

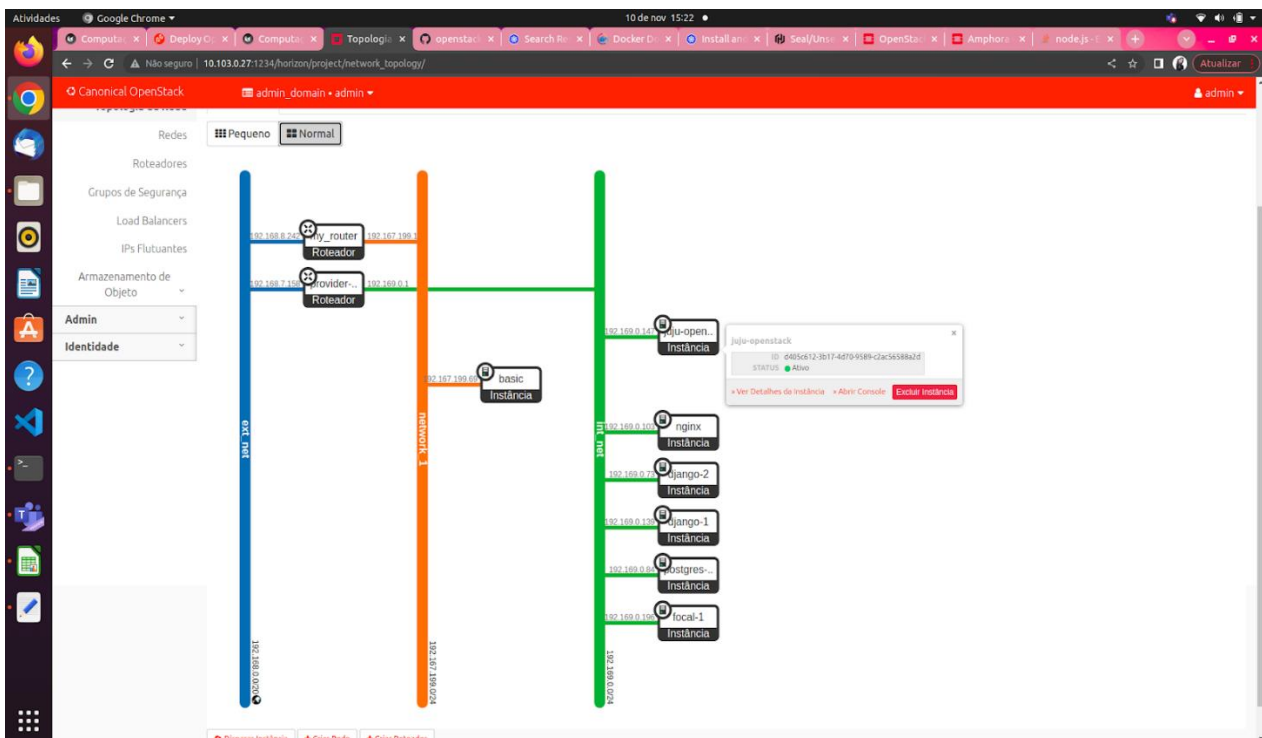
ROTEIRO 5

Alunos: Bernardo Cunha Capoferri e Livia Sayuri Makuta.

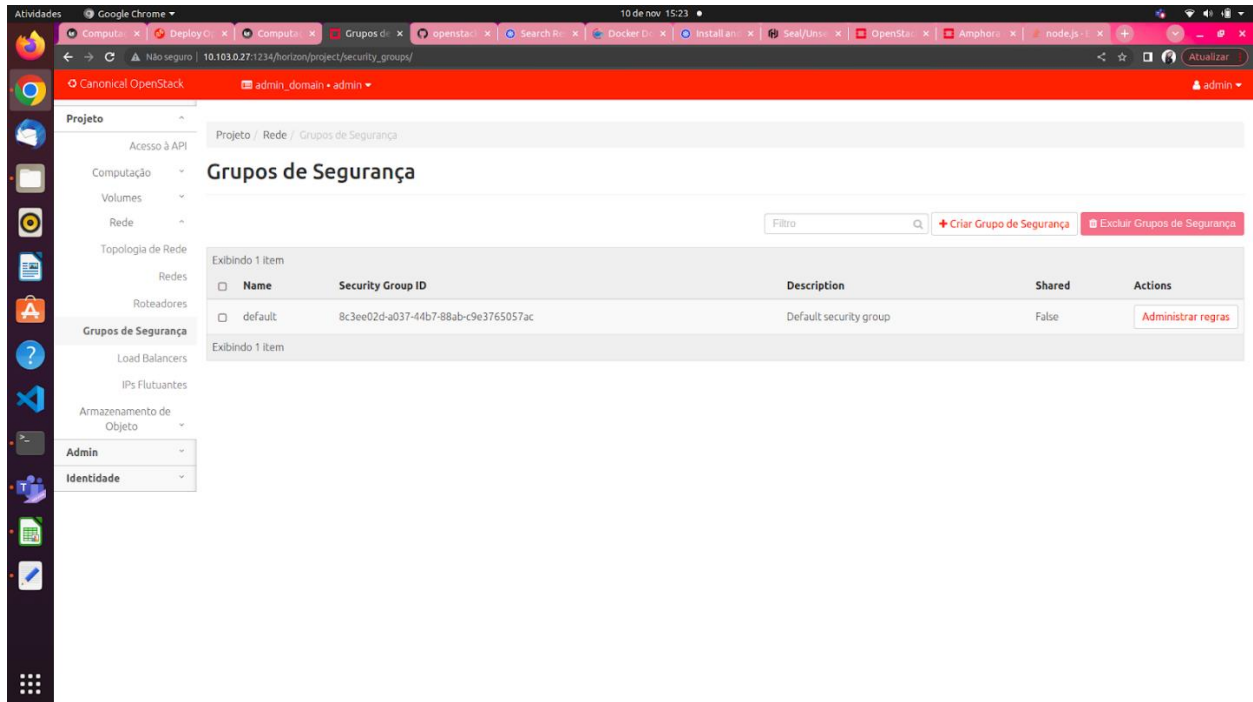
QUESTÕES

CHECKPOINT-1 WEBSERVER

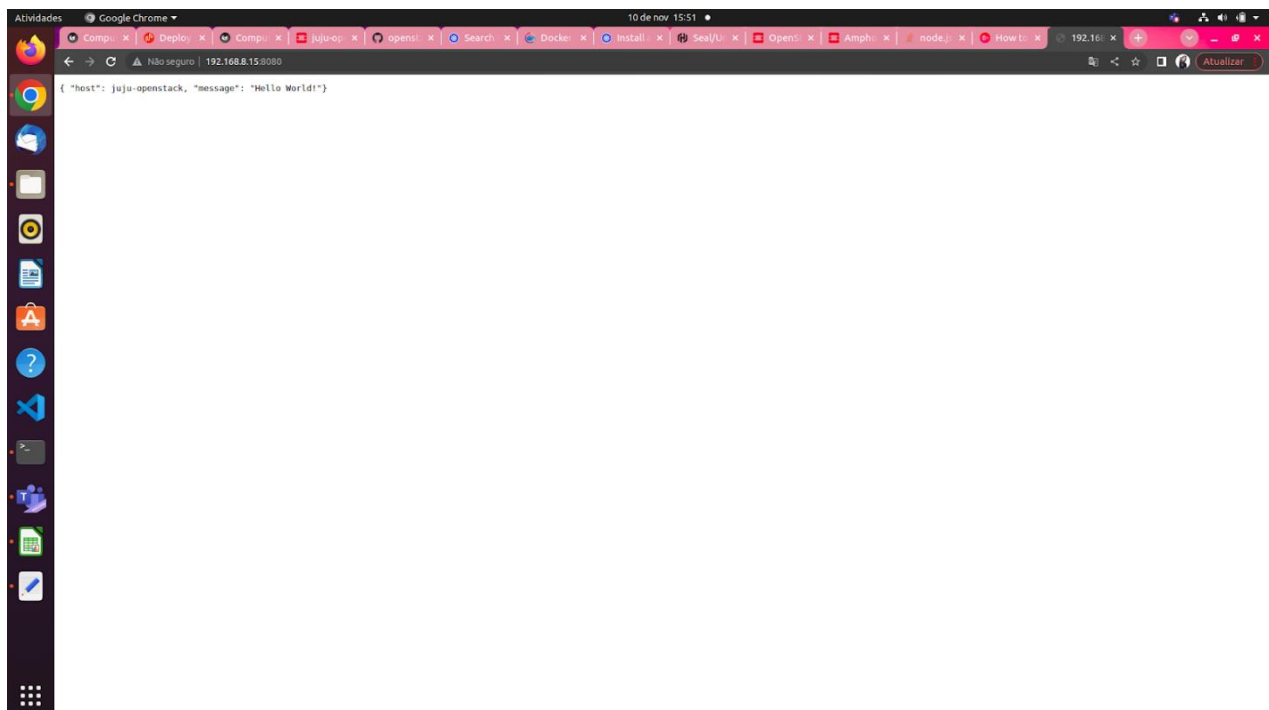
1. De um print das Telas abaixo:
2. Da aba *network topology* no OpenStack.



3. Das regras do Security Group utilizado

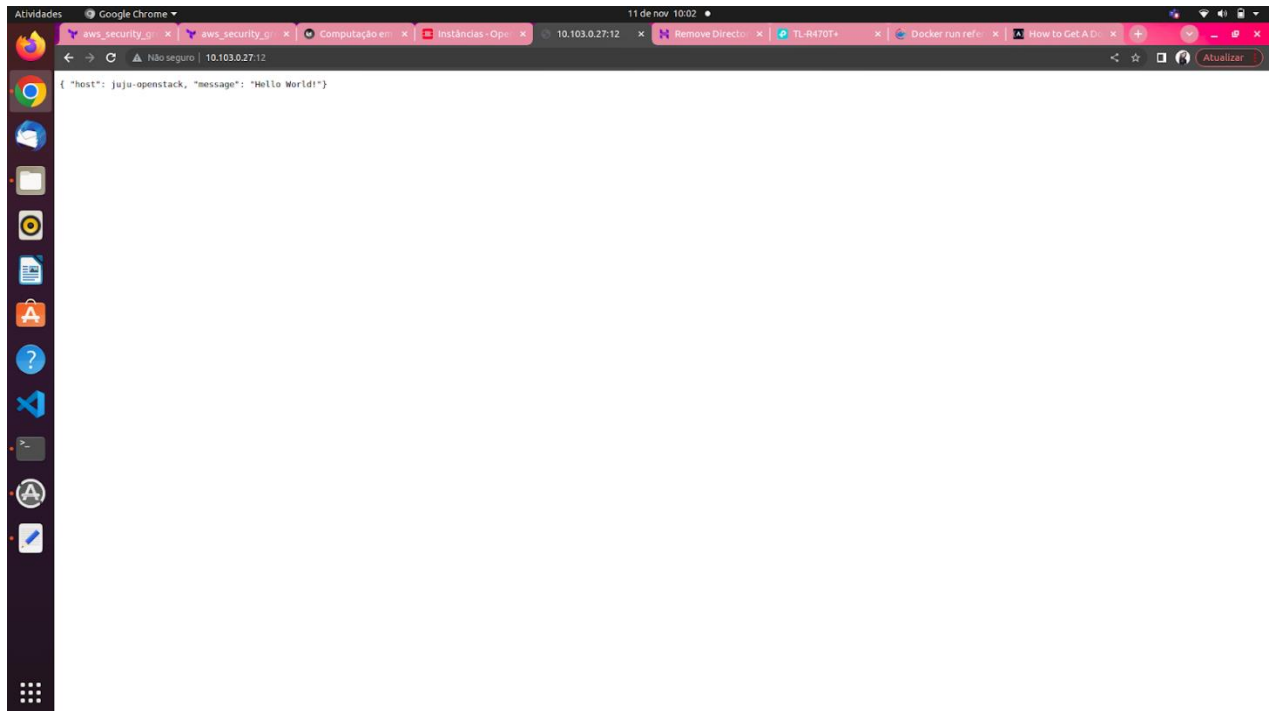


4. Da Aba do navegador com o WebServer Rodando.



CHECKPOINT-2 DOCKER

1. De um print das Telas abaixo:
2. WebServer Rodando no navegador.
- 3.



QUESTÕES-1

1. O que é Docker Swarm?

O Docker Swarm é uma ferramenta de orquestração de container que executa a aplicação do Docker. Essa ferramenta foi configurada para operar junto com a aplicação do Docker em um cluster, sendo que as atividades do cluster são controladas pelo gerenciador do swarm e as máquinas que fazem parte do cluster são referenciadas como nós. Dessa forma, há vários nós trabalhadores (workers) e um gerenciador (manager) que é responsável por entregar recursos aos nós trabalhadores e garantir o funcionamento do cluster de maneira eficiente.

Sintetizando, podemos dizer que o Docker Swarm abrange um grupo de máquinas físicas ou virtuais que estão rodando a aplicação do Docker e que foram configuradas para operar juntas em um cluster. Uma das grandes vantagens de usar essa ferramenta é o alto nível de disponibilidade oferecido para as aplicações (high availability), além de que o Docker Swarm permite o usuário conectar containers a múltiplos hosts de maneira parecida ao Kubernetes e de ter dois tipos de serviços que são replicáveis e globais.

2. O que é Docker Registry?

O Docker Registry é um armazenamento e sistema de distribuição para imagens nomeadas do Docker. A mesma imagem pode ter várias versões diferentes que são identificadas por suas tags. Sendo que, o Docker registry é organizado em repositórios do Docker, onde um repositório contém todas as versões de uma imagem específica. O registry permite que os usuários do Docker façam pull de imagens localmente, bem como push de novas imagens para o registry (considerando as permissões de acesso adequadas). E por padrão, o engine do Docker interage com o DockerHub, que é a instância pública de registry do Docker.

3. O que é Docker Engine?

O Docker Engine é uma tecnologia de containerização open source para construir (fazer o build) e containerizar as aplicações do usuário, de tal forma que o Docker Engine age como uma aplicação do tipo cliente-servidor (client-server) com: um servidor que roda um processo daemon (o dockerd) de longa duração, APIs que especificam interfaces que os programas podem usar para se comunicar e instruir o Docker daemon, e que possui uma interface de comando de linha (CLI). Importante ressaltar que o CLI utiliza as APIs do Docker para controlar ou interagir com o Docker daemon através de scripts ou comandos do CLI. Ademais, é o daemon que cria e gerencia objectos do Docker, como imagens, containers, redes e volumes.

CHECKPOINT-3 DASHBOARD KUBERNETES

1. De um print das Telas abaixo:
2. Do dashboard Kubernetes com as 5 replicas

The screenshot shows a web browser with the Kubernetes Dashboard. The dashboard has a sidebar with 'Roteiros' (Routes) and a main content area with 'Fazendo um Deploy' (Making a Deploy) and 'Acessando o Deploy' (Accessing the Deploy). The 'Fazendo um Deploy' section includes instructions and a terminal window showing the execution of 'kubectl run hello-node' and 'kubectl get all'. The terminal output shows a table of pods and their status.

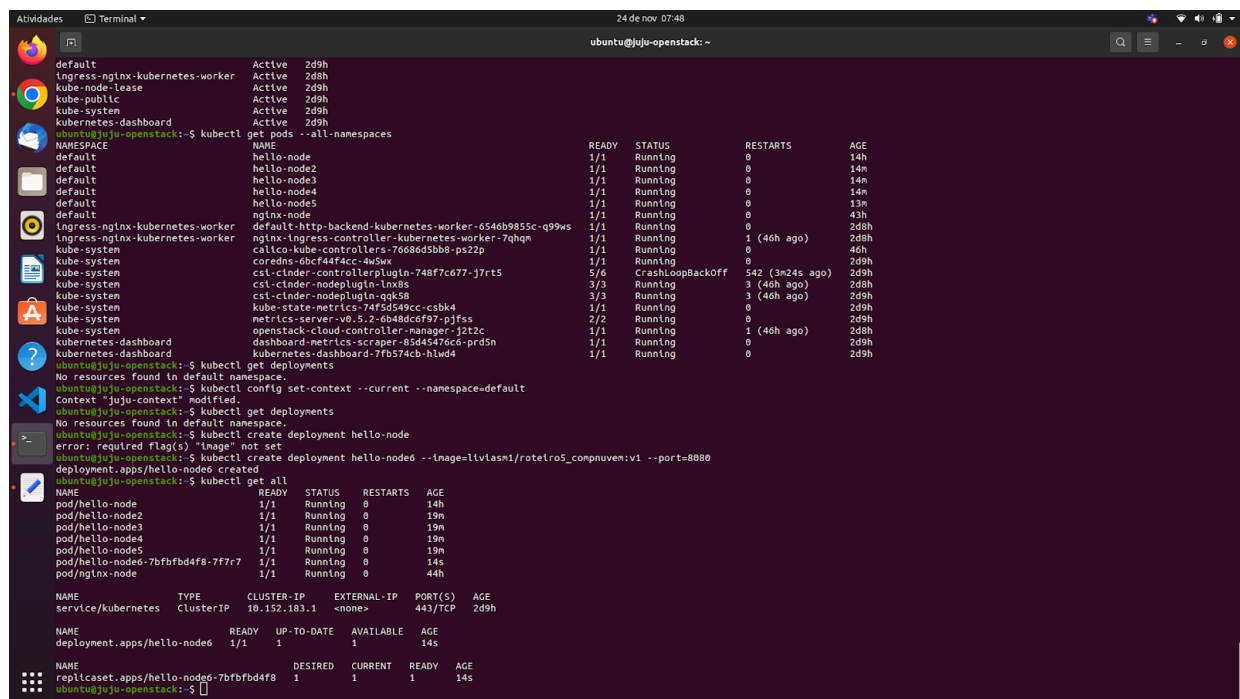
NAME	READY	STATUS	RESTARTS	AGE
pod/hello-node	1/1	Running	0	14h
pod/hello-node2	1/1	Running	0	28s
pod/hello-node3	1/1	Running	0	13s
pod/hello-node4	1/1	Running	0	8s
pod/hello-node5	1/1	Running	0	4s
pod/nginx-node	1/1	Running	0	43h

The 'Acessando o Deploy' section shows the command 'kubectl expose deployment hello-node --type=ClusterIP' and the creation of an ingress resource. The terminal output shows the command 'kubectl run hello-node' and the resulting pod status.

Observação: A princípio eu havia criado os pods através do comando run, entretanto esse comando não cria mais o deploy (deprecated), por isso, através do comando deploy criei mais dois pods (hello-node6 e hello-node7) que foram usados nos checkpoints a seguir.

CHECKPOINT-4 EXPOSE SERVICE

1. De um print das Telas abaixo:
2. Do comando -> kubectl get all



```

24 de nov 07:48
ubuntu@juju-openstack: ~$ kubectl get pods --all-namespaces
NAMESPACE      NAME                                READY   STATUS    RESTARTS   AGE
default         hello-node                          1/1     Running   0           14h
default         hello-node2                         1/1     Running   0           14h
default         hello-node3                         1/1     Running   0           14h
default         hello-node4                         1/1     Running   0           14h
default         hello-node5                         1/1     Running   0           13m
default         nginx-node                          1/1     Running   0           43h
ingress-nginx   ingress-nginx-kubernetes-worker     1/1     Running   0           2d8h
ingress-nginx   ingress-nginx-kubernetes-worker     1/1     Running   1 (46h ago)  2d8h
kube-system     calico-kube-controllers-76086d5bb8-ps2zp  1/1     Running   0           46h
kube-system     coredns-6bcf44f4cc-4w5wx           1/1     Running   0           2d9h
kube-system     csi-cinder-controllerplugin-748f7c677-j7rt5  5/6     CrashLoopBackOff  542 (3m24s ago)  2d9h
kube-system     csi-cinder-nodeplugin-lnx8s         3/3     Running   3 (46h ago)  2d8h
kube-system     csi-cinder-nodeplugin-qql58        3/3     Running   3 (46h ago)  2d9h
kube-system     kube-state-metrics-74f5d549cc-csbk4  1/1     Running   0           2d9h
kube-system     metrics-server-v0.5.2-6b48dcf97-pjffs  2/2     Running   0           2d9h
kube-system     openstack-cloud-controller-manager-j2t2c  1/1     Running   1 (46h ago)  2d8h
kubernetes-dashboard  dashboard-metrics-scraper-85d45476c0-prd5n  1/1     Running   0           2d9h
kubernetes-dashboard  kubernetes-dashboard-7fb574cb-hlwd4  1/1     Running   0           2d9h

ubuntu@juju-openstack:~$ kubectl get deployments
No resources found in default namespace.

ubuntu@juju-openstack:~$ kubectl config set-context --current --namespace=default
Context "juju-context" modified.

ubuntu@juju-openstack:~$ kubectl get deployments
No resources found in default namespace.

ubuntu@juju-openstack:~$ kubectl create deployment hello-node6
error: required flag(s) "image" not set

ubuntu@juju-openstack:~$ kubectl create deployment hello-node6 --image=lvlnas1/roterio5_compuvme:v1 --port=8088
deployment.apps/hello-node6 created

ubuntu@juju-openstack:~$ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/hello-node                      1/1     Running   0           14h
pod/hello-node2                     1/1     Running   0           19m
pod/hello-node3                     1/1     Running   0           19m
pod/hello-node4                     1/1     Running   0           19m
pod/hello-node5                     1/1     Running   0           19m
pod/hello-node6-7bfbfd4f8-7f7r7    1/1     Running   0           14s
pod/nginx-node                      1/1     Running   0           44h

NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/kubernetes  ClusterIP   10.152.183.1  <none>        443/TCP    2d9h

NAME      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hello-node6  1/1      1             1           14s

NAME      DESIRED   CURRENT   READY   AGE
replicaset.apps/hello-node6-7bfbfd4f8  1         1         1       14s

```

CHECKPOINT-4 ACESSAR O DEPLOY

1. De um print das Telas abaixo:
2. Do comando -> kubectl get all
3. WebServer Rodando no navegador utilizando a porta aberta.

```
Atividades Terminal 24 de nov 08:26
ubuntu@juju-openstack: ~
calico          go1.13.8      active      2      calico          stable      54      no      Calico is active
containerd      3.0.1        active      2      containerd      stable      41      no      Container runtime available
easyrsa         3.4.5        active      1      easyrsa         stable      26      no      Certificate Authority connected.
etcd            1.25.4       active      1      etcd            stable      718     no      Healthy with 1 known peer
kubernetes-control-plane 1.25.4       active      1      kubernetes-control-plane  stable      284     yes     Kubernetes control-plane running.
kubernetes-worker 1.25.4       waiting     1/2    kubernetes-worker  stable      70      yes     waiting for machine
openstack-integrator xena         active      1      openstack-integrator  stable      45      no      Ready

Machine State Address Inst id Series AZ Message
0 started 192.169.0.137 62ffda71-7bae-4bfa-9348-fbd59f37e944 focal nova ACTIVE
1 started 192.169.0.198 89df1asc-54fd-4d2a-bc41-789ac9c17853 focal nova ACTIVE
2 started 192.169.0.169 82cc30ea-daba-4764-a6d1-d8fce56b33fe focal nova ACTIVE
3 started 192.169.0.135 1d6ad57b-7e9e-47c7-b224-68e2b332f310 focal nova ACTIVE
4 pending 192.169.0.91 8e70e3bd-2f2a-46e6-b089-57f86a913139 focal nova ACTIVE

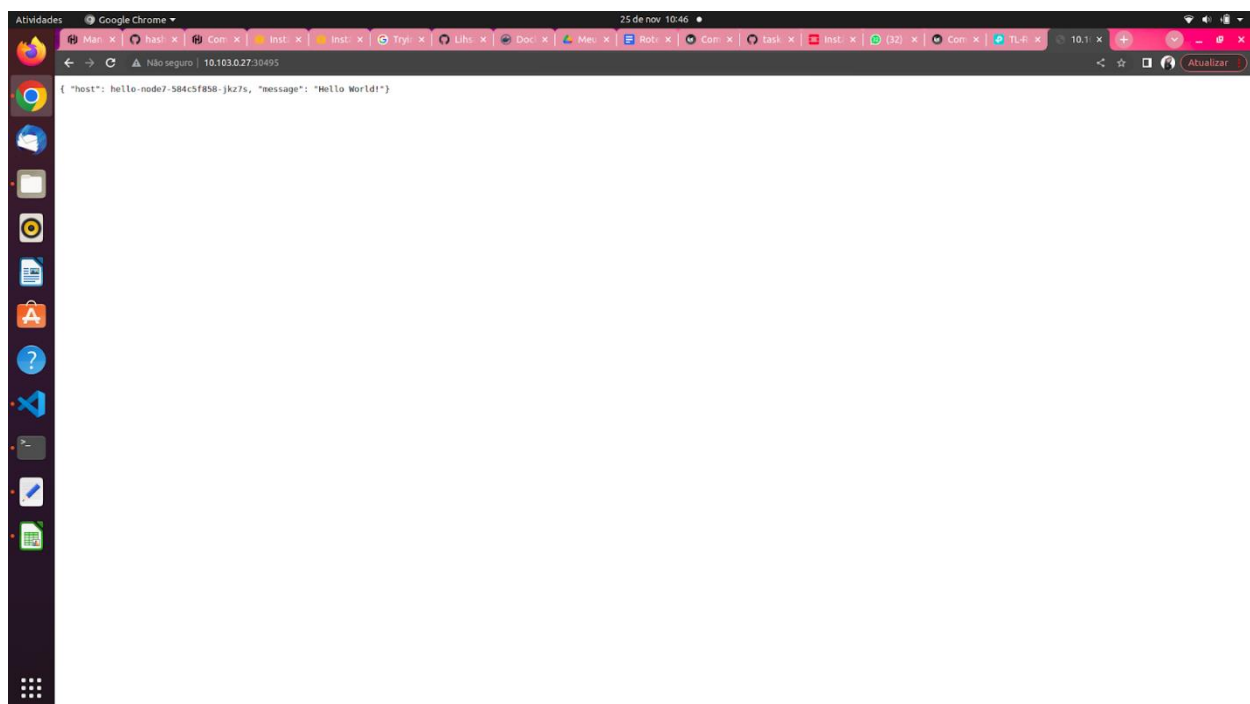
ubuntu@juju-openstack:~$ juju run --unit kubernetes-worker/0 "open-port 30495"

ubuntu@juju-openstack:~$
ubuntu@juju-openstack:~$ kubectl get all
NAME                                READY    STATUS    RESTARTS   AGE
pod/hello-node                      1/1      Running   0           15h
pod/hello-node2                     1/1      Running   0           57m
pod/hello-node3                     1/1      Running   0           56m
pod/hello-node4                     1/1      Running   0           56m
pod/hello-node5                     1/1      Running   0           56m
pod/hello-node6-7bfbfbd4f8-7f7r7  1/1      Running   0           37m
pod/hello-node7-584c5f858-jkz7s    1/1      Running   0           15m
pod/nginx-node                      1/1      Running   0           44h

NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
service/hello-node6                 ClusterIP    10.152.183.51 <none>          8080/TCP        35m
service/hello-node7                 NodePort     10.152.183.52 <none>          8080:30495/TCP 15m
service/kubernetes                   ClusterIP    10.152.183.1  <none>          443/TCP        2d9h

NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/hello-node6         1/1      1              1            37m
deployment.apps/hello-node7         1/1      1              1            15m

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/hello-node6-7bfbfbd4f8 1          1          1        37m
replicaset.apps/hello-node7-584c5f858  1          1          1        15m
ubuntu@juju-openstack:~$
```



CHECKPOINT-5 MODIFICACAO

1. De um print das Telas abaixo:
2. Do comando -> kubectl get all
3. WebServer modificado Rodando no navegador utilizando a porta aberta.

Comando do get all para o hello-node-teste criado:

```
Atividades Terminal 24 de nov 09:55
ubuntu@juju-openstack: ~/kubernetes-core

ubuntu@juju-openstack: ~/kubernetes-core$ kubectl get all
NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/hello-node-teste    1/1      1              1            46m
deployment.apps/hello-node6         1/1      1              1            121m
deployment.apps/hello-node7         1/1      1              1            98m

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/hello-node-teste-58fb74dc5    0          0          0        11m
replicaset.apps/hello-node-teste-646cf9b479    1          1          1        46m
replicaset.apps/hello-node6-7bfbfbd4f8        1          1          1        121m
replicaset.apps/hello-node7-584c5f858         1          1          1        98m

ubuntu@juju-openstack:~/kubernetes-core$ juju run --untt kubernetes-worker/0 "open-port 8080"

ubuntu@juju-openstack:~/kubernetes-core$ kubectl get all
NAME                                READY    STATUS      RESTARTS    AGE
pod/hello-node                        1/1      Running    0            16h
pod/hello-node-teste-646cf9b479-5bcj8    1/1      Running    0            51m
pod/hello-node2                       1/1      Running    0            146m
pod/hello-node3                       1/1      Running    0            145m
pod/hello-node4                       1/1      Running    0            145m
pod/hello-node5                       1/1      Running    0            145m
pod/hello-node6-7bfbfbd4f8-7f7r7        1/1      Running    0            126m
pod/hello-node7-584c5f858-jkz7s         1/1      Running    0            104m
pod/nginx-node                         1/1      Running    0            46h

NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
service/hello-node6                  ClusterIP    10.152.183.51  <none>         8080/TCP         124m
service/hello-node7                  NodePort     10.152.183.52  <none>         8080:30495/TCP   192m
service/kubernetes                    ClusterIP    10.152.183.1   <none>         443/TCP          2d11h

NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/hello-node-teste    1/1      1              1            51m
deployment.apps/hello-node6         1/1      1              1            126m
deployment.apps/hello-node7         1/1      1              1            104m

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/hello-node-teste-58fb74dc5    0          0          0        17m
replicaset.apps/hello-node-teste-646cf9b479    1          1          1        51m
replicaset.apps/hello-node6-7bfbfbd4f8        1          1          1        124m
replicaset.apps/hello-node7-584c5f858         1          1          1        104m

ubuntu@juju-openstack:~/kubernetes-core$
```

O server.js modificado para rodar uma nova mensagem:

```
Atividades Terminal 25 de nov 09:54
ubuntu@juju-openstack: ~/hello-node

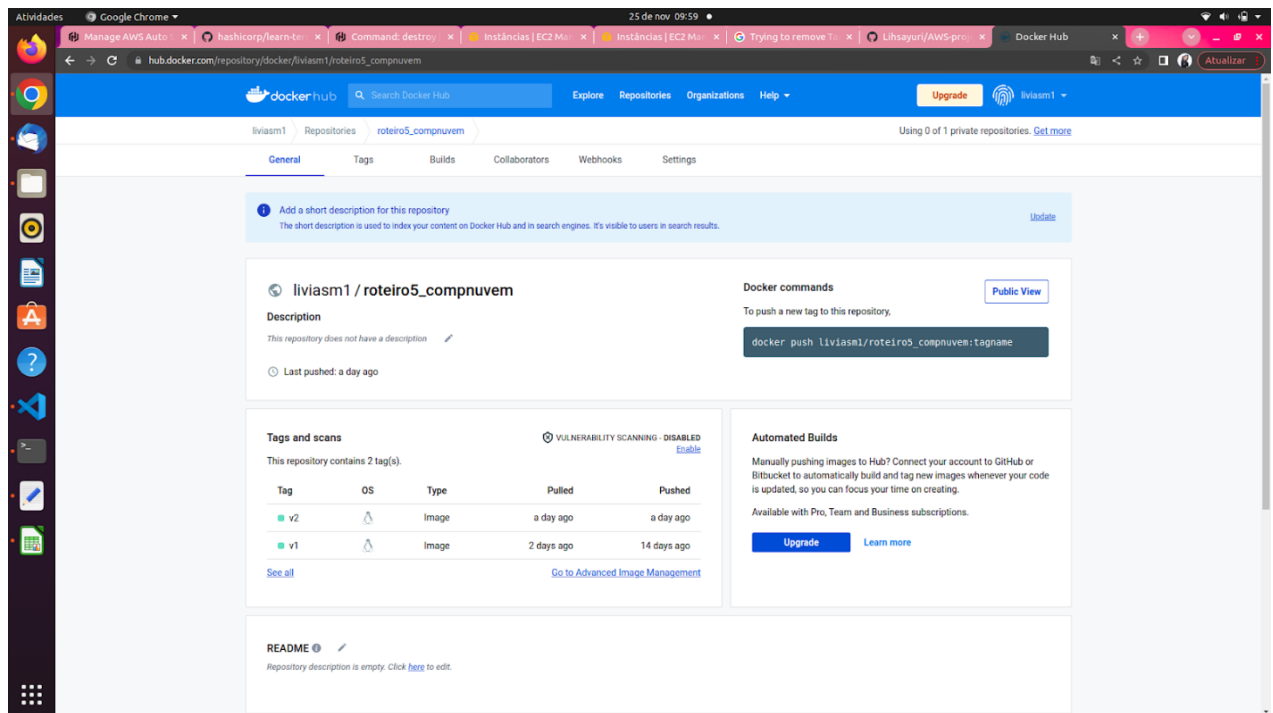
ubuntu@juju-openstack: ~/hello-node$ cat server.js
#!/usr/bin/env node

var http = require('http');
var os = require('os');

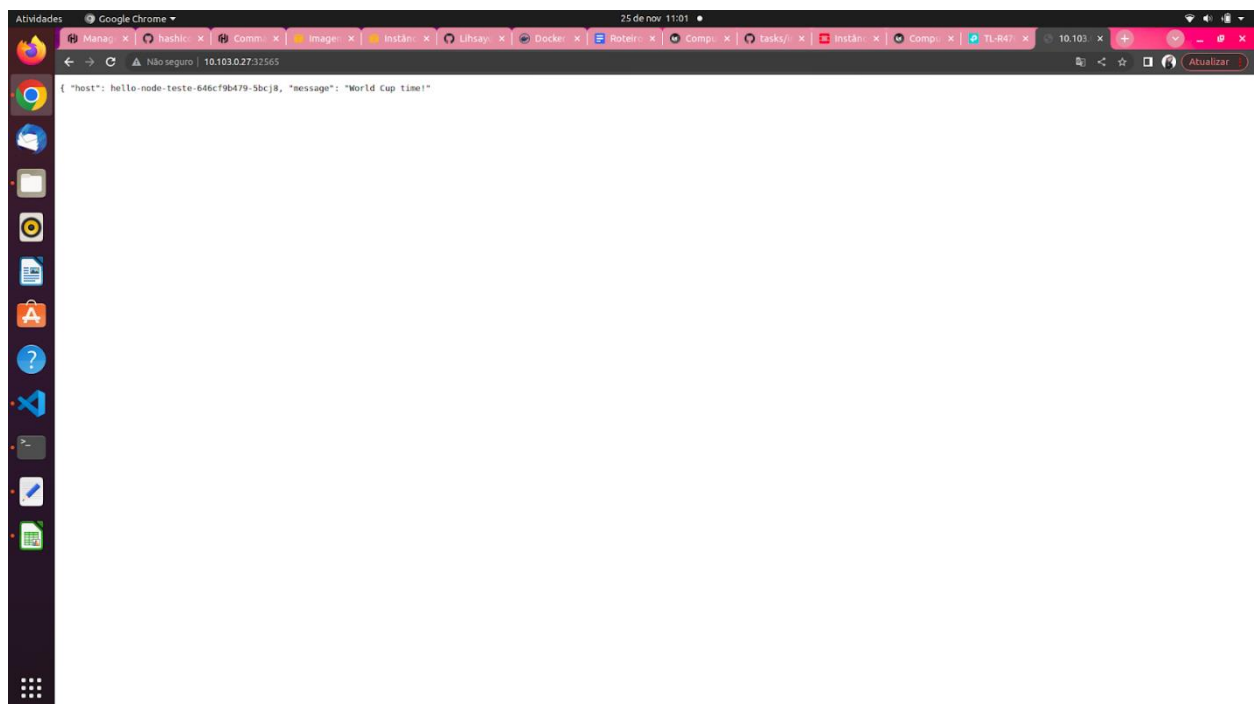
var handleRequest = function(request, response) {
  console.log('Received request for URL: ' + request.url);
  response.writeHead(200);
  response.end('{"host": ' + os.hostname() + ', "message": "World Cup time!"}');
};

var www = http.createServer(handleRequest);
www.listen(8080);
console.log('Server running at http://localhost:8080/');
```


O Docker com a nova imagem em uma nova tag: v2.



Web server rodando com a nova mensagem: "World Cup time".



OBSERVAÇÕES FINAIS:

O Dashboard do Kubernetes não estava rodando. O professor Tiago Demay tentou nos ajudar mudando algumas relações, mas não funcionou. Abaixo,

após já termos modificado algumas relações, está o que estava retornando na porta 8080:

