



Kubernetes Labs - 1

- *Doc created by → Lucas Brito Rodrigues dos Santos*
- *Esta doc foi baseada nos seguintes cursos alura:*

Curso Online Kubernetes: Pods, Services e ConfigMaps | 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

 <https://cursos.alura.com.br/course/kubernetes-pods-services-configmaps>



curso Online

**ubernetes: Pods, Services e
ConfigMaps**

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

 <https://kubernetes.io/>

kuberr

- *Busco demonstrar em laboratórios práticos realizados com base nesse curso as funções e a utilização do Kubernetes*

Summary

- *Conhecendo o Kubernetes*
 - *Criando o cluster*
 - *Criando e entendendo pods*
 - *Expondo pods com services*
 - *Aplicando services ao projeto*
 - *Definindo variáveis de ambiente*
-

▼ Conhecendo o Kubernetes

▼ O que é Kubernetes

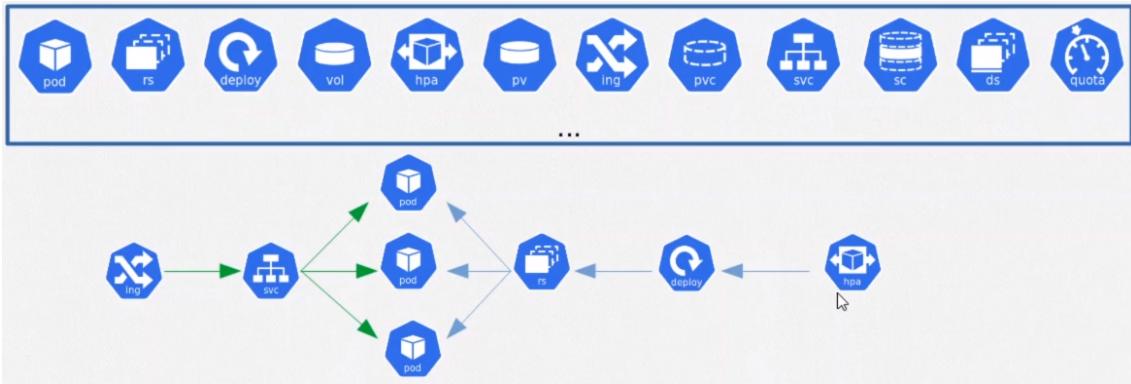
- Kubernetes é uma plataforma de código aberto que ajuda a implantar, gerenciar e escalar aplicativos em contêineres. Ele garante que seus aplicativos funcionem de maneira confiável e escalável, dividindo-os em partes menores chamadas de contêineres e organizando seu funcionamento em vários computadores. O Kubernetes também oferece recursos automáticos de escalonamento para lidar com a demanda crescente.
- O Kubernets é um orquestrador de containers
 - Permite criar e gerenciar um cluster permitindo que as aplicações nos containers sejam escaláveis
 - Permite a autodisponibilidade de containers sendo criar, replicar, e religar caso uma aplicação falhe

▼ Entendendo o Kubernetes

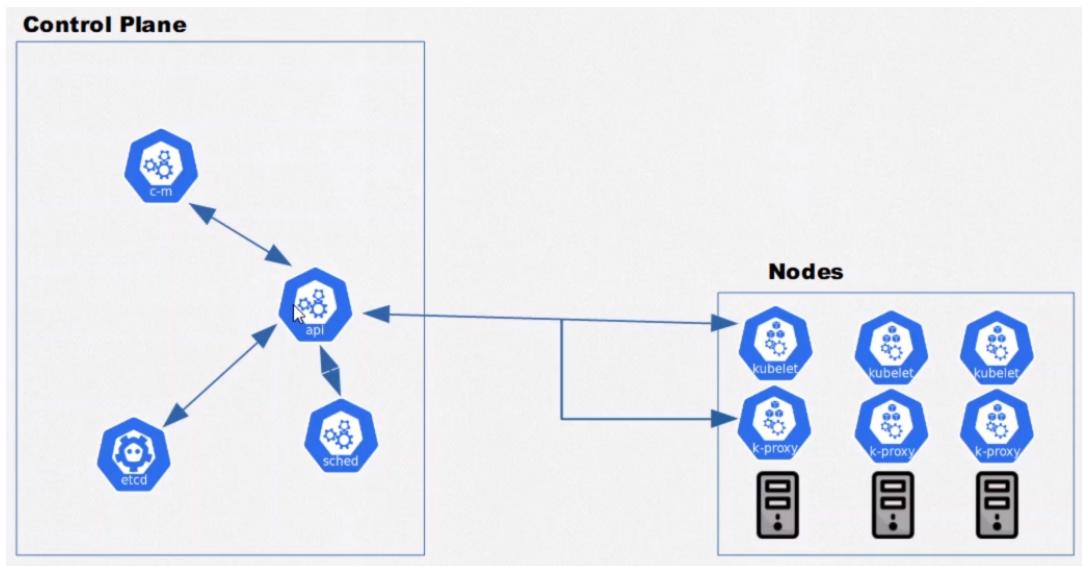
- **O Kubernetes é capaz de reiniciar aplicações automaticamente em caso de falhas.**

▼ A Arquitetura do Kubernetes

- O Kubernetes possui recursos que compõem sua arquitetura e permitem configurar nossa aplicação em containers da melhor forma



- Entendendo o funcionamento do Cluster



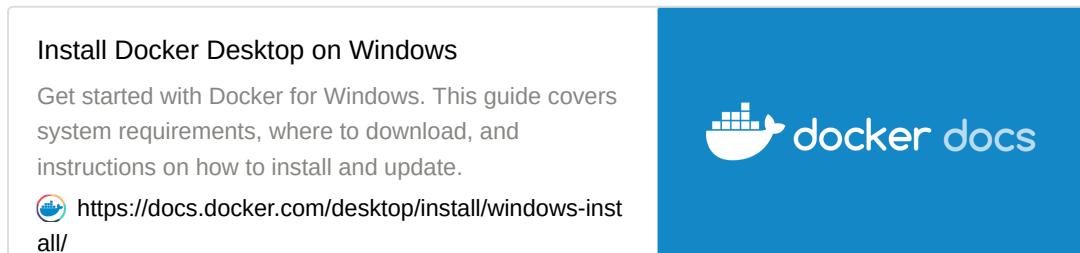
- Master (Control plane) -> Gerencia o cluster, mantem e atualiza o estado desejado, receber e executar novos comandos
 - Componentes da master
 - API - recebe os comandos
 - Kubectl - cria, le, att, remove
 - c-m - mantem e atualiza os estados
 - sched - determina as execuções do cluster
 - etcd - armazena os dados do cluster (chave : valor)
 - Nodes -> Executa as aplicações

- Componentes dos nodes
 - kubelet - agent de execução
 - k-proxy - comunicação dos nodes

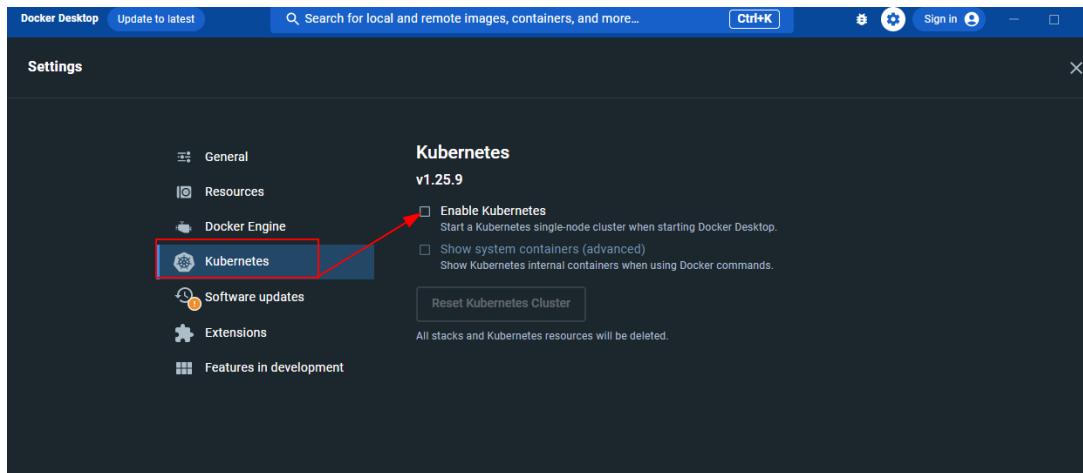
▼ Criando o cluster

▼ Inicializando o cluster no Windows

- O proprio docker desktop do windows possui uma ferramenta que permite utilizar o kubernets



- Realize a instalação normal conforme o instalador, após instalação reinicie o computador
- Vamos abrir o docker desktop normalmente, e vamos em configurações > Kubernetes e habilitar



- Após habilitar, se formos no proprio powershell poderemos validar que o cluster já se encontra em funcionamento utilizando o comando `kubectl get nodes`, irá mostrar o docker-desktop como master

▼ Kubernetes e windows

- Utilizando o próprio Docker Desktop, conseguimos inicializar um cluster.

▼ Instalando as ferramentas no linux

Os comandos necessários para a instalação e inicialização do cluster no Linux podem ser obtidos abaixo:

Primeiro para o kubectl:

- `sudo apt-get update`

```
lk47@Latitude-E6410:~$ sudo apt-get update
Hit:1 http://br.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://br.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://br.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:5 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:6 http://packages.microsoft.com/repos/code stable InRelease
Hit:7 https://packages.cloud.google.com/apt cloud-sdk InRelease
Hit:8 https://packages.microsoft.com/repos/azure-cli jammy InRelease
Hit:9 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:10 https://packages.microsoft.com/ubuntu/22.04/prod jammy InRelease
Reading package lists... Done
```

- `curl -LO "https://dl.k8s.io/release/ $(curl -L -s https://dl.k8s.io/release/stable.txt) /bin/linux/amd64/kubectl"`

```
lk47@Latitude-E6410:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
% Total    % Received % Xferd  Average Speed   Time   Time     Time Current
          Dload  Upload Total Spent   Left Speed
100  138  100  138    0     0   474      0 --:--:-- --:--:-- 475
100 46.9M  100 46.9M    0     0  7733k      0  0:00:06  0:00:06 --:--:-- 9697k
```

- `curl -LO "https://dl.k8s.io/ $(curl -L -s https://dl.k8s.io/release/stable.txt) /bin/linux/amd64/kubectl.sha256"`

```
lk47@Latitude-E6410:~$ curl -LO "https://dl.k8s.io/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
% Total    % Received % Xferd  Average Speed   Time   Time     Time Current
          Dload  Upload Total Spent   Left Speed
100  138  100  138    0     0   459      0 --:--:-- --:--:-- 458
100   64  100   64    0     0   122      0 --:--:-- --:--:-- 122
```

- `echo "$(cat kubectl.sha256) kubectl" | sha256sum --check`

```
lk47@Latitude-E6410:~$ echo "$(cat kubectl.sha256) kubectl" | sha256sum --check
kubectl: OK
```

- `sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl`

```
lk47@Latitude-E6410:~$ kubectl version --client
WARNING: This version information is deprecated and will be replaced with the output from kubectl version --short. Use --output=yaml|json to get the full version.
Client Version: version.Info{Major:"1", Minor:"27", GitVersion:"v1.27.3", GitCommit:"25b4e43193bcda6c7328a6d147b1fb73a33f1598", GitTreeState:"clean", BuildDate:"2023-06-14T09:53:42Z", GoVersion:"go1.20.5", Compiler:"gc", Platform:"linux/amd64"}
Kustomize Version: v5.0.1
```

Agora para o minikube:

- `curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64`

```
lk47@Latitude-E6410:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
% Total    % Received % Xferd  Average Speed   Time   Time     Current
          Dload  Upload   Total Spent    Left  Speed
100 80.0M  100 80.0M    0     0  4355k      0  0:00:18  0:00:18  4530k
```

- `sudo install minikube /usr/local/bin/`

```
lk47@Latitude-E6410:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

▼ Inicializando o cluster no linux

- Precisaremos instalar manualmente o kubectl
- Podemos seguir o procedimento conforme o proprio site do kubernetes

Install and Set Up kubectl on Linux

Before you begin You must use a kubectl version that is within one minor version difference of your cluster. For example, a v1.27 client can communicate with v1.26,

 <https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/>



- Após instalação precisaremos instalar o minikube pois não temos o cluster ainda

Hello Minikube

This tutorial shows you how to run a sample app on Kubernetes using minikube. The tutorial provides a container image that uses NGINX to echo back all the

 <https://kubernetes.io/docs/tutorials/hello-minikube/>



- Minikube cria o ambiente virtualizado com o cluster para nós já utilizarmos
- Podemos seguir a instalação conforme o site

```
minikube start
minikube is local Kubernetes
🌐 https://minikube.sigs.k8s.io/docs/start/
```

- Agora após a instalação do minikube, podemos iniciar a criação de um cluster usando o comando `minikube start`, porém precisaremos especificar um drive de virtualização, no caso temos o virtualbox já instalado em nossa maquina de laboratorio, sendo assim vamos especificar o driver no comando

- `minikube start --vm-driver=virtualbox`

```
lk47@Latitude-E6410:~$ minikube start --vm-driver=virtualbox
🕒 minikube v1.30.1 on Ubuntu 22.04
🌟 Using the virtualbox driver based on user configuration
🕒 Downloading VM boot image ...
> minikube-v1.30.1-amd64.iso....: 65 B / 65 B [=====] 100.00% ? p/s 0s
> minikube-v1.30.1-amd64.iso: 282.84 MiB / 282.84 MiB 100.00% 5.73 MiB p/
🕒 Starting control plane node minikube in cluster minikube
🕒 Downloading Kubernetes v1.26.3 preload ...
> preloaded-images-k8s-v18-v1...: 397.02 MiB / 397.02 MiB 100.00% 4.79 Mi
🕒 Creating virtualbox VM (CPUs=2, Memory=2200MB, Disk=20000MB) ...
🕒 Preparing Kubernetes v1.26.3 on Docker 20.10.23 ...
🕒 Generating certificates and keys ...
🕒 Booting up control plane ...
🕒 Configuring RBAC rules ...
🕒 Configuring bridge CNI (Container Networking Interface) ...
🕒 Using image gcr.io/k8s-minikube/storage-provisioner:v5
🕒 Verifying Kubernetes components...
🕒 Enabled addons: storage-provisioner, default-storageclass
🕒 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

- Após a instalação podemos rodar o `kubectl get nodes` que irá no trazer as informações do nó

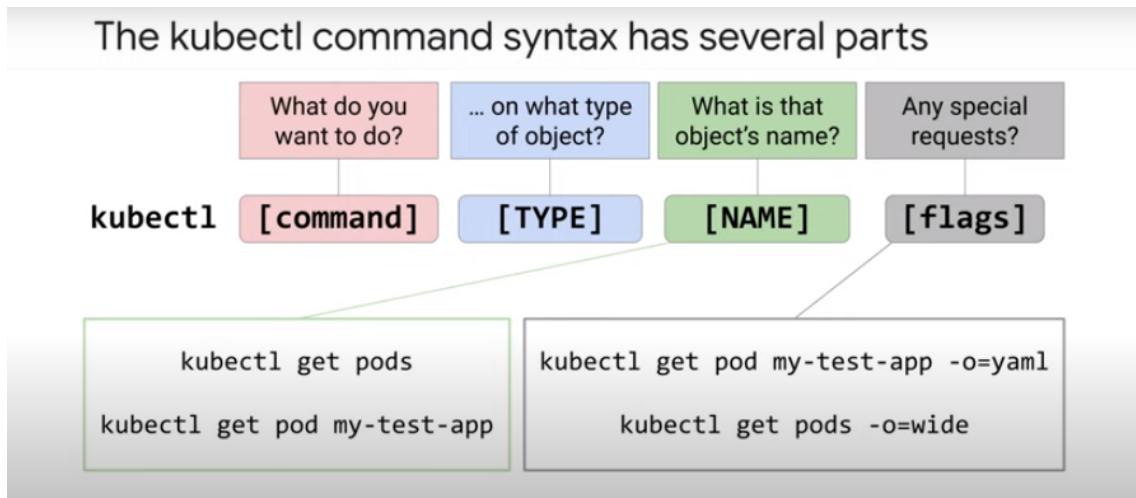
```
lk47@Latitude-E6410:~$ kubectl get nodes
NAME      STATUS   ROLES      AGE      VERSION
minikube  Ready    control-plane  73s    v1.26.3
```

- No caso a unica diferença com o windows em sí é que sempre que reiniciar o computador teremos que usar o comando `minikube start --vm-drive=<Nome do driver de virtualização>`

▼ Kubernetes e linux

- Precisamos instalar o minikube e o kubectl manualmente.

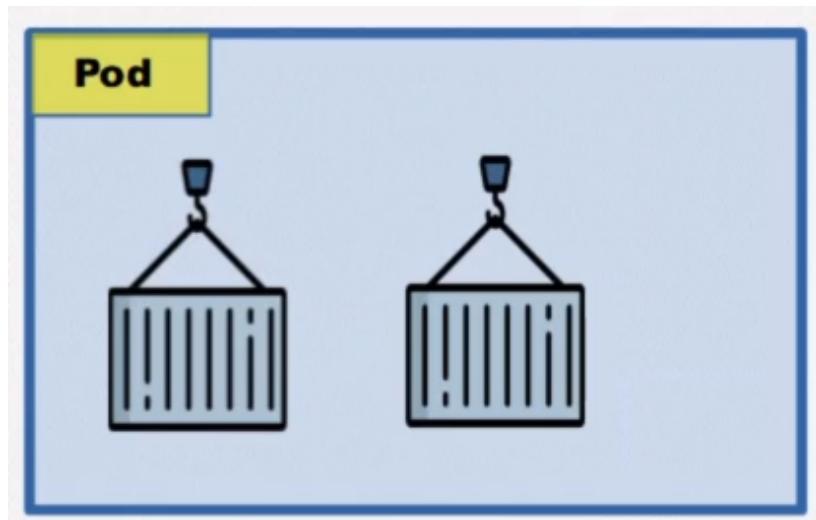
▼ Comandos Kubectl



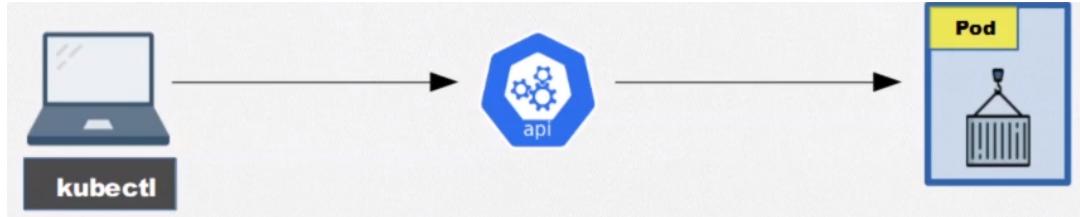
▼ Criando e entendendo pods

▼ Entendendo o que são pods

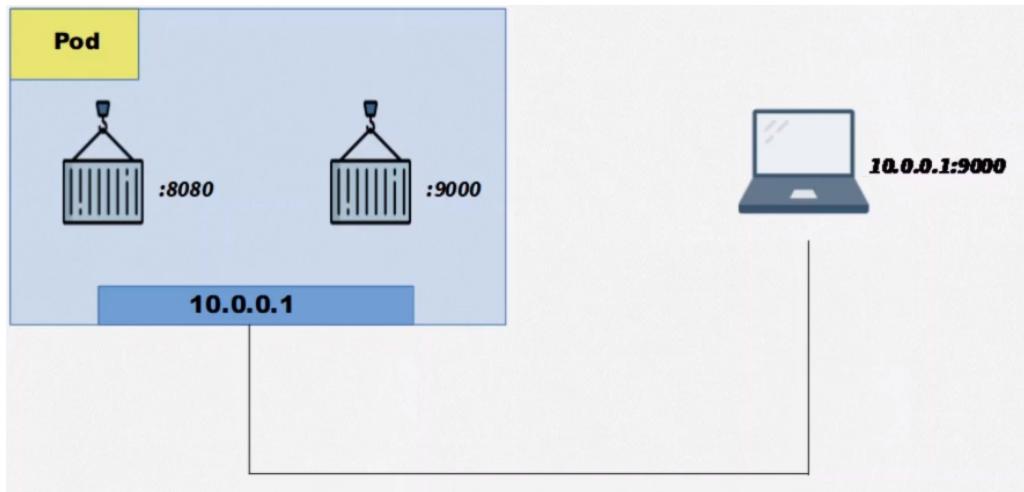
- No kubernetes não gerenciamos os containers e sim os pods
- Pods são conjuntos de um ou mais containers



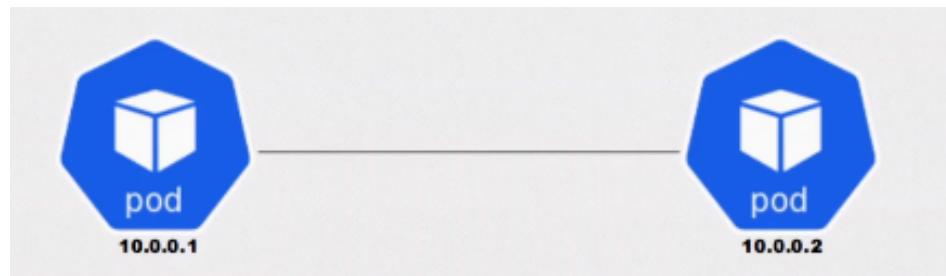
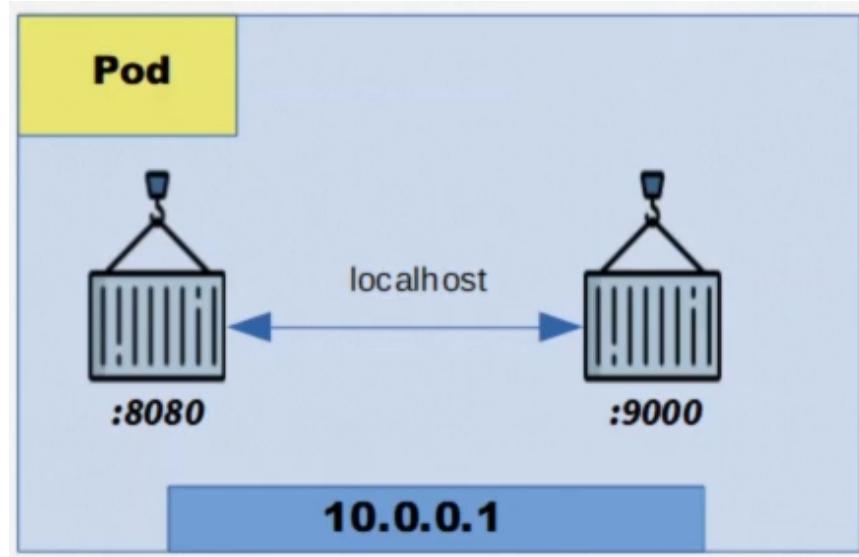
- A comunicação através do kubectl com a API irá direcionar para um pod e não para o container em si



- Cada pod irá possuir um IP (No caso não será mais o container em si)
 - Dessa forma podemos ter um pod com um IP específico (ex: 10.0.0.1) e no pod poderá haver dois containers em portas diferentes (ex: container-1 na porta 8080 e container-2 na porta 9000)



- Caso um container falhe, o pod irá parar definitivamente, e o kubernetes poderá subir um novo que terá um novo IP
- Os pods são efêmeros, ou seja podem ser substituídos a qualquer momento
- Caso um pod que contenha 2 containers e um dos containers falhe, o pod será considerado com a saúde ruim, porém caso os 2 containers venham a falhar o pod irá finalizar e outro será criado em seu lugar
- Um pouco mais sobre redes
 - Compartilham namespaces de rede e IPC
 - Ou seja os containers podem se comunicar diretamente via localhost ou com outros pods que se incluem na rede na qual se localiza



- Os pods também podem compartilhar volumes

▼ Pods e redes

- Como possuem IPs diferentes, containers em pods diferentes podem utilizar o mesmo número de porta.
- Containers dentro de um mesmo pod conseguem se comunicar via localhost.

▼ O primeiro pod

- Vamos usar o comando para criarmos um pod
 - `kubectl run nginx-pod --image=nginx:latest` (é um pouco semelhante ao docker run) → No caso com este comando estaremos criando um pod com a versão mais recente do nginx

```
lk47@Latitude-E6410:~$ kubectl run nginx-pod --image=nginx:latest
pod/nginx-pod created
```

- Rodando o comando `kubectl get pods` iremos obter os pods que estão sendo criados ou que estão em andamento

```
lk47@Latitude-E6410:~$ kubectl get pods
NAME        READY   STATUS            RESTARTS   AGE
nginx-pod   0/1    ContainerCreating   0          19s
```

- `kubectl get pods --watch` nos mostra em tempo real

```
lk47@Latitude-E6410:~$ kubectl get pods --watch
NAME        READY   STATUS            RESTARTS   AGE
nginx-pod   0/1    ContainerCreating   0          26s
nginx-pod   1/1    Running           0          29s
```

- `kubectl describe pod nginx-pod` → Irá nos trazer informações sobre o pod, como processo de criação, IP, labels, nome, versão etc.

```
lk47@Latitude-E6410:~$ kubectl describe pod nginx-pod
Name:           nginx-pod
Namespace:      default
Priority:      0
Service Account: default
Node:          minikube/192.168.59.100
Start Time:    Thu, 15 Jun 2023 09:59:42 -0300
Labels:         run=nginx-pod
Annotations:   <none>
Status:        Running
IP:            10.244.0.3
IPs:
  IP: 10.244.0.3
Containers:
  nginx-pod:
    Container ID: docker://b165977c9ef5f06c6fce03c9e35889f1aa07f93dfb4dc2014ccb3bd49366eb0
    Image:        nginx:latest
    Image ID:    docker-pullable://nginx@sha256:593dac25b7733ffb7afe1a72649a43e574778bf025ad60514ef40f6b5d60
    Ports:        6247/TCP
    Port:        <none>
    Host Port:   <none>
    State:       Running
      Started:   Thu, 15 Jun 2023 10:00:10 -0300
    Ready:       True
    Restart Count: 0
    Environment: <none>
```

Events:					Etapas da criação
Type	Reason	Age	From	Message	
Normal	Scheduled	107s	default-scheduler	Successfully assigned default/nginx-pod to minikube	
Normal	Pulling	105s	kubelet	Pulling image "nginx:latest"	
Normal	Pulled	79s	kubelet	Successfully pulled image "nginx:latest" in 26.235623546s (26.235839s including waiting)	
Normal	Created	79s	kubelet	Created container nginx-pod	
Normal	Started	79s	kubelet	Started container nginx-pod	

- `kubectl edit pod nginx-pod` → Permite editarmos algumas informações do pod
 - Ele irá abrir um bloco de notas, com varias informações no formato YAML, no caso vamos alterar a versão de latest para 1.0 em image, no

caso do linux ele irá abrir o vim

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2023-06-15T12:59:42Z"
  labels:
    run: nginx-pod
  name: nginx-pod
  namespace: default
  resourceVersion: "503"
  uid: 08fc83ce-die6-4258-8215-7986c359ffda
spec:
  containers:
    - image: nginx:latest
      imagePullPolicy: Always
      name: nginx-pod
      resources: {}
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
    volumeMounts:
      - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
        name: kube-api-access-7cd4b
        readOnly: true
```

```
spec:
  containers:
    - image: nginx:latest
      imagePullPolicy: Always
      name: nginx-pod
      resources: {}
```

```
spec:
  containers:
    - image: nginx:1.0
      imagePullPolicy: Always
      name: nginx-pod
      resources: {}
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
    volumeMounts:
      - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
        name: kube-api-access-7cd4b
        readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: minikube
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
:wq!
```

- Validando novamente do `kubectl get pods` podemos ver que o pod quebrou

NAME	READY	STATUS	RESTARTS	AGE
nginx-pod	0/1	ImagePullBackOff	0 (58s ago)	12m

- Se rodarmos o `kubectl describe pod` podemos verificar que ele tentou baixar a imagem porém teve um erro

Events:				
Type	Reason	Age	From	Message
Normal	Scheduled	13m	default-scheduler	Successfully assigned default/nginx-pod to minikube
Normal	Pulling	13m	kubelet	Pulling image "nginx:latest"
Normal	Pulled	12m	kubelet	Successfully pulled image "nginx:latest" in 26.2356s (26.235645839s including waiting)
Normal	Created	12m	kubelet	Created container nginx-pod
Normal	Started	12m	kubelet	Started container nginx-pod
Normal	Killing	96s	kubelet	Container nginx-pod definition changed, will be restarted
Normal	Pulling	48s (x3 over 96s)	kubelet	Pulling image "nginx:1.0"
Warning	Failed	47s (x3 over 94s)	kubelet	Failed to pull image "nginx:1.0": rpc error: code = Unknown desc = Error response from daemon: manifest for nginx:1.0 not found: manifest unknown: manifest unknown
Warning	Failed	47s (x3 over 94s)	kubelet	Error: ErrImagePull
Warning	BackOff	19s (x2 over 34s)	kubelet	Back-off restarting failed container nginx-pod in pod nginx-pod_default(08fc83ce-d1e6-4258-8215-7986c359ffda)
Normal	BackOff	8s (x3 over 93s)	kubelet	Back-off pulling image "nginx:1.0"
Warning	Failed	8s (x3 over 93s)	kubelet	Error: ImagePullBackOff

- No caso essa forma de implatação em nosso cluster se chama Imperativa no qual definimos o pod e sua versão logo no primeiro contato.
 - `kubectl run`
- A outra forma é denominada como declarativa que seria uma declaração da imagem na qual criamos, juntamente com as especificações que definimos ou seja seria um pod configurável

▼ Comandos do kubectl

- kubectl run, kubectl describe, kubectl edit**
 - Cria, informa, e edit tudo referente ao pod

▼ Onde imagens são armazenadas?

- O scheduler definirá em qual node a imagem será armazenada. As imagens podem ser baixadas de repositórios como o docker hub, que serão armazenadas localmente em cada node, não sendo compartilhada por padrão entre todos os membros do cluster

▼ Criando pods de maneira declarativa

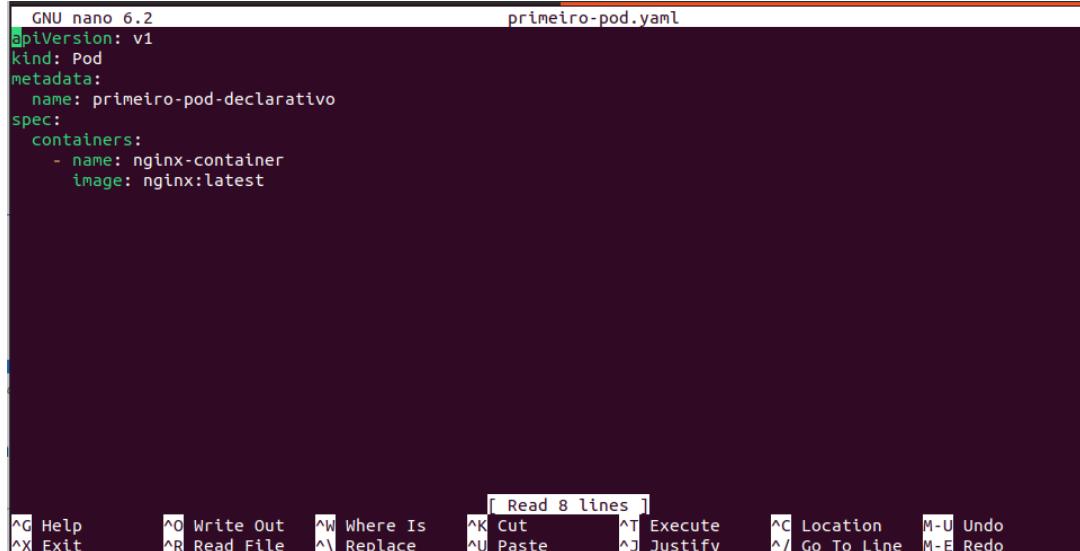
- Vamos criar o arquivo denominado 'primeiro-pod' o mesmo pode ser YAML ou JSON

- No caso é mais comum usarmos o YAML

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ touch primeiro-pod.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ ls
primeiro-pod.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ █
```

- Vamos definir da seguinte forma as informações do YAML
 - **apiversion:** v1 -> na doc podemos ver que ela foi dividida em diversas partes (alfa, beta, estável), vamos usar a stable v1
 - **kind:** pod -> recurso que queremos criar
 - **metadada:** -> Informações referente ao pod
 - **name:** primeiro-pod-declarativo
 - **spec:** -> especificações
 - containers:**
 - **name:** nginx-container
 - image:** nginx:latest -> Imagem do repo

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ nano primeiro-pod.yaml █
```



```
GNU nano 6.2                                         primeiro-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: primeiro-pod-declarativo
spec:
  containers:
    - name: nginx-container
      image: nginx:latest
```

The screenshot shows the nano text editor displaying the YAML configuration for the first pod. The file content is as follows:

```
GNU nano 6.2                                         primeiro-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: primeiro-pod-declarativo
spec:
  containers:
    - name: nginx-container
      image: nginx:latest
```

The bottom of the screen shows the nano editor's command-line interface with various keyboard shortcuts for navigation and editing.

- No caso criamos o arquivo com a ferramenta nano do linux, mas é possível utilizar outros editores de texto para a criação do objeto sem problemas.

- O arquivo ficará da seguinte forma

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ cat primeiro-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: primeiro-pod-declarativo
spec:
  containers:
    - name: nginx-container
      image: nginx:latest
```

- Agora para subirmos o pod usamos o comando `kubectl apply -f primeiro-pod.yaml`

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl apply -f primeiro-pod.yaml
pod/primeiro-pod-declarativo created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$
```

- Usando o `kubectl get pods` podemos acompanhar

NAME	READY	STATUS	RESTARTS	AGE
nginx-pod	0/1	ImagePullBackOff	0 (22m ago)	34m
primeiro-pod-declarativo	1/1	Running	0	34s

- Agora com o arquivo de definição vamos na versão da imagem e vamos definir como 1.0, assim a imagem será alterada

```
apiVersion: v1
kind: Pod
metadata:
  name: primeiro-pod-declarativo
spec:
  containers:
    - name: nginx-container
      image: nginx:1.0
```

- Rodamos novamente o comando `kubectl apply -f`

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl apply -f primeiro-pod.yaml
pod/primeiro-pod-declarativo configured
```

- E acompanhamos com o `kubectl get pods --watch`

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl get pods --watch
NAME                  READY   STATUS      RESTARTS   AGE
nginx-pod             0/1     ImagePullBackOff  0 (30m ago)  41m
primeiro-pod-declarativo  0/1     ImagePullBackOff  0 (3m6s ago)  8m1s
primeiro-pod-declarativo  0/1     CrashLoopBackOff  0 (3m7s ago)  8m2s
```

- Podemos ver que irá ocorre um erro, então vamos alterar novamente agora para uma versão que exista no caso 'stable'

```
apiVersion: v1
kind: Pod
metadata:
  name: primeiro-pod-declarativo
spec:
  containers:
    - name: nginx-container
      image: nginx:stable
```

- Agora vamos aplicar e validar novamente

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl apply -f primeiro-pod.yaml
pod/primeiro-pod-declarativo configured
```

- Feito isso o mesmo voltará a ficar funcional

```
^L^C lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl get pods --watch
NAME                  READY   STATUS      RESTARTS   AGE
nginx-pod             0/1     ImagePullBackOff  0 (31m ago)  43m
primeiro-pod-declarativo  1/1     Running    1 (4m53s ago)  9m48s
```

- Porém agora vamos validar as informações do pod em events com o `kubectl describe pod`

```
Events:
Type  Reason  Age           From            Message
----  -----  --            --              --
Normal Scheduled  11m          default-scheduler  Successfully assigned default/primeiro-pod-declarativo to minikube
Normal Pulling   11m          kubelet         Pulling image "nginx:latest"
Normal Pulled    11m          kubelet         Successfully pulled image "nginx:latest" in 1.48499652s (1.485048769s including waiting)
Normal Created   11m          kubelet         Created container nginx-container
Normal Started   11m          kubelet         Started container nginx-container
Normal Killing   6m11s        kubelet         Container nginx-container definition changed, will be restarted
Normal Backoff   5m43s (x2 over 6m9s)  kubelet         Back-off pulling image "nginx:1.0"
Warning Failed   5m43s (x2 over 6m9s)  kubelet         Error: ImagePullBackOff
Normal Pulling   4m45s (x4 over 6m11s)  kubelet         Pulling image "nginx:1.0"
Warning Failed   4m43s (x4 over 6m9s)  kubelet         Failed to pull image "nginx:1.0": rpc error: code = Unknown desc = Error response from daemon: manifest for nginx:1.0 not found: manifest unknown: manifest unknown
Warning Failed   4m43s (x4 over 6m9s)  kubelet         Error: ErrImagePull
Warning Backoff  4m (X5 over 5m14s)   kubelet         Back-off restarting failed container nginx-container in pod primeiro-pod-declarativo default/c6753f1a-3694-46d5-8812-5071a1f35427)
```

- E podemos ver as informações sendo pontuadas do erro até o retorno da funcionalidade do pod

▼ Analisando arquivos YAML

- Podemos concluir com o YAML abaixo

```
apiVersion: v1
kind: Pod
spec:
  containers:
    - name: container-pod-1
      image: nginx:latest
```

- Não funcionará, faltou definir o `metadata` e o `name`.

▼ Iniciando o projeto

- `kubectl delete pod` -> Permite deletarmos pods existentes

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl get pods
NAME           READY   STATUS        RESTARTS   AGE
nginx-pod      0/1     ImagePullBackOff  0 (34m ago)  46m
primeiro-pod-declarativo 1/1     Running       1 (7m45s ago) 12m
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl delete pod nginx-pod
pod "nginx-pod" deleted
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$
```

- `kubectl delete -f <arquivo de definição>` irá excluir o pod através do arquivo de definição

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl delete -f primeiro-pod.yaml
pod "primeiro-pod-declarativo" deleted
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl get pods
No resources found in default namespace.
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$
```

- Vamos criar um portal de notícias
- Primeiramente vamos criar um yaml chamado portal-noticias

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ touch portal-noticias.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ nano portal-noticias.yaml
```

- Vamos definir

```

apiVersion: v1
kind: Pod
metadata:
  name: portal-noticias
spec:
  containers:
    - name: portal-noticias-container
      image: aluracursos/portal-noticias:1

```

```

GNU nano 6.2                                         portal-noticias.yaml *
apiVersion: v1
kind: Pod
metadata:
  name: portal-noticias
spec:
  containers:
    - name: portal-noticias-container
      image: aluracursos/portal-noticias:1

```

- Feito isso vamos aplicar e criar

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl apply -f portal-noticias.yaml
pod/portal-noticias created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
portal-noticias  0/1     ContainerCreating   0          19s
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl get pods --watch
NAME        READY   STATUS    RESTARTS   AGE
portal-noticias  0/1     ContainerCreating   0          26s
portal-noticias  1/1     Running   0          47s

```

- Vamos validar o IP do pod com o describe

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
portal-noticias  1/1     Running   0          110s
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl describe pod portal-noticias
Name:           portal-noticias
Namespace:      default
Priority:       0
Service Account: default
Node:          minikube/192.168.59.100
Start Time:    Thu, 15 Jun 2023 10:52:48 -0300
Labels:         <none>
Annotations:   <none>
Status:        Running
IP:            10.244.0.5
IPs:
  IP: 10.244.0.5

```

- Vamos copiar o IP e tentar executar o IP no browser, no caso iremos tomar um erro



This site can't be reached

10.244.0.5 took too long to respond.

Try:

- Checking the connection
- Checking the proxy and the firewall

ERR_CONNECTION_TIMED_OUT

[Details](#) [Reload](#)

- Pelo IP do pod no navegador não conseguiremos ver a aplicação pois o IP se trata de algo interno do cluster (no caso o minikube em si)
- Para validarmos se está tudo funcionando dentro do pod vamos usar o comando

- `kubectl exec -it portal-noticias -- bash`

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl exec -it portal-noticias -- bash
root@portal-noticias:/var/www/html#
```

- Assim estaremos acessando o pod, vamos enviar uma requisição para o localhost com o curl e ele irá exibir todo o conteúdo

```
root@portal-noticias:/var/www/html# curl localhost
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Alura | Notícias</title>
  <!-- Tell the browser to be responsive to screen width -->
  <meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" name="viewport">
  <!-- Bootstrap 3.3.7 -->
  <link rel="stylesheet" href="theme/bower_components/bootstrap/dist/css/bootstrap.min.css">
  <!-- Font Awesome -->
  <link rel="stylesheet" href="theme/bower_components/font-awesome/css/font-awesome.min.css">
  <!-- Ionicons -->
  <link rel="stylesheet" href="theme/bower_components/Ionicons/css/ionicons.min.css">
  <!-- Theme style -->
  <link rel="stylesheet" href="theme/dist/css/AdminLTE.min.css">
  <!-- AdminLTE Skins. Choose a skin from the css/skins
       folder instead of downloading all of them to reduce the load. -->
  <link rel="stylesheet" href="theme/dist/css/skins/_all-skins.min.css">

  <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->
  <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
  <!--[if lt IE 9]>
    <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
    <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
  <![endif]-->

  <!-- Google Font -->
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,600,700,300italic,400italic,600ita
">
</head>
<!-- ADD THE CLASS layout-top-nav TO REMOVE THE SIDEBAR. -->
<body class="hold-transition skin-blue layout-top-nav">
<div class="wrapper">
```

- Para que possamos expor corretamente, teremos que usar os services

▼ Pods e contêineres

- O pod, só é considerado encerrado quando todos os contêineres dentro do pod param de funcionar

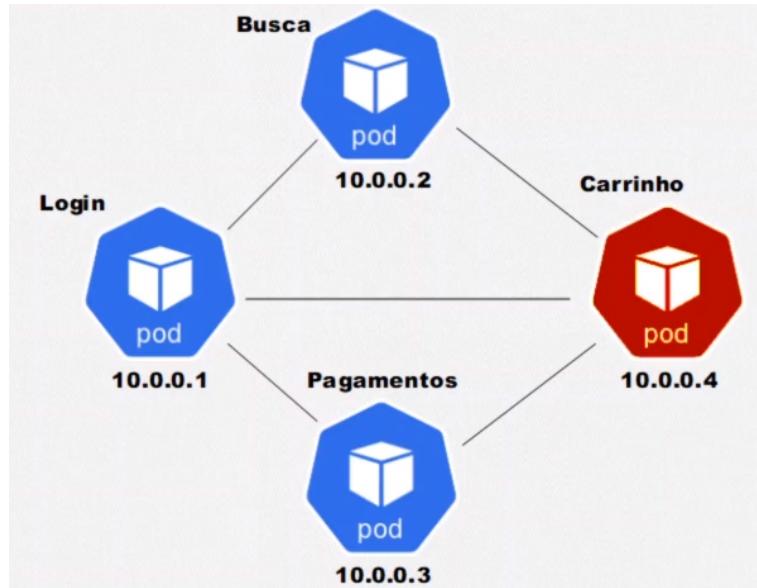
▼ Expondo pods com services

▼ Conhecendo services

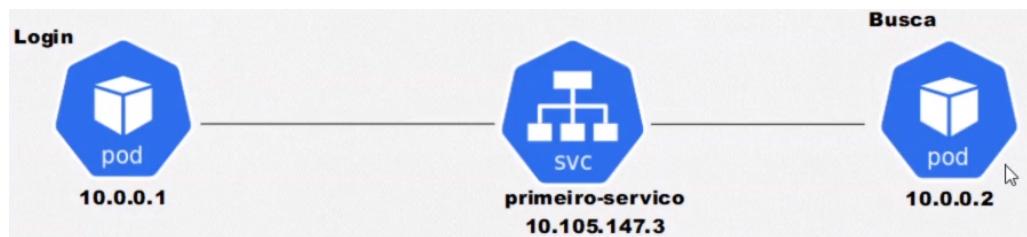
- No nosso cluster poderá ocorrer comunicação entre os pods através da rede existente
- Caso realizarmos a exclusão de um pod o IP pode mudar
 - Usando o comando kubectl get pod -o wide -> obtemos informações abrangentes do pod
 - Vamos excluir o pod e criar novamente
 - Podemos validar que o IP do pod foi alterado

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE     IP           NODE   NOMINATED NODE   READINESS GATES
portal-noticias   1/1    Running   0          7m30s   10.244.0.5   minikube   <none>        <none>
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl delete pod portal-noticias
pod "portal-noticias" deleted
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl apply -f portal-noticias.yaml
pod/portal-noticias created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE     IP           NODE   NOMINATED NODE   READINESS GATES
portal-noticias   1/1    Running   0          4s     10.244.0.6   minikube   <none>        <none>
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$
```

- Caso uma aplicação comunique entre pods, como saber se o pod permanecerá se comunicando com os demais caso o mesmo seja recriado?



- Para isso o Kubernetes possui o services (SVC)
 - Abstrações para expor aplicações executando em um ou mais pods
 - Proveem IP's fixos para comunicação
 - Proveem um DNS para um ou mais pods
 - São capazes de fazer balanceamento de carga
- Em um exemplo
 - O pod login para se comunicar sempre com o pod de busca em uma aplicação, teremos um SVC que será o intermediario dessa comunicação, impedindo que caso o pod seja recriado o mesmo para de se comunicar com os demais



- SVC possui 3 tipos
 - ClusterIP
 - NodePort

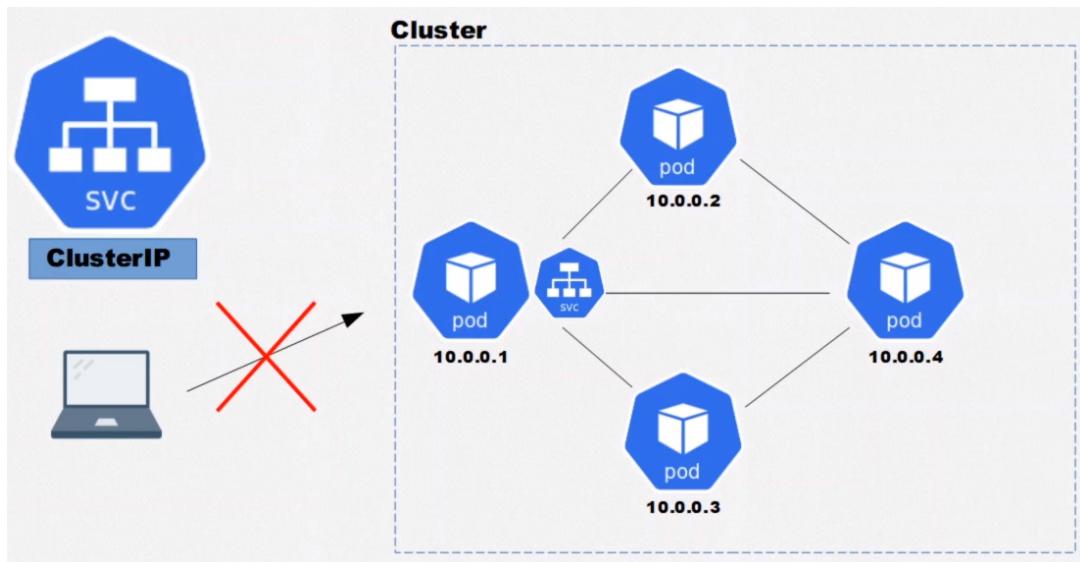
- LoadBalancer

▼ Vantagens de services

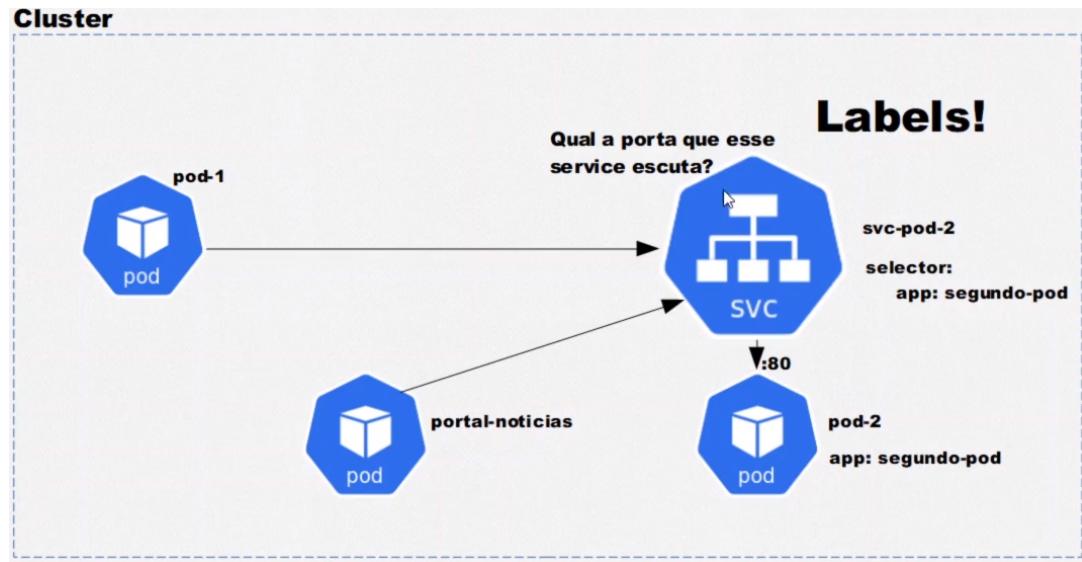
- Fazem o balanceamento de carga.
- Proveem IP's fixos para comunicação.

▼ Criando um cluster IP

- Cluster IP serve para fazer a comunicação de diferentes pods dentro de um mesmo cluster
 - Os Services não funcionam como mão dupla, se um pod possui um service e os outros não, o pod que possui o svc pode ser acessado pelos outros pods porém os que não possuem o SVC não poderão ter o acesso
 - Se tentarmos acessar o pod de fora do cluster usando o SVC não será possível pois se trata de uma comunicação interna do cluster



- Vamos criar dois pods nginx, o objetivo será criar um SVC que pertencerá ao segundo pod, e os demais terão acesso a este pod pelo SVC



- Vamos criar dois arquivos pod1.yaml e pod2.yaml
- Pod1
 - No caso estamos especificando agora em 'ports' o 'containerPort' no qual nossa aplicação irá rodar que no caso é a porta 80

```

apiVersion: v1
kind: Pod
metadata:
  name: pod1
spec:
  containers:
    - name: container-pod1
      image: nginx:latest
      ports:
        - containerPort: 80
  
```

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ ls
pod1.yaml  pod2.yaml  primeiro-pod.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ cat pod1.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod1
spec:
  containers:
    - name: container-pod1
      image: nginx:latest
      ports:
        - containerPort: 80
  
```

No caso criei esta pasta pods para salvar os yaml's desse projeto.

- Pod2:

- Já neste segundo pod adicionamos uma label, que nada mais é do que uma marcação na qual irá servir para que possamos pontuar no YAML do SVC para onde as requisições deverão ser encaminhadas, portanto é de extrema importância utilizá-la

```
apiVersion: v1
kind: Pod
metadata:
  name: pod2
  labels:
    app: segundo-pod
spec:
  containers:
    - name: container-pod2
      image: nginx:latest
      ports:
        - containerPort: 80
```

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ cat pod2.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod2
  labels:
    app: segundo-pod
spec:
  containers:
    - name: container-pod2
      image: nginx:latest
      ports:
        - containerPort: 80
```

- Agora seguiremos com a criação do YAML do SVC

```
apiVersion: v1
kind: Service
metadata:
  name: svc-pod2
spec:
  type: ClusterIP
  selector:
    app: segundo-pod
  ports:
    - port: 80
```

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ nano svc-pod-2.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ cat svc-pod-2.yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-pod2
spec:
  type: ClusterIP
  selector:
    app: segundo-pod
  ports:
    - port: 80
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ █

```

Criada pasta para os svcs desse projeto

- Neste caso adicionamos o selector apontando para a label
- E adicionamos a ports que irá apontar para em qual porta as requisições deverão ser encaminhadas no pod
- Com os arquivos criados vamos aplicar os objetos no cluster

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ kubectl apply -f pod1.yaml
pod/pod1 created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ kubectl apply -f pod2.yaml
pod/pod2 created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ cd ..
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ cd svcs/
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl apply -f svc-pod-2.yaml
service/svc-pod2 created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
pod1     1/1     Running   0          33s
pod2     1/1     Running   0          26s
portal-noticias 1/1     Running   1 (24h ago) 26h
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get svc
NAME      TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE
kubernetes  ClusterIP  10.96.0.1  <none>        443/TCP   27h
svc-pod2  ClusterIP  10.110.144.74 <none>        80/TCP    18s
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ █

```

- Feito isso podemos validar o describe do pod2, perceba a label aparente

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl describe pod pod2
Name:           pod2
Namespace:      default
Priority:       0
Service Account: default
Node:           minikube/192.168.59.100
Start Time:     Fri, 16 Jun 2023 13:18:19 -0300
Labels:         app=segundo-pod
Annotations:    <none>
Status:         Running
IP:             10.244.0.10
IPs:
  IP: 10.244.0.10
Containers:

```

- Podemos rodar o comando `kubectl get svc` ou `service` que irá apontar os services do cluster

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	27h
svc-pod2	ClusterIP	10.110.144.74	<none>	80/TCP	4m33s

- Kubernetes é padrão
- Lá está nosso SVC com a port 80 definida conforme especificado
- Vamos acessar o pod1 e vamos enviar uma requisição para o pod2 através do IP do SVC `10.110.144.74` e a porta `80`

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get svc
NAME      TYPE      CLUSTER-IP        EXTERNAL-IP   PORT(S)    AGE
kubernetes  ClusterIP  10.96.0.1        <none>        443/TCP   27h
svc-pod2   ClusterIP  10.110.144.74    <none>        80/TCP    4m33s

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl exec -it pod1 -- bash
root@pod1:/# curl 10.110.144.74:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br />
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

- Vamos realizar o mesmo repetir o processo no pod do portal-noticias

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl exec -it portal-noticias -- bash
root@portal-noticias:/var/www/html# curl 10.110.144.74:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@portal-noticias:/var/www/html#
```

- Vamos realizar a exclusão do pod2 e vamos tentar encaminhar uma requisição novamente

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
pod1        1/1     Running   0          9m48s
pod2        1/1     Running   0          9m41s
portal-noticias  1/1     Running   1 (24h ago)  26h
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl delete pod pod2
pod "pod2" deleted
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl exec -it pod1 -- bash
root@pod1:/# curl 10.110.144.74:80
curl: (7) Failed to connect to 10.110.144.74 port 80 after 0 ms: Couldn't connect to server
root@pod1:/#
```

- Perceba que irá ocorrer um erro, que no caso é devido a falta do pod em si
- Se subirmos o pod novamente com a mesma label, o direcionamento ocorrerá novamente

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ kubectl apply -f pod2.yaml
pod/pod2 created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
pod1     1/1     Running   0          11m
pod2     1/1     Running   0          12s
portal-noticias 1/1     Running   1 (24h ago) 26h
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ kubectl exec -it pod1 -- bash
root@pod1:/# curl 10.110.144.74:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p><a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@pod1:/#
```

- A comunicação via DNS também é possível

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP      27h
svc-pod2     ClusterIP   10.110.144.74    <none>        80/TCP       11m
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl exec -it pod1 -- bash
root@pod1:/# curl svc-pod2
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>

```

- Podemos definir outra porta em nosso SVC sendo diferente da especificada no YAML do pod, que no caso precisaremos colocar a porta que o service irá ouvir e a porta na qual o mesmo irá despachar, ficando da seguinte forma
 - No caso definimos a porta de escuta do SVC a 9000 e a de destino a 80 conforme especificado no pod

```

apiVersion: v1
kind: Service
metadata:
  name: svc-pod-2
spec:
  type: ClusterIP
  selector:
    app: segundo-pod
  ports:
    - port: 9000
      targetPort: 80

```

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ ls
svc-pod-2.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ nano svc-pod-2.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ cat svc-pod-2.yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-pod-2
spec:
  type: ClusterIP
  selector:
    app: segundo-pod
  ports:
    - port: 9000
      targetPort: 80
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$
```

- Recriando novamente o SVC através do arquivo

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP     27h
svc-pod2   ClusterIP   10.110.144.74   <none>        80/TCP      17m
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl delete svc svc-pod2
service "svc-pod2" deleted
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP     27h
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl apply -f svc-pod-2.yaml
service/svc-pod-2 created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP     27h
svc-pod-2   ClusterIP   10.97.4.76      <none>        9000/TCP    4s
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$
```

- Vamos acessar o pod1 novamente e vamos encaminhar a requisição para a porta 9000

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get svc
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes  ClusterIP  10.96.0.1   <none>        443/TCP   27h
svc-pod-2  ClusterIP  10.97.4.76   <none>        9000/TCP   4s
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl exec -it pod1 -- bash
root@pod1:/# curl 10.97.4.76:9000
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@pod1:/#

```

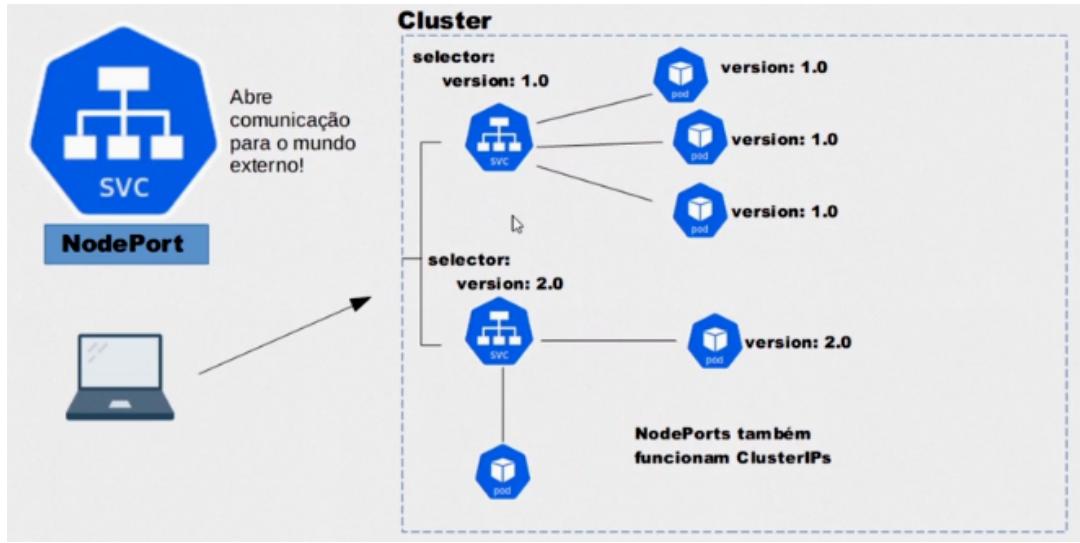
- No caso o IP do pod2 ouve na porta 80, mas o SVC na porta 9000, dessa forma precisamos especificar a 9000 em caso de encaminhamento requisição pelo service

▼ Linkando services e pods

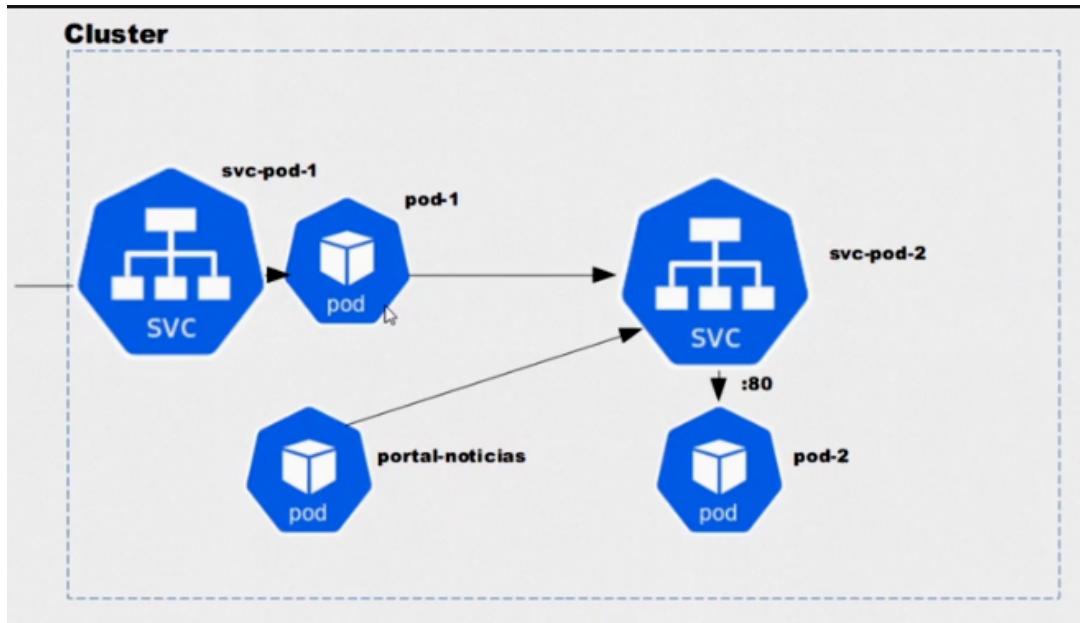
- Como um service sabe quais pods deve gerenciar?
 - Através de `labels` definidas no `metadata` e utilizando o campo `selector` no service.

▼ Criando um node port

- NodePort é o serviço que permite a comunicação com o externo



- Assim será possível acessar uma aplicação através do navegador
- O nodeport também funcionam como um clusterIP
- Com base no projeto anterior vamos criar um SVC que irá expor o nosso pod1 para o externo



- Vamos criar svc-pod-1
 - Vamos definir o port 80 e alteraremos o type em spec vamos add o selector como primeiro-pod

```
apiVersion: v1
kind: Service
metadata:
  name: svc-pod1
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
  selector:
    app: primeiro-pod
```

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ nano svc-pod-1.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ cat svc-pod-1.yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-pod1
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
  selector:
    app: primeiro-pod
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ █
```

- O YAML do pod1 deverá possuir a label `app:primeiro-pod`

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ nano pod1.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ cat pod1.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod1
  labels:
    app: primeiro-pod
spec:
  containers:
    - name: container-pod1
      image: nginx:latest
      ports:
        - containerPort: 80
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$
```

- Feito isso vamos subir novamente o pod1 e em seguida o svc-pod1

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ kubectl apply -f pod1.yaml
pod/pod1 configured
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
pod1     1/1     Running   0          118m
pod2     1/1     Running   0          107m
portal-noticias 1/1     Running   1 (26h ago) 28h
```

Só reconfiguramos para add a label

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/pods$ kubectl edit pod pod1
```

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"labels":{"app":"primeiro-pod"},"name":"pod1"},"spec":{"containers":[{"image":"nginx:latest","name":"container-pod1","ports":[{"containerPort":80}]}]}}
  creationTimestamp: "2023-06-16T16:18:12Z"
  labels:
    app: primeiro-pod
  name: pod1
  namespace: default
  resourceVersion: "17988"
  uid: e5e3e4cc-cc9c-44a8-9869-abff90065875
spec:
  containers:
  - image: nginx:latest
    imagePullPolicy: Always
    name: container-pod1
    ports:
    - containerPort: 80
      protocol: TCP
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
```

- Subindo o SVC

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ ls
svc-pod-1.yaml svc-pod-2.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl apply -f svc-pod-1.yaml
service/svc-pod1 created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get svc
NAME      TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
kubernetes  ClusterIP  10.96.0.1    <none>       443/TCP       29h
svc-pod-2  ClusterIP  10.97.4.76   <none>       9000/TCP      102m
svc-pod1   NodePort   10.96.106.236 <none>       80:30810/TCP  8s
```

- Vamos acessar o pod do portal-noticias e vamos encaminhar uma requisição para o pod1, irá apresentar funcionalidade semelhante ao clusterIP

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes  ClusterIP  10.96.0.1    <none>        443/TCP     29h
svc-pod-2  ClusterIP  10.97.4.76    <none>        9000/TCP     103m
svc-pod1   NodePort   10.96.106.236  <none>        80:30810/TCP  104s
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl exec -it portal-noticias -- bash
root@portal-noticias:/var/www/html# curl 10.96.106.236:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@portal-noticias:/var/www/html#

```

- Agora externamente precisaremos validar o IP Externo do nó e utilizar a porta que é bind da 80
 - No caso :

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes  ClusterIP  10.96.0.1    <none>        443/TCP     29h
svc-pod-2  ClusterIP  10.97.4.76    <none>        9000/TCP     103m
svc-pod1   NodePort   10.96.106.236  <none>        80:30810/TCP  104s

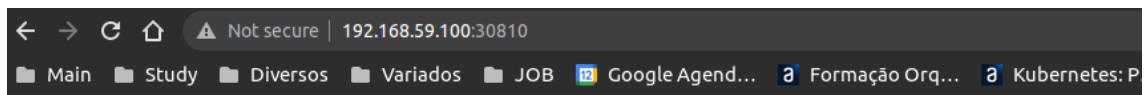
```

- Sendo assim se acessar o localhost + a porta bind, seremos encaminhados para o pod, isso no windows pois o docker-desktop irá configurar para o localhost
- No linux com o minikube usamos o comando `kubectl get nodes -o wide` e pegamos o Internal-IP + a porta bind

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get nodes -o wide
NAME      STATUS      ROLES      AGE      VERSION      INTERNAL-IP      EXTERNAL-IP      OS-IMAGE      KERNEL-VERSION      CONTAINER-RUNTIME
minikube  Ready      control-plane  29h      v1.26.3   192.168.59.100  <none>        Buildroot 2021.02.12  5.10.57       docker://
20.10.23
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ 

```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

- Podemos definir a porta bind adicionando a seguinte instrução no YAML

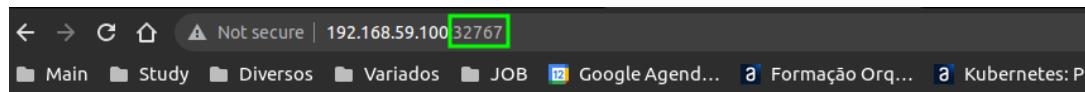
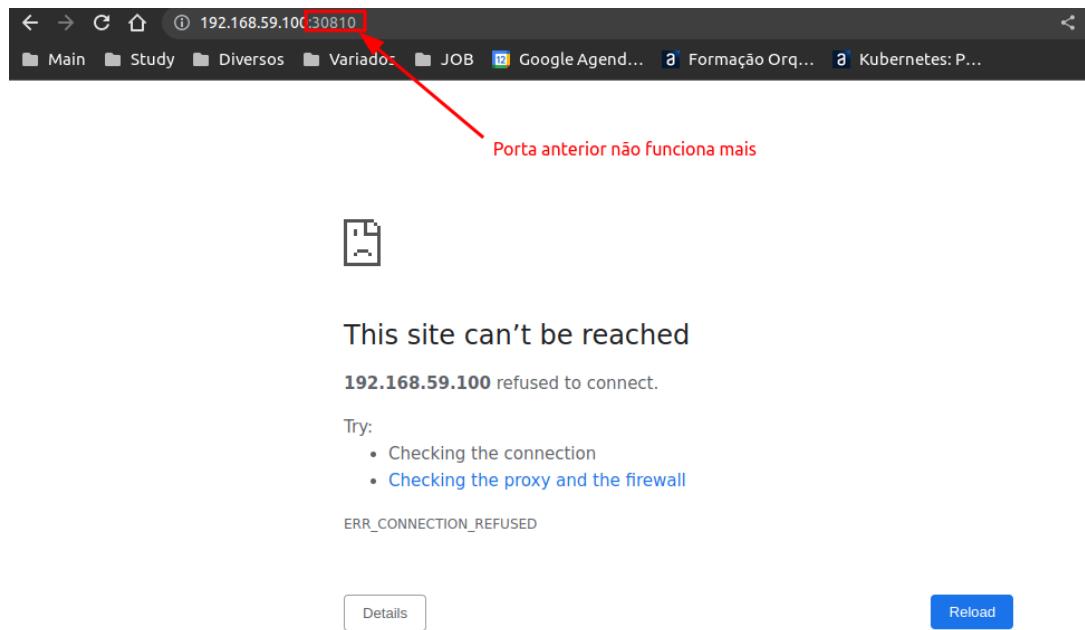
- `port: 80`

```
nodePort: # Podemos colocar de 3000 a 32767
```

```
GNU nano 6.2                                         svc-pod-1.yaml *
apiVersion: v1
kind: Service
metadata:
  name: svc-pod1
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 32767
  selector:
    app: primeiro-pod
```

- Aplicando novamente podemos ver que a porta bind irá mudar

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl apply -f svc-pod-1.yaml
service/svc-pod1 configured
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/svcs$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes  ClusterIP  10.96.0.1      <none>        443/TCP      29h
svc-pod-2  ClusterIP  10.97.4.76      <none>        9000/TCP      115m
svc-pod1   NodePort   10.96.106.236  <none>        80:32767/TCP  13m
```



▼ Praticando NodePort's

NAME	TYPE	CLUSTER-IP	PORT(S)
svc-1	NodePort	10.101.214.22	80:30000/TCP

- Dentro do cluster o service escuta na porta 80, enquanto fora do cluster escuta na porta 30000.
- Utilizamos o IP do nó para acessar o service através da porta 30000.

▼ Load balancer

- No caso o LoadBalancer é um recurso de rede que distribui cargas de acesso, comumente presente na nuvem como Azure, GCP, AWS, OCI, etc.

- No kubernetes este recurso nada mais é do que um cluster IP que irá realizar este procedimento de distribuição de carga, porém utilizará o recurso de loadbalancer da nuvem
 - Utilizam automaticamente os balanceadores de carga de cloud providers
- No caso os serviços podem estar acessíveis de fora do cluster, sem expor diretamente os pods
 - Por serem um Load Balancer, também são um NodePort e ClusterIP ao mesmo tempo.
- No caso está doc não será voltada para as clouds, sendo assim para demais informações:

Services, Load Balancing, and Networking

Concepts and resources behind networking in Kubernetes.

 <https://kubernetes.io/docs/concepts/services-networking/>



▼ Aplicando services ao projeto

▼ Acessando o portal

- Vamos primeiramente apagar todos os pods com o comando `kubectl delete pods --all`
- E apagar todos os serviços com o `kubectl delete svc --all`

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
pod1        1/1     Running   0          3h28m
pod2        1/1     Running   0          3h18m
portal-noticias 1/1     Running   1 (27h ago) 29h
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl delete pods --all
pod "pod1" deleted
pod "pod2" deleted
pod "portal-noticias" deleted
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes  ClusterIP  10.96.0.1      <none>        443/TCP      30h
svc-pod-2  ClusterIP  10.97.4.76      <none>        9000/TCP      3h10m
svc-pod1   NodePort   10.96.106.236   <none>        80:32767/TCP  88m
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ kubectl delete svc --all
service "kubernetes" deleted
service "svc-pod-2" deleted
service "svc-pod1" deleted

```

- Criado uma pasta chamada projeto-noticias onde armazenaremos os arquivos deste projeto

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ mkdir projeto-noticias
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ ls
pods  portal-noticias.yaml  projeto-noticias  svcs
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ mv portal-noticias.yaml projeto-noticias/
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ ls
pods  projeto-noticias  svcs
lk47@Latitude-E6410:~/Desktop/alura-kubernetes$ 

```

- No YAML do nosso portal de noticias vamos definir uma label chamada `app: portal-noticias`

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ cat portal-noticias.yaml
apiVersion: v1
kind: Pod
metadata:
  name: portal-noticias
spec:
  containers:
    - name: portal-noticias-container
      image: aluracursos/portal-noticias:1
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ nano portal-noticias.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ cat portal-noticias.yaml
apiVersion: v1
kind: Pod
metadata:
  name: portal-noticias
  labels:
    app: portal-noticias
spec:
  containers:
    - name: portal-noticias-container
      image: aluracursos/portal-noticias:1

```

- Vamos criar um SVC para o pod

```
apiVersion: v1
kind: Service
metadata:
  name: svc-portal-noticias
spec:
  type: NodePort
  ports:
    - port: 80
      #targetPort: 80
      nodePort: 30000
  selector:
    app: portal-noticias
```

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ nano svc-portal-noticias.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ ls
portal-noticias.yaml  svc-portal-noticias.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ cat svc-portal-noticias.yaml

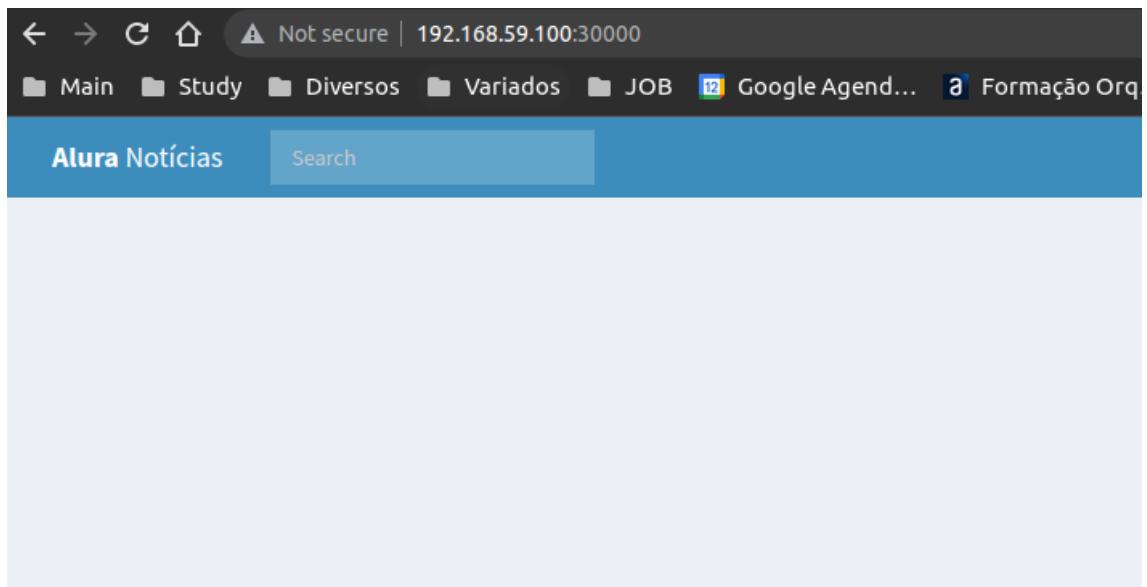
apiVersion: v1
kind: Service
metadata:
  name: svc-portal-noticias
spec:
  type: NodePort
  ports:
    - port: 80
      #targetPort: 80
      nodePort: 30000
  selector:
    app: portal-noticias
```

- Vamos aplicar e criar o pod e o svc

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl apply -f portal-noticias.yaml
pod/portal-noticias created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl apply -f svc-portal-noticias.yaml
service/svc-portal-noticias created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
portal-noticias 1/1     Running   0          11s
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl get svc
NAME            TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1   <none>        443/TCP       11m
svc-portal-noticias  NodePort   10.103.62.155  <none>        80:30000/TCP  12s
```

- Agora vamos tentar acessar através do IP do minikube e será possível normalmente

```
svc-portal-noticias  NodePort  10.103.62.155  <none>        80:30000/TCP  12s
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/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  31h   v1.26.3   192.168.59.100  <none>        Buildroot-2021.02.12  5.10.57   docker:///
20.10.23
```

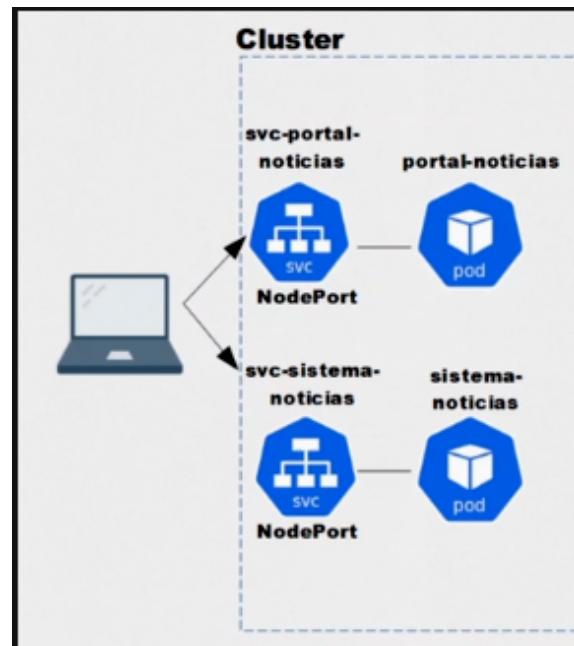


▼ Definindo portas

- O campo `nodePort` e `targetPort` serão definidos implicitamente.

▼ Subindo o sistema

- No caso já temos o portal agora precisaremos de um sistema para cadastrar e exibir as notícias no portal



- Vamos criar um pod e um SVC para este sistema de notícias

- Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: sistema-noticias
  labels:
    app: sistema-noticias
spec:
  containers:
    - name: sistema_noticias-container
      image: aluracursos/sistema-noticias:1
      ports:
        - containerPort: 80
```

- SVC

```
apiVersion: v1
kind: Service
metadata:
  name: svc-sistema-noticias
spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 30001
  selector:
    app: sistema-noticias
```

- Como já temos uma aplicação na porta bind 3000, está segunda ficará na porta 30001

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ nano pod-sistema-noticias.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ cat pod-sistema-noticias.yaml
apiVersion: v1
kind: Pod
metadata:
  name: sistema-noticias
  labels:
    app: sistema-noticias
spec:
  containers:
    - name: sistema-noticias-container
      image: aluracursos/sistema-noticias:1
      ports:
        - containerPort: 80
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ nano svc-sistema-noticias.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ cat svc-sistema-noticias.yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-sistema-noticias
spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 30001
  selector:
    app: sistema-noticias
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ 

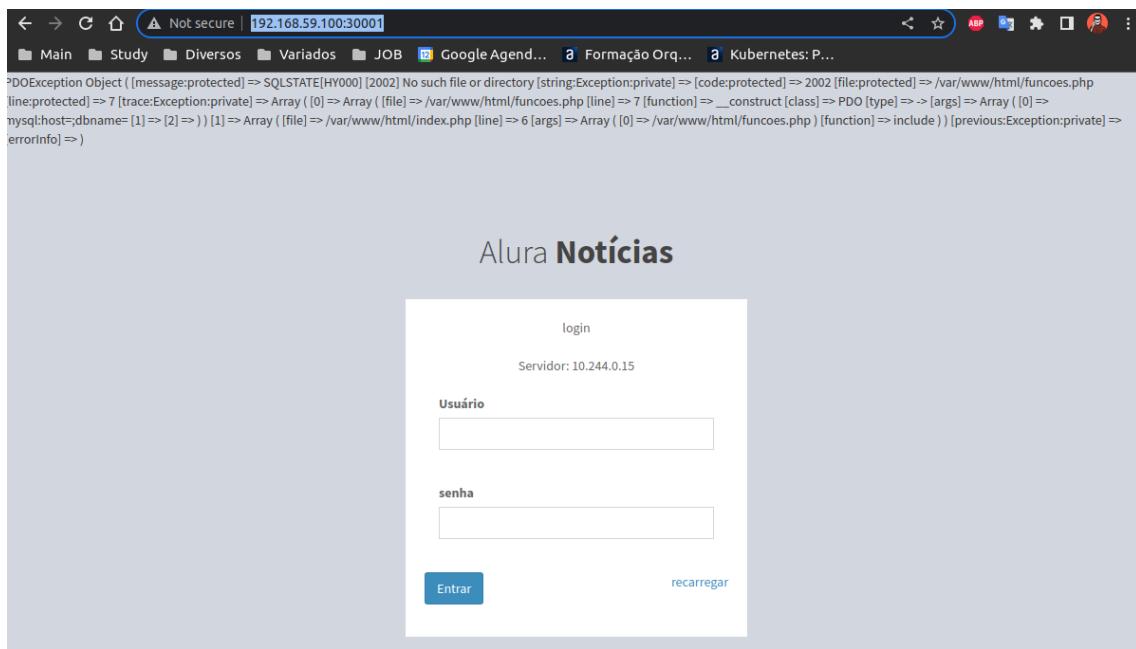
```

- Vamos aplicar e validar a funcionalidade

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl apply -f pod-sistema-noticias.yaml
pod/sistema-noticias created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl apply -f svc-sistema-noticias.yaml
service/svc-sistema-noticias created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
portal-noticias 1/1     Running   0          13m
sistema-noticias 1/1     Running   0          15s
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl get svc
NAME            TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1   <none>        443/TCP       24m
svc-portal-noticias  NodePort   10.103.62.155 <none>        80:30000/TCP  13m
svc-sistema-noticias  NodePort   10.96.4.78   <none>        80:30001/TCP  9s
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/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  31h   v1.26.3   192.168.59.100  <none>        Buildroot 2021.02.12   5.10.57        docker://
20.10.23
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ 

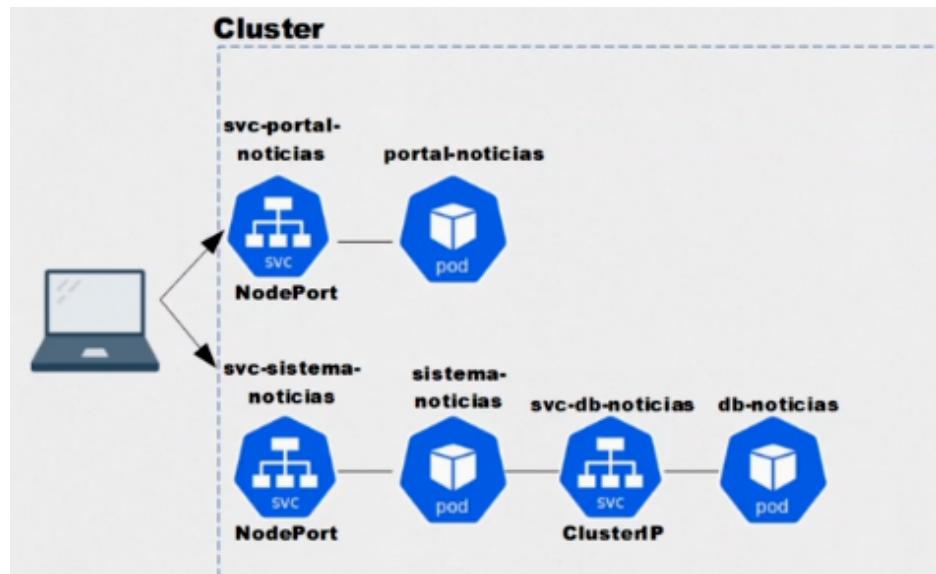
```



- Agora precisaremos de um banco para que as informações possam ser armazenadas

▼ Subindo o banco

- Agora precisarmos criar um banco para que possamos guardar as notícias e acessar nosso sistema



- O banco não deverá ser acessível externamente dessa forma iremos criar um pod e um clusterIP

- POD

```
apiVersion: v1
kind: Pod
metadata:
  name: db-noticias
  labels:
    app: db-noticias
spec:
  containers:
    - name: db-noticias-container
      image: aluracursos/mysql-db:1
      ports:
        - containerPort: 3306
```

- SVC

```
apiVersion: v1
kind: Services
metadata:
  name: svc-db-noticias
spec:
  type: ClusterIP
  ports:
    - port: 3306
  selector:
    app: db-noticias
```

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ nano pod-db-noticias.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ cat pod-db-noticias.yaml
apiVersion: v1
kind: Pod
metadata:
  name: db-noticias
  labels:
    app: db-noticias
spec:
  containers:
    - name: db-noticias-container
      image: aluracursos/mysql-db:1
      ports:
        - containerPort: 3306
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ nano svc-db-noticias.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ cat svc-db-noticias.yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-db-noticias
spec:
  type: ClusterIP
  ports:
    - port: 3306
  selector:
    app: db-noticias
```

- Feito isso vamos aplicar os objetos e validar

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl apply -f pod-db-noticias.yaml
pod/db-noticias created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl apply -f svc-db-noticias.yaml
service/svc-db-noticias created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl get pods
NAME           READY   STATUS      RESTARTS   AGE
db-noticias    0/1     CrashLoopBackOff   1 (9s ago) 12s
portal-noticias 1/1     Running      0          32m
sistema-noticias 1/1     Running      0          19m
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl get svc
NAME            TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE
kubernetes      ClusterIP  10.96.0.1    <none>        443/TCP     43m
svc-db-noticias ClusterIP  10.100.171.92 <none>        3306/TCP    11s
svc-portal-noticias NodePort   10.103.62.155 <none>        80:30000/TCP 32m
svc-sistema-noticias NodePort   10.96.4.78   <none>        80:30001/TCP 19m
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$
```

- Vemos que o pod do banco está com um erro

```
Events:
Type  Reason  Age           From            Message
----  -----  --            --             -----
Normal Scheduled  80s          default-scheduler  Successfully assigned default/db-noticias to minikube
Normal Pulled   38s (x4 over 79s)  kubelet        Container image "aluracursos/mysql-db:1" already present on machine
Normal Created   38s (x4 over 79s)  kubelet        Created container db-noticias-container
Normal Started   38s (x4 over 79s)  kubelet        Started container db-noticias-container
Warning BackOff  12s (x7 over 77s)  kubelet        Back-off restarting failed container db-noticias-container in pod db-noticias_default(f9e4a5c6-a11d-444b-8005-38073d492923)
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$
```

- Validando a doc do mysql vemos que é necessário definirmos variáveis de ambiente

mysql - Official Image | Docker Hub

MySQL is a widely used, open-source relational database management system (RDBMS).

 https://hub.docker.com/_/mysql

Environment Variables

When you start the `mysql` image, you can adjust the configuration of the MySQL instance by passing one or more environment variables on the `docker run` command line. Do note that none of the variables below will have any effect if you start the container with a data directory that already contains a database: any pre-existing database will always be left untouched on container startup.

See also <https://dev.mysql.com/doc/refman/5.7/en/environment-variables.html> for documentation of environment variables which MySQL itself respects (especially variables like `MYSQL_HOST`, which is known to cause issues when used with this image).

`MYSQL_ROOT_PASSWORD`

This variable is mandatory and specifies the password that will be set for the MySQL `root` superuser account. In the above example, it was set to `my-secret-pw`.

`MYSQL_DATABASE`

This variable is optional and allows you to specify the name of a database to be created on image startup. If a user/password was supplied (see below) then that user will be granted superuser access ([corresponding to GRANT ALL](#)) to this database.

`MYSQL_USER` , `MYSQL_PASSWORD`

- Dessa forma precisaremos analisar outros pontos para podermos aplicar no kubernetes este banco

▼ Definindo variáveis de ambiente

▼ Utilizando variáveis de ambiente

- Com base na documentação do docker MySQL vamos definir as envs em nosso YAML

mysql - Official Image | Docker Hub

MySQL is a widely used, open-source relational database management system (RDBMS).

 https://hub.docker.com/_/mysql

```
apiVersion: v1
kind: Pod
metadata:
  name: db-noticias
  labels:
    app: db-noticias
spec:
  containers:
```

```

- name: db-noticias-container
  image: aluracursos/mysql-db:1
  ports:
    - containerPort: 3306
  env:
    - name: "MYSQL_ROOT_PASSWORD"
      value: "q1w2e3r4"
    - name: "MYSQL_DATABASE"
      value: "empresa"
    - name: "MYSQL_PASSWORD"
      value: "q1w2e3r4"

```

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ nano pod-db-noticias.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ cat pod-db-noticias.yaml
apiVersion: v1
kind: Pod
metadata:
  name: db-noticias
  labels:
    app: db-noticias
spec:
  containers:
    - name: db-noticias-container
      image: aluracursos/mysql-db:1
      ports:
        - containerPort: 3306
      env:
        - name: "MYSQL_ROOT_PASSWORD"
          value: "q1w2e3r4"
        - name: "MYSQL_DATABASE"
          value: "empresa"
        - name: "MYSQL_PASSWORD"
          value: "q1w2e3r4"

```

- Dessa forma vamos excluir o pod antigo e recriar novamente com base no arquivo atualizado

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl delete -f pod-db-noticias.yaml
pod "db-noticias" deleted
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl apply -f pod-db-noticias.yaml
pod/db-noticias created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl get pods
Command 'kubectl' not found, did you mean:
  command 'kubectl' from snap kubectl (1.27.3)
See 'snap info <snapname>' for additional versions.
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
db-noticias   1/1     Running   0          20s
portal-noticias 1/1     Running   0          46m
sistema-noticias 1/1     Running   0          32m

```

- Agora podemos ver que o pod estará apresentando funcionalidade
- Vamos acessar o pod e realizar um acesso no banco

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl exec -it db-noticias -- bash
root@db-noticias:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.6.48 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

- Dentro do banco vamos validar se os bancos se encontram
 - show databases;
 - use empresa;
 - show tables;
 - select * from usuario;

```
mysql> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| empresa        |
| mysql          |
| performance_schema |
+-----+
4 rows in set (0.00 sec)

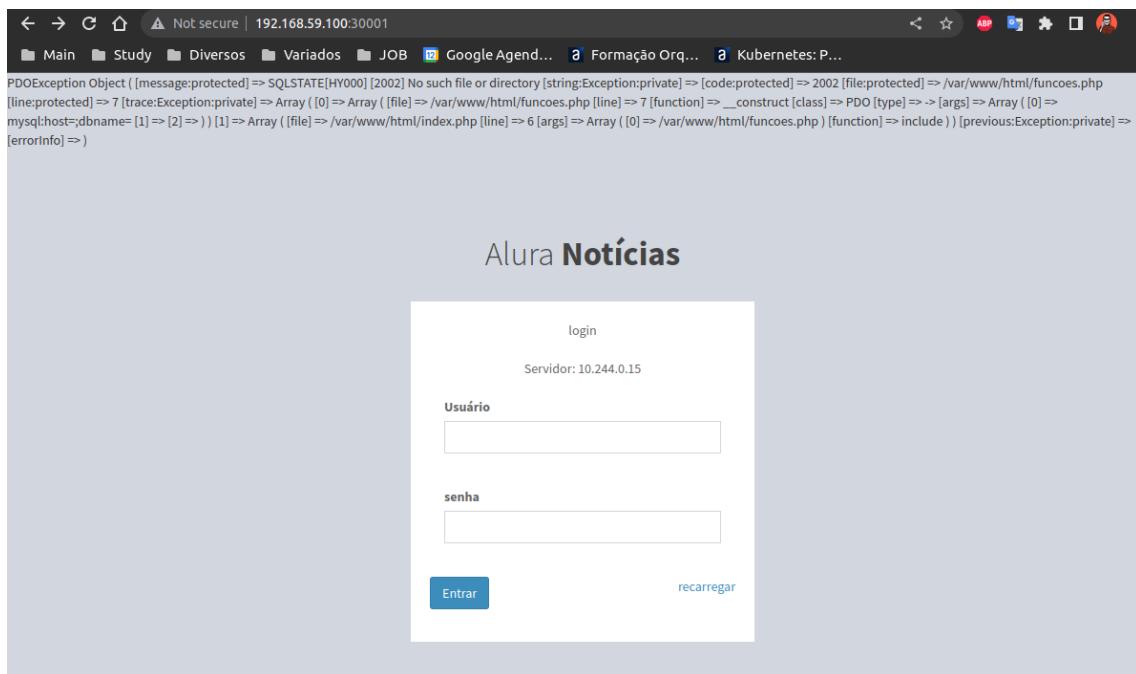
mysql> use empresa
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_empresa |
+-----+
| noticias          |
| usuario           |
+-----+
2 rows in set (0.00 sec)

mysql> select * from usuario;
+-----+-----+-----+
| idusuario | login | senha |
+-----+-----+-----+
|       1 | admin | admin |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> █
```

- Validando o sistema-noticias em nosso navegador podemos ver que ele ainda não se encontra funcional



- Acessando o pod do sistema noticias podemos ver que a um arquivo chamado bancodedados.php que há algumas variaveis que não estão especificadas

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ kubectl exec -it sistema-noticias -- bash
root@sistema-noticias:/var/www/html# ls
Dockerfile      docker-compose.yml  index.php      notificacao.sh    php.ini-production upload.php
arquivos.php    excluir.php       inserir_noticias.php  php.ini        sair.php      uploadClass.
bancodedados.php funcoes.php     noticias.php      php.ini-development theme          uploads
root@sistema-noticias:/var/www/html# cat bancodedados.php
<?php
$host = getenv("HOST_DB");
$usuario = getenv("USER_DB");
$senha = getenv("PASS_DB");
$banco = getenv("DATABASE_DB");
?>
root@sistema-noticias:/var/www/html#
```

- No caso podemos ver que o nosso YAML está meio bagunçado pois estamos misturando definições com configurações que é o caso das envs, isso deverá ser ajustado como forma de boas práticas

```
apiVersion: v1
kind: Pod
metadata:
  name: db-noticias
  labels:
    app: db-noticias
spec:
  containers:
    - name: db-noticias-container
      image: aluracursos/mysql-db:1
      ports:
        - containerPort: 3306
      env:
        - name: "MYSQL_ROOT_PASSWORD"
          value: "q1w2e3r4"
        - name: "MYSQL_DATABASE"
          value: "empresa"
        - name: "MYSQL_PASSWORD"
          value: "q1w2e3r4"
```

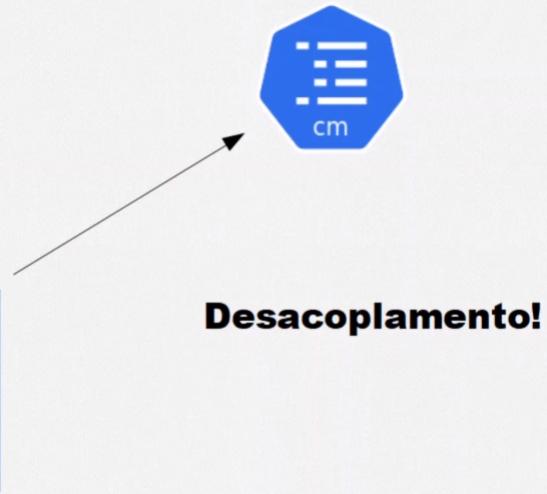
▼ Variáveis e definições

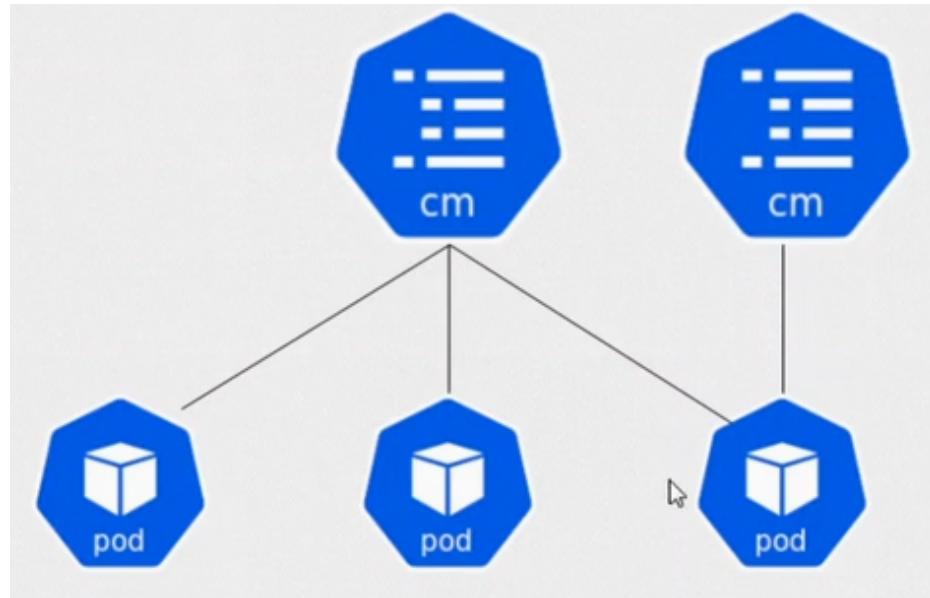
- Podemos usar o campo `env` para definir uma ou mais variáveis.

▼ Criando um ConfigMap

- O config map armazena as informações existentes em um objeto, e permite a reutilização, acoplamento e desacoplamento

```
apiVersion: v1
kind: Pod
metadata:
  name: db-noticias
  labels:
    app: db-noticias
spec:
  containers:
    - name: db-noticias-container
      image: aluracursos/mysql-db:1
      ports:
        - containerPort: 3306
      env:
        - name: "MYSQL_ROOT_PASSWORD"
          value: "q1w2e3r4"
        - name: "MYSQL_PASSWORD"
          value: "q1w2e3r4"
        - name: "MYSQL_DATABASE"
          value: "empresa"
```





- Ou seja podemos usar pods acessando o mesmo ou diferentes config-maps
- Antes de tudo na pasta de nosso projeto, organizamos da seguinte forma
 - Criamos uma pasta para os pods, para os svcs e agora para os configmaps que criaremos

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ ls
configmap  pods  svcs
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias$ █
```

- Vamos criar um db-configmap.yaml
 - Perceba que utilizamos as informações existentes no YAML do banco em env no campo data em nosso config map

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: db-configmap
data:
  MYSQL_ROOT_PASSWORD: q1w2e3r4
  MYSQL_DATABASE: empresa
  MYSQL_PASSWORD: q1w2e3r4
```

- As informações contidas no campo data são atribuídas em formato 'chave: valor'

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$ nano db-configmap.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$ cat db-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: db-configmap
data:
  MYSQL_ROOT_PASSWORD: q1w2e3r4
  MYSQL_DATABASE: empresa
  MYSQL_PASSWORD: q1w2e3r4
```

- Vamos aplicar agora em nosso cluster

- Podemos usar o `kubectl get configmap` e `kubectl describe configmap <nome>`

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$ kubectl apply -f db-configmap.yaml
configmap/db-configmap created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$ kubectl get configmap
NAME        DATA   AGE
db-configmap 3      119s
kube-root-ca.crt 1      2d2h
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$ kubectl describe configmap db-configmap
Name:         db-configmap
Namespace:    default
Labels:       <none>
Annotations: <none>

Data
====
MYSQL_DATABASE:
-----
empresa
MYSQL_PASSWORD:
-----
q1w2e3r4
MYSQL_ROOT_PASSWORD:
-----
q1w2e3r4

BinaryData
====

Events:  <none>
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$
```

- Dessa forma o proximo passo será configurar o nosso YAML do banco

▼ Definindo ConfigMaps

```
apiVersion: v2
kind: ConfigMap
metadata:
  name: config-data
spec:
  MYSQL_ROOT_PASSWORD: q1w2e3r4
  MYSQL_DATABASE: empresa
  MYSQL_PASSWORD: q1w2e3r4
```

- Ele não funcionará, a versão da API está errada!
- Ele não funcionará, no lugar de `spec` o certo seria `data`.

▼ Aplicando o ConfigMap ao projeto (LAB)

- Vamos importar agora as informações do configmap para o yaml do pod do banco

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ ls
pod-db-noticias.yaml  pod-portal-noticias.yaml  pod-sistema-noticias.yaml
lk47@latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ cat pod-db-noticias.yaml
apiVersion: v1
kind: Pod
metadata:
  name: db-noticias
  labels:
    app: db-noticias
spec:
  containers:
    - name: db-noticias-container
      image: aluracursos/mysql-db:1
      ports:
        - containerPort: 3306
  env:
    - name: "MYSQL_ROOT_PASSWORD"
      value: "q1w2e3r4"
    - name: "MYSQL_DATABASE"
      value: "empresa"
    - name: "MYSQL_PASSWORD"
      value: "q1w2e3r4"
  lremos declarar aqui as informações do
  configmap
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ █
```

- No caso podemos fazer de dois jeitos
 - Variavel por variavel, no caso iremos declarar cada campo ficando da seguinte forma

```
env:
  - name: MYSQL_ROOT_PASSWORD
    valueFrom:
      configMapKeyRef:
        name: db-configmap
        key: MYSQL_ROOT_PASSWORD
```

- Dessa forma permite utilizarmos campos específicos do configmap
- O configMap inteiro

```
envFrom:
  - configMapRef:
      name: db-configmap
```

- Dessa forma permite utilizarmos todo o configmap sem precisar especificar variavel por variavel

- Vamos usar o configMap inteiro deixando o YAML do nosso banco da seguinte forma

```

apiVersion: v1
kind: Pod
metadata:
  name: db-noticias
  labels:
    app: db-noticias
spec:
  containers:
    - name: db-noticias-container
      image: aluracursos/mysql-db:1
      ports:
        - containerPort: 3306
      envFrom:
        - configMapRef:
            name: db-configmap

```

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ nano pod-db-noticias.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ cat pod-db-noticias.yaml
apiVersion: v1
kind: Pod
metadata:
  name: db-noticias
  labels:
    app: db-noticias
spec:
  containers:
    - name: db-noticias-container
      image: aluracursos/mysql-db:1
      ports:
        - containerPort: 3306
      envFrom:
        - configMapRef:
            name: db-configmap

```

- Feito isso vamos apagar e recriar o pod novamente

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
db-noticias   1/1     Running   1 (19m ago)  18h
portal-noticias 1/1     Running   1 (19m ago)  19h
sistema-noticias 1/1     Running   1 (19m ago)  19h
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl delete pod db-noticias
pod "db-noticias" deleted

```

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl apply -f pod-db-noticias.yaml
pod/db-noticias created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
db-noticias   1/1     Running   0          4s
portal-noticias 1/1     Running   1 (22m ago)  19h
sistema-noticias 1/1     Running   1 (22m ago)  19h

```

- Agora vamos validar se o banco permanece funcional

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
db-noticias   1/1     Running   0          5m52s
portal-noticias 1/1     Running   1 (28m ago) 19h
sistema-noticias 1/1     Running   1 (28m ago) 19h
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl exec -it db-noticias -- bash
root@db-noticias:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.48 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

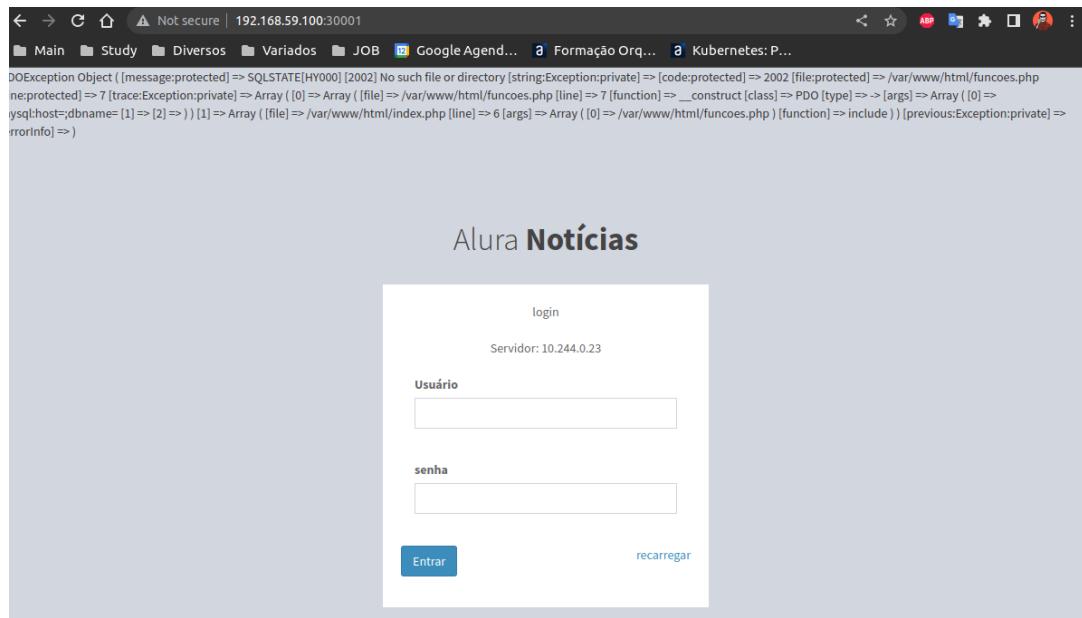
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| empresa        |
| mysql          |
| performance_schema |
+-----+
4 rows in set (0.01 sec)

mysql> 

```

- Feito isso, validando o nosso sistema, veremos que o mesmo permanece não funcional



- Vamos acessar o pod sistema-noticias e veremos que o arquivo bancodedados.php se encontra com algumas variaveis que precisam ser declaradas

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pod$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
db-noticias 1/1     Running   0          11m
portal-noticias 1/1     Running   1 (34m ago) 19h
sistema-noticias 1/1     Running   1 (34m ago) 19h
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl exec -it sistema-noticias -- bash
root@sistema-noticias:/var/www/html# ls
Dockerfile      docker-compose.yml  index.php      notificacao.sh      php.ini-production  upload.php
arquivos.php    excluir.php       inserir_noticias.php  php.ini           sair.php        uploadClass.php
bancodedados.php  funcoes.php     noticias.php     php.ini-development theme           uploads
root@sistema-noticias:/var/www/html# cat bancodedados.php
<?php

$host = getenv("HOST_DB");
$usuario = getenv("USER_DB");
$senha = getenv("PASS_DB");
$banco = getenv("DATABASE_DB");

?>
root@sistema-noticias:/var/www/html# 

```

- Sendo assim vamos criar um segundo configmap para este pod

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: sistema-configmap
data:
  HOST_DB: svc-db-noticias:3306 #Podemos usar o IP do SVC também
  USER_DB: root
  PASS_DB: q1w2e3r4
  DATABASE_DB: empresa

```

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$ nano sistema-noticias-configmap.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$ cat sistema-noticias-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: sistema-configmap
data:
  HOST_DB: svc-db-noticias:3306 #Podemos usar o IP do SVC também
  USER_DB: root
  PASS_DB: q1w2e3r4
  DATABASE_DB: empresa

```

- Vamos modificar o YAML do sistema-noticias adicionando o configmap

```

envFrom:
  - configMapRef:
      name: sistema-configmap

```

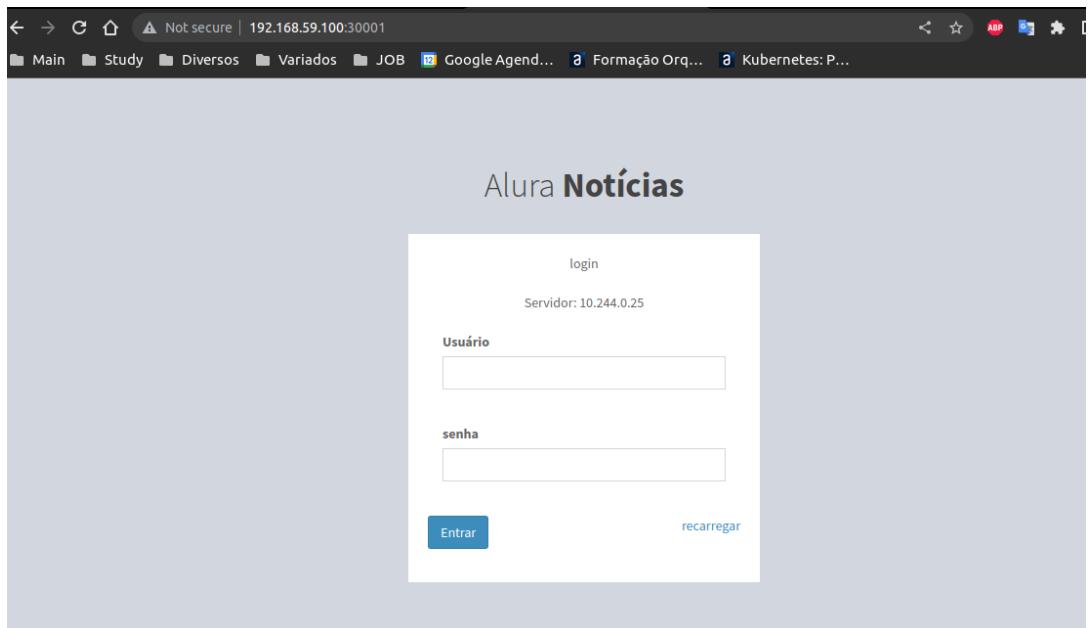
```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ nano pod-sistema-noticias.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ cat pod-sistema-noticias.yaml
apiVersion: v1
kind: Pod
metadata:
  name: sistema-noticias
  labels:
    app: sistema-noticias
spec:
  containers:
    - name: sistema-noticias-container
      image: aluracursos/sistema-noticias:1
      ports:
        - containerPort: 80
      envFrom:
        - configMapRef:
            name: sistema-configmap
```

- Vamos subir o novo configmap e vamos recriar o pod de sistema-noticias

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$ kubectl apply -f sistema-noticias-configmap.yaml
configmap/sistema-configmap created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$ kubectl get configmap
NAME          DATA   AGE
db-configmap  3      25m
kube-root-ca.crt 1      2d2h
sistema-configmap 4      14s
```

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
db-noticias    1/1     Running   0          18m
portal-noticias 1/1     Running   1 (40m ago) 19h
sistema-noticias 1/1     Running   1 (40m ago) 19h
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl delete pod sistema-noticias
pod "sistema-noticias" deleted
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl apply -f pod-sistema-noticias.yaml
pod/sistema-noticias created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
db-noticias    1/1     Running   0          19m
portal-noticias 1/1     Running   1 (41m ago) 19h
sistema-noticias 1/1     Running   0          37s
```

- Feito isso podemos ver que será possível acessar o sistema sem problemas



- Vamos usar o user e senha admin/admin conforme apontado no banco

```
mysql> select * from usuario;
+-----+-----+-----+
| idusuario | login | senha |
+-----+-----+-----+
|       1 | admin | admin |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Alura Notícias

login

Servidor: 10.244.0.25

Usuário

admin

senha

.....

Entrar recarregar

Not secure | 192.168.59.100:30001/inserir_noticias.php

Main Study Diversos Variados JOB Google Agenda... Formação Orq... Kubernetes: P...

Alura Notícias Search Servidor: 10.244.0.25 Sair

Portal de notícias Alura

Nova Notícia

Título

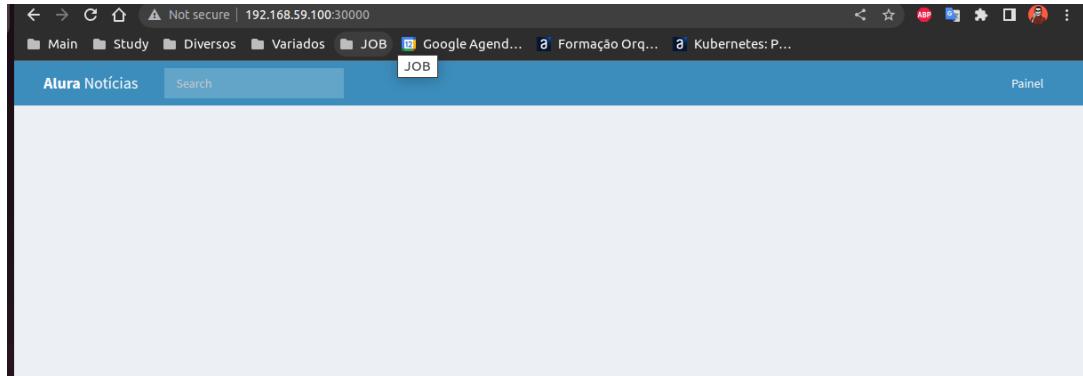
Ação

- Vamos adicionar uma notícia

The image consists of three vertically stacked screenshots of a web application interface. All three screenshots have a blue header bar with the text 'Alura Notícias' on the left, 'Search' in the center, and 'Servidor: 10.244.0.25 Sair' on the right.

- Screenshot 1:** Shows the main dashboard titled 'Portal de notícias Alura'. At the bottom right is a blue button labeled 'Nova Notícia'. A green arrow points from the top right towards this button.
- Screenshot 2:** A modal window titled 'Nova Notícia' is open. It contains three input fields: 'Título' (Title) with the value 'Venha conhecer meu github!!', 'Notícia' (Content) with the value 'Lá posto meus projetos e estudos a respeito das áreas de DevOps e Dev!', and 'Foto' (Photo) with a file input field containing 'Choose File venhaconhermeugithub.png'. At the bottom are two buttons: 'Cancelar' (Cancel) on the left and 'Salvar' (Save) on the right.
- Screenshot 3:** Shows the main dashboard again, but now with the new news item listed. The 'Título' field is highlighted with a red box and a green arrow pointing to it. The new news item is visible in the list, and at the bottom right is a red button labeled 'Excluir' (Delete).

- Agora queremos que está notícia seja exibida em nosso portal de noticias, se acessar o pod do portal-noticias veremos um arquivo chamado `configuracao.php` no qual nele especifica a necessidade de informação em uma variavel



```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
db-noticias 1/1     Running   0          33m
portal-noticias 1/1     Running   1 (56m ago) 20h
sistema-noticias 1/1     Running   0          15m
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl exec -it portal-noticias -- bash
root@portal-noticias:/var/www/html# ls
configuracao.php content.html docker-compose.yml index.php theme
root@portal-noticias:/var/www/html# cat configuracao.php
<?php
$urlnoticias = getenv("IP_SISTEMA");
?>
root@portal-noticias:/var/www/html#
```

- Sendo assim vamos criar um novo config map indicando o pod sistema-noticias para o portal

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: portal-configmap
data:
  IP_SISTEMA: http://192.168.59.100:30001 #No caso é o IP do internal minikube
```

```
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$ nano portal-noticias-configmap.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$ cat portal-noticias-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: portal-configmap
data:
  IP_SISTEMA: http://192.168.59.100:30001
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$ kubectl get configmap
NAME        DATA   AGE
db-configmap 3     95m
kube-root-ca.crt 1     2d3h
portal-configmap 1     99s
sistema-configmap 4     69m
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/configmap$
```

No windows colocaríamos http://localhost:30001 devido ao docker desktop, este IP se refere ao minikube (kubectl get nodes -o wide)

- Feito isso vamos adicionar agora o envFrom em nosso YAML do portal-noticias

```

envFrom:
  - configMapRef:
      name: portal-configmap

```

```

lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ nano pod-portal-noticias.yaml
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ cat pod-portal-noticias.yaml
apiVersion: v1
kind: Pod
metadata:
  name: portal-noticias
  labels:
    app: portal-noticias
spec:
  containers:
    - name: portal-noticias-container
      image: aluracursos/portal-noticias:1
      envFrom:
        - configMapRef:
            name: portal-configmap

```

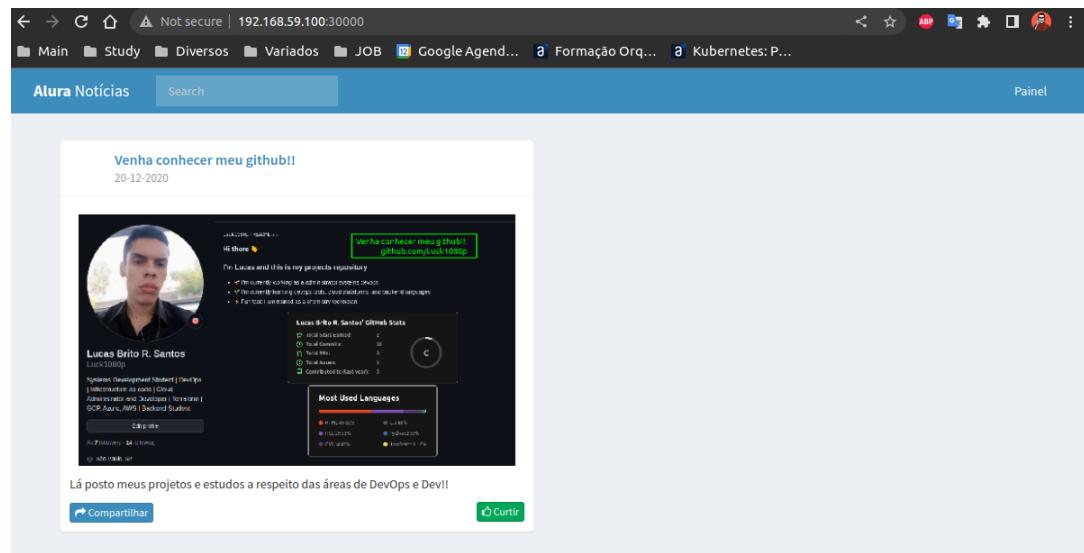
- Feito isso vamos recriar o pod portal-noticias

```

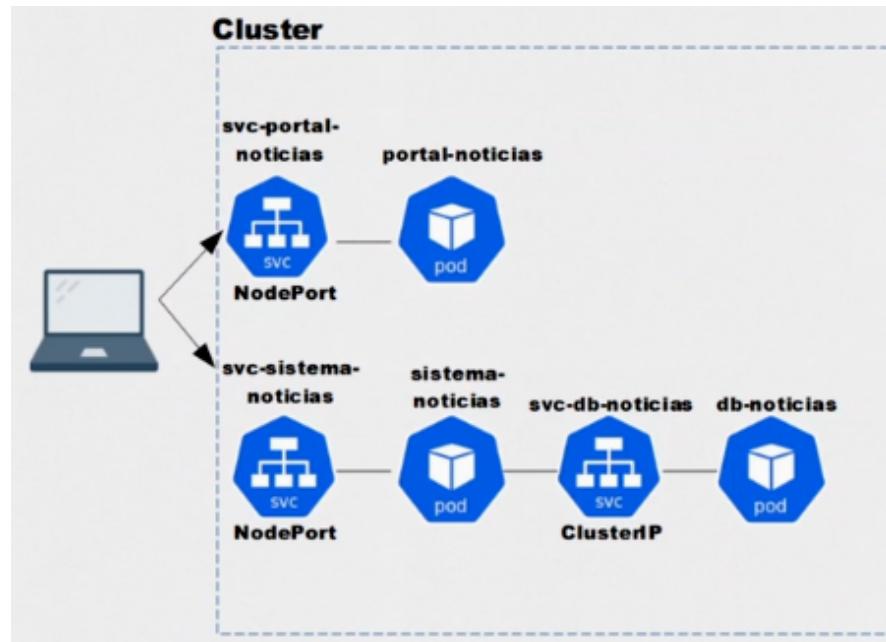
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
db-noticias 1/1     Running   1 (2m54s ago)  48m
portal-noticias 1/1     Running   1 (2m54s ago)  9m41s
sistema-noticias 1/1     Running   1 (2m54s ago)  30m
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl delete pod portal-noticias
pod "portal-noticias" deleted
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl apply -f pod-portal-noticias.yaml
pod/portal-noticias created
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
db-noticias 1/1     Running   1 (3m39s ago)  49m
portal-noticias 1/1     Running   0          4s
sistema-noticias 1/1     Running   1 (3m39s ago)  30m
lk47@Latitude-E6410:~/Desktop/alura-kubernetes/projeto-noticias/pods$ 

```

- Voltando ao navegador, vamos recarregar o portal e veremos que a notícia irá aparecer conforme adicionamos no sistema



- Dessa forma criamos um exemplo simples de uma arquitetura de microserviços utilizando o kubernetes



▼ Linkando Pods e ConfigMaps

- Como podemos fazer um pod utilizar dados de um ConfigMap?
 - Utilizando os campos `env` OU `envFrom`

▼ NodePort e IP's

Caso tivéssemos múltiplos nodes em nosso cluster, tudo funcionaria da mesma maneira, pois as portas mapeadas pelo NodePort são compartilhadas entre os IP's de todos os nodes.

Service

Expose an application running in your cluster behind a single outward-facing endpoint, even when the workload is split across multiple backends.

 <https://kubernetes.io/docs/concepts/services-networking/service/#nodeport>

kuberr