

# Hands-on com Kubernetes no Cluster Nacional

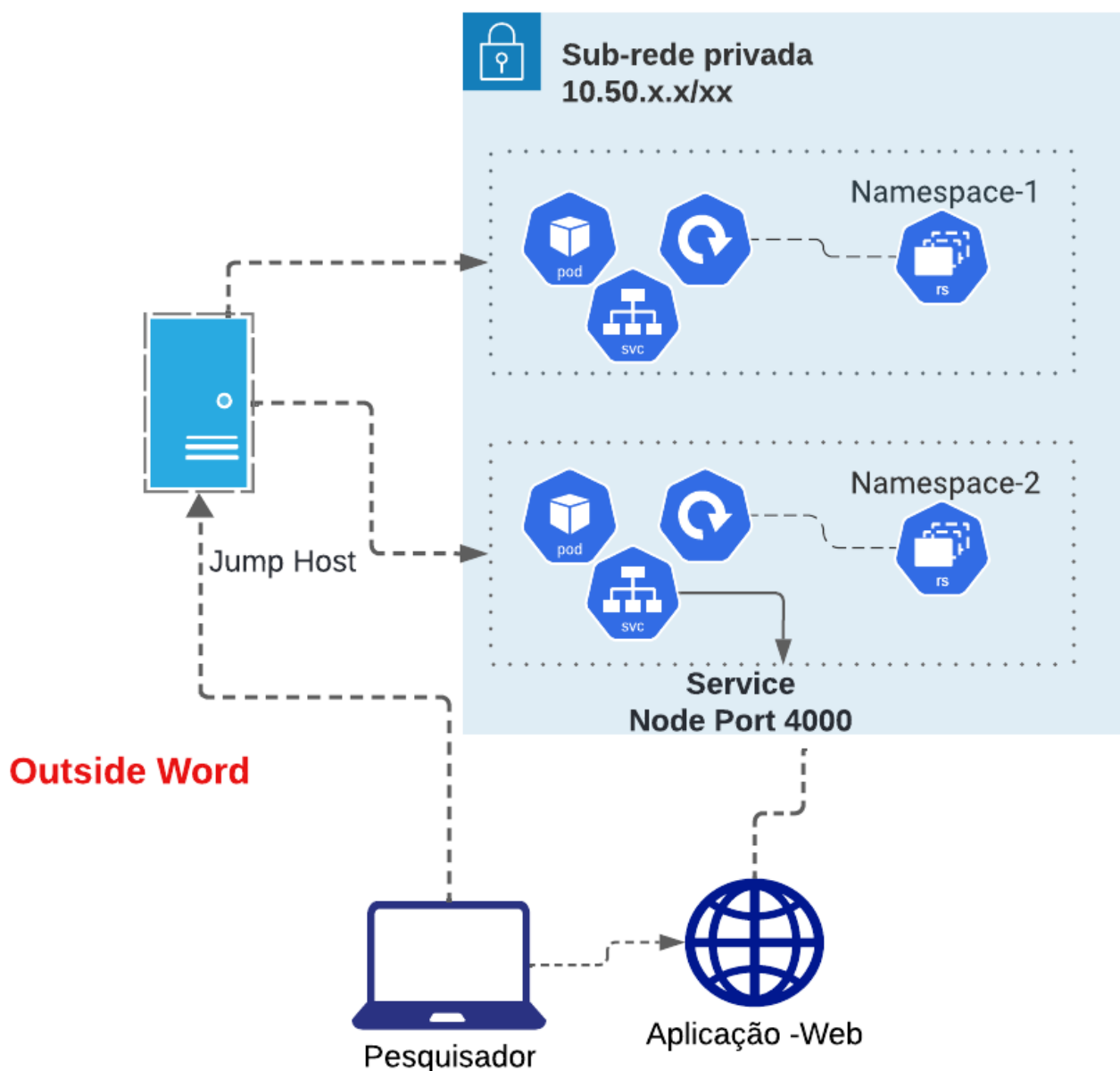
- [Instanciar objetos no Cluster Nacional utilizando o Kubernetes](#)

# Instanciar objetos no Cluster Nacional utilizando o Kubernetes

Nesta atividade iremos explorar algumas possibilidades utilizando o Kubernetes para instanciar objetos no Cluster Nacional.

Iremos utilizar alguns conceitos mencionados anteriormente para demonstrar em um ambiente real, como podemos subir pods, deployments ,services, ConfigMap...

Como exemplo iremos subir uma aplicação web utilizando deployments, services e demais configurações, de forma que a mesma fique acessível externamente.



Para o acesso ao Namespace iremos utilizar o seguinte comando :

```
ssh -i .ssh/{USER} {USER}@xx.xx.xx.xx -p xx
```

Os arquivos que se encontram nos diretórios dos namespaces Não devem ser alterados!

## Parte-01

# 1- Subindo nosso Primeiro Pod no Cluster Nacional

Navegue até o diretório `wtestbeds-2025` .

```
cd WTESTBEDS-2025
```

As etapas foram estruturadas em partes 01 , 02 e 03.

Navegue até o dir `parte-01` para darmos início as primeiras atividades.

```
cd parte-01
```

Temos 3 arquivos : `pod.yml`, `deployment.yml`, `service.yml`

Utilize o comando `cat` para visualizar o arquivo `pod-v1.yml`

```
cat pod.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: wtestbeds
  labels:
    app: wtestbeds
spec:
  containers:
    - name: wtestbeds
      image: nginx:latest
```

A aplicação que iremos executar trata-se de um:

## 2- Subindo nosso primeiro deployment no Cluster Nacional

Utilize o comando `cat` para visualizar o arquivo `deployment-v1.yml`

```
cat deployment.yml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wtestbeds
  labels:
    app: wtestbeds
spec:
  replicas: 5
  selector:
    matchLabels:
      app: wtestbeds
  template:
    metadata:
      name: wtestbeds
      labels:
        app: wtestbeds
    spec:
      containers:
        - name: wtestbeds
          image: nginx:latest
```

Para criamos o nosso primeiro deployment utilizamos o comando abaixo:

```
kubectl apply -f deployment.yml
```

Observe que o deployment foi criado corretamente:

```
kubectl get pods
```

Para uma descrição mais detalhada utilize:

```
kubectl get pods -o wide
```

Para criamos o nosso primeiro pod utilizamos o comando abaixo:

```
kubectl apply -f pod.yml
```

Observe que o pod foi criado corretamente:

```
kubectl get pods
```

Para uma descrição mais detalhada utilize:

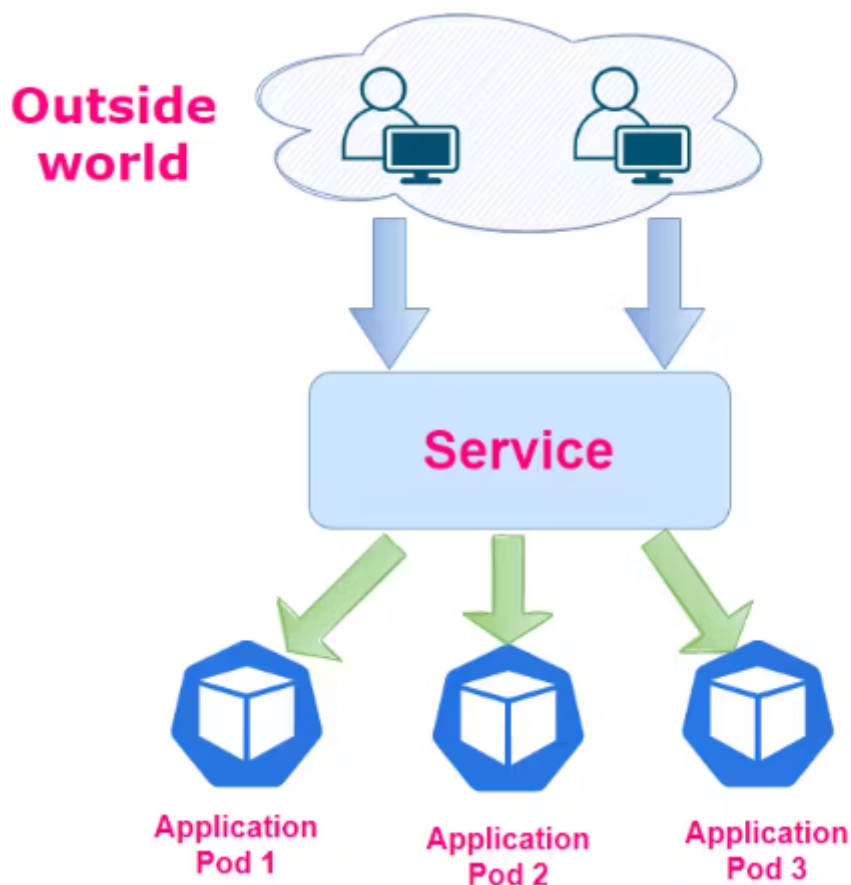
```
kubectl get pods -o wide
```

Digite o comando abaixo para deletarmos o pod que criamos:

```
kubectl delete -f pod.yaml
```

## 3- Subindo nosso primeiro Service no Cluster Nacional

Agora que o nosso deployment foi criado vamos criar um serviço para que possamos acessar a aplicação web.



Vamos criar um service do tipo NodePort , veja o arquivo service-v1.yaml:

```
apiVersion: v1
kind: Service
metadata:
  name: svc-wtestbeds
  labels:
    app: svc-wtestbeds
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: xxxxx
  selector:
    app: wtestbeds
```

Para criamos o nosso primeiro service utilizamos o comando abaixo:

```
Kubectl apply -f service.yml
```

```
Kubectl get service
```

## #Parte-02

# 1- Configurando o NGINX utilizando objetos do tipo deployment.

Utilizaremos a mesma aplicação, porém, faremos algumas atualizações no Deployment .

Navegue até o dir parte-02 para darmos continuidade as atividades

```
cd parte-02
```

Observe que dentro da pasta temos os seguintes arquivos:

comands.txr , deployment.yml, homepage.html, nginx.config

No arquivo homepage.html foi implementado uma página simples para demonstrar como podemos subir aplicações web no Cluster Nacional utilizando a orquestração em Kubernetes:

```
cat homepage.html
```

```
<!-- ##### THIS IS A COMMENT - Visible only in the source editor #####-->
<h1 style="text-align: center;"></h1>
<h1 style="text-align: center;">Bem-vindo ao WTestbeds 2025!</h1>
<p style="font-size: 1.5em; text-align: center;">Parab&eacute;s! Voc&ecirc; acaba de
provisionar um servi&ccedil;o no cluster kubernetes da RNP!</p>
<p style="font-size: 1.5em; text-align: center;">&nbsp;<strong data-start="206" data-
end="249">Explore o Futuro com os Laborat&oacute;rio Nacional Multiusu&aacute;rio da
RNP</strong></p>
<p style="font-size: 1.5em; text-align: center;">Tem uma ideia inovadora em Redes,
Computa&ccedil;&atilde;o, Blockchain, IoT ou Seguran&ccedil;a Cibern&eacute;tica?<br data-
start="331" data-end="334" /> A RNP oferece a infraestrutura ideal para transformar sua
pesquisa em realidade com&nbsp;ambientes de teste isolados, escal&aacute;veis e seguros,
projetados para <strong data-start="574" data-end="608">experimenta&ccedil;&atilde;o em larga
escala</strong>, <strong data-start="610" data-end="644">avalia&ccedil;&atilde;o de novas
tecnologias</strong> e <strong data-start="647" data-end="698">valida&ccedil;&atilde;o de
solu&ccedil;&otilde;es cient&iacute;ficas ou de mercado</strong>.</p>
<p style="text-align: center;">Contate-nos por meio do atendimento@rnp.br.</p>
```

Iremos apontar essa página no nosso arquivo de configuração nginx.config:

```
cat nginx.config:
```

```
events {}

http {
    server {
        listen 80;

        location / {
            root /usr/share/nginx/html;
            index homepage.html;
```



```
    }  
  }  
}
```

```
cat deployment-v2.yaml
```

Para que o funcionamento da página ocorra com sucesso, precisamos realizar a criação de um Config Map , para isso iremos utilizar o comando localizado arquivo de texto comands.txt:

```
cat comands.txt
```

```
kubectl create configmap wtestbeds-nginx-config --from-file=nginx.conf --from-  
file=homepage.html
```

Após isso podemos criar o deployment para implementar a página:

```
cat deployment.yml
```

```
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: wtestbeds  
  labels:  
    app: wtestbeds  
spec:  
  replicas: 5  
  selector:  
    matchLabels:  
      app: wtestbeds  
  template:  
    metadata:  
      name: wtestbeds  
      labels:  
        app: wtestbeds  
    spec:  
      containers:  
        - name: wtestbeds  
          image: nginx:latest  
          volumeMounts:  
            - name: nginx-config-volume
```

```
    mountPath: /etc/nginx/nginx.conf
    subPath: nginx.conf
  - name: nginx-config-volume
    mountPath: /usr/share/nginx/html/homepage.html
    subPath: homepage.html
volumes:
  - name: nginx-config-volume
    configMap:
      name: wtestbeds-nginx-config
```

Aplica o comando abaixo para instanciar o deployment:

```
Kubectl apply -f deployment.yml
```

Observe se os pods foram criados corretamente de acordo com a quantidade de replicas indicada :

```
Kubectl get pod
```

Para uma descrição mais detalhada utilize:

```
kubectl get pods -o wide
```

Para observarmos a criação dos pods em tempo real podemos utilizar o seguinte comando :

```
watch ' kubectl get pod -o wide
```

## #Parte-03

Nessa etapa final iremos configurar o Ingress do NGINX .

Essa configuração do Ingress NGINX no Kubernetes permitirá a implementação de um controlador de entrada (**Ingress Controller**). Ele é responsável por gerenciar o acesso externo (via HTTP/HTTPS) aos serviços internos do cluster, funcionando semelhante a um **proxy reverso**.

Observe o arquivo de configuração e modifique o endereço host inserindo

**userX.wtestbeds.rnp.br:**

```
cat nginx-ingress.yml
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wtestbeds
  annotations:
    ingressClassName: "nginx"
spec:
  rules:
  - host: wtestbeds.rnp.br
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: wtestbeds
            port:
              number: 80
```

Aplique as configurações :

```
kubectl apply -f nginx-ingress.yml
```

Agora é possível acessar a página criada através do endereço:

userX.wtestbeds.rn.br:**Porta\_do\_serviço**

A porta utilizada deve ser a Node Port utilizada para instanciar o serviço anteriormente.

Ex: 300xx