Cluster Kubernetes Local

Sinopse:

Neste capitulo vamos dar continuidade da primeira aula que utilizávamos o terminal dentro do Lens, e faremos algumas configurações para facilitar nossos estudos.

1. **Seguindo da etapa anterior vamos realizar o deploy de um manifesto, com o arquivo aberto de vamos dar uma olhada**

```yaml
# Aplicativo de monitoramento de rastreamento veiculos
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tracking-car
spec:
  replicas: 3
  selector:
   matchLabels:
    app: tracking-car
  template:
    metadata:
      labels:
        app: tracking-car
    spec:
      containers:
      - name: tracking-car
        image: traccar/traccar:latest
        ports:
        - containerPort: 8082
```

Temos aqui um Deployment de uma aplicação WEB, como declaramos em replicas ele irá conter 3 Pods rodando se uma falhar automaticamente irá subir um novo Pod por conta da declaração que fizemos matchLabels ele vai validar se existe 3 pods com a labels tracking-car, que declarei dentro de template. Seguindo temos o nome do container que estamos atribuindo e a imagem da aplicação que no caso ele ira buscar no docker hub, e por fim declaramos a porta 8082.
Cada metada e o nome dentro deles vamos abordar mais adiante, agora veremos um SVC/ service NodePort.

```yaml
---
# NodePort para o tracking-car
apiVersion: v1
kind: Service
metadata:
  name: tracking-port
spec:
  type: NodePort
  selector:
    app: tracking-car
  ports:
    - port: 80
#---
```

Temos aqui o service NodePort ele vai nos ajudar a realizar o acesso entre os Pods para conexões externas podendo ser de um microsserviço diferente ou um acesso ao mundo externo.

Nota: dentro do manifesto temos um outro microsserviço comentado para ele segue a mesma premissa da aplicação acima para fazer o deploy é só descomentar.

2. **Vamos ao Deploy, no terminal do Lens vá até o diretorio que está o ym de manifesto e digite o seguinte comando=> kubectl apply -f manifesto.yml.**

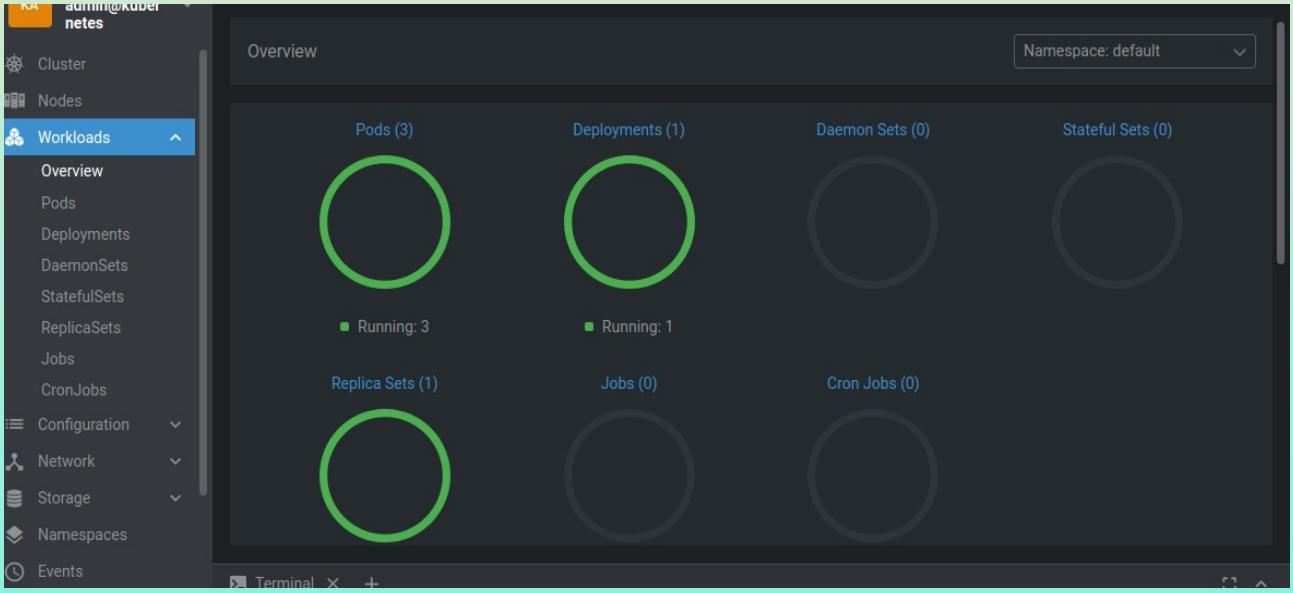
Verifique se tudo ocorreu certo você irá ver essa imagem que aparece acima.

3. **Agora vamos ver o que temos em nosso ambiente pelo terminal digitando o comando a seguir, "kubectl get all"**

```
mestrer@mestrer-VirtualBox:~/amb$ kubectl get all
NAME                                READY   STATUS    RESTARTS        AGE
pod/tracking-car-6848f66cd5-br6m6   1/1     Running   0               8m56s
pod/tracking-car-6848f66cd5-ntnqk   1/1     Running   0               8m56s
pod/tracking-car-6848f66cd5-wcnrf   1/1     Running   1 (2m51s ago)   8m56s

NAME                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/kubernetes      ClusterIP   10.96.0.1       <none>        443/TCP        5h38m
service/tracking-port   NodePort    10.110.156.88   <none>        80:30179/TCP   8m56s

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/tracking-car   3/3     3            3           8m56s

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/tracking-car-6848f66cd5   3         3         3       8m56s
```

Note que agora temos muito mais coisas em nosso ambiente, como não criamos nenhum outro namespace ele criou no Default, mas não se preocupe criaremos mais pra frente. Agora vamos falar daqueles metadatas então cada um tinha um name que nesse caso aqui é de cada um desses recursos que você esta vendo, o metadata de Deployment tem um nome de tracking-car que atribuimos no yaml pode-se ver na imagem que ele tem o nome que colocamos e assim segue o nome dos Pods e Service.

4. **Vamos dar uma olhada com o Lens no nosso cluster**



Repare que temos todos eles rodando no namespace Default no canto superior da imagem veja o que temos um dropdown que você pode selecionar o namespace no caso como não criamos nenhum você vai ver que só a os que o kubernetes usa para gerenciar o cluster e o default

5. **Neste momento faremos uma configuração essencial para o passo 6 que você consiga visualizar suas aplicações WEBs com o Lens siga as instruções que é"easy-peasy"e facilitará para futuros estudos.**

    - **Aqui vou usar o VS-Code porque vamos editar um aquivo e já usar o terminal, abra a pasta onde esta o seu vagratfile e abra o terminal nele execute o seguinte comando, "vagrant ssh-config master worker1 >sshconfig.conf", visualize o arquivo criado.**

sshconfig.conf

```
Host master
  HostName 127.0.0.1
  User vagrant
  Port 2200
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile /home/nata.martini/Documentos/ProjetoK8S/.vagrant/machines/master/virtual
  IdentitiesOnly yes
  LogLevel FATAL

Host worker1
  HostName 127.0.0.1
  User vagrant
  Port 2201
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile /home/nata.martini/Documentos/ProjetoK8S/.vagrant/machines/worker1/virtua
  IdentitiesOnly yes
  LogLevel FATAL
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL

    - **Modifique para os IP's das respectivas maquinas e a porta para a 22, após isso**

execute o seguinte comando “sudo cp sshconfig.conf /etc/ssh/ssh_config.d/sshconfig.conf”

- **A partir de agora quando acessarmos usaremos o comando ssh “nome-da-vm”.**

```
/Documentos/ProjetoK8S$ ssh master
```

acesse as VM’s para configurarmos a interface de rede e desabilitarmos as interfaces NAT no virtualbox.

```
vagrant@master:~$ sudo nano /etc/netplan/50-vagrant.yaml
```

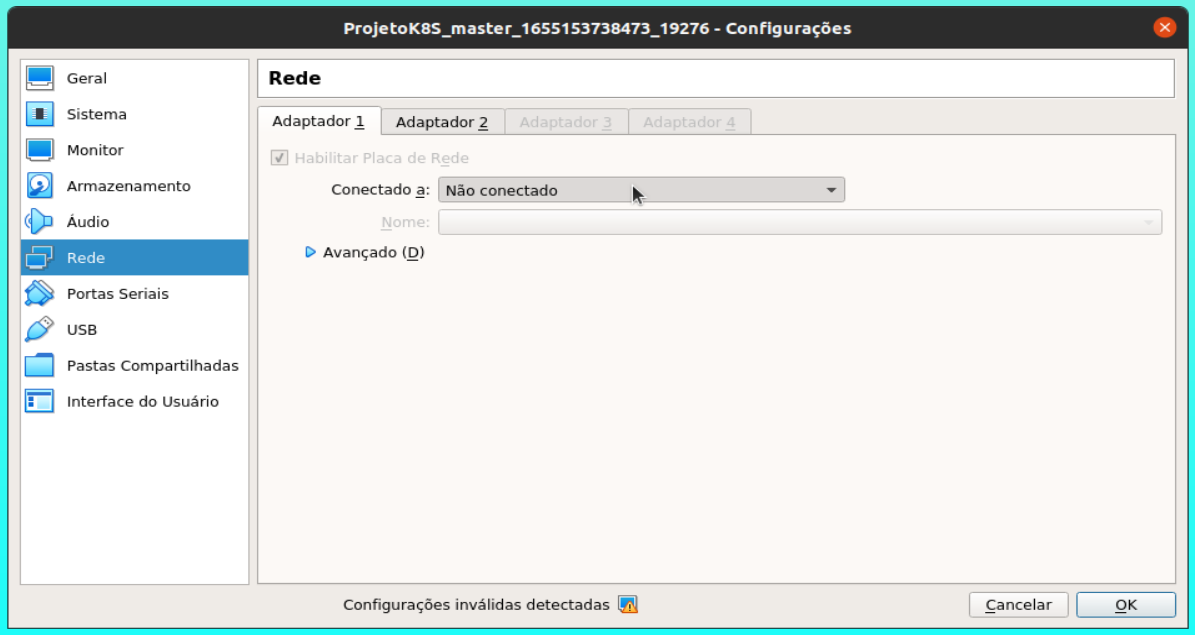
Acrescente no arquivo;

```
gateway4: SeuGateway
nameservers:
   addresses: [8.8.8.8,8.8.4.4]
```

No caso das nossa master ficou assim;

```
---
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s8:
      addresses:
      - 192.168.10.127/24
      gateway4: 192.168.10.1
      nameservers:
        addresses: [8.8.8.8,8.8.4.4]
```

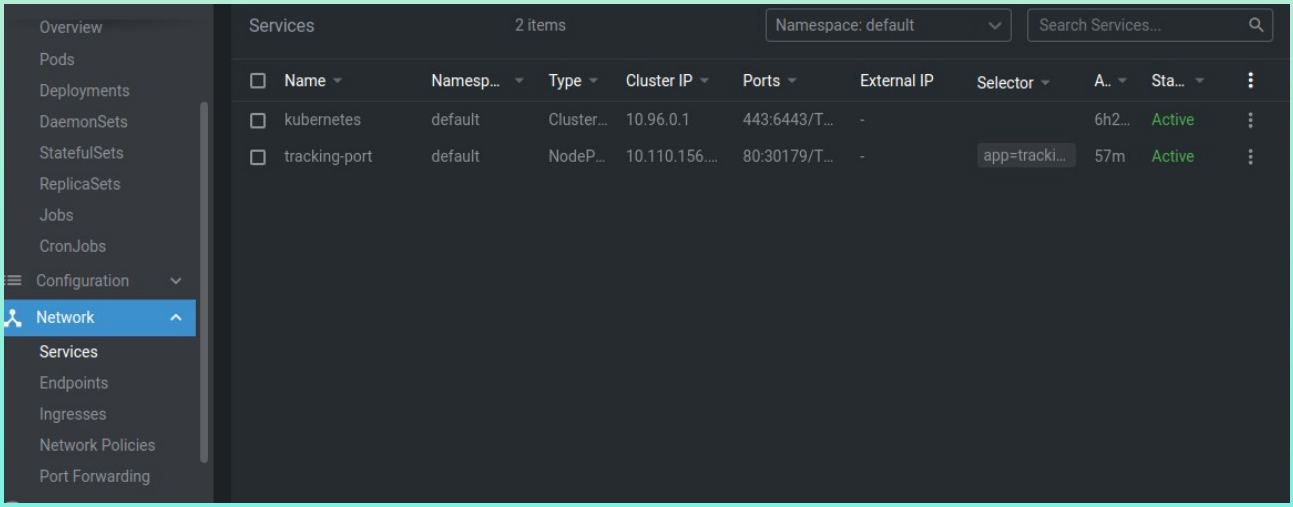
- **Salve o arquivo e execute o comando “sudo netplan apply”, e só acessar o virtualbox desativar a interface NAT. E de volta a VM execute “sudo systemctl restart kubelet”.**



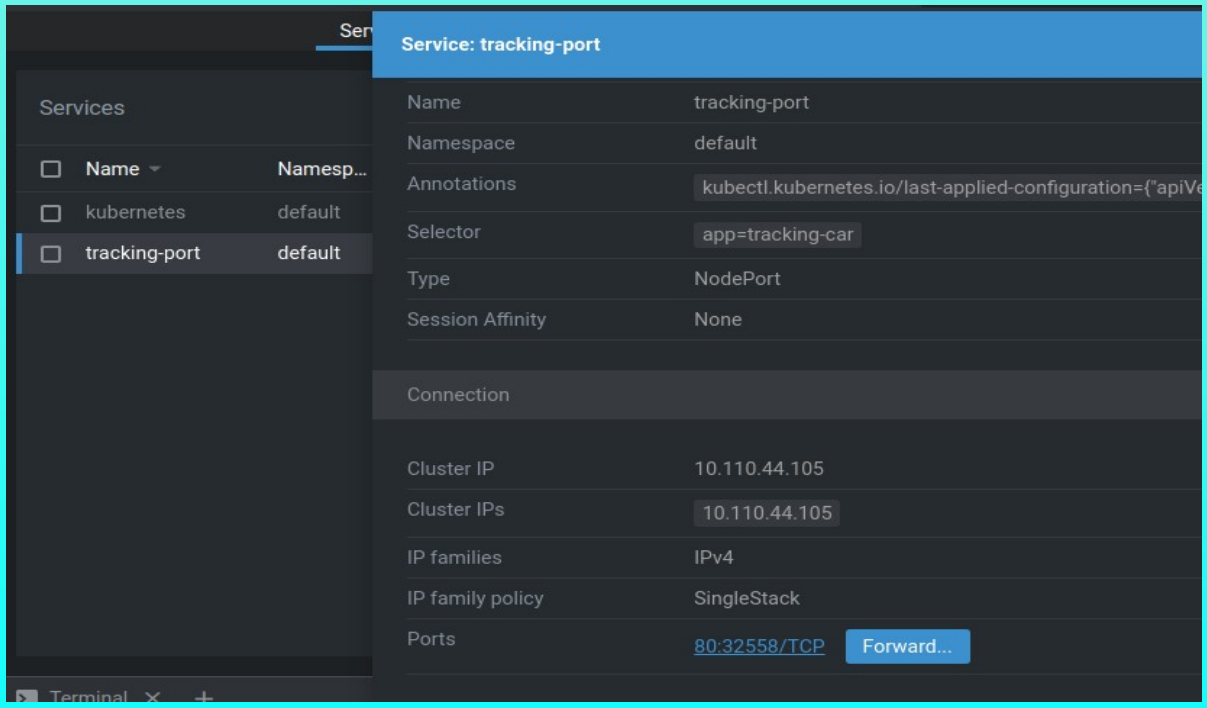
Nota: Agora as maquina nós ligaremos manualmente e não com o comando “vagrant up”

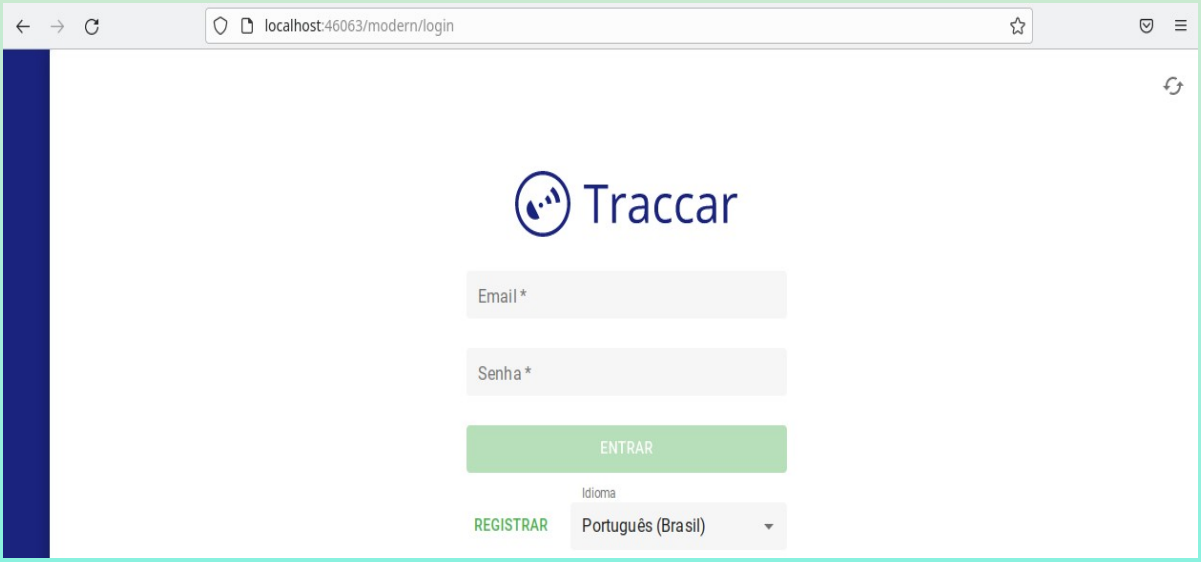
Depois dessa trama voltamos ao Lens, resumo rapido do “por quê?” desta configuração é que usando o vagrant no Virtuabox as VMs ficam com o IP 10.0.2.15 e isso nos da um problema para o passo 6. Dica de uma olhada nos Pods do namespace kube-system eles estão com o IP 10.0.2.15 exceto Pods de DNS, após realizarmos essas configurações irá funcionar.

6. **Agora navegue até o dropdown network vamos ver o nosso service NodePort para ver a aplicação WEB no navegador**

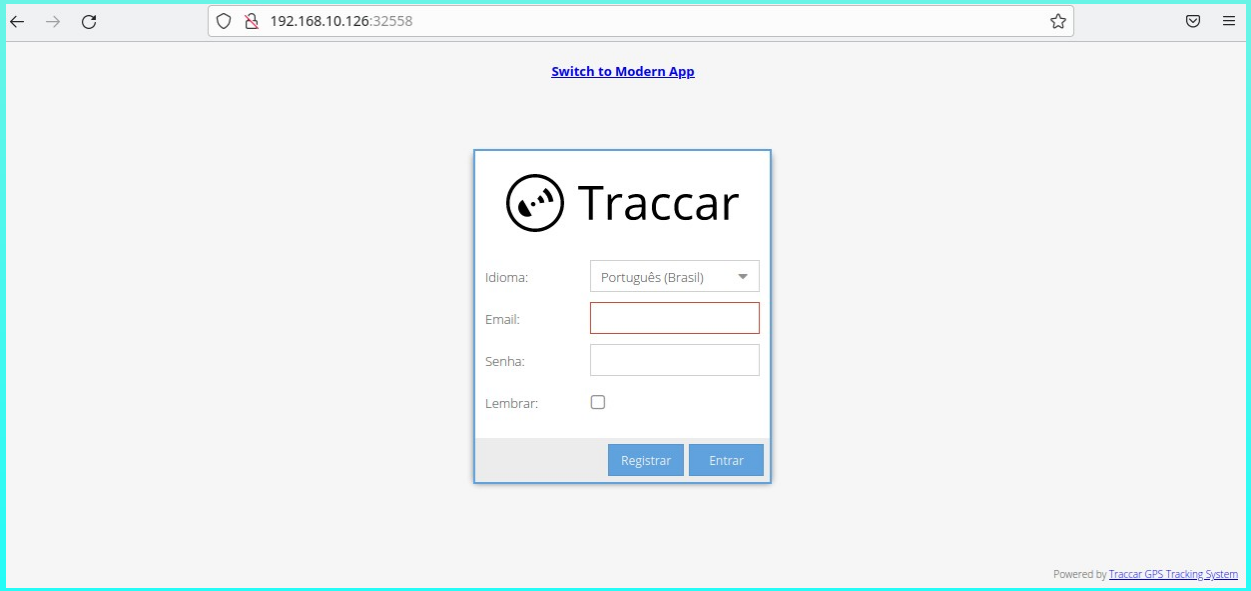


Clique no traking-port e vai expandir uma aba da direita, nela role para baixo até ports depois clique em forward e quando abrir a janela aperte o start, você sera redirecionado ao browser.





7. **Da mesma forma que verificamos no lens como ver as aplicações WEB, como estamos com o cluster local o NodePort publicou uma porta aleatória que no caso foi a 32558 como podemos ver na imagem se você quiser verificar pelo terminal execute “kubectl get svc” com o IP da VM worker1 que no caso aqui é 192.168.10.126. Vá para browser digite o ip com a porta 32558.**



8. **Vamos colocar recriar agora esse sistema em um namespace e vamos conecta-lo a um banco de dados MYSQL.**