# Wordpress no Kubernetes

## Introdução

Os arquivos YAML neste repositório são usados para definir diversos tipos de objetos Kubernetes, como Pods, Deployments, Services, ConfigMaps, Secrets, entre outros. Ao utilizar esses arquivos, você pode facilmente criar, atualizar e remover recursos em seu cluster Kubernetes.

## Premissas

- Ter o K3s instalado na máquina
- Ter o Kubectl instalado na máquina
- Descompactar os arquivos deste projeto ou clonar o repositório GIT
- Entrar na pasta raiz para executar os códigos

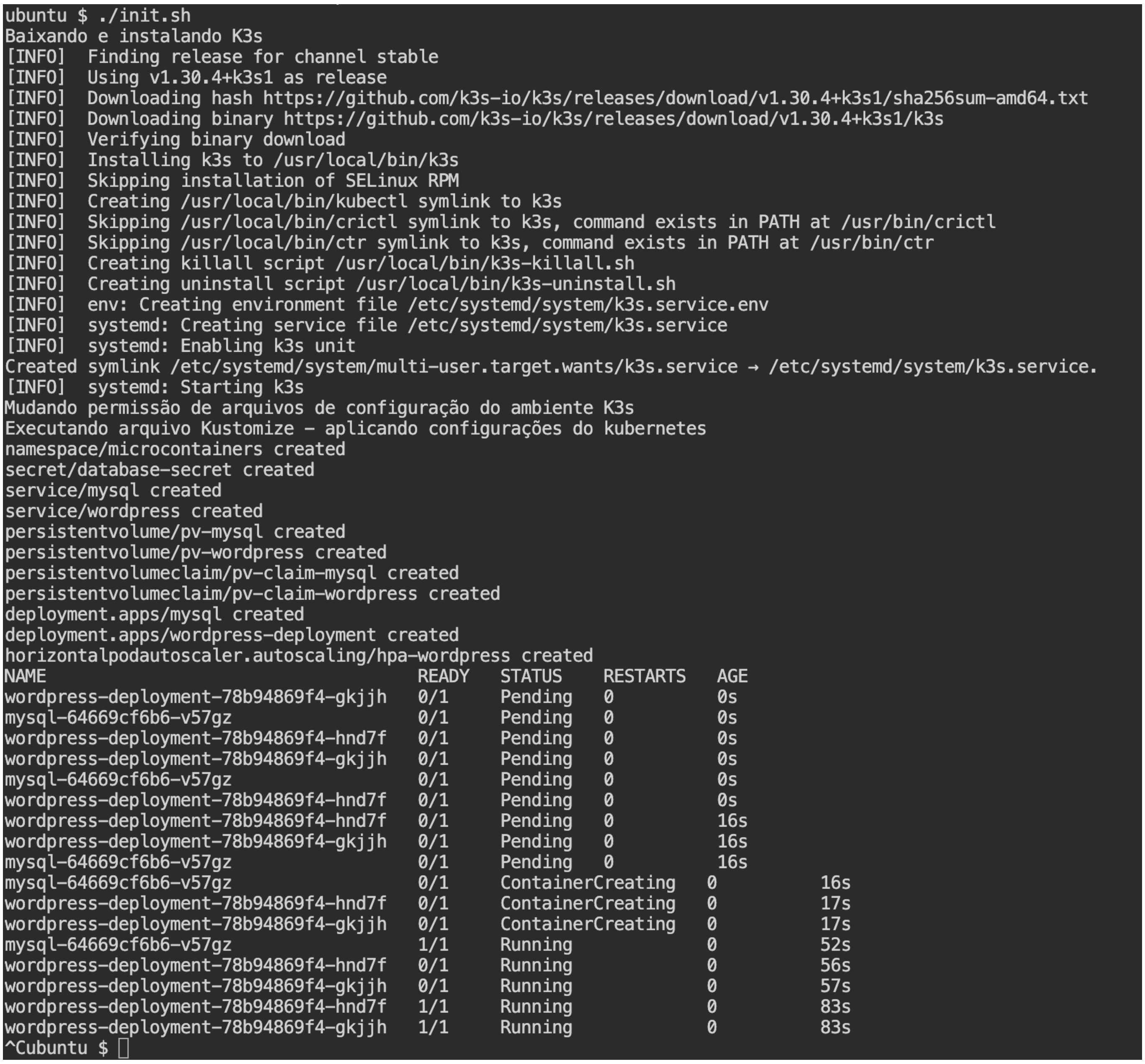
Na seção **Executando os passos preparatórios** ao final deste documento, você encontra as instruções para executar o passo-a-passo para criar o ambiente.

## Rodando automação no Linux Ubuntu

A maneira mais simples de executar e testar todo o provisionamento do Wordpress com banco de dados MySql em um ambiente Linux Ubuntu é executando o *shell script* **init.sh** localizado na raiz do projeto. Ele garante que a instalação do K3s exista na máquina e executa o arquivo *kustomization.yaml* responsável por aplicar as configurações no kubernetes.

```
# Executando automação
./init.sh
```

Ao final do processo, um comando iniciará em modo vigilante (watch) para validar se os containers foram criados e estão rodando corretamente. Assim que o banco de dados MySQL e duas instâncias do Wordpress estiverem rodando, poderemos abrir no navegador e iniciar as configurações do novo blog.



```
ubuntu $ ./init.sh
Baixando e instalando K3s
[INFO]  Finding release for channel stable
[INFO]  Using v1.30.4+k3s1 as release
[INFO]  Downloading hash https://github.com/k3s-io/k3s/releases/download/v1.30.4+k3s1/sha256sum-amd64.txt
[INFO]  Downloading binary https://github.com/k3s-io/k3s/releases/download/v1.30.4+k3s1/k3s
[INFO]  Verifying binary download
[INFO]  Installing k3s to /usr/local/bin/k3s
[INFO]  Skipping installation of SELinux RPM
[INFO]  Creating /usr/local/bin/kubectl symlink to k3s
[INFO]  Skipping /usr/local/bin/crictl symlink to k3s, command exists in PATH at /usr/bin/crictl
[INFO]  Skipping /usr/local/bin/ctr symlink to k3s, command exists in PATH at /usr/bin/ctr
[INFO]  Creating killall script /usr/local/bin/k3s-killall.sh
[INFO]  Creating uninstall script /usr/local/bin/k3s-uninstall.sh
[INFO]  env: Creating environment file /etc/systemd/system/k3s.service.env
[INFO]  systemd: Creating service file /etc/systemd/system/k3s.service
[INFO]  systemd: Enabling k3s unit
Created symlink /etc/systemd/system/multi-user.target.wants/k3s.service → /etc/systemd/system/k3s.service.
[INFO]  systemd: Starting k3s
Mudando permissão de arquivos de configuração do ambiente K3s
Executando arquivo Kustomize - aplicando configurações do kubernetes
namespace/microcontainers created
secret/database-secret created
service/mysql created
service/wordpress created
persistentvolume/pv-mysql created
persistentvolume/pv-wordpress created
persistentvolumeclaim/pv-claim-mysql created
persistentvolumeclaim/pv-claim-wordpress created
deployment.apps/mysql created
deployment.apps/wordpress-deployment created
horizontalpodautoscaler.autoscaling/hpa-wordpress created
NAME                                   READY   STATUS    RESTARTS   AGE
wordpress-deployment-78b94869f4-gkjjh   0/1     Pending   0          0s
mysql-64669cf6b6-v57gz                  0/1     Pending   0          0s
wordpress-deployment-78b94869f4-hnd7f   0/1     Pending   0          0s
wordpress-deployment-78b94869f4-gkjjh   0/1     Pending   0          0s
mysql-64669cf6b6-v57gz                  0/1     Pending   0          0s
wordpress-deployment-78b94869f4-hnd7f   0/1     Pending   0          0s
wordpress-deployment-78b94869f4-hnd7f   0/1     Pending   0          16s
wordpress-deployment-78b94869f4-gkjjh   0/1     Pending   0          16s
mysql-64669cf6b6-v57gz                  0/1     Pending   0          16s
mysql-64669cf6b6-v57gz                  0/1     ContainerCreating   0          16s
wordpress-deployment-78b94869f4-hnd7f   0/1     ContainerCreating   0          17s
wordpress-deployment-78b94869f4-gkjjh   0/1     ContainerCreating   0          17s
mysql-64669cf6b6-v57gz                  1/1     Running             0          52s
wordpress-deployment-78b94869f4-hnd7f   0/1     Running             0          56s
wordpress-deployment-78b94869f4-gkjjh   0/1     Running             0          57s
wordpress-deployment-78b94869f4-hnd7f   1/1     Running             0          83s
wordpress-deployment-78b94869f4-gkjjh   1/1     Running             0          83s
^Cubuntu $
```

Para sair do modo vigilante, basta pressionar `CTRL+C`.

Se todo o ambiente estiver pronto, é possível executar diretamente o *kustomization.yaml* da raiz do projeto. O comando já executa todas as etapas descritas em detalhes no passo-a-passo a seguir.

```
# Executando o kustomization.yaml
kubectl apply -k .
```

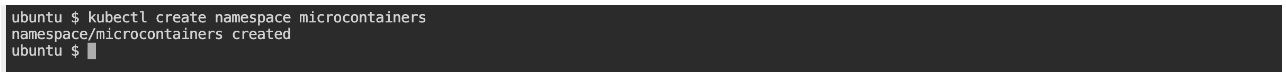
## Passo-a-passo

Caso você queira subir manualmente, abaixo está descrito cada uma das etapas e seus respectivos comandos, desde que já tenha executado todos os passos descritos na premissa anterior.

### Configuração do ambiente Wordpress

1. Criar o namespace dentro do cluster Kubernetes:

```
# Criando namespaces
kubectl create namespace microcontainers
```

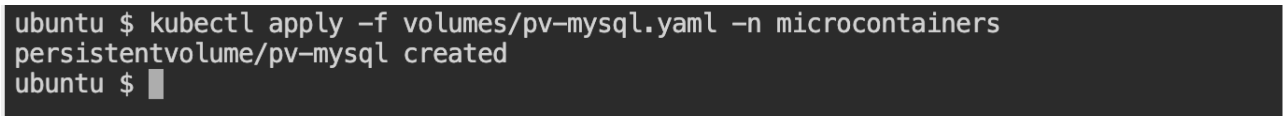


```
ubuntu $ kubectl create namespace microcontainers
namespace/microcontainers created
ubuntu $ 
```

2. Criar os volumes utilizados pelas PODs:

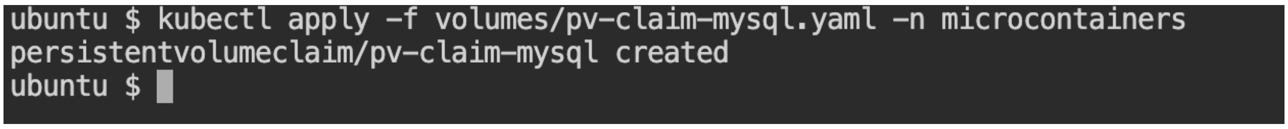
- **PersistentVolume(PV):** volume de armazenamento fisico, e é idependente do ciclo de vida da pod ou do namespace
- **PersistentVolumeClaim(PVC):** solicitação de armazenamento persistente feito pela pod, é feito um bind junto a PV

```
# Criar PV do banco de dados MYSQL:
kubectl apply -f volumes/pv-mysql.yaml -n microcontainers
```



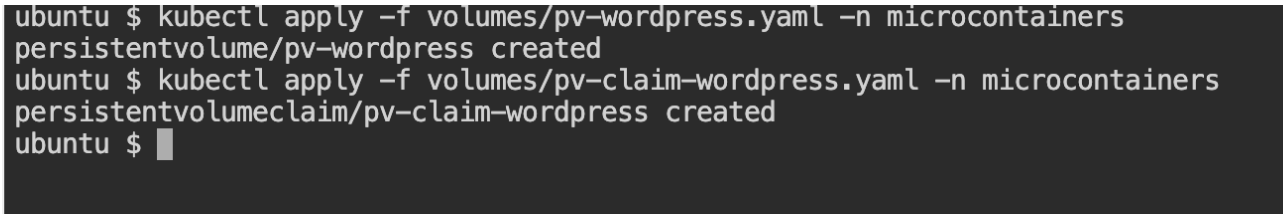
```
ubuntu $ kubectl apply -f volumes/pv-mysql.yaml -n microcontainers
persistentvolume/pv-mysql created
ubuntu $ 
```

```
# Criar PVC do banco de dados MYSQL:
kubectl apply -f volumes/pv-claim-mysql.yaml -n microcontainers
```



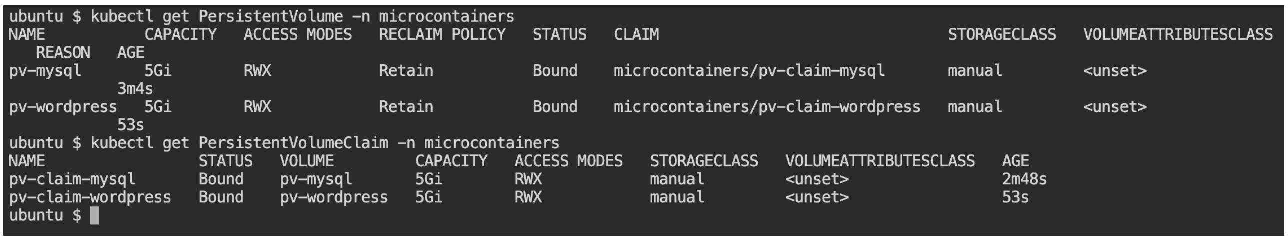
```
ubuntu $ kubectl apply -f volumes/pv-claim-mysql.yaml -n microcontainers
persistentvolumeclaim/pv-claim-mysql created
ubuntu $ 
```

```
# Criar PV e PVC do Wordpress:
kubectl apply -f volumes/pv-wordpress.yaml -n microcontainers
kubectl apply -f volumes/pv-claim-wordpress.yaml -n microcontainers
```



```
ubuntu $ kubectl apply -f volumes/pv-wordpress.yaml -n microcontainers
persistentvolume/pv-wordpress created
ubuntu $ kubectl apply -f volumes/pv-claim-wordpress.yaml -n microcontainers
persistentvolumeclaim/pv-claim-wordpress created
ubuntu $ 
```

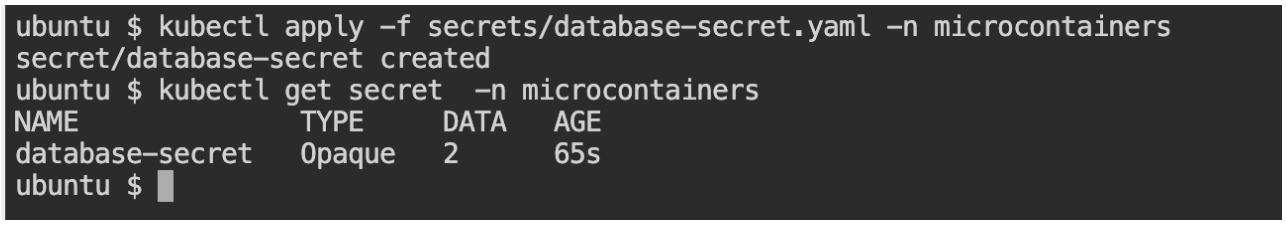
```
# Consultar volumes criados
kubectl get PersistentVolume -n microcontainers
kubectl get PersistentVolumeClaim -n microcontainers
```



```
ubuntu $ kubectl get PersistentVolume -n microcontainers
NAME           CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                                STORAGECLASS   VOLUMEATTRIBUTESCLASS
   REASON   AGE
pv-mysql       5Gi        RWX            Retain           Bound    microcontainers/pv-claim-mysql       manual         <unset>
            3m4s
pv-wordpress   5Gi        RWX            Retain           Bound    microcontainers/pv-claim-wordpress   manual         <unset>
            53s
ubuntu $ kubectl get PersistentVolumeClaim -n microcontainers
NAME                 STATUS   VOLUME         CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
pv-claim-mysql       Bound    pv-mysql       5Gi        RWX            manual         <unset>                 2m48s
pv-claim-wordpress   Bound    pv-wordpress   5Gi        RWX            manual         <unset>                 53s
ubuntu $
```

3. Criar Secret com credenciais de acesso ao banco de dados:

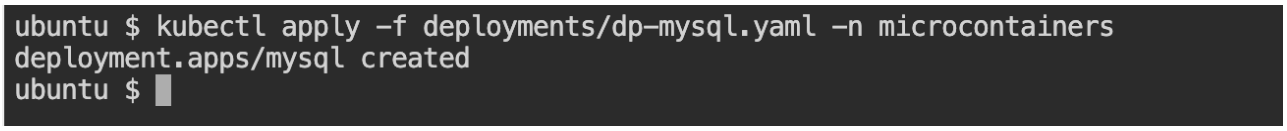
```
# Criar secret
kubectl apply -f secrets/database-secret.yaml -n microcontainers

# Consultar secret criado
kubectl get secret  -n microcontainers
```



```
ubuntu $ kubectl apply -f secrets/database-secret.yaml -n microcontainers
secret/database-secret created
ubuntu $ kubectl get secret  -n microcontainers
NAME              TYPE     DATA   AGE
database-secret   Opaque   2      65s
ubuntu $
```

4. Deploy do banco de dados MySQL:

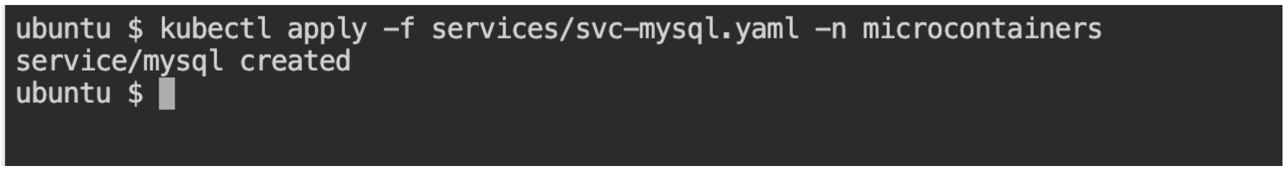
```
# Criando as pods do MySQL
kubectl apply -f deployments/dp-mysql.yaml -n microcontainers
```



```
ubuntu $ kubectl apply -f deployments/dp-mysql.yaml -n microcontainers
deployment.apps/mysql created
ubuntu $
```

5. Criar service do MySQL:

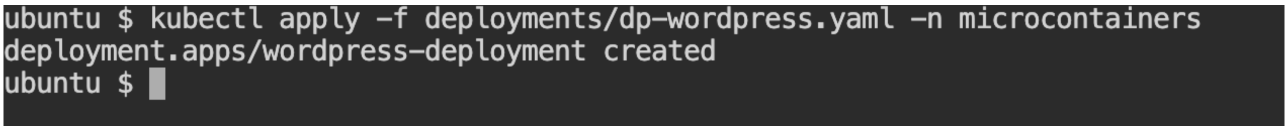
```
# Service do tipo ClusterIP: permite acesso somente de dentro do
cluster
kubectl apply -f services/svc-mysql.yaml -n microcontainers
```



```
ubuntu $ kubectl apply -f services/svc-mysql.yaml -n microcontainers
service/mysql created
ubuntu $
```

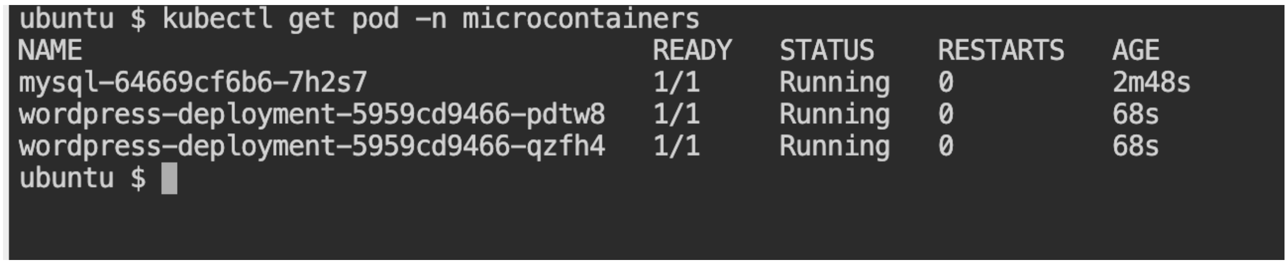
6. Deploy do Wordpress:

```
# Criando as pods do WP
kubectl apply -f deployments/dp-wordpress.yaml -n microcontainers
```



```
ubuntu $ kubectl apply -f deployments/dp-wordpress.yaml -n microcontainers
deployment.apps/wordpress-deployment created
ubuntu $
```

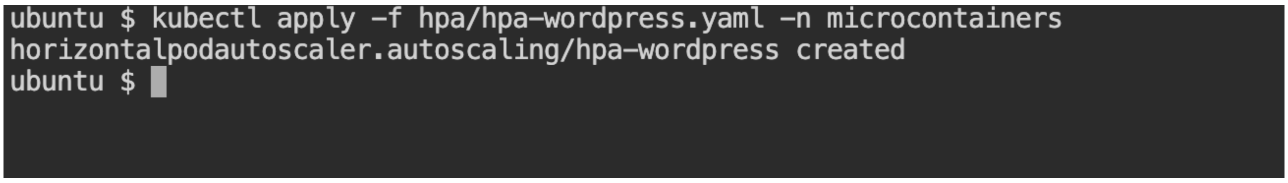
```
# Consultar PODs criadas(Wordpress e Mysql)
kubectl get pod -n microcontainers
```



```
ubuntu $ kubectl get pod -n microcontainers
NAME                                    READY   STATUS    RESTARTS   AGE
mysql-64669cf6b6-7h2s7                  1/1     Running   0          2m48s
wordpress-deployment-5959cd9466-pdtw8   1/1     Running   0          68s
wordpress-deployment-5959cd9466-qzfh4   1/1     Running   0          68s
ubuntu $
```

7. Regra de escalabilidade das PODs do Wordpress:

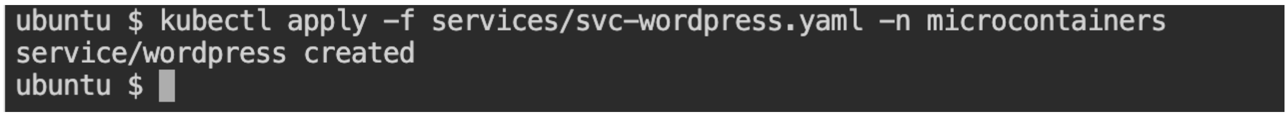
```
kubectl apply -f hpa/hpa-wordpress.yaml -n microcontainers
```



```
ubuntu $ kubectl apply -f hpa/hpa-wordpress.yaml -n microcontainers
horizontalpodautoscaler.autoscaling/hpa-wordpress created
ubuntu $
```

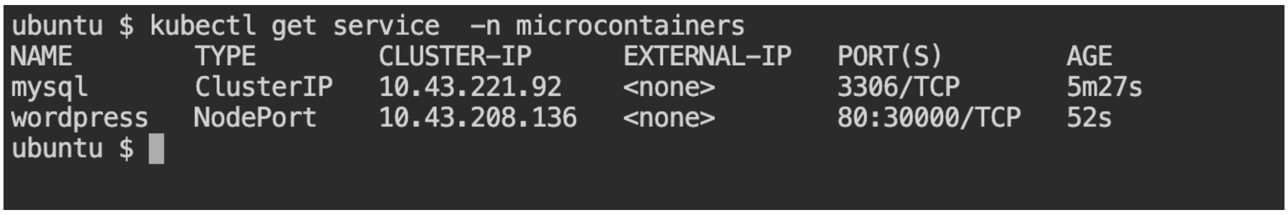
8. Criar service do Wordpress:

```
# Service do tipo NodePort: fazendo um bind da porta 8080 para a porta
80 dentro do cluster (rodando em ambiente local)
kubectl apply -f services/svc-wordpress.yaml -n microcontainers
```



```
ubuntu $ kubectl apply -f services/svc-wordpress.yaml -n microcontainers
service/wordpress created
ubuntu $
```

```
# Consultar services criados e portas expostas
kubectl get service  -n microcontainers
```



```
ubuntu $ kubectl get service  -n microcontainers
NAME        TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
mysql       ClusterIP   10.43.221.92    <none>        3306/TCP       5m27s
wordpress   NodePort    10.43.208.136   <none>        80:30000/TCP   52s
ubuntu $
```

## Acesso ao Wordpress

Depois do ambiente provisionado, teremos que acessar a página do Wordpress na porta correspondente.

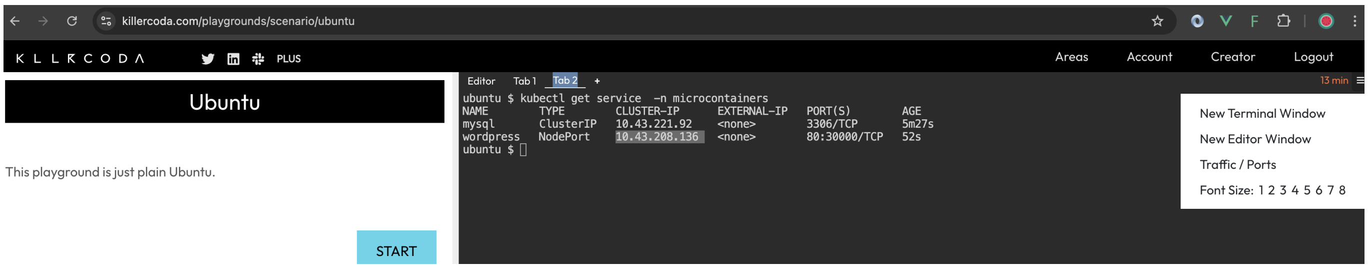
### Ambiente Local

Se o tutorial for executado em ambiente local, acessar a url http://localhost:8080 no navegador para abrir a página do Wordpress.

### Killercoda

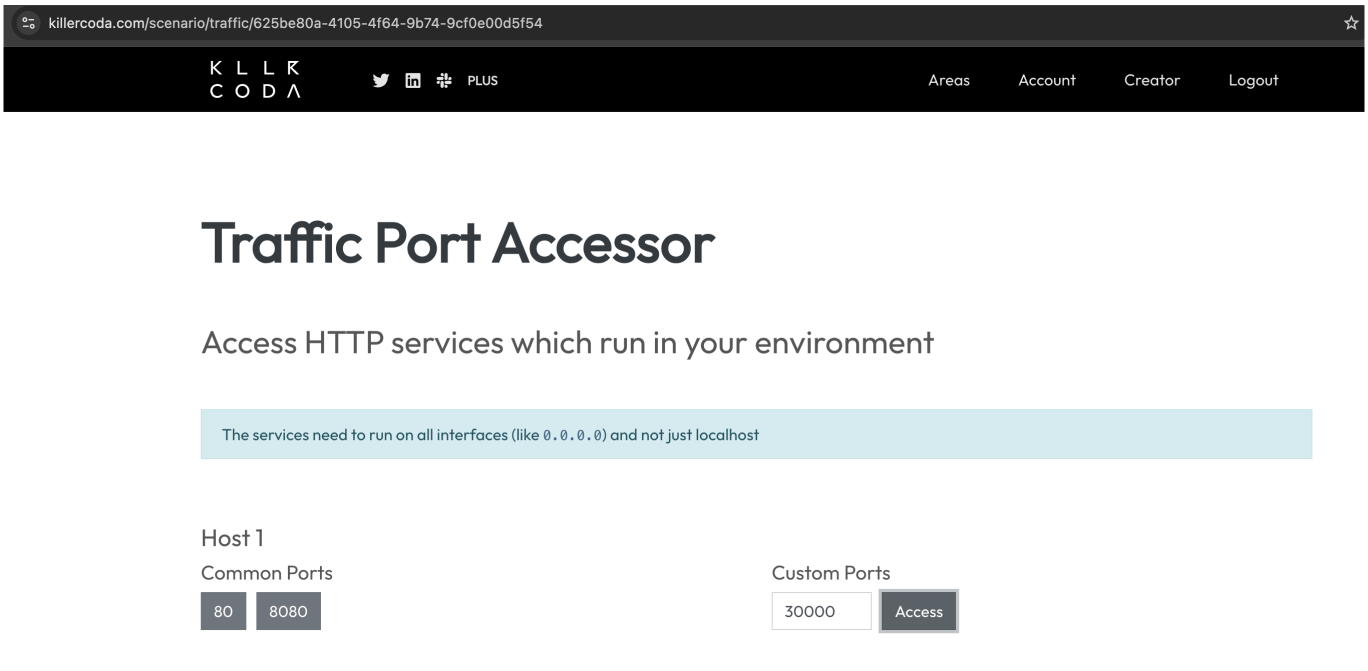
Se o tutorial for executado em uma máquina linux virtual, como no KillerCoda (https://killercoda.com/playgrounds/scenario/ubuntu), é necessário fazer a configuração abaixo:

Acessar item **Traffic Ports** no menu da direita



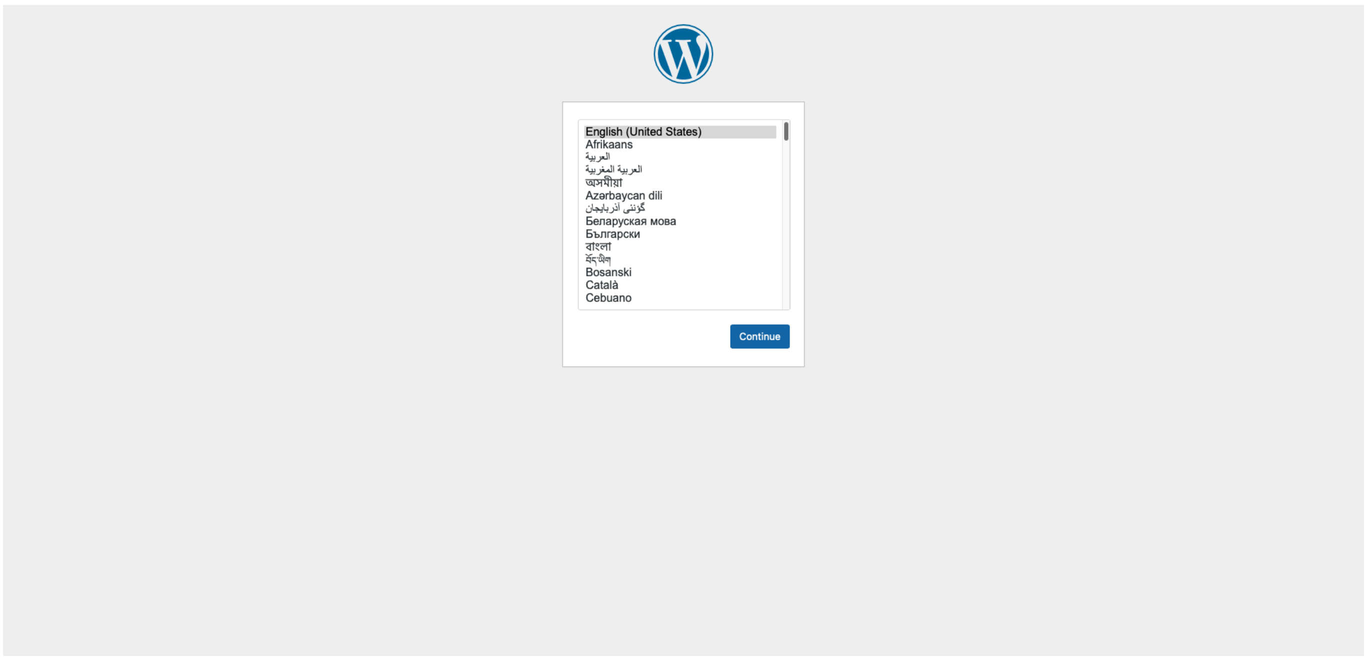
```
ubuntu $ kubectl get service  -n microcontainers
NAME        TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)         AGE
mysql       ClusterIP   10.43.221.92    <none>        3306/TCP        5m27s
wordpress   NodePort    10.43.208.136   <none>        80:30000/TCP    52s
ubuntu $ 
```

Configurar uma porta 30000 customizada e clicar em Acessar

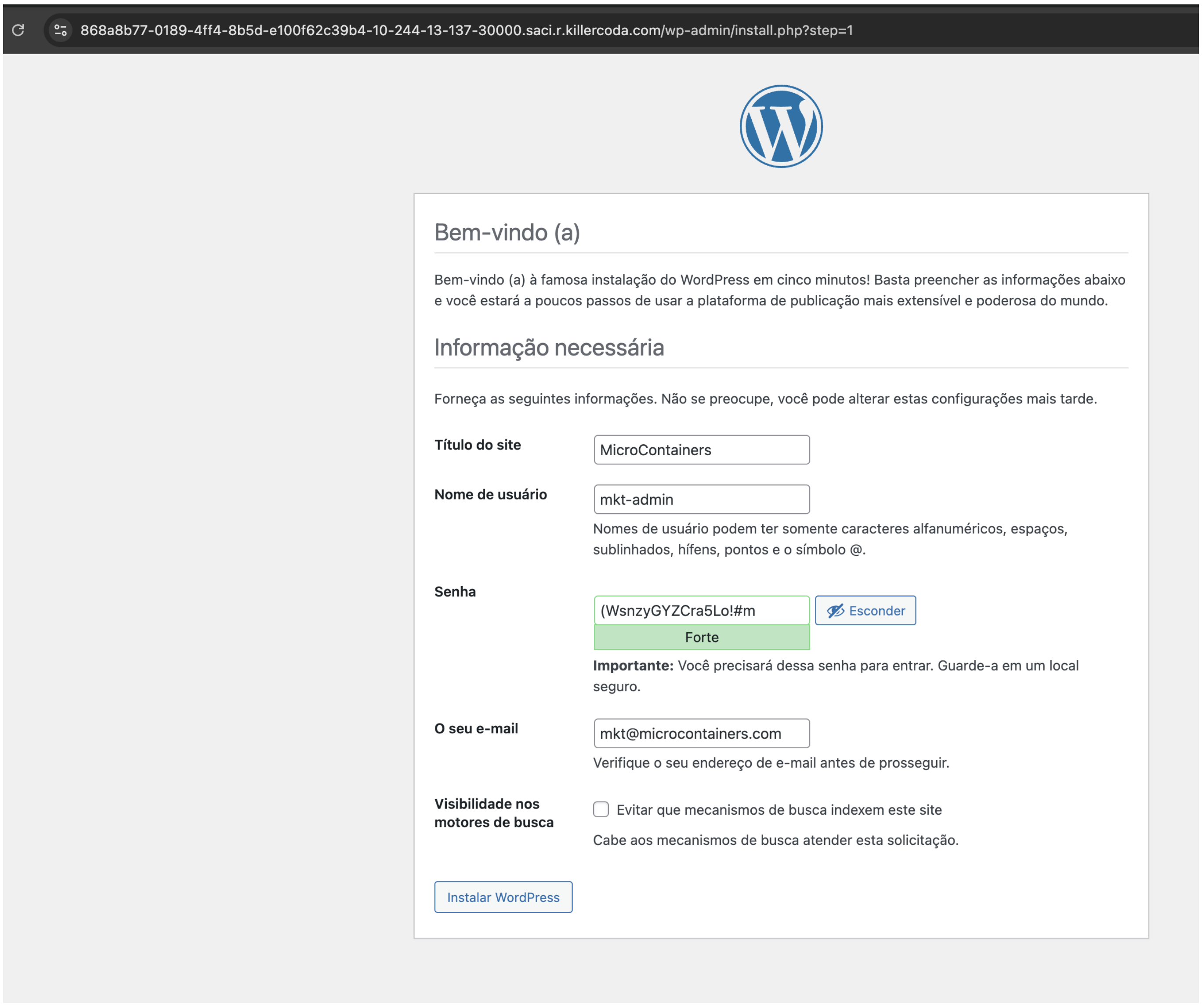

killercoda.com/scenario/traffic/625be80a-4105-4f64-9b74-9cf0e00d5f54
PLUS
Areas
Account
Creator
Logout
Traffic Port Accessor
Access HTTP services which run in your environment
The services need to run on all interfaces (like 0.0.0.0) and not just localhost
Host 1
Common Ports
80
8080
Custom Ports
30000
Access

### Configurando o Wordpress

No primeiro acesso deverá visualizar a seguinte página, onde é possível configurar *Português do Brasil* ou qualquer outra língua de seu interesse:



Abaixo algumas telas de Configurações do WordPress e páginas do Painel de Administração:



868a8b77-0189-4ff4-8b5d-e100f62c39b4-10-244-13-137-30000.saci.r.killercoda.com/wp-admin/install.php?step=1

Bem-vindo (a)

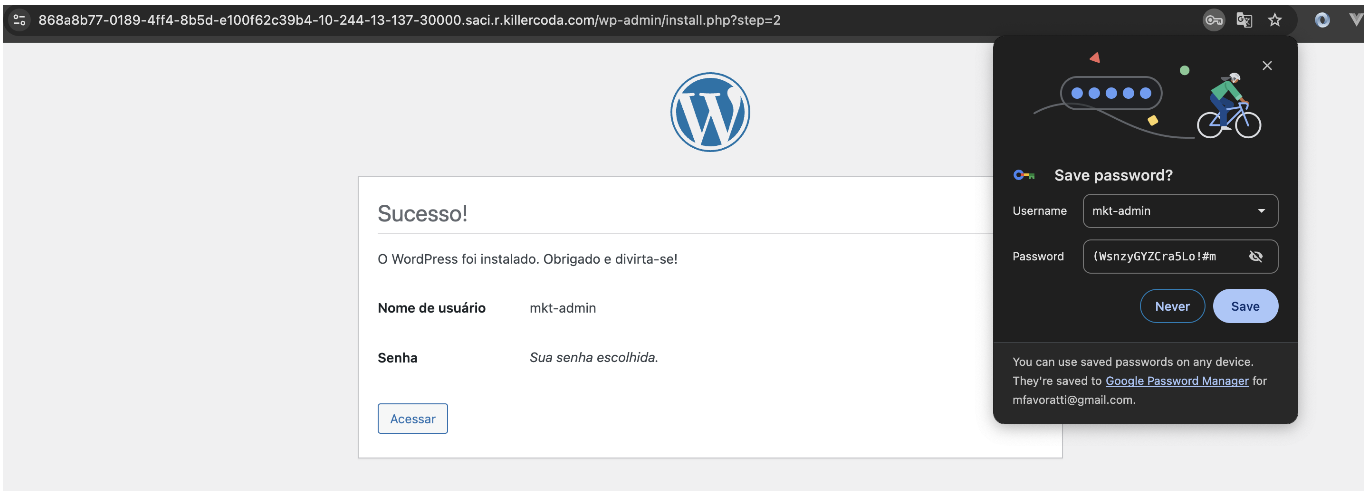
Bem-vindo (a) à famosa instalação do WordPress em cinco minutos! Basta preencher as informações abaixo e você estará a poucos passos de usar a plataforma de publicação mais extensível e poderosa do mundo.

Informação necessária

Forneça as seguintes informações. Não se preocupe, você pode alterar estas configurações mais tarde.

| | |
|---|---|
| **Título do site** | MicroContainers |
| **Nome de usuário** | mkt-admin<br>Nomes de usuário podem ter somente caracteres alfanuméricos, espaços, sublinhados, hífens, pontos e o símbolo @. |
| **Senha** | (WsnzyGYZCra5Lo!#m Esconder<br>Forte<br>**Importante:** Você precisará dessa senha para entrar. Guarde-a em um local seguro. |
| **O seu e-mail** | mkt@microcontainers.com<br>Verifique o seu endereço de e-mail antes de prosseguir. |
| **Visibilidade nos motores de busca** | ☐ Evitar que mecanismos de busca indexem este site<br>Cabe aos mecanismos de busca atender esta solicitação. |

Instalar WordPress



868a8b77-0189-4ff4-8b5d-e100f62c39b4-10-244-13-137-30000.saci.r.killercoda.com/wp-admin/install.php?step=2

Sucesso!

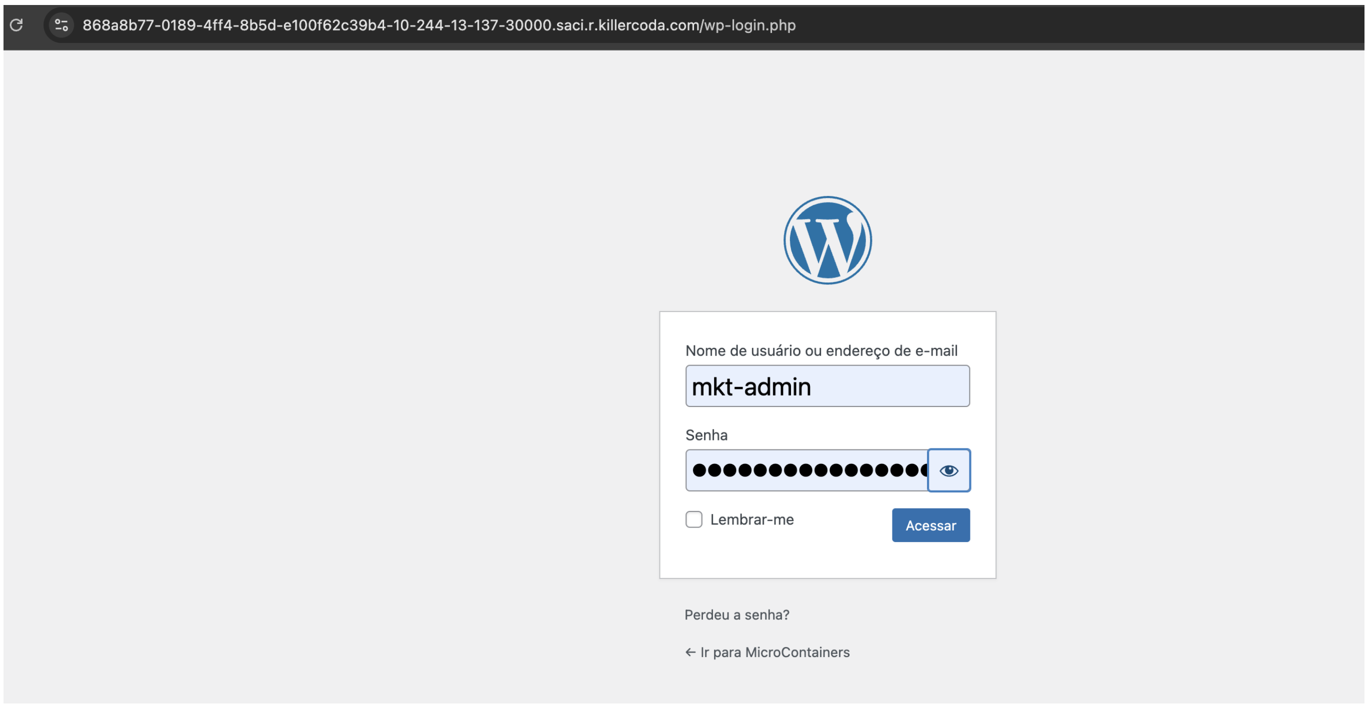
O WordPress foi instalado. Obrigado e divirta-se!

| | |
|---|---|
| **Nome de usuário** | mkt-admin |
| **Senha** | *Sua senha escolhida.* |

Save password?

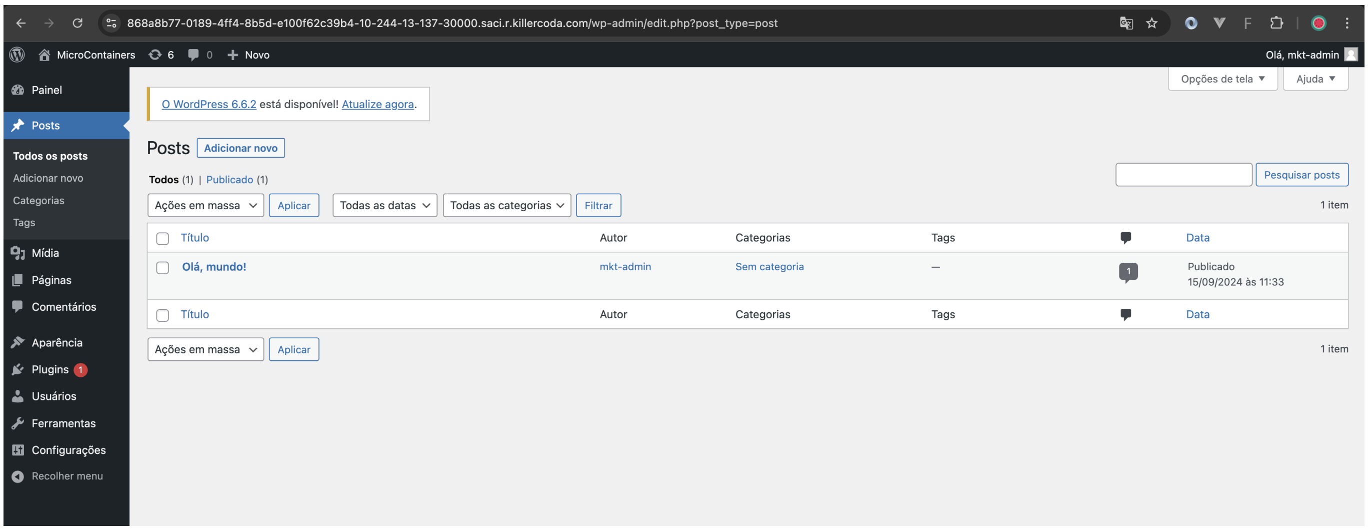
| | |
|---|---|
| Username | mkt-admin |
| Password | (WsnzyGYZCra5Lo!#m |

You can use saved passwords on any device. They're saved to Google Password Manager for mfavoratti@gmail.com.



868a8b77-0189-4ff4-8b5d-e100f62c39b4-10-244-13-137-30000.saci.r.killercoda.com/wp-login.php

| | |
|---|---|
| Nome de usuário ou endereço de e-mail | mkt-admin |
| Senha | ●●●●●●●●●●●●●●● |



868a8b77-0189-4ff4-8b5d-e100f62c39b4-10-244-13-137-30000.saci.r.killercoda.com/wp-admin/edit.php?post_type=post

O WordPress 6.6.2 está disponível! Atualize agora.

Posts

Todos (1) | Publicado (1)

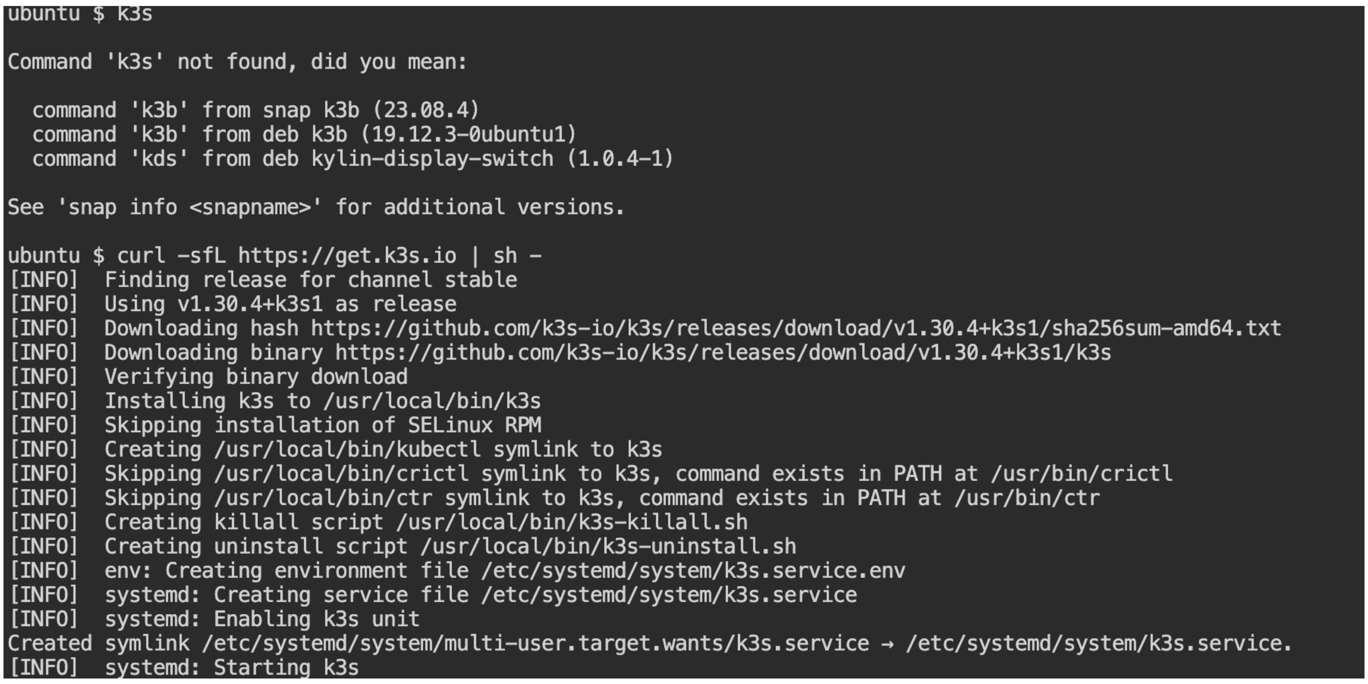
| Título | Autor | Categorias | Tags | | Data |
|---|---|---|---|---|---|
| Olá, mundo! | mkt-admin | Sem categoria | — | 1 | Publicado 15/09/2024 às 11:33 |

1 item

## Executando os passos preparatórios

### Ter o K3s instalado na máquina

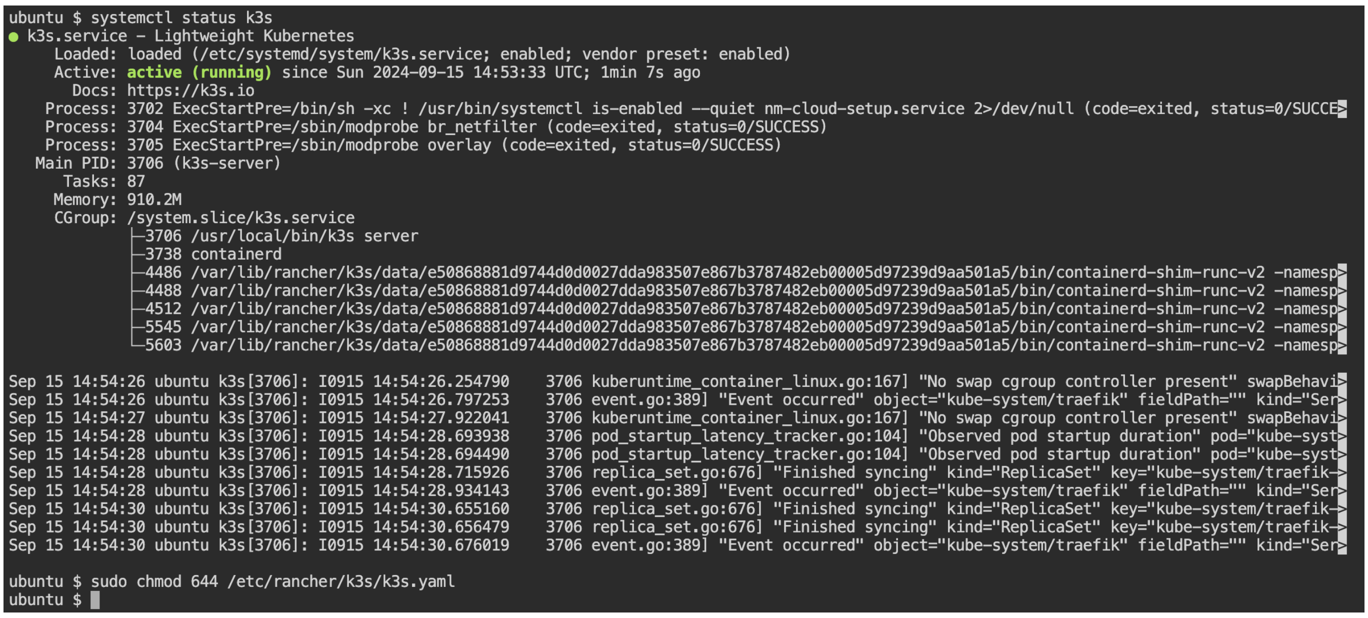
```
curl -sfL https://get.k3s.io | sh -
```



```
ubuntu $ k3s

Command 'k3s' not found, did you mean:

  command 'k3b' from snap k3b (23.08.4)
  command 'k3b' from deb k3b (19.12.3-0ubuntu1)
  command 'kds' from deb kylin-display-switch (1.0.4-1)

See 'snap info <snapname>' for additional versions.

ubuntu $ curl -sfL https://get.k3s.io | sh -
[INFO]  Finding release for channel stable
[INFO]  Using v1.30.4+k3s1 as release
[INFO]  Downloading hash https://github.com/k3s-io/k3s/releases/download/v1.30.4+k3s1/sha256sum-amd64.txt
[INFO]  Downloading binary https://github.com/k3s-io/k3s/releases/download/v1.30.4+k3s1/k3s
[INFO]  Verifying binary download
[INFO]  Installing k3s to /usr/local/bin/k3s
[INFO]  Skipping installation of SELinux RPM
[INFO]  Creating /usr/local/bin/kubectl symlink to k3s
[INFO]  Skipping /usr/local/bin/crictl symlink to k3s, command exists in PATH at /usr/bin/crictl
[INFO]  Skipping /usr/local/bin/ctr symlink to k3s, command exists in PATH at /usr/bin/ctr
[INFO]  Creating killall script /usr/local/bin/k3s-killall.sh
[INFO]  Creating uninstall script /usr/local/bin/k3s-uninstall.sh
[INFO]  env: Creating environment file /etc/systemd/system/k3s.service.env
[INFO]  systemd: Creating service file /etc/systemd/system/k3s.service
[INFO]  systemd: Enabling k3s unit
Created symlink /etc/systemd/system/multi-user.target.wants/k3s.service → /etc/systemd/system/k3s.service.
[INFO]  systemd: Starting k3s
```

## Validando o status do k3s

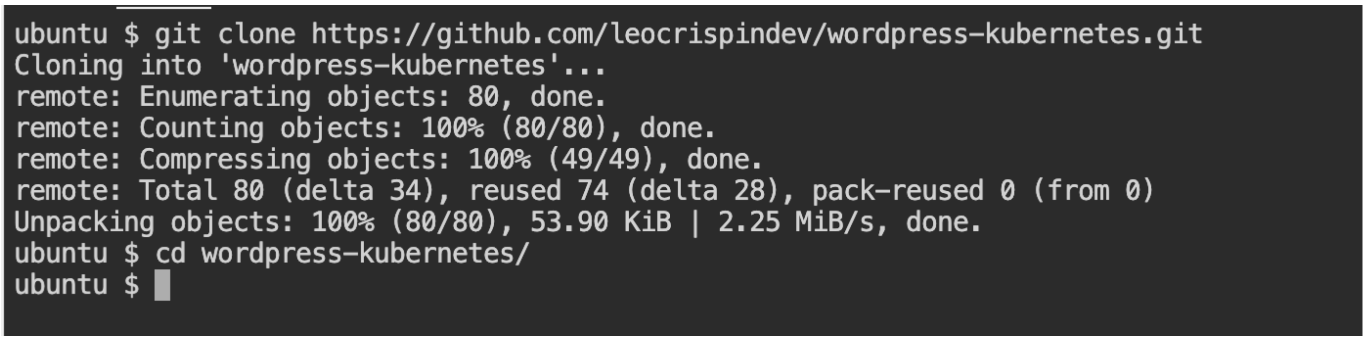
```
systemctl status k3s
```



```
ubuntu $ systemctl status k3s
● k3s.service - Lightweight Kubernetes
     Loaded: loaded (/etc/systemd/system/k3s.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-09-15 14:53:33 UTC; 1min 7s ago
       Docs: https://k3s.io
    Process: 3702 ExecStartPre=/bin/sh -xc ! /usr/bin/systemctl is-enabled --quiet nm-cloud-setup.service 2>/dev/null (code=exited, status=0/SUCCE>
    Process: 3704 ExecStartPre=/sbin/modprobe br_netfilter (code=exited, status=0/SUCCESS)
    Process: 3705 ExecStartPre=/sbin/modprobe overlay (code=exited, status=0/SUCCESS)
   Main PID: 3706 (k3s-server)
      Tasks: 87
     Memory: 910.2M
     CGroup: /system.slice/k3s.service
             ├─3706 /usr/local/bin/k3s server
             ├─3738 containerd
             ├─4486 /var/lib/rancher/k3s/data/e50868881d9744d0d0027dda983507e867b3787482eb00005d97239d9aa501a5/bin/containerd-shim-runc-v2 -namesp>
             ├─4488 /var/lib/rancher/k3s/data/e50868881d9744d0d0027dda983507e867b3787482eb00005d97239d9aa501a5/bin/containerd-shim-runc-v2 -namesp>
             ├─4512 /var/lib/rancher/k3s/data/e50868881d9744d0d0027dda983507e867b3787482eb00005d97239d9aa501a5/bin/containerd-shim-runc-v2 -namesp>
             ├─5545 /var/lib/rancher/k3s/data/e50868881d9744d0d0027dda983507e867b3787482eb00005d97239d9aa501a5/bin/containerd-shim-runc-v2 -namesp>
             └─5603 /var/lib/rancher/k3s/data/e50868881d9744d0d0027dda983507e867b3787482eb00005d97239d9aa501a5/bin/containerd-shim-runc-v2 -namesp>

Sep 15 14:54:26 ubuntu k3s[3706]: I0915 14:54:26.254790    3706 kuberuntime_container_linux.go:167] "No swap cgroup controller present" swapBehavi>
Sep 15 14:54:26 ubuntu k3s[3706]: I0915 14:54:26.797253    3706 event.go:389] "Event occurred" object="kube-system/traefik" fieldPath="" kind="Ser>
Sep 15 14:54:27 ubuntu k3s[3706]: I0915 14:54:27.922041    3706 kuberuntime_container_linux.go:167] "No swap cgroup controller present" swapBehavi>
Sep 15 14:54:28 ubuntu k3s[3706]: I0915 14:54:28.693938    3706 pod_startup_latency_tracker.go:104] "Observed pod startup duration" pod="kube-syst>
Sep 15 14:54:28 ubuntu k3s[3706]: I0915 14:54:28.694490    3706 pod_startup_latency_tracker.go:104] "Observed pod startup duration" pod="kube-syst>
Sep 15 14:54:28 ubuntu k3s[3706]: I0915 14:54:28.715926    3706 replica_set.go:676] "Finished syncing" kind="ReplicaSet" key="kube-system/traefik->
Sep 15 14:54:28 ubuntu k3s[3706]: I0915 14:54:28.934143    3706 event.go:389] "Event occurred" object="kube-system/traefik" fieldPath="" kind="Ser>
Sep 15 14:54:30 ubuntu k3s[3706]: I0915 14:54:30.655160    3706 replica_set.go:676] "Finished syncing" kind="ReplicaSet" key="kube-system/traefik->
Sep 15 14:54:30 ubuntu k3s[3706]: I0915 14:54:30.656479    3706 replica_set.go:676] "Finished syncing" kind="ReplicaSet" key="kube-system/traefik->
Sep 15 14:54:30 ubuntu k3s[3706]: I0915 14:54:30.676019    3706 event.go:389] "Event occurred" object="kube-system/traefik" fieldPath="" kind="Ser>

ubuntu $ sudo chmod 644 /etc/rancher/k3s/k3s.yaml
ubuntu $ 
```

## Clonando o repositório GIT e entrando na pasta raiz

```
git clone https://github.com/leocrispindev/wordpress-kubernetes.git && cd
wordpress-kubernetes
```



```
ubuntu $ git clone https://github.com/leocrispindev/wordpress-kubernetes.git
Cloning into 'wordpress-kubernetes'...
remote: Enumerating objects: 80, done.
remote: Counting objects: 100% (80/80), done.
remote: Compressing objects: 100% (49/49), done.
remote: Total 80 (delta 34), reused 74 (delta 28), pack-reused 0 (from 0)
Unpacking objects: 100% (80/80), 53.90 KiB | 2.25 MiB/s, done.
ubuntu $ cd wordpress-kubernetes/
ubuntu $
```