Tutorial Kubernets WordPress

REPO: https://github.com/hugobarbato/k8s-wordpress/tree/main

Para provisionar um site WordPress com um banco de dados MySQL no Kubernetes usando K3S, siga os passos abaixo.

** Este guia é projetado para ser seguido em uma VM Ubuntu e assume conhecimento básico de Kubernetes e uso de linha de comando.

1. Preparação do Ambiente

Para começarmos vamos preparar nosso ambiente linux para implementação desse lab.

Instalação do K3S no Ubuntu

1. **Atualize o sistema**: Antes de começar, é sempre uma boa prática atualizar os pacotes do seu sistema Ubuntu.

```
# Atualizar o software
sudo apt update && sudo apt upgrade -y
```

2. **Instale o K3S**: O K3S é uma distribuição leve do Kubernetes fácil de instalar. Execute o seguinte comando para instalá-lo.

```
# Instalar K3S com kubectl
curl -sfL https://get.k3s.io | sh -
# Habilitar permissão para usuário ubuntu obter configuração de acesso ao K8S
echo K3S_KUBECONFIG_MODE=\"644\" >> /etc/systemd/system/k3s.service.env
# Restart do serviço k3s para carregar nova config
systemctl restart k3s
```

3. **Verifique a instalação**: Após a instalação, confirme se o K3S está rodando corretamente.

```
# Logar com usuário ubuntu (caso pedir senha, utilizar `ubuntu`)
su ubuntu
# Obter informações dos nós do cluster K8S
kubectl get nodes
# Observe que possuímos uma única instância de K8S
```

Você deve ver o nó do seu servidor listado, indicando que o K3S está rodando.

---

2. Preparação dos Arquivos de Configuração do Kubernetes

Você precisará configurar o MySQL e WordPress no Kubernets para isso precisaremos da configuração de alguns recursos. Você poderá clonar este repositorio ou criar os arquivos por linha de comando seguindo o padrão:

```
nano <NOME_DO_ARQUIVO>.yml
```

MySQL

Para configurar o MySQL no Kubernetes envolve a criação de um conjunto de configurações, incluindo `Secret`, `PersistentVolumeClaim` (PVC), `Deployment` e `Service`.

1. Secret

Uma boa pratica em Kubernets é armazenar segredos em cofres seguros ao inves do codigo, para criar a secret execute o seguinte comando:

```
cat <<EOF >./kustomization.yaml
secretGenerator:
- name: mysql-pass
  literals:
  - password=YOUR_PASSWORD
EOF
```

Esse comando criou um arquivo `kustomization.yaml` que irá ser nosso arquivo de configuração principal.

2. PersistentVolumeClaim - mysql.yml

O PersistentVolumeClaim é um recurso que representa um volume persistente no cluster, adicione ao arquivo mysql.yml o seguinte trecho:

```
--- #PVC
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi #Espaço disponibilizado para o MySQL
```

O PVC(PersistentVolumeClaim) é usado pelo pod do MySQL para requisitar o armazenamento físico.

3. Deployment - mysql.yml

O Deployment gerencia o pod do MySQL, adicione ao arquivo mysql.yml o seguinte trecho:

```
--- # Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: mysql
    spec:
      containers:
      - image: mysql:8.0
        name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password
        - name: MYSQL_DATABASE
          value: wordpress
        - name: MYSQL_USER
          value: wordpress
        - name: MYSQL_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
        resources:
          requests:
            memory: "200Mi"
            cpu: "100m" #(0.1 vCPU)
```

```
            limits:
              memory: "1000Mi"
              cpu: "500m" #(0.5 vCPU)
        volumes:
        - name: mysql-persistent-storage
          persistentVolumeClaim:
            claimName: mysql-pv-claim
```

4. Service - mysql.yml

O Service define como expor o aplicativo MySQL, geralmente dentro do cluster, permitindo que outros serviços ou pods se conectem ao MySQL, adicione ao arquivo mysql.yml o seguinte trecho:

```
--- # Service
apiVersion: v1
kind: Service
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  ports:
    - port: 3306
  selector:
    app: wordpress
    tier: mysql
  clusterIP: None
```

5. Considerações

- **Segurança**: Garanta que a senha do MySQL (`MYSQL_ROOT_PASSWORD`) não seja armazenada em texto claro nos arquivos. Edite as secrets do Kubernetes utilizando `kubectl edit secret mysql-root-password`.
- **Storage**: Ajuste o tamanho do PV (`resources.requests.storage`) conforme necessário.
- **Versão do MySQL**: Certifique-se de que a versão do MySQL (`image: mysql:8`) esteja alinhada com seus requisitos de aplicação WordPress.

WordPress

Para configurar o WordPress no Kubernetes envolve a criação de um conjunto de configurações, incluindo `PersistentVolumeClaim` (PV), `Deployment` e `Service`.

1. PersistentVolumeClaim - wordpress.yml

O PersistentVolume é um recurso que representa um volume persistente no cluster, adicione ao arquivo wordpress.yml o seguinte trecho:

```
--- #PVC
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi #Espaço disponibilizado para o WordPress
```

O PVC(PersistentVolumeClaim) é usado pelo pod do WordPress para requisitar o armazenamento físico.

2. Deployment - wordpress.yml

O Deployment gerencia pods do WordPress, garantindo que o número desejado de instâncias esteja rodando e atualiza os pods de forma declarativa, adicione ao arquivo wordpress.yml o seguinte trecho:

```
--- # Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  replicas: 3
  selector:
    matchLabels:
      app: wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: frontend
    spec:
      containers:
      - image: wordpress:6.2.1-apache
        name: wordpress3
        env:
        - name: WORDPRESS_DB_HOST
          value: wordpress-mysql
        - name: WORDPRESS_DB_PASSWORD
          valueFrom:
```

```
            secretKeyRef:
              name: mysql-pass
              key: password
        - name: WORDPRESS_DB_USER
          value: wordpress
        ports:
        - containerPort: 80
          name: wordpress
        volumeMounts:
        - name: wordpress-persistent-storage
          mountPath: /var/www/html
        resources:
          requests:
            memory: "200Mi"
            cpu: "100m" #(0.1 vCPU)
          limits:
            memory: "1000Mi"
            cpu: "500m" #(0.5 vCPU)
      volumes:
      - name: wordpress-persistent-storage
        persistentVolumeClaim:
          claimName: wp-pv-claim
```

3. Service - wordpress.yml

O Service define como expor o aplicativo WordPress, geralmente dentro do cluster, permitindo que outros serviços ou pods se conectem ao WordPress, adicione ao arquivo wordpress.yml o seguinte trecho:

```
--- # Service
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  ports:
    - port: 80
  selector:
    app: wordpress
    tier: frontend
  type: LoadBalancer
```

Aplicando as configurações

1. Adicione os arquivos criados ao `kustomization.yaml` como recursos a serem criados, rodando o comando a seguir:

```
cat <<EOF >>./kustomization.yaml
resources:
  - mysql.yml
  - wordpress.yml
EOF
```

- Caso não tenha criado os arquivos pode optar por baixar eles por meio do seguinte comando

```
curl -LO https://raw.githubusercontent.com/hugobarbato/k8s-
wordpress/main/mysql.yml
curl -LO https://raw.githubusercontent.com/hugobarbato/k8s-
wordpress/main/wordpress.yml
```

1. Aplique o arquivo `kustomization.yaml` atraves do comando:

```
  kubectl apply -k ./
```

2. Você pode confirmar a criação dos recursos utilizando o seguinte comando:

```
  kubectl get pv,pvc,secret,deployment,pods,service
```

3. Acesso ao WordPress

Após a aplicação dos arquivos de configuração, o WordPress estará acessível através do endereço IP do Serviço. Você pode encontrar este endereço IP e Porta com o seguinte comando:

```
kubectl get service
```

Agora, abra um navegador e acesse o WordPress usando o endereço IP e Porta encontrado.

4. Testando a resiliencia

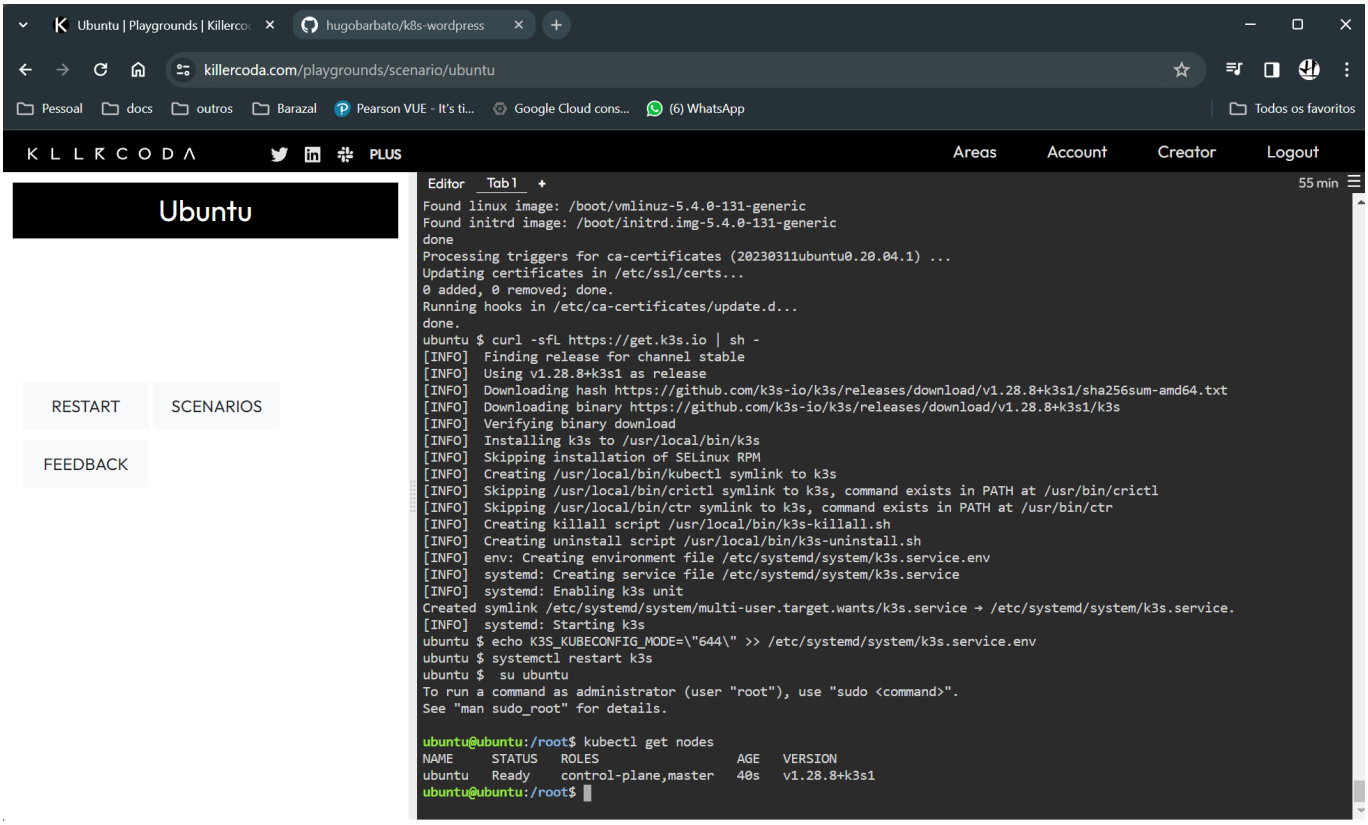
Ao matar qualquer pod da solução, por conta da `strategy` estar como `Recreate` nos deployments ele serão automaticamente recriados sem perder as informações por conta do `PersistentVolume`.

Teste usando o comando:

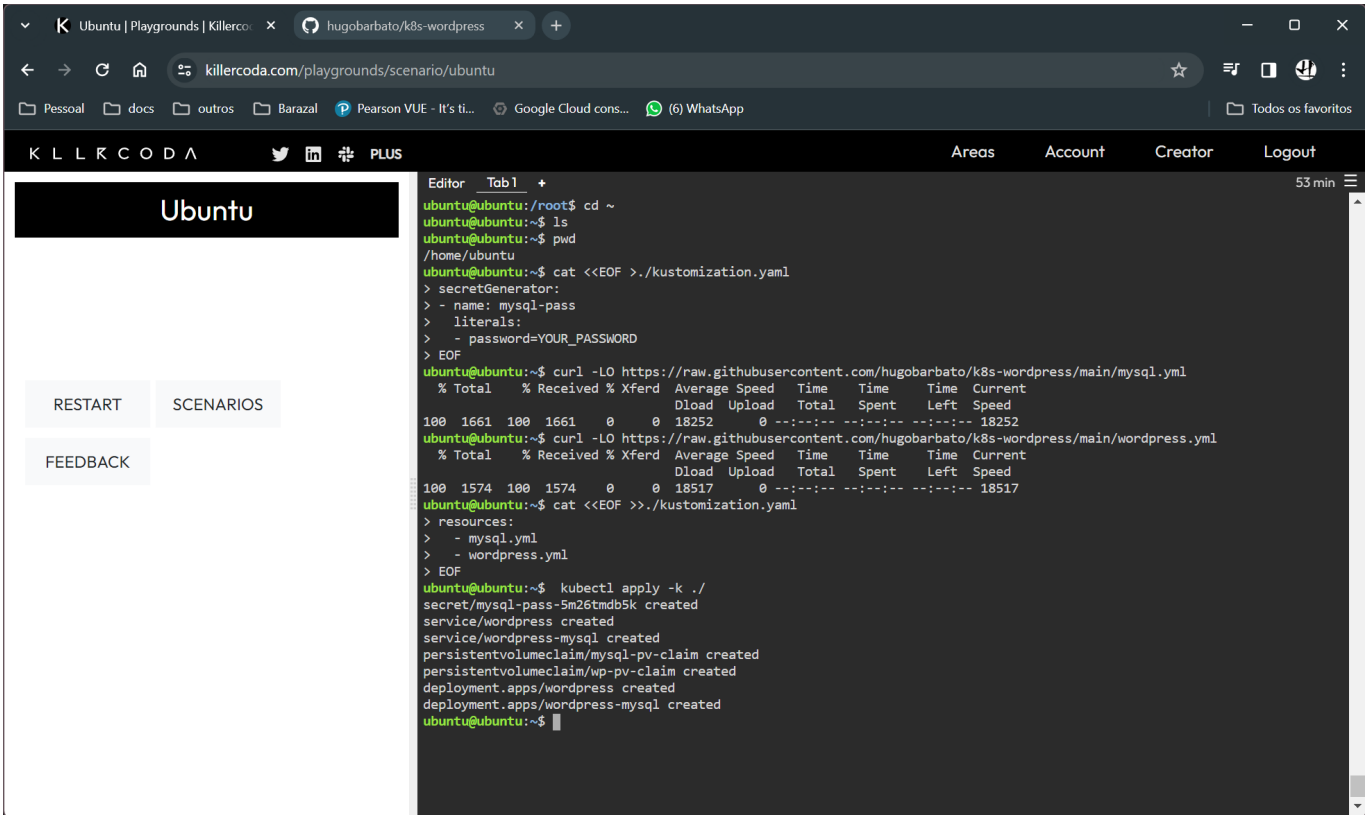
```
kubectl delete <POD>
```

5. Prints da Execução no KillerCoda

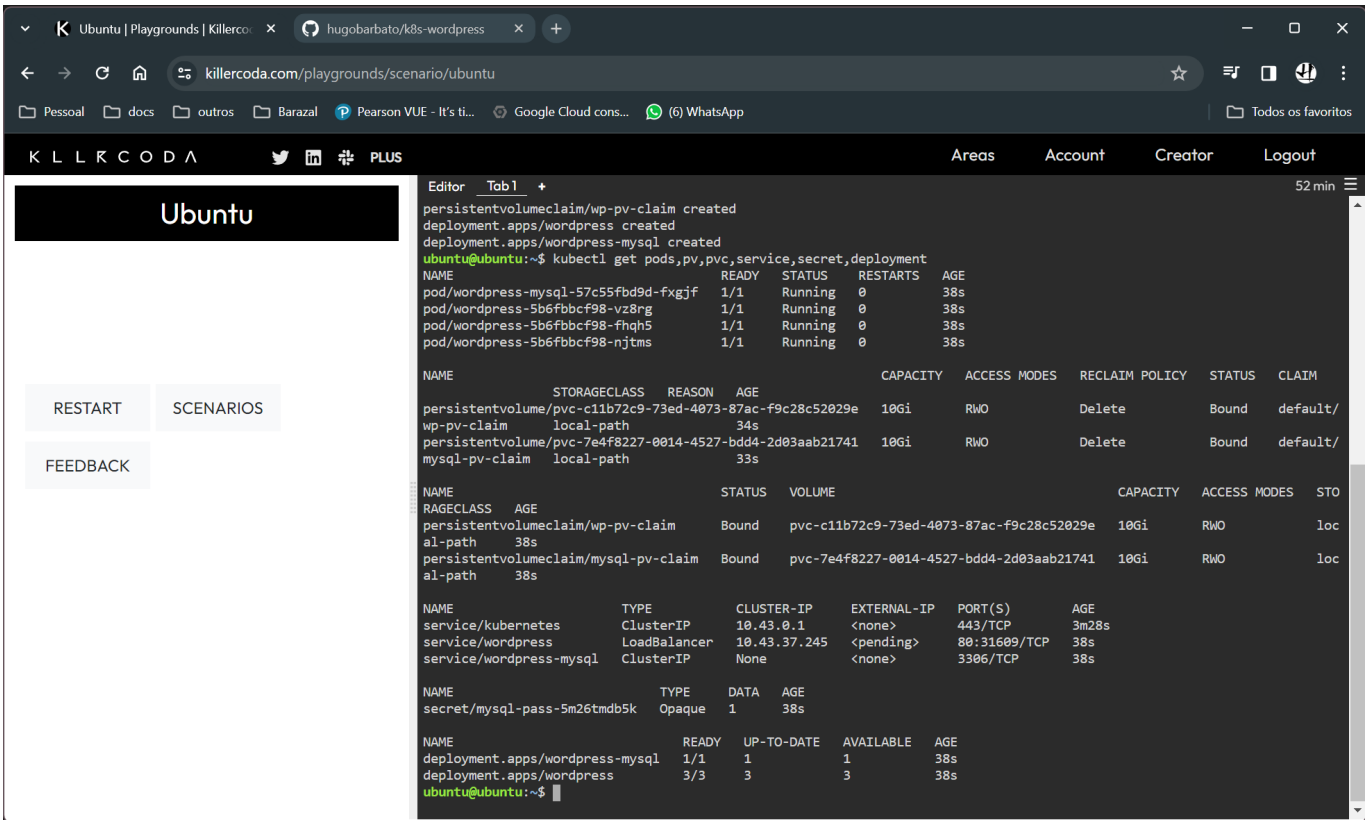
1º Configurando o K8S no Ubuntu:



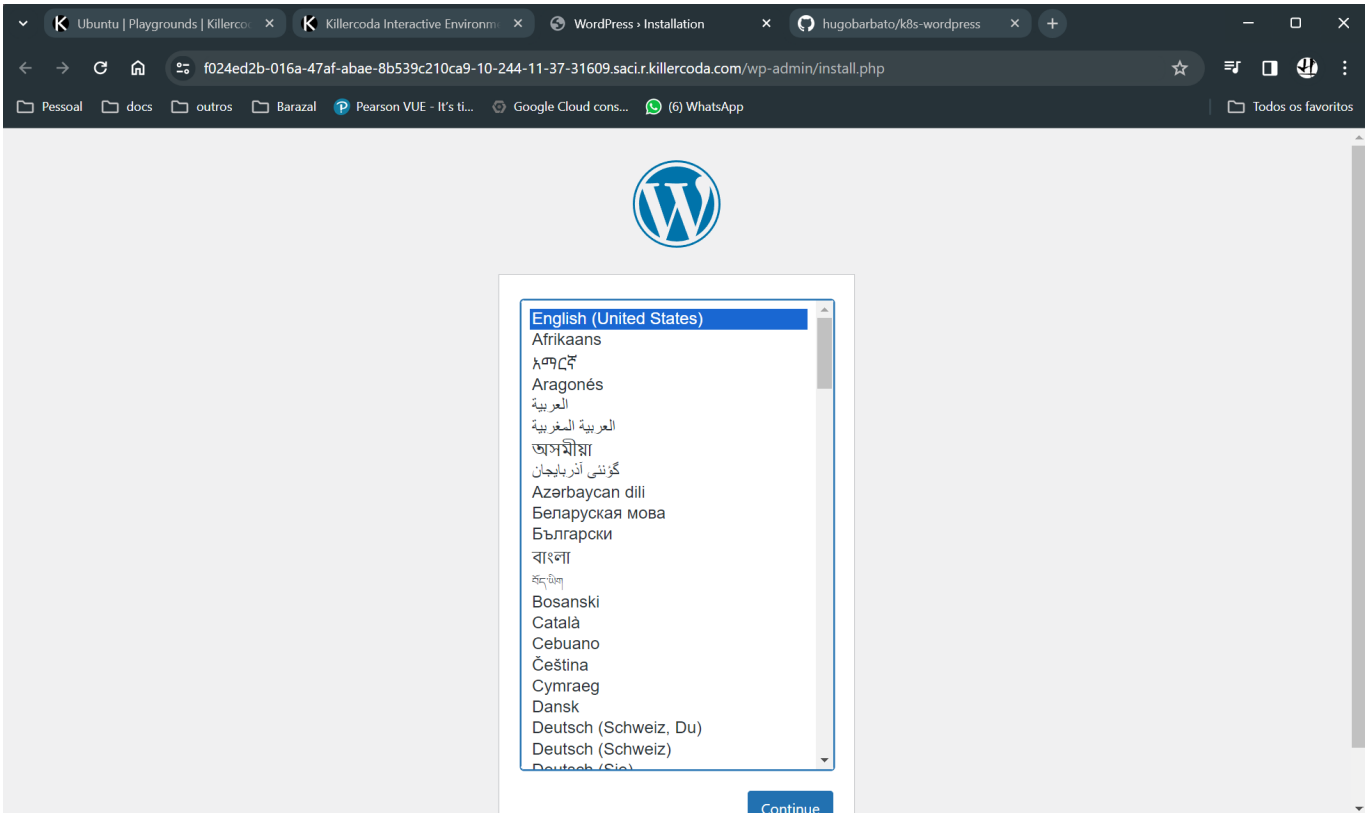
2º Configurando e subindo o Wordpress no K8S:



3º Serviços Wordpress e MySQL rodando no Kubernets:



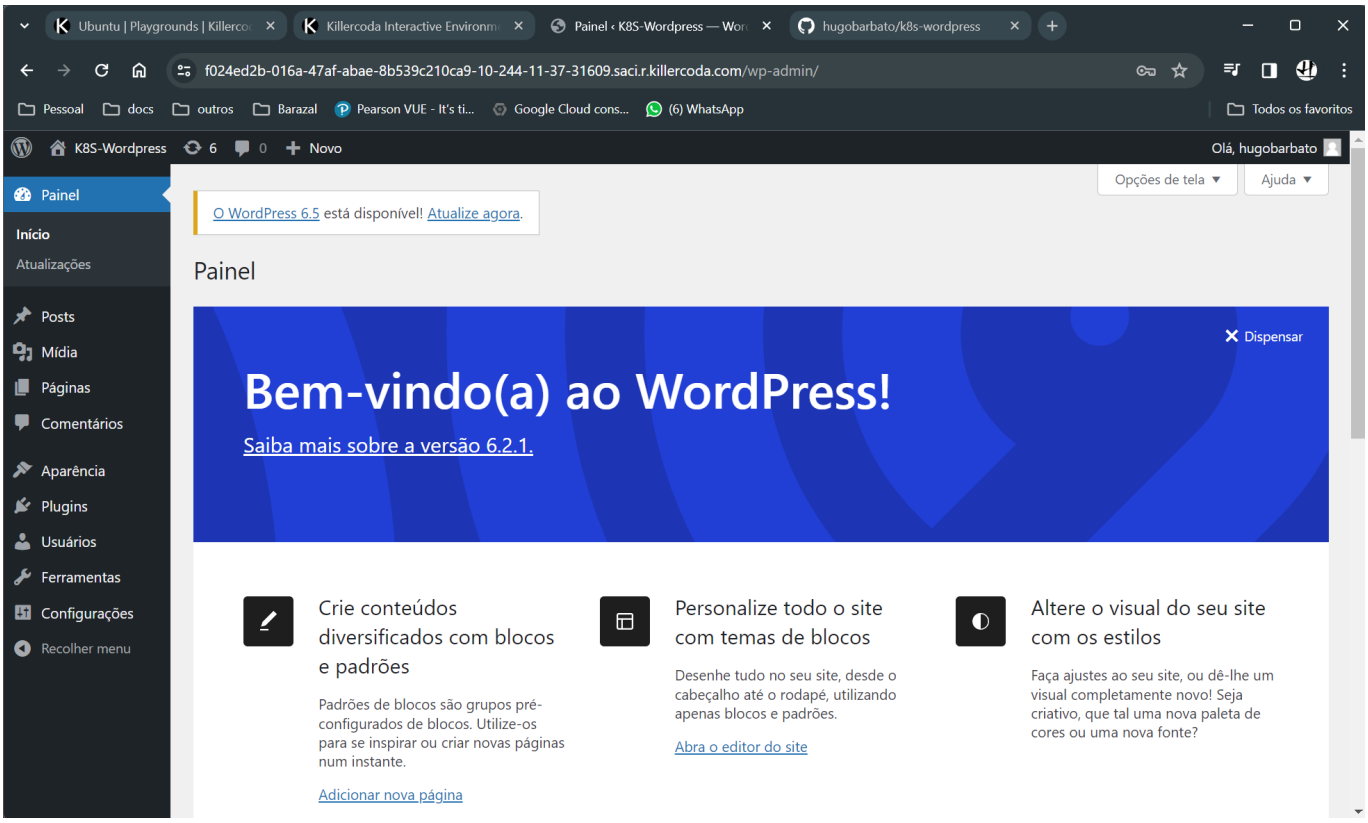
4º Testando o serviço Wordpress:



5º Executando o comando de delete do POD do MySQL:

Print do terminal com os comandos de deleção do pod do MySQL, posterior os serviços executando.

6º Painel Admin do Wordpress rodando apos delete do pod do MySQL:


Ubuntu | Playgrounds | Killercoda
Killercoda Interactive Environm
Painel ‹ K8S-Wordpress — Wor
hugobarbato/k8s-wordpress
f024ed2b-016a-47af-abae-8b539c210ca9-10-244-11-37-31609.saci.r.killercoda.com/wp-admin/
Pessoal
docs
outros
Barazal
Pearson VUE - It's ti...
Google Cloud cons...
(6) WhatsApp
Todos os favoritos
K8S-Wordpress
6
0
Novo
Olá, hugobarbato
Painel
Início
Atualizações
Posts
Mídia
Páginas
Comentários
Aparência
Plugins
Usuários
Ferramentas
Configurações
Recolher menu
Opções de tela
Ajuda
O WordPress 6.5 está disponível! Atualize agora.
Painel
Dispensar
Bem-vindo(a) ao WordPress!
Saiba mais sobre a versão 6.2.1.
Crie conteúdos diversificados com blocos e padrões
Padrões de blocos são grupos pré-configurados de blocos. Utilize-os para se inspirar ou criar novas páginas num instante.
Adicionar nova página
Personalize todo o site com temas de blocos
Desenhe tudo no seu site, desde o cabeçalho até o rodapé, utilizando apenas blocos e padrões.
Abra o editor do site
Altere o visual do seu site com os estilos
Faça ajustes ao seu site, ou dê-lhe um visual completamente novo! Seja criativo, que tal uma nova paleta de cores ou uma nova fonte?