

Cluster Kubernetes Local

Requisitos:

- Mínimo de 8GB RAM (recomendável: 16GB)
- VS Code (não obrigatório)
- VirtualBox
- Kubectl
- Vagrant (o uso dele é para subir a infra)
- Lens (IDE para uma familiarização gráfica, não será abordado a fundo para melhor estudos indico o uso da documentação “docs.k8slens.dev/main/”)

Considerações:

O SO hospedeiro utilizado nesse tutorial é o Linux, porém pode ser utilizado qualquer outro de sua preferência. Por heterogeneidade, o Hypervisor utilizado será o VirtualBox.

Sinopse:

Neste laboratório abordaremos a criação de um cluster k8s local onde teremos 3 Ubuntu 18.04, uma control-plane e duas workers. Vamos implantar micro-serviços, explorar comandos e abordaremos o uso básico de funcionalidades que o Kubernetes tem a oferecer para melhor gerenciamento.

Inicialização do laboratório:

1. Presumindo que você já esteja com os programas que abordamos em **Requisitos** instalados, vamos abrir o vagrantfile no VSCode e realize as alterações dos IPs das máquinas para o ip da sua rede:

```
config.vm.define "vm2" do |vm2|  
  vm2.vm.network "public_network", ip: "192.168.10.127"
```

No meu caso o range de IP aqui é 192.168.10.0

2. Agora abra o **master.sh** localize e altere o IP do **--apiserver-advertise-address=** para o mesmo da vm “master” que será nossa control-plane:

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=192.168.10.126
```

3. Abra o próprio terminal do VSCode e execute o comando **=> vagrant up master** ele irá iniciar a vm master e dará a opção de interface de rede, no caso aqui usarei a WI-FI então escolherei a 1 como bridge

```
Bringing machine 'master' up with 'virtualbox' provider...  
==> master: Importing base box 'ubuntu/bionic64'...  
==> master: Matching MAC address for NAT networking...  
==> master: Checking if box 'ubuntu/bionic64' version '20220107.0.'  
to date...  
==> master: Setting the name of the VM: ProjetoK8S_master_16549015  
5  
==> master: Fixed port collision for 22 => 2222. Now on port 2200.  
==> master: Clearing any previously set network interfaces...  
==> master: Available bridged network interfaces:  
1) wlp0s20f3  
2) eno1  
3) docker0  
==> master: When choosing an interface, it is usually the one that  
==> master: being used to connect to the internet.  
==> master:  
master: Which interface should the network bridge to? 1
```

Nota: Se existir apenas uma interface de rede essa etapa será avançada, porque não existe outras para escolher.

4. Ele vai começar a instalar as dependências de tudo que precisamos para criar nosso cluster Kubernetes assim que ele finalizar vamos voltar ao vagrantfile para realizarmos alterações de memória das VM's workers, devemos comentar onde está declarado na vm master as configurações de memória e CPU:

```
# MASTER será a Control-plane
config.vm.define "master" do |master|
#   config.vm.provider "virtualbox" do |v|
#       v.memory = 2048
#       v.cpus = 2
#   end
end
master.vm.hostname = "master"
```

5. Salve e no terminal do VSCode execute o comando => `vagrant up worker1 worker2`

```
er2
Bringing machine 'worker1' up with 'virtualbox' provider...
Bringing machine 'worker2' up with 'virtualbox' provider...
==> worker1: Importing base box 'ubuntu/bionic64'...
1
```

Siga o mesmo processo de escolha da interface que fizemos na master.

Nota:(se sua máquina for 8GB de RAM execute apenas uma `vagrant up worker1`).

6. Assim que instalarem acesse as workers com o comando => `vagrant ssh worker1`

```
~/Documentos/ProjetoK8S$ vagrant ssh worker1
15 (CPU) (linux 4.15.0-166-generic) x86_64)
```

(para a worker2 é só substituir o nome)

7. Agora vá até o diretório `cd /vagrant` verifique se existe um executável chamado `join.sh`

```
vagrant@worker1:~$ cd /vagrant
vagrant@worker1:/vagrant$ ls
Documentação.odt  join.sh  master.sh  ubuntu-bionic-18.04-cloudimg-console.log  vagrantfile  worker.sh
```

Execute o `join.sh` ele retornará a imagem:

```
vagrant@worker1:/vagrant$ sudo sh join.sh
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Nota: repita o mesmo processo 6 e 7 para a worker2.

8. Para sair da worker é só executar `exit`

```
vagrant@worker1:/vagrant$ exit
logout
Connection to 127.0.0.1 closed.
```


Comemore você configurou seu cluster Kubernetes!!!

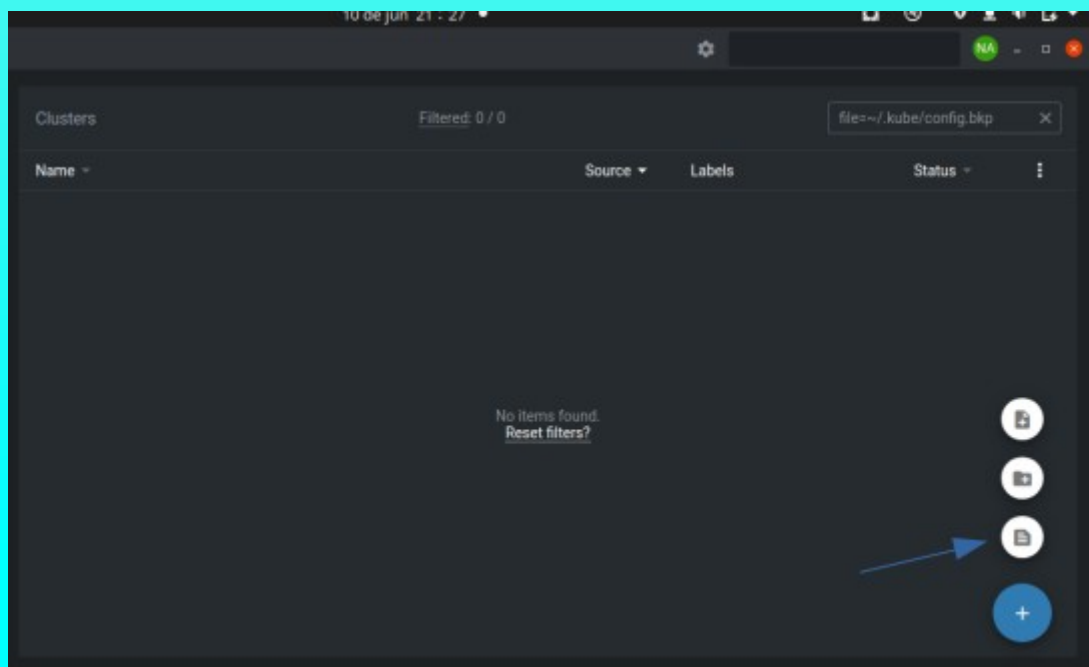
Nota: Para este passo 9 aqui vou usar uma VM ubuntu-desktop para simular de como você usaria para conectar em um cluster remoto, Claro que se você fosse utilizar em Cloud terá configurações diferentes da que estamos fazendo aqui. E se você estiver no Lens em sua própria maquina precisa do kubectl instalado.

9. Acesse a VM master com o comando do passo 6, vamos pegar o arquivo de configuração para executar o kubectl na IDE LENS, execute “cd .kube” e abra arquivo “config”. Com o mouse selecione tudo o que ele exibiu como na imagem abaixo.

```
vagrant@master:~$ cd .kube
vagrant@master:~/kube$ cat config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUMvakNDQWVhZG
    F3SUJBZ0lCQURBTklna3Foa2lH0XcwQkFRc0ZBREWFwTVJNd0VRWURWUWFERXdwcmRXSmwKY201bGRHVnpNQjRY
    RFRJeU1EwXHNREL6TVRBek9Wb1hEVE15TURZd056S3pNVEF6TlZvd0ZURVRNc0kVHQTFRVQpBeE1LYTNWVpYSr
    VawFJ3Y3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVB0R0tRUE4R0ExVWRFd0VCCi93UUZNU1Q
    NEdLUXFJQUdteFZpYlB1RndmRS9oTmPyK3JUTk1ZZXBxL3J1QUF1bWtDRmJwZnR5bnZVoV2sKbDl4ODJiIbTg1S
    9qYUdLZDBJdnk0Y2V4clEwYjNncDBzTGl2SFN0UTE5YlVJZXBKMEZiSVVKcU5IUmF0ZEt2awo4M21xdXE1NXBp
    bmFNT091VURrbUkwTDFsS0J6NEF6UWhBZHE4Y3l2VFRJbWwNa3VJRWxH0WJERmtXTWxmcnNlCnFnVzdISkdCUH
    Z0aG1ia05Za1E4M0tnSudvbkpZU2w5b1B0R1A0YW1XMlgrUEptRz14QVQxdVJvYw0ajl2dUQKSjRTR2UydVp
    VTcxbThHdGk5cFpJUVNVmW1iVW9kwUfKS3BqN1BVC1lwVjZ3TXU3dktIVkMvRzZxMzFLQXJaMQpMSkI3YnB6L
    YrTGdQWVXU1ZjQ0F3RUFBYU5aTUZjd0RnWURWUjBQOVFILOJBUURBZ0trTUE4R0ExVWRFd0VCCi93UUZNU1Q
    QWY4d0hRURWUjBQOVFILOJBUURBZ0trTUE4R0ExVWRFd0VCCi93UUZNU1Q
    FZbVZ5Ym1wMfPjYTXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVB0R0tRUE4R0ExVWRFd0VCCi93UUZNU1Q
    MwZyazZPYks2Q0x2Wk9VUEM3QWZVY3Ntr3l6SmtPVGJYKzYrY0xCajNYOUJmY2svWmNGWC9VnKZrClJlY2RTMT
    VDbGFKMUp5eG5MSU1TVGRXcnExamdZTXNNR0ZlTkpYzV5RUR6ekNTM1pkS1cySwV6eEF5NVZaeVYKb3VEbFMy
    a2JaSlzSlZaeUI4WWhjQkNnYw1UMkQ0Y2lhwVmg4SzdqbGRtaGJkYWpDTkttSEd3dzVEbHkvWmZmQWp0aCtNND
    QvWktlbG1mNlV5NmVGSUpwZTQwcGwxR24zcnJ0R0dBRTYxRzRoSmZ0QXl5M0FGKzVEc1dXZVpLV2UzCkMzaUk3
    RUdVdUplRVVoL3ZTYktkNy8vb0l5azNFdERPZkgvOFdGalVJUmhoWEEzTzdXZl3hPY1U4NHJlMmwrR08KaTk0P
    otLS0tLUVORCBDRVJUSUZJQ0FURSB0tLS0tCg==
    server: https://192.168.10.127:6443
  name: kubernetes
contexts:
- context:
```

Nota: a imagem está cortada mas você deve selecionar tudo que ele está exibindo.


10. Após você ter copiado o abra seu LENS passe a seta no 
Selecione o file e cole dentro da caixa o arquivo copiado igual na imagem abaixo



Clusters added here are **not** merged into the `~/.kube/config` file. [Read more about adding clusters.](#)

```
1  apiVersion: v1
2  clusters:
3  - cluster:
4      certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1J
5      server: https://192.168.10.127:6443
6      name: kubernetes
7  contexts:
8  - context:
9      cluster: kubernetes
10     user: kubernetes-admin
11     name: kubernetes-admin@kubernetes
12  current-context: kubernetes-admin@kubernetes
13  kind: Config
14  preferences: {}
15  users:
16  - name: kubernetes-admin
17     user:
18         client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1J
19         client-key-data: LS0tLS1CRUdJTiBSU0EgUUFJJVkFURSBLRVktLS0tLQpNSU1F
```

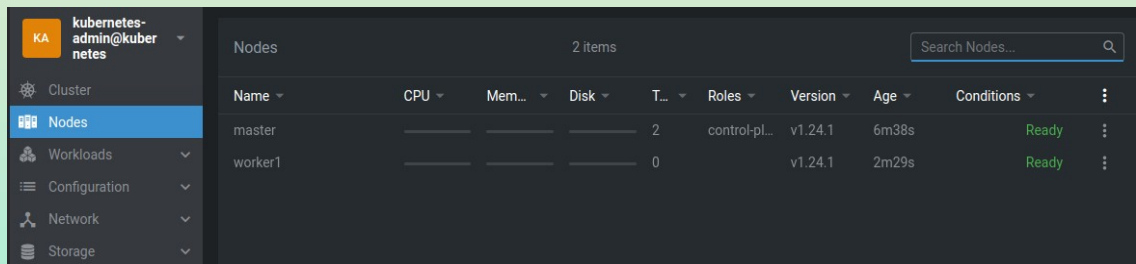
Role para baixo e clique em “add cluster”, você vai ver que ele foi adicionado

Clusters		1 item	Search...		
Name ▾	Source ▾	Labels	Status ▾		
 kubernetes-admin@kubernetes	local		connected		

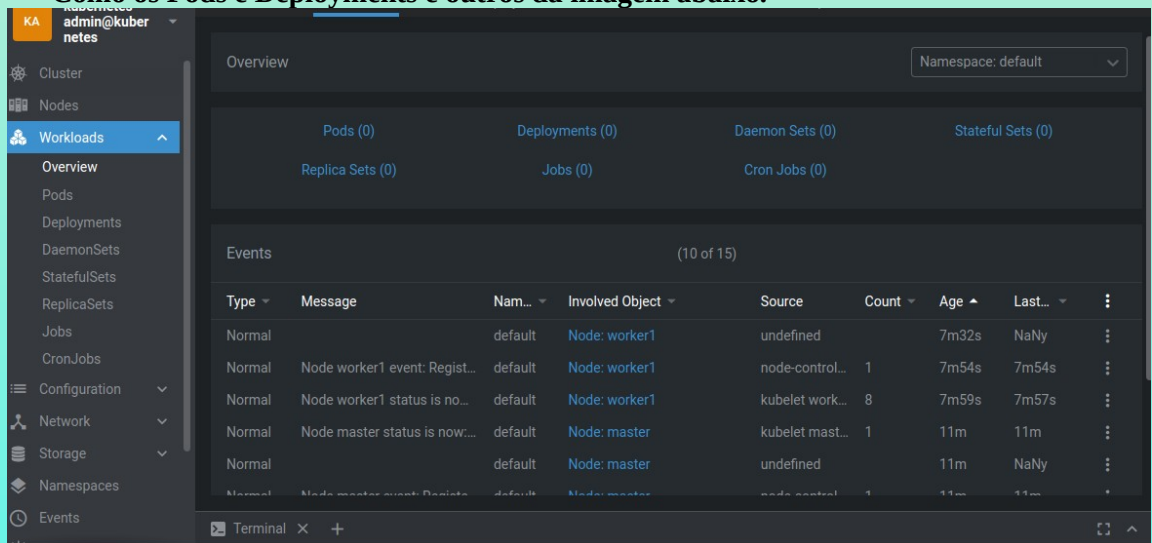
Clique nele para conectar ao cluster.

Agora vamos brincar um pouco!!!

11. Ao iniciar conectar no cluster com o Lens você irá ver que temos diversos temas a abordar, mas aqui vamos aprender o básico como havia explicado no início, clique em “Nodes” teremos listado a nossas vms que estão no cluster.

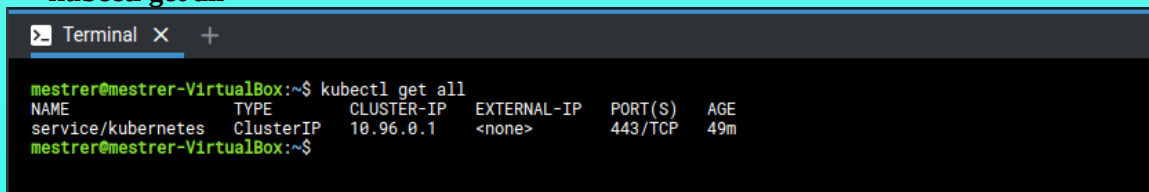


12. Em Workloads podemos ver graficamente o que usaremos no dia-a-dia para gerenciar Como os Pods e Deployments e outros da imagem abaixo.



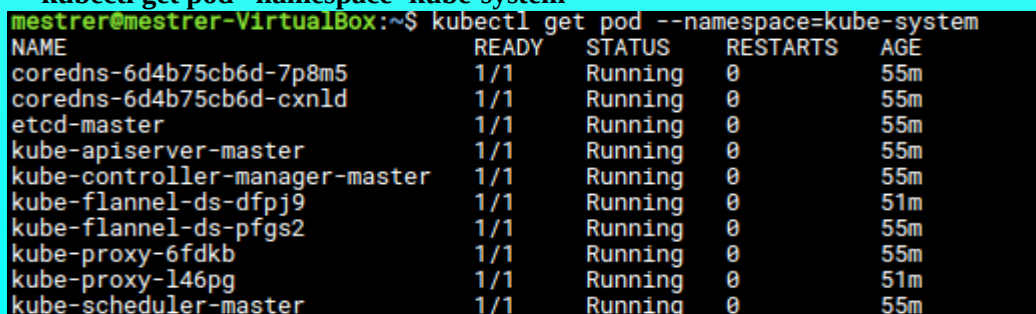
Note que no inferior da tela do Lens temos um “Terminal” vamos abrir e vamos executar alguns comandos.

13. Execute os seguintes comandos;
kubectl get all



Aqui visualizamos tudo que temos no namespace default como não criamos nenhum outro então aparecerá somente isso.

kubectl get pod --namespace=kube-system



Aqui são os pods que fazem o seu cluster kubernetes funcionar, esta tudo OK!!

**TO BE
CONTINUED...→**