experts club

# Crie um cluster Kubernetes com Raspberry Pi 4

24/01/2022

# Agenda

experts club

- Sobre mim e a minha relação com o código;
- Sobre a aula e o que será entregue no final;
- Requisitos, ambiente e recursos;
- Considerações sobre o Raspberry Pi 4;
- Instalação do Sistema Operacional e dos requerimentos;
- Configurações específicas;
- Acesso e utilização;

# Sobre mim e a minha relação com o código

experts club

- **Sergio Siqueira;**
- **Engenheiro Eletricista com ênfase em eletrônica;**
- **35 anos de experiência em TI, infraestrutura e hardware;**
- **Desenvolvimento de software como hobby e recentemente parte do trabalho;**
- **Head of devops e consultor em tecnologia;**
- **Redes sociais:**
  - https://app.rocketseat.com.br/me/sergio-siqueira-05693;
  - https://www.linkedin.com/in/snsergio/;
  - https://github.com/snsergio;

# Sobre a aula e o que será entregue no final

experts club

- **Breve descrição do Kubernetes e porque utilizar;**
- **Hardware Raspberry Pi e suas características;**
- **Diferenças entre o cluster em Raspberry Pi e servidores e PC x86/x64;**
- **Acesso e exemplo de utilização do cluster;**

  - https://github.com/rocketseat-experts-club/cluster-kubernetes-raspberry-pi-2022-01-28
  - Arquivos disponíveis no github:
    - Passo-a-passo
    - Links para os recursos utilizados

# Requisitos, ambiente e recursos

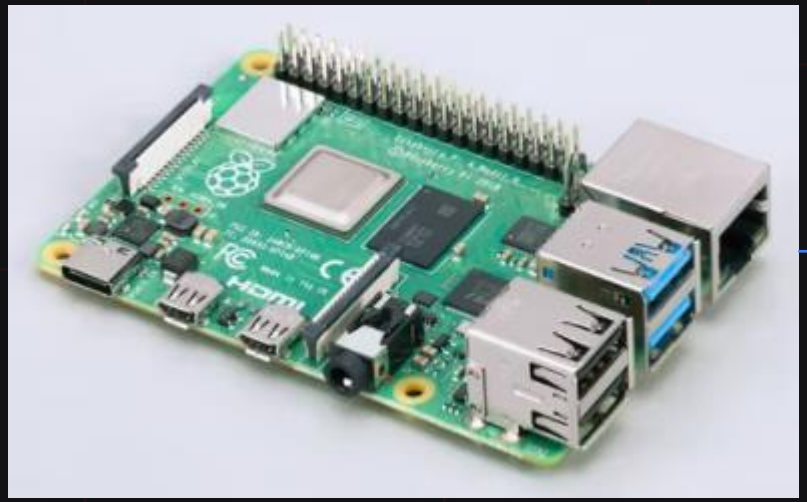
experts club

- **Requisitos para um melhor aproveitamento da aula:**
  - PC Windows 10 com as ferramentas:
    - Raspberry Pi Imager (https://www.raspberrypi.com/software/)
    - Conhecimento básico de Kubernetes
    - Conhecimento básico de Ubuntu Linux
- **Ambiente e recursos necessários:**
  - Raspberry Pi (https://www.raspberrypi.com/)
    - Fonte de alimentação
    - Cartão micro-SD para carregar o SO no Raspberry Pi
    - Conexão Ethernet por cabo (comentários sobre Wifi)
  - Aplicativo de Terminal para acesso remoto (cliente SSH):
    - PuTTY (https://www.putty.org/)
    - MobaXterm (https://mobaxterm.mobatek.net/)
  - Editor de textos

# Tópico relacionado ao conteúdo

experts club

## Raspberry Pi 4



Nó único

Vários nós

## Cluster Kubernetes

```
ubuntu@pimaster:~$ kubectl get nodes
NAME       STATUS   ROLES                  AGE   VERSION
pimaster   Ready    control-plane,master   16h   v1.23.3
ubuntu@pimaster:~$
```

```
ubuntu@pimaster:~$ kubectl get nodes
NAME       STATUS   ROLES                  AGE   VERSION
pimaster   Ready    control-plane,master   16d   v1.23.1
pinode1    Ready    <none>                 16d   v1.23.1
pinode2    Ready    <none>                 16d   v1.23.1
ubuntu@pimaster:~$
```

## Exemplo (Arquivo YAML)

```
 GNU nano 4.8                                  utils.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: utilities
  labels:
    app: utilities
spec:
  replicas: 1
  selector:
    matchLabels:
      app: utilities
  template:
    metadata:
      labels:
        app: utilities
    spec:
      containers:
      - name: utilities
        image: snsergio/monitor:util_armv2
        # Just spin & wait forever
        command: [ "/bin/bash", "-c", "--" ]
        args: [ "while true; do sleep 30; done;" ]
        resources:
          requests:
            cpu: 30m
            memory: 64Mi
          limits:
            cpu: 100m
```

experts club

# Obrigado!

**Sergio Siqueira**
sergio@tecnosiq.dev
https://app.rocketseat.com.br/me/sergio-siqueira-05693
https://www.linkedin.com/in/snsergio
https://github.com/snsergio

# Comandos para preparar o cluster

experts club

- **Editar o arquivo `cmdline.txt` no cartão microSD (system-boot)**
  Use um editor de texto para modificar o arquivo cmdline.txt do microSD
  Adicione um espaço em branco no final da linha existente e adicione o texto seguinte:
  ```
  cgroup_enable=cpuset cgroup_enable=memory cgroup_memory=1 ip=<IP do Raspberry Pi 4>::<IP do default gateway da sua rede>:<máscara da sua rede>:<hostname do Raspberry Pi>:<nome da interface de rede do Raspberry Pi>:off
  ```
  Exemplo:
  ```
  cgroup_enable=cpuset cgroup_enable=memory cgroup_memory=1 ip=192.168.0.50::192.168.0.1:255.255.255.0:pimaster:eth0:off
  ```

- **Após editar e salvar o arquivo, insira o microSD no Raspberry Pi, ligue o cabo de rede e a energia do Raspberry Pi**
- **Em alguns minutos, você deve conseguir conectar ao Raspberry Pi através de SSH:**
  ```
  ssh ubuntu@<endereço IP do seu Raspberry Pi>
  ```
  Exemplo:
  ```
  ssh ubuntu@192.168.0.50
  ```

- **Ativar a sincronização do Raspberry Pi com NTP e ajustar o TimeZone:**
  ```
  systemctl status systemd-timesyncd.service
  timedatectl status
  timedatectl list-timezones
  sudo timedatectl set-timezone "America/Sao_Paulo"
  ```
  Edite o arquivo timesyncd.conf do diretório /etc/systemd
  ```
  sudo nano /etc/systemd/timesyncd.conf
  ```
  Procure a linha #NTP= (se tiver o # no início da linha, retire), deve ficar assim:
  ```
  NTP=b.ntp.br
  ```
  Salve o arquivo e saia do editor (no caso do nano: <ctrl-x> y)
  ```
  timedatectl set-ntp on
  ```

# Comandos para preparar o cluster (continuação)

experts club

- **Editar o nome do host (Raspberry Pi):**

```
hostnamectl set-hostname <nome do host>
Exemplo:
hostnamectl set-hostname pimaster
```

- **Execute os comandos a seguir para atualizar o sistema operacional, limpar o iptables e rebotar o Raspberry Pi:**

```
sudo iptables -F
sudo su -
sudo apt update
sudo apt -y upgrade
sudo systemctl reboot
```

- **Instale os componentes do Kubernetes e ao final, verifique a versão:**

```
sudo apt -y install curl apt-transport-https
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt update
sudo apt -y install vim git curl wget kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
kubectl version --client && kubeadm version
```

- **Desative o swap de memória:**

```
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
sudo swapoff -a
sudo modprobe overlay
sudo modprobe br_netfilter
```

- **Ajuste a configuração de rede do Kubernetes:**

```
sudo tee /etc/sysctl.d/kubernetes.conf<<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```

# Comandos para preparar o cluster (continuação)

experts club

- **Aplique as modificações e faça a instalação do Docker e dos componentes necessários:**

```
sudo sysctl --system
sudo apt update
sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt update

curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

- **Faça as configuração do Docker:**

```
sudo sh -eux <<EOF
# Install newuidmap & newgidmap binaries
apt-get install -y uidmap
EOF

dockerd-rootless-setuptool.sh install
sudo mkdir -p /etc/systemd/system/docker.service.d

sudo tee /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF

sudo systemctl daemon-reload
sudo systemctl restart docker
sudo systemctl enable docker
```

- **Adicione o usuário ubuntu ao grupo Docker para não precisar executar com sudo:**

```
sudo usermod -aG docker ubuntu
logout
login
```

# Comandos para preparar o cluster (continuação)

experts club

- **Complete a configuração do Kubernetes executando os comandos:**

```
sudo kubeadm config images pull
sudo kubeadm config images pull --cri-socket /var/run/docker.sock
```

- **As etapas até aqui devem ser feitas em todos os nós do cluster Kubernetes, a partir daqui execute somente no nó principal (master):**
- **Inicialize o cluster Kubernetes:**

```
sudo kubeadm init --pod-network-cidr=10.10.0.0/16
```

- **Complete a configuração do nó master:**

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

- **Configure a rede do cluster Kubernetes**

```
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

- **Configure o nó master para atuar como nó de trabalho também:**

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

- **Caso queira adicionar novos nós de trabalho ao cluster, gere o token no nó master:**

```
kubeadm token create --print-join-command
```

E execute o comando no nó de trabalho:

```
kubeadm join <IP do nó master>:6443 --token <token> \
        --discovery-token-ca-cert-hash sha256:<discovery hash>
```

Exemplo:

```
kubeadm join 192.168.15.40:6443 --token xmsy45.qev6w0muhfdplnn8 \
        --discovery-token-ca-cert-hash sha256:8b4497363a2d248a65f97c69618e098d23d5f331a0592351f2312f5763f5699c
```

# Exemplo de utilização do cluster

experts club

- **Com o cluster criado e em funcionamento:**

```
kubectl get nodes
kubectl apply –f nginx.yaml
kubectl get pod
```

Verifique se a resposta é algo parecido (os números/letras finais podem variar):

```
NAME                         READY   STATUS    RESTARTS   AGE
app-nginx-5d7b66b8c8-6t2k8   1/1     Running   0          24m
```

É importante que o STATUS esteja em Running

- **Execute o comando abaixo para redirecionar a porta do Raspberry Pi para o POD do nginx:**

```
kubectl port-forward --address 0.0.0.0 svc/app-nginx-service 8088:80
```

port-forward é o commando do Kubernetes para apontar a porta do cluster para a porta do POD

--address 0.0.0.0 indica que será aceita chamadas de qualquer endereço IP

svc/app-nginx-service é o serviço (no YAML) que indica como o POD se comunica

você pode ver o nome do serviço com o comando kubectl get svc

8088 é a porta do Raspberry Pi que será encaminhada para o POD

80 é a porta do POD (do serviço) que receberá o conteúdo da porta do Raspberry Pi

- **Acesse com um browser o IP do Raspberry Pi na porta indicada acima (8088):**

```
http://192.168.0.50:8088
Você deve receber uma mensagem: 500 Internal Server Error / nginx/1.21.6
Porquê? (veja a resposta no vídeo da aula)
```

# Exemplo de utilização do cluster

experts club

- **Crie uma página web inicial:**
  `index.html`
  Copie esta página para o POD do Nginx:
  `kubectl cp index.html app-nginx-5d7b66b8c8-6t2k8:/usr/share/nginx/html/`
  - cp é o comando de cópia do kubectl
  - index.html é a página que acabou de ser criada
  - app-nginx-5d7b66b8c8-6t2k8 é o nome do POD do Nginx (deve ser diferente na sua instalação)
  - /usr/share/nginx/html/ é o diretório no POD onde deve estar a página index.html

- **Execute novamente o port-forward:**
  `kubectl port-forward --address 0.0.0.0 svc/app-nginx-service 8088:80`

- **Acesse novamente a página pelo browser (ou faça o recarregamento/reload da página):**
  `http://192.168.0.50:8088`
  `Agora você deve receber o conteúdo criado na página index.html ☺`