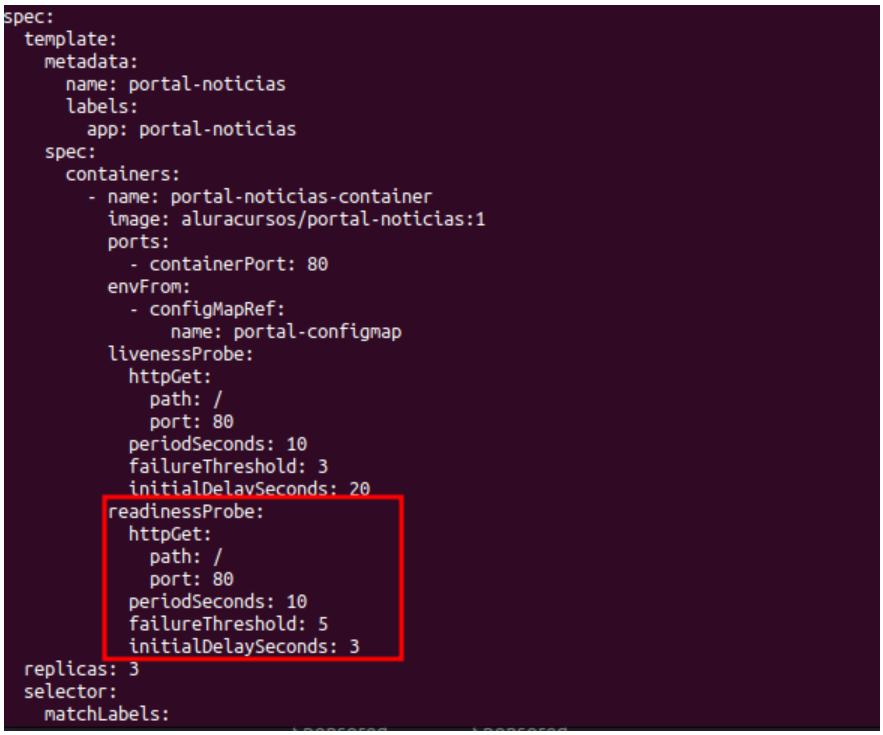
- Caso o pod por ventura falhe temos o replicaset e o deployment, caso o status code falhe temos a prova de vida que reiniciará o pode, agora precisamos de um mecanismos que permita definirmos um momento em que a aplicação estará pronta para receber requisições



- Para isso temos o readiness probes
- No caso vamos aplicar para o portal e sistema-noticias
  - Portal

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: portal-noticias-deployment
spec:
  template:
    metadata:
      name: portal-noticias
      labels:
        app: portal-noticias
    spec:
      containers:
        - name: portal-noticias-container
          image: aluracursos/portal-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: portal-configmap
```

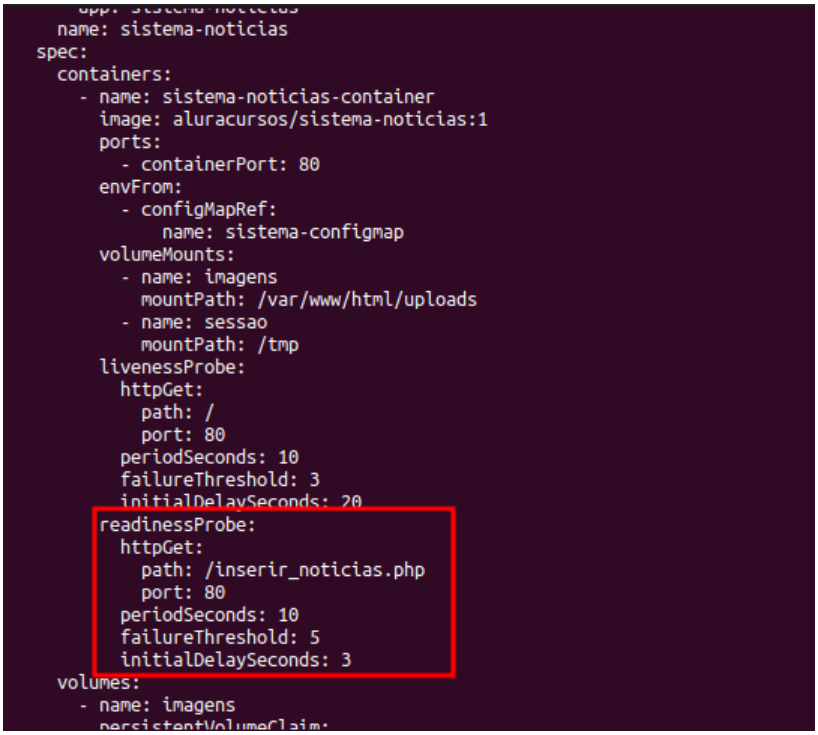
```
          livenessProbe:
            httpGet:
              path: /
              port: 80
            periodSeconds: 10
            failureThreshold: 3
            initialDelaySeconds: 20
          readinessProbe:
            httpGet:
              path: /
              port: 80
            periodSeconds: 10
            failureThreshold: 5
            initialDelaySeconds: 3
  replicas: 3
  selector:
    matchLabels:
      app: portal-noticias
```



- Sistema

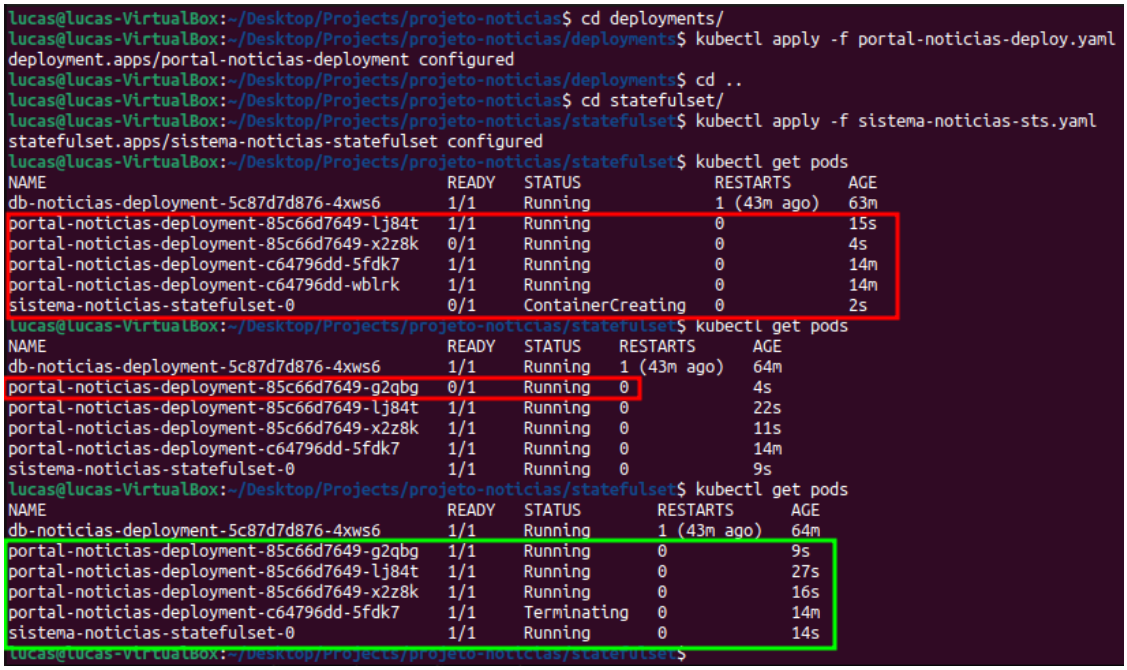
```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sistema-noticias-statefulset
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: sistema-noticias
      name: sistema-noticias
    spec:
      containers:
        - name: sistema-noticias-container
          image: aluracursos/sistema-noticias:1
```

```
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: sistema-configmap
          volumeMounts:
            - name: imagens
              mountPath: /var/www/html/uploads
            - name: sessao
              mountPath: /tmp
          livenessProbe:
            httpGet:
              path: /
              port: 80
            periodSeconds: 10
            failureThreshold: 3
            initialDelaySeconds: 20
          readinessProbe:
            httpGet:
              path: /inserir_noticias.php
              port: 80
            periodSeconds: 10
            failureThreshold: 5
            initialDelaySeconds: 3
      volumes:
        - name: imagens
          persistentVolumeClaim:
            claimName: imagens-pvc
        - name: sessao
          persistentVolumeClaim:
            claimName: sessao-pvc
  selector:
    matchLabels:
      app: sistema-noticias
```



- Se assemelha como o liveness e possui os mesmos conceitos porém só alteramos os valores, e colocamos o caminho especifico no caso inserir_noticias.php

- Vamos aplicar o redeploy e no caso o delay de inicialização já será aplicado



- Assim como utilizamos o `httpGet` podemos usar o `tcpSocket`
- Portanto essa é a maneira que o kubernetes permite manter as aplicações funcionais em nosso cluster

▼ Probes e suas insturações

```
readinessProbe:
  httpGet:
    path: /
    port: 80
  periodSeconds: 10
  initialDelaySeconds: 3
```

- **Ele começará a executar os testes apenas 3 segundos depois do container ser criado.**
- **Ele fará os testes utilizando a porta 80 da aplicação.**

▼ Startup Probes

- **Há um terceiro tipo de probe voltado para aplicações legadas, o Startup Probe.**
- **Algumas aplicações legadas exigem tempo adicional para inicializar na primeira vez. Nem sempre Liveness ou Readiness Probes vão conseguir resolver de maneira simples os problemas de inicialização de aplicações legadas.**

Configure Liveness, Readiness and Startup Probes

This page shows how to configure liveness, readiness and startup probes for containers. The kubelet uses liveness probes to know when to restart a container. For example, liveness probes could catch a deadlock, where an application is running, but unable to

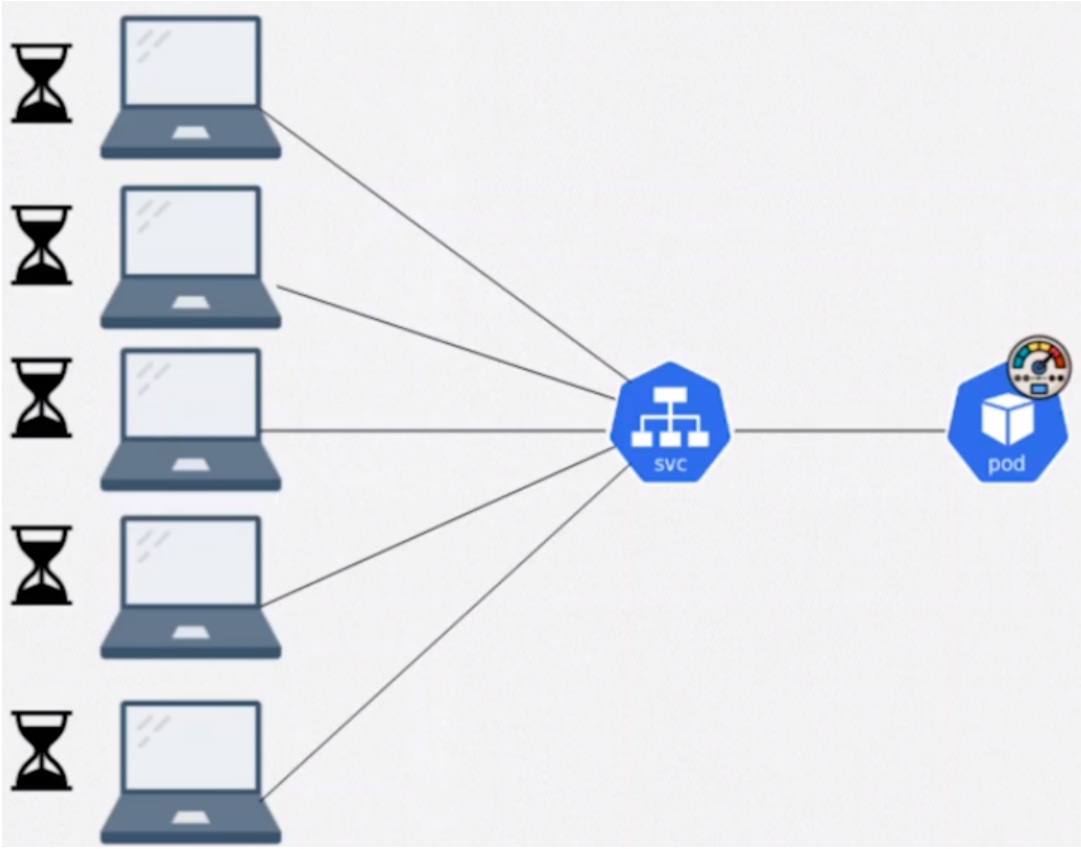
https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/#define-startup-probes

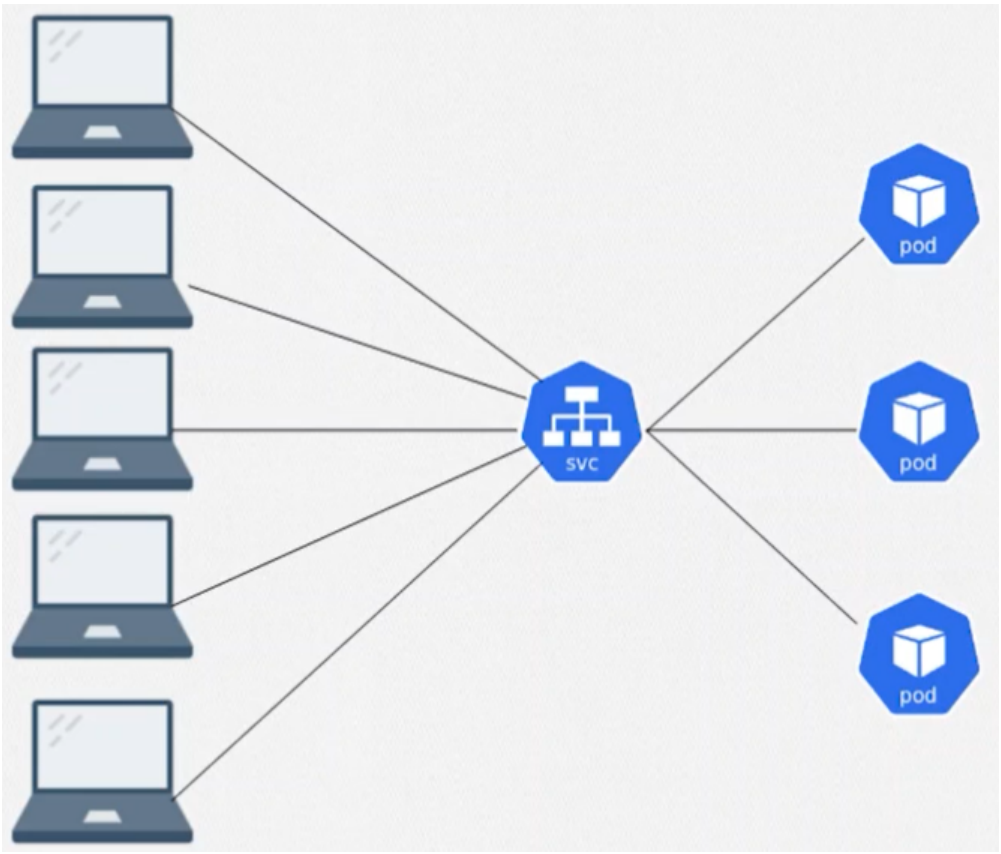
kubern

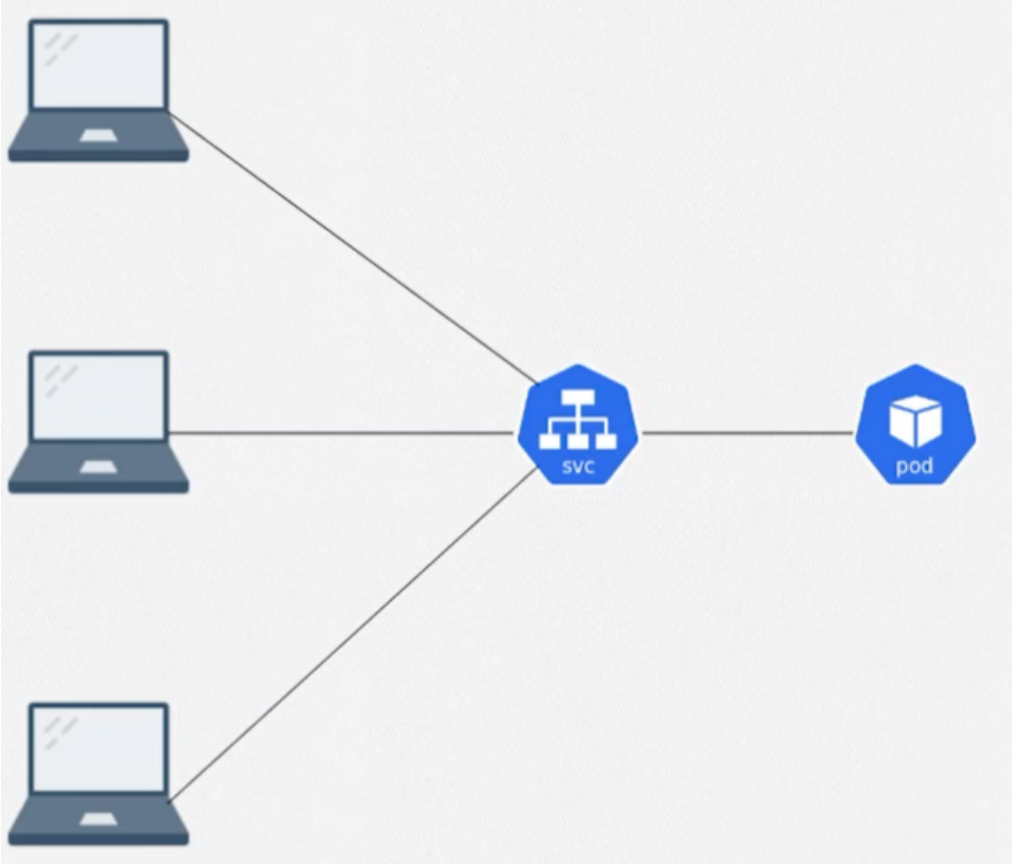
▼ **Como escalar com o Horizontal Pod Autoscales**

▼ **Escalando pods automaticamente**

- Quanto maior maior a quantidade de acessos ao nosso portal noticias, mais processamento e utilização de recursos os nosso pods requiriram, e para isso o ideal é aumentar a quantidade de pods para que fique distribuida toda a carga não sobrecarregando um determinado nó em questão
- Inversamente segue o mesmo raciocinio, se não estamos tendo muitos acessos não será necessario uma grande quantidade de pods, tendo em vista que as requisições não estão tão altas



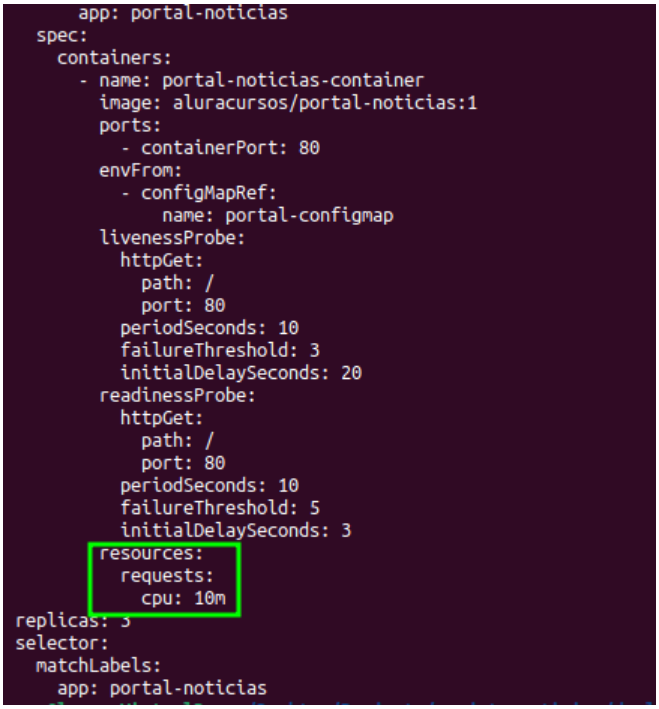

svc
pod
pod
pod



- Sendo assim a melhor forma de trabalhar com escalabilidade no kubernetes de forma automatizada, é utilizando um recurso chamado HPA (Horizontal pod autoscaler)
- Ele escala o numero de pods com base na observação de CPU utilizada
- Para isso as requisições de CPU devem estar declaradas no YAML
- Vamos estruturar nosso pod portal-sistemas-noticias

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: portal-noticias-deployment
spec:
  template:
    metadata:
      name: portal-noticias
      labels:
        app: portal-noticias
    spec:
      containers:
        - name: portal-noticias-container
          image: aluracursos/portal-noticias:1
          ports:
            - containerPort: 80
          envFrom:
            - configMapRef:
                name: portal-configmap
          livenessProbe:
            httpGet:
              path: /
              port: 80
```

```
          periodSeconds: 10
          failureThreshold: 3
          initialDelaySeconds: 20
        readinessProbe:
          httpGet:
            path: /
            port: 80
          periodSeconds: 10
          failureThreshold: 5
          initialDelaySeconds: 3
        resources:
          requests:
            cpu: 10m
replicas: 3
selector:
  matchLabels:
    app: portal-noticias
```



- Campo Resources → Definição de utilização de CPU

```
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias/deployments$ kubectl apply -f portal-noticias-deploy.yaml
deployment.apps/portal-noticias-deployment created
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias/deployments$ kubectl get deploy
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
db-noticias-deployment       1/1     1            1           110m
portal-noticias-deployment   2/3     3            2           9s
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias/deployments$
```

- No caso vamos criar um novo arquivo no qual iremos declarar o nosso HPA
  - No curso é demonstrado utilizar o seguinte codigo

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: portal-noticias-hpa
```

```
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: portal-noticias-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
    - type: Resource
      resource:
        name: cpu
        target:
          type: Utilization
          averageUtilization: 50
```

    - No caso utilizando o `autoscaling/v2beta2`, porém no nosso caso realizei de forma diferente, utilizando o `v1`

  - Ficando da seguinte forma

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: portal-noticias-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: portal-noticias-deployment
  minReplicas: 1
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

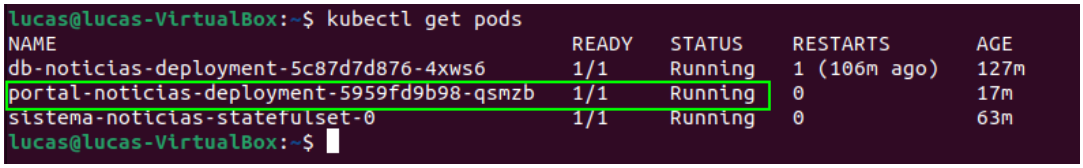
  - No caso estamos utilizando a apiversion autosclaing/2beta2, mas também temos a versão v1
  - Declaramos o target, o tipo de apiVersion, tipo de recurso, nome e min e max de 1 e 10 em replicas
  - Metric, declaramos o recurso e a quantidade utilização + o limite da CPU no caso, baseado na utilização dele, uso médio de 50%
  - Ou seja se passar de 50% ele vai escalar

- Dessa forma vamos aplicar

```
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$ cat portal-noticias-hpa.yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: portal-noticias-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: portal-noticias-deployment
  minReplicas: 1
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50

lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$ kubectl apply -f portal-noticias-hpa.yaml
horizontalpodautoscaler.autoscaling/portal-noticias-hpa unchanged
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$ kubectl get hpa
NAME                REFERENCE                                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
portal-noticias-hpa   Deployment/portal-noticias-deployment   10%/50%   1         10        3          26s
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$
```

- Podemos ver que os pods já se encontram conforme a definição do HPA, no caso o portal-noticias está definido na implantação para que suba 3 pods, porém como defimos os requisitos do HPA (min 1 e max 10), não irá precisar criar os 3 pods sendo que a quantidade processamento está abaixo de 50%



- No caso o HPA pode não identificar o processamento devido a ausencia do metric server que é o responsavel por fazer está aferição de utilização de CPU, nesse caso precisamos definir uma configuração adicional para que este recurso seja habilitado, no caso ela irá diferenciar do linux(minikube) pro windows(Docker)
  - No nosso caso estamos realizando os laboratorios em ambiente linux, portanto a demonstração será neste amabiente

---

WINDOWS - Metric Server - Docker

- No caso utilizamos o Linux em especifico para realização desses laboratios, mas abaixo segue a forma de configuração no windows
- Utilizaremos o YAML deste repositorio, para configuração do metric server

  https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.3.7/components.yaml
  - Utilizamos a flag `--kubelet-insecure-tls` pois não utilizamos um certificado CA
  - Feito isso salve como components.yaml e aplique no kubernetes
- Irá demorar um pouco e depois ele começara a identificas as metricas, é bom acompanhar através do `describe HPA` ou através do `get HPA` na coluna `TARGETS`
- Em seguida poderá utilizar o script shell abaixo para mandar sequencias de requisições para aumentar a taxa de processamento do pod, recomendo utilizar o git bash conforme o curso ou então o WSL

```
#!/bin/bash
for i in {1..10000}; do
  curl localhost:30000
  sleep $1
done
```

  - Crie um arquivo .sh
  - Executar o comando no git bash ou WSL `sh <nomedoarquivo>.sh 0.001 > out.txt`
- Acompanhe a quantidade de pods através do kubectl get pods

LINUX - Metric Server - minikube

- O minikube prove varios recursos, utilizando o comando `minikube addons list` podemos ver a lista de `ADDONS` disponíveis

```
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$ minikube addons list
|-----------------------------|----------|--------------|--------------------------------|
|         ADDON NAME          | PROFILE  |    STATUS    |           MAINTAINER           |
|-----------------------------|----------|--------------|--------------------------------|
| ambassador                  | minikube | disabled     | 3rd party (Ambassador)         |
| auto-pause                  | minikube | disabled     | Google                         |
| cloud-spanner               | minikube | disabled     | Google                         |
| csi-hostpath-driver         | minikube | disabled     | Kubernetes                     |
| dashboard                   | minikube | disabled     | Kubernetes                     |
| default-storageclass        | minikube | enabled ✅   | Kubernetes                     |
| efk                         | minikube | disabled     | 3rd party (Elastic)            |
| freshpod                    | minikube | disabled     | Google                         |
| gcp-auth                    | minikube | disabled     | Google                         |
| gvisor                      | minikube | disabled     | Google                         |
| headlamp                    | minikube | disabled     | 3rd party (kinvolk.io)         |
| helm-tiller                 | minikube | disabled     | 3rd party (Helm)               |
| inaccel                     | minikube | disabled     | 3rd party (InAccel             |
|                             |          |              | [info@inaccel.com])            |
| ingress                     | minikube | disabled     | Kubernetes                     |
| ingress-dns                 | minikube | disabled     | Google                         |
| istio                       | minikube | disabled     | 3rd party (Istio)              |
| istio-provisioner           | minikube | disabled     | 3rd party (Istio)              |
| kong                        | minikube | disabled     | 3rd party (Kong HQ)            |
| kubevirt                    | minikube | disabled     | 3rd party (KubeVirt)           |
| logviewer                   | minikube | disabled     | 3rd party (unknown)            |
| metallb                     | minikube | disabled     | 3rd party (MetalLB)            |
| metrics-server              | minikube | enabled ✅   | Kubernetes                     |
| nvidia-driver-installer     | minikube | disabled     | Google                         |
| nvidia-gpu-device-plugin    | minikube | disabled     | 3rd party (Nvidia)             |
| olm                         | minikube | disabled     | 3rd party (Operator Framework) |
| pod-security-policy         | minikube | disabled     | 3rd party (unknown)            |
| portainer                   | minikube | disabled     | 3rd party (Portainer.io)       |
| registry                    | minikube | disabled     | Google                         |
| registry-aliases            | minikube | disabled     | 3rd party (unknown)            |
| registry-creds              | minikube | disabled     | 3rd party (UPMC Enterprises)   |
| storage-provisioner         | minikube | enabled ✅   | Google                         |
```

- No caso já habilitamos o metric server, porém é muito simples de habilitar usando o comando `minikube addons enable metrics-server`

```
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$ minikube addons enable metrics-server
💡  metrics-server is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image registry.k8s.io/metrics-server/metrics-server:v0.6.3
🌟  The 'metrics-server' addon is enabled
```

- Ele irá sincronizar, demora um pouco, e assim que ele concluir poderemos acompanhar as metricas do `TARGET HPA`, para isso podemos usar os comandos `kubectl describe HPA <Nome do HPA>` ou `kubectl get HPA` e acompanhar a verificação pela coluna `TARGETS`

```
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$ kubectl get hpa
NAME                 REFERENCE                                   TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
portal-noticias-hpa   Deployment/portal-noticias-deployment   10%/50%   1         10        1          9m32s
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$ kubectl describe hpa portal-noticias-hpa
Name:                                                  portal-noticias-hpa
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Tue, 18 Jul 2023 17:07:22 -0300
Reference:                                             Deployment/portal-noticias-deployment
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  10% (1m) / 50%
Min replicas:                                          1
Max replicas:                                          10
Deployment pods:                                       1 current / 1 desired
Conditions:
  Type            Status  Reason              Message
  ----            ------  ------              -------
  AbleToScale     True    ReadyForNewScale    recommended size matches current size
  ScalingActive   True    ValidMetricFound    the HPA was able to successfully calculate a replica count from
uest)
  ScalingLimited  False   DesiredWithinRange  the desired count is within the acceptable range
Events:
  Type    Reason             Age    From                       Message
  ----    ------             ----   ----                       -------
  Normal  SuccessfulRescale  4m40s  horizontal-pod-autoscaler  New size: 1; reason: All metrics below target
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$
```

- Feito isso agora podemos rodar o seguinte script abaixo

```
#!/bin/bash
for i in {1..10000}; do
  curl <IP do node>:30000
  sleep $1
done
```

  - No caso conforme pontuado alteramos pelo IP do nosso node e salvamos como um arquivo `.sh`

```
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$ cat > test-hpa.sh
#!/bin/bash
for i in {1..10000}; do
  curl <IP do node>:30000
  sleep $1
done
^C
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$
```

```
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$ kubectl get nodes -o wide
NAME       STATUS   ROLES           AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
minikube   Ready    control-plane   28h   v1.26.3   192.168.49.2   <none>        Ubuntu 20.04.5 LTS   5.19.0-46-generic   docker://23.0.2
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$ nano test-hpa.sh
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$ cat test-hpa.sh
#!/bin/bash
for i in {1..10000}; do
  curl 192.168.49.2:30000
  sleep $1
done
```

- Agora podemos executar o arquivo usando o comando `bash test-hpa.sh 0.001 > out.txt`

```
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$ bash test-hpa.sh 0.001 > out.txt
```

  - Dessa forma ele irá mandar varias requisições para o nosso portal noticias

```
100  5938  100  5938    0     0   492k      0 --:--:-- --:--:-- --:--:--  527k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0   664k      0 --:--:-- --:--:-- --:--:--  724k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0  2585k      0 --:--:-- --:--:-- --:--:-- 2899k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0   718k      0 --:--:-- --:--:-- --:--:--  828k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0  1013k      0 --:--:-- --:--:-- --:--:-- 1159k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0  1248k      0 --:--:-- --:--:-- --:--:-- 1449k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0  3323k      0 --:--:-- --:--:-- --:--:-- 5798k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0  2699k      0 --:--:-- --:--:-- --:--:-- 5798k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0   473k      0 --:--:-- --:--:-- --:--:--  483k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0  1739k      0 --:--:-- --:--:-- --:--:-- 1932k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0  1137k      0 --:--:-- --:--:-- --:--:-- 1449k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0   521k      0 --:--:-- --:--:-- --:--:--  579k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0  1659k      0 --:--:-- --:--:-- --:--:-- 1932k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Up
```

- Vamos acompanhar o HPA e a quantidade de replicas dos pods usando `watch kubectl get hpa`

```
Every 2,0s: kubectl get hpa                                                    lucas-Virt

NAME                REFERENCE                                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
portal-noticias-hpa   Deployment/portal-noticias-deployment   10%/50%   1         10        1          33m
```

- Podemos ver que a quantidade de replicas está aumentando

```
Every 2,0s: kubectl get hpa                                                    lucas-Virtuali

NAME                REFERENCE                                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
portal-noticias-hpa   Deployment/portal-noticias-deployment   150%/50%   1         10        3          35m
```

```
NAME                REFERENCE                                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
portal-noticias-hpa   Deployment/portal-noticias-deployment   60%/50%   1         10        8          38m
```

- Vamos validar os pods

```
lucas@lucas-VirtualBox:~$ kubectl get pods
NAME                                        READY   STATUS    RESTARTS      AGE
db-noticias-deployment-5c87d7d876-4xws6     1/1     Running   1 (129m ago)   150m
portal-noticias-deployment-5959fd9b98-7cd89   1/1     Running   0             104s
portal-noticias-deployment-5959fd9b98-qsmzb   1/1     Running   0             39m
portal-noticias-deployment-5959fd9b98-xmddh   1/1     Running   0             104s
sistema-noticias-statefulset-0              1/1     Running   0             86m
lucas@lucas-VirtualBox:~$
```

```
lucas@lucas-VirtualBox:~$ kubectl get pods
NAME                                          READY   STATUS    RESTARTS       AGE
db-noticias-deployment-5c87d7d876-4xws6       1/1     Running   1 (130m ago)   151m
portal-noticias-deployment-5959fd9b98-7cd89   1/1     Running   0              2m41s
portal-noticias-deployment-5959fd9b98-bk8sv   1/1     Running   0              41s
portal-noticias-deployment-5959fd9b98-f689n   1/1     Running   0              41s
portal-noticias-deployment-5959fd9b98-fs6zf   1/1     Running   0              26s
portal-noticias-deployment-5959fd9b98-k4ptv   1/1     Running   0              41s
portal-noticias-deployment-5959fd9b98-qsmzb   1/1     Running   0              40m
portal-noticias-deployment-5959fd9b98-xmddh   1/1     Running   0              2m41s
portal-noticias-deployment-5959fd9b98-zhz8z   1/1     Running   0              26s
sistema-noticias-statefulset-0                1/1     Running   0              87m
lucas@lucas-VirtualBox:~$
```

- Agora vamos cancelar as requisições e vamos acompanhar novamente

```
100  5938  100  5938    0     0  2629k      0 --:--:-- --:--:-- --:--:-- 2899k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0  3548k      0 --:--:-- --:--:-- --:--:-- 5798k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0  2355k      0 --:--:-- --:--:-- --:--:-- 2899k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5938  100  5938    0     0   530k      0 --:--:-- --:--:-- --:--:--  579k
^C
lucas@lucas-VirtualBox:~/Desktop/Projects/projeto-noticias$
```

- Assim como a utilização voltou a ficar abaixo de 50%, a quantidade de replicas diminui

```
lucas@lucas-VirtualBox:~$ kubectl get hpa
NAME              REFERENCE                                   TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
portal-noticias-hpa   Deployment/portal-noticias-deployment   10%/50%   1         10        1          51m
lucas@lucas-VirtualBox:~$ kubectl get pods
NAME                                          READY   STATUS    RESTARTS       AGE
db-noticias-deployment-5c87d7d876-4xws6       1/1     Running   1 (144m ago)   165m
portal-noticias-deployment-5959fd9b98-qsmzb   1/1     Running   0              54m
sistema-noticias-statefulset-0                1/1     Running   0              101m
lucas@lucas-VirtualBox:~$
```

- Sendo assim essa é a forma na qual configuramos o HPA

▼ Entendendo erros

- *Por qual motivo o `HorizontalPodAutoscaler` pode não conseguir identificar o consumo de recursos dos pods gerenciados?*
  - **Não foi definido um servidor de métricas para que haja o funcionamento da maneira esperada.**

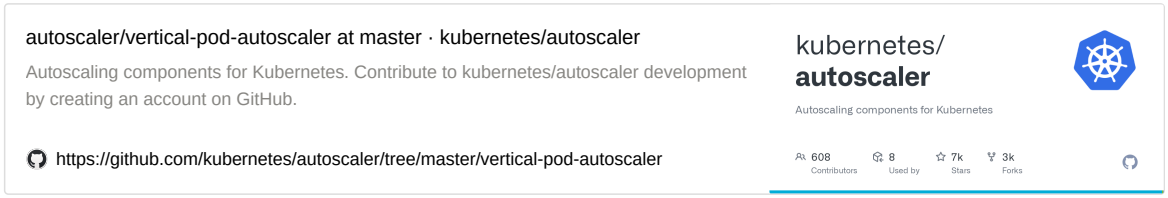
▼ Definindo HPAs

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: primeiro-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: primeiro-deployment
  minReplicas: 1
  maxReplicas: 5
  metrics:
    - type: Resource
      resource:
```

```
      name: cpu
      target:
        type: Utilization
        averageUtilization: 20
```

- *Quais informações podemos extrair sobre o arquivo declarativo*
  - **O HPA visa manter o consumo médio de CPU o mais próximo de 20%.**

▼ **VerticalPodAutoscales**



- O Kubernetes possui o VerticalPodAutoscaler, um recurso customizado que automatiza o ajuste dos consumos de recursos do sistema, como CPU e memória, com base na utilização em cada nó. Isso elimina a necessidade de definir limites e pedidos manualmente.
- Configurações adicionais são necessárias para usar o VerticalPodAutoscaler, e mais informações podem ser encontradas no link fornecido.
- Também é possível encontrar detalhes específicos para provedores de nuvem como GCP e AWS.

Para gabi_bagneti, meu pilar,
Obrigado por todo o apoio durante este projeto. Sua presença foi essencial para meu sucesso. Sou grato por todo o amor e encorajamento. Com todo meu amor!
