



# **Desenvolvimento de aplicação com integração do orquestrador *Kubernetes***

Trabalho Laboratorial nº 2 (TL2)  
Laboratório de Tecnologias de Informação

LEI 2020/21

Grupo 5

Filipe Alexandre Coelho Almeida, 2171302

Gonçalo Miguel Conceição Vicente, 2172131

Leiria, junho de 2021



# Resumo

O presente documento insere-se no âmbito da disciplina de Laboratório de Tecnologias de Informação (LTI), pertencente à Licenciatura em Engenharia Informática e descreve a implementação de uma aplicação Web para gerir os recursos disponibilizados através do orquestrador de *containers Kubernetes*.

O orquestrador de *containers Kubernetes* é uma ferramenta *Open Source* para automatizar a gestão de aplicações que correm em *containers* e, de forma a implementar este projeto, utilizamos a framework Laravel juntamente com a framework Vue.js, para construir a aplicação Web, que irá funcionar como *front-end*. Através da framework Vue.js é possível comunicar com o servidor Linux, onde se encontra instalado o *Kubernetes*, através de pedidos RESTFull (GET, POST, PUT, DELETE).

O trabalho laboratorial descrito neste documento, encontra-se dividido em três partes distintas. Na primeira parte é feito um estudo acerca das vantagens e desvantagens da solução utilizada relativamente a outras possíveis soluções existentes. Na segunda parte está documentada todo o processo de instalação de três máquinas Linux, com o sistema operativo Ubuntu 18.04 LTS, de forma a formar um *cluster*. Por fim, na terceira parte são abordadas funcionalidades existentes numa aplicação de *front-end*, desenvolvida com base no projeto laboratorial 1, onde é possível realizar um conjunto de operações permitidas através do uso da API *Kubernetes*.

**Palavras-chave:** Virtualização, *Kubernetes*, *RESTFull*, API, *Containers*

# Abstract

The following document is part of the discipline of Information Technology Laboratory (LTI), belonging to the Degree in Computer Engineering and explains the implementation of a web application to manage the resources made available through the Kubernetes container orchestrator.

Kubernetes Container Orchestrator is an *OpenSource* tool to automate the management of applications that run in containers and, in order to implement this project, use a Laravel framework together with a Vue.js framework to build a web application that will work as front-end. Through the Vue.js framework it is possible to communicate with the Linux server, where Kubernetes is installed, through *RESTFull* requests (GET, POST, PUT, DELETE).

The laboratory work described in this document is divided into three distinct parts. In the first part, a study is made of the advantages and disadvantages of the solution based on other existing solutions. The second part is documented the entire process of installing three Linux machines, with the operating system Ubuntu 18.04 LTS, in order to form a cluster. Finally, in the third part, features existing in a front-end application, developed based on the laboratory project 1, where it is possible to perform a set of operations allowed through the use of the Kubernetes API, are discussed.

**Keywords:** Virtualization, Kubernetes, RESTFull, API, Containers

# Índice

Resumo.....	iii
Abstract.....	iv
Lista de Figuras.....	vii
Lista de Tabelas.....	viii
1. Introdução .....	1
2. Análise das soluções do Kubernetes.....	2
2.1. Integração <i>OpenStack</i> com <i>Kubernetes</i> .....	4
3. Implementação da solução <i>Kubernetes</i> .....	5
3.1. Configuração do <i>Master Node</i> .....	6
3.2. Configuração dos Worker Nodes .....	8
3.3. Proxy.....	9
4. Aplicação desenvolvida .....	11
4.1. Componente goFor .....	11
4.2. Componente definirIPKubernetes .....	12
4.3. Componente loginKubernetes .....	13
4.4. Componente dashboard .....	14
4.5. Componente <i>namespaces</i> .....	15
4.6. Componente <i>nodes</i> .....	16
4.7. Componente Cluster Roles .....	17
4.8. Componente <i>pods</i> .....	18
4.9. Componente <i>Deployment</i> .....	21

<b>4.10.</b>	<b>Componente <i>Services</i>.....</b>	<b>23</b>
<b>4.11.</b>	<b>Componente <i>endpoints</i> .....</b>	<b>25</b>
<b>4.12.</b>	<b>Dificuldades encontradas.....</b>	<b>27</b>
<b>4.13.</b>	<b>URIs utilizados.....</b>	<b>28</b>
<b>5.</b>	<b>Análise crítica e proposta de melhorias.....</b>	<b>29</b>
<b>6.</b>	<b>Conclusão .....</b>	<b>30</b>
	<b>Bibliografia.....</b>	<b>31</b>

# Lista de Figuras

Figura 1 - Exemplo de arquitetura <i>Kubernetes</i> com <i>Kubeadm</i> .....	3
Figura 2 - Configuração de <i>NAT Network</i> .....	5
Figura 3 - <i>Port Forwarding SSH</i> .....	6
Figura 4 - Escolha da aplicação a utilizar .....	12
Figura 5 - Definir o endereço IP e o porto a utilizar .....	13
Figura 6 - Regras de <i>Port Forwarding</i> .....	13
Figura 7 - Formulário de login .....	14
Figura 8 - Vista do componente <i>dashboard.vue</i> .....	15
Figura 9 - Formulário para criação de um novo <i>namespace</i> .....	16
Figura 10 - Lista de todos os <i>namespaces</i> .....	16
Figura 11 - Lista de todos os <i>nodes</i> .....	17
Figura 12 - Lista de todos os <i>cluster's roles</i> .....	18
Figura 13 - <i>Namespaces</i> possíveis para seleção .....	19
Figura 14 - Lista de todos os <i>Pods</i> .....	19
Figura 15 - Portos utilizados por um determinado <i>pod</i> .....	20
Figura 16 - Informações sobre <i>resources</i> e volumes de um <i>pod</i> .....	20
Figura 17 - Formulário para a criação de um <i>pod</i> .....	21
Figura 18 - Lista de todos os <i>deployments</i> .....	22
Figura 19 - Portos utilizados por um determinado <i>deployment</i> .....	22
Figura 20 - Formulário para a criação de um <i>deployment</i> .....	23
Figura 21 - Portos utilizados por um determinado <i>service</i> .....	24
Figura 22 - Lista de todos os <i>services</i> .....	24
Figura 23 - Formulário para a criação de um <i>service</i> .....	25
Figura 25 - Portos utilizados por um determinado <i>endpoint</i> .....	26
Figura 26- Lista de todos os <i>endpoints</i> .....	26
Figura 27 - <i>Subset address</i> de um determinado <i>endpoint</i> .....	26
Figura 28 - Ficheiro <i>kube-apiserver.yaml</i> .....	27
Figura 29 - <i>Script</i> em <i>perl</i> para auxiliar na ativação do <i>proxy</i> .....	28

# Lista de Tabelas

Tabela 1 - Lista de todos pedidos RESTFull utilizados no componente <code>dashboard.vue</code> .....	14
Tabela 2 - Lista de todos pedidos RESTFull utilizados no componente <code>namespaces.vue</code> .....	16
Tabela 3 - Lista de todos pedidos RESTFull utilizados no componente <code>nodes.vue</code> .....	17
Tabela 4 - Lista de todos pedidos RESTFull utilizados no componente <code>clusterRoles.vue</code> .....	18
Tabela 5 - Lista de todos pedidos RESTFull utilizados no componente <code>Pods.vue</code> .....	21
Tabela 6 - Lista de todos pedidos RESTFull utilizados no componente <code>deployment.vue</code> .....	23
Tabela 7 - Lista de todos pedidos RESTFull utilizados no componente <code>services.vue</code> .....	25
Tabela 8 - Lista de todos pedidos RESTFull utilizados no componente <code>endpoint.vue</code> .....	27
Tabela 9 - Lista de todos pedidos RESTFull utilizados .....	28



# 1. Introdução

O presente documento insere-se na licenciatura em Engenharia Informática, mais concretamente na unidade curricular Laboratório de Tecnologias de Informação decorrente no ano letivo 2020/2021.

Neste documento, é descrito como foi implementado o trabalho laboratorial 2 bem como as funcionalidades existentes.

O trabalho laboratorial descrito neste documento, encontra-se dividido em três partes distintas. Na primeira parte é feito um estudo acerca das vantagens e desvantagens da solução utilizada relativamente a outras possíveis soluções existentes.

Na segunda parte está documentada todo o processo de instalação de três máquinas Linux, com o sistema operativo Ubuntu 18.04 LTS, a instalação do *Docker* e a instalação do *Kubernetes* de forma a formar um *cluster*.

Por fim, na terceira parte são abordadas funcionalidades existentes numa aplicação de *front-end*, desenvolvida com base no projeto laboratorial 1, onde é possível realizar um conjunto de operações permitidas através do uso da API *Kubernetes*.

Nos dias de hoje, a utilização de *containers* tem vindo a crescer cada vez mais, e a utilização de um orquestrador como o *Kubernetes* acaba por ter uma grande utilidade, uma vez que permite automatizar processos que teriam que ser feitos manualmente através da utilização do comando *Docker*. Além disso, uma aplicação que permita fazer um vasto conjunto de operações através de um interface gráfico ao invés da utilização de comandos complexo acaba por ser uma mais-valia.

## 2. Análise das soluções do *Kubernetes*

O *Kubernetes* é uma ferramenta *OpenSource* de orquestração de *containers* para automatizar a implementação, o dimensionamento e a gestão de aplicações a correr em *containers*.

O sistema *Kubernetes* tem por base o *Docker*. Este é, também, uma plataforma *OpenSource* de desenvolvimento, provisionamento e execução de aplicações que tem como principal objetivo facilitar a criação e gestão de ambientes isolados com recurso aos *containers*.

Com a utilização do sistema *Kubernetes* como orquestrador, é possível automatizar todas as medidas que um administrador teria de fazer manualmente através de pedidos com o comando *docker*.

De forma a escolher qual a melhor opção para instalar o *Kubernetes*, foi necessário fazer uma pesquisa sobre o que existia disponível. Chegamos à conclusão de que existem várias soluções, cada uma com diferentes requisitos. Por exemplo, existem opções que apresentam uma melhor facilidade de manutenção face a outras, melhor segurança ou controlo, bem como existem aplicações que requerem mais ou menos conhecimentos/experiência. Com a nossa pesquisa, foi também possível apurar que existe a possibilidade de implementar *clusters* de *Kubernetes* em diferentes ambientes, como é o caso de máquinas locais, *cloud's* ou *datacenters*.

Resumidamente, a implementação de um *cluster* em *Kubernetes* resume-se a dois grandes tipos: ambiente de testes e ambiente de produção, mudando com isso os recursos necessários.

Existem então várias soluções para instalação do *Kubernetes* como o *MiniKube*, *Kind*, *Kudeamd*, *Kubespray* (não se adapta ao ambiente de testes). Entre estas soluções existem várias diferenças, principalmente os requisitos mínimos e o desempenho de cada uma das soluções.

Uma vez excluída a solução *Kubespray*, sobraram três outras soluções, sendo necessário visualizar em pormenor as suas características. A nível de recursos necessários tanto a solução *MiniKube* como a *Kubeamd* apresentam características bastante idênticas:

necessidade de 2 CPUs e 2GB de RAM, porém a solução *Kind* apresenta uma grande discrepância em relação à RAM, necessita de 8GB. Uma vez que a solução está a ser desenvolvida diretamente nos nossos computadores portáteis, não é viável ter uma máquina a consumir 8GB de RAM face à quantidade de recursos que temos.

Assim, limitamos a nossa escolha entre as soluções *MiniKube* e *Kubeadm*. Apesar de o *MiniKube* ser a solução mais utilizada a nível de testes e ser uma solução bastante fácil de implementar, consideramos que a solução *Kubeadm* é uma solução que permite explorar o verdadeiro potencial do *Kubernetes* e a escalabilidade que oferece, uma vez que caso se pretenda adicionar mais um *Worker* é apenas necessário configurar uma nova máquina virtual e fazer o *join* ao *Master*. Na figura 1, é possível visualizar o esquema de implementação da solução *Kubeadm* para o âmbito deste projeto: 1 *Master Node* e 2 *Worker Nodes*.

Relativamente ao *Kubeadm*, a principal desvantagem que consideramos existir reflete-se na necessidade de adicionar mais que 2 *Worker Nodes*. Com a adição de mais uma máquina virtual, já ficará pesado para um ambiente de testes em que deverá existir uma menor quantidade recursos quando comparado com um ambiente de produção.

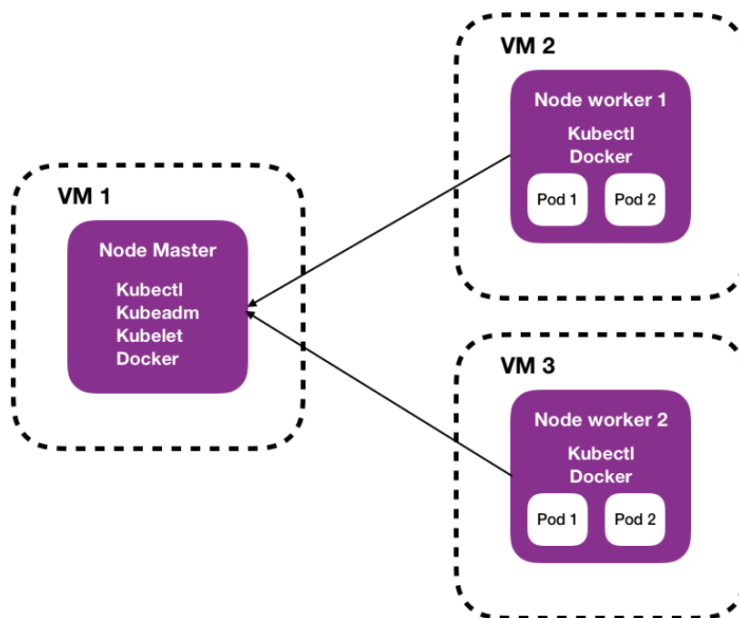


Figura 1 - Exemplo de arquitetura *Kubernetes* com *Kubeadm*

## 2.1. Integração *OpenStack* com *Kubernetes*

O *Kubernetes* e o *OpenStack* estão profundamente integradas, sendo essa integração o resultado de anos de desenvolvimento das duas plataformas, onde os recursos de computação, rede e armazenamento são consumidos do *OpenStack* pelos *clusters Kubernetes*.

A plataforma de desenvolvimento *OpenStack* foi projetada estar implementada num *datacenter*, permitindo assim que os *clusters Kubernetes* em execução no *OpenStack* sejam redimensionados conforme necessário.

É possível implementar o *Kubernetes* no *OpenStack* através de duas opções: *Kuryr CNI and Neutron tenant networks* ou *Kubernetes SDN and Neutron Provider Networks*.

Neste projeto, não foi realizada a integração do *Kubernetes* com o *OpenStack*, uma vez que o nosso cenário de implementação está a correr diretamente nas nossas máquinas que não tem tantos recursos como acontece em servidores.

### 3. Implementação da solução *Kubernetes*

De forma a implementarmos o *Kubernetes* num *cluster*, como era um dos requerimentos do projeto, procedeu-se à instalação de três máquinas virtuais, com o sistema operativo Linux, Ubuntu 18.04 LTS, de forma a possibilitar um *Master Node* e dois *Worker Nodes*. Antes de passar para a instalação do orquestrador *Kubernetes*, instala-se o *Docker*.

Para a criação e instalação das máquinas virtuais foi utilizado o *hypervisor VirtualBox*.

Assim, foi necessário criar uma máquina virtual com: 4GB de RAM, 2 vCPUs, disco de 100 GB dinamicamente alocados de forma a evitar estar a ocupar à priori a totalidade do disco sem que exista necessidade e uma placa de rede em modo *NAT Network*.

O mesmo procedimento segue-se para as outras duas máquinas virtuais, apenas com a diferença de terem apenas 2GB de RAM.

Como não existia ainda nenhuma rede *NAT Network*, foi necessário criar uma *pool*. Para tal foi necessário ir às definições do *hypervisor* e na opção *Network* adicionar uma *NAT Network*, com as configurações presentes na figura 2.

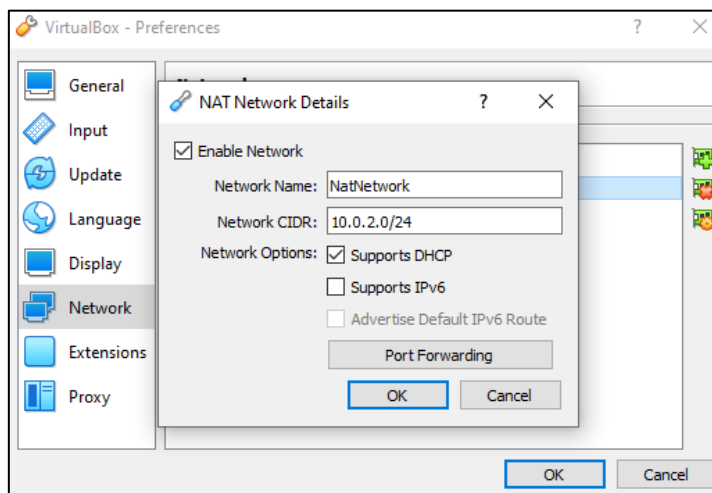
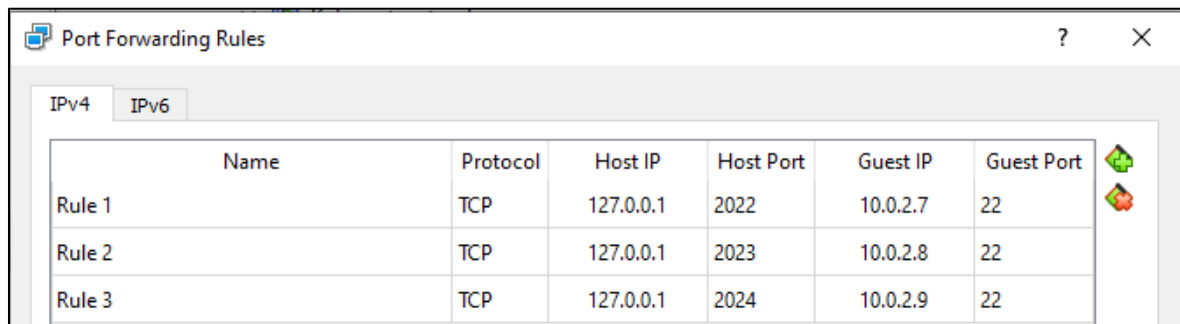


Figura 2 - Configuração de NAT Network

Prosseguindo com a instalação do sistema operativo Linux, *Ubuntu* 18.04 LTS, é necessário definir como nome de utilizador, *password* e *hostname* como **ubuntu**. De forma a facilitar a utilização da consola para a inserção de diversos comandos mais à frente é necessário seleccionar a opção *OpenSSH Server*, de forma a ser possível aceder à máquina virtual via SSH.

Após a instalação, é necessário realizar o comando `sudo apt update && sudo apt upgrade -y`, bem como aceder ao ficheiro `00-installer-config.yaml` que se encontra na diretoria `/etc/netplan/` e definir o endereço IP de forma estática. Neste caso, foram definidos endereços IP entre 10.0.2.7-10.0.2.9 e como *default gateway* o endereço 10.0.2.2.

Posto isto, iremos aceder via ssh para fazer os restantes comandos de forma mais facilitada, mas para tal é necessário ir às preferências do *VirtualBox* e realizar o *port forwarding*. Tendo em conta que existem três máquinas diferentes que pretendem ser acedidas, não é possível mapear tudo pelo mesmo porto, assim irá ficar mapeado entre o porto 2022-2024, como é possível observar na figura 3.



Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
Rule 1	TCP	127.0.0.1	2022	10.0.2.7	22
Rule 2	TCP	127.0.0.1	2023	10.0.2.8	22
Rule 3	TCP	127.0.0.1	2024	10.0.2.9	22

Figura 3 - Port Forwarding SSH

### 3.1. Configuração do *Master Node*

A máquina virtual que foi criada com 4GB de RAM irá ser a escolhida para funcionar como *Master Node*. Para tal, iremos aceder à máquina através de SSH, especificando o endereço IP 127.0.0.1 e o porto 2022. Após fazer o login, é necessário fazer `sudo su`.

Neste momento, todos os comandos que iremos escrever irão ser efetuados com permissões de administrador, razão pela qual é necessário ter cuidados redobrados. Posteriormente, é necessário garantir que não existem versões antigas do *Docker*, sendo necessário efetuar o comando `apt remove docker docker-engine docker.io` e de forma a assegurar que existem todos os pacotes necessários para a instalação do *Docker*, efetua-se o comando `apt install ca-certificates software-properties-common gnupg curl apt-transport-https`. Ainda antes de instalar o *Docker*, iremos alterar o nome da máquina para `hostnamectl set-hostname kmaster`, sendo necessário reiniciar para que a alteração seja efetuada. Após a máquina reiniciar, iremos instalar o

*Docker*, através do comando *apt – get install docker.io*. De seguida, é necessário ativar o serviço bem como fazer o seu *start*, e para tal são utilizados os comandos *systemctl enable docker.service* e *service docker start*.

Após se encontrar instalado o *Docker*, é necessário proceder à instalação do orquestrador *Kubernetes*. Recomenda-se fazer um *snapshot* neste ponto, e seguidamente fazer *snapshots* com frequência.

Primeiramente é necessário adicionar o repositório do *Kubernetes*, e para isso utiliza-se os comandos:

- *curl – s https://packages.cloud.google.com/apt/doc/apt – key.gpg | sudo apt – key add*
- *apt – add – repository "deb http://apt.kubernetes.io/ kubernetes – xenial main"*
- *apt – get update*

Neste momento, já se encontra o repositório do *Kubernetes* na máquina, sendo de seguida necessário desativar o *swap*, com recurso ao comando *swapoff – a*. De seguida, irá ser instalado o *Kubernetes* com o comando *apt – get install kubeadm kubelet kubectl – y*. Posteriormente, é necessário desativar a *firewall* através do comando *ufw disable*. Seguidamente, é necessário ir ao ficheiro *hosts* presente na diretoria */etc* e adicionar uma nova entrada: *10.0.2.7 localhost*, onde *10.0.2.7* representa o endereço IP da máquina em questão. Por fim, é necessário descarregar os ficheiros de configuração do orquestrador *Kubernetes* através do comando *kubeadm config images pull*.

Findo estes passos, é necessário voltar à conta de utilizador normal, onde se irá executar o comando *sudo kubeadm init – –apiserver – advertise – address = 10.0.2.7 – –pod – network – cidr = 10.244.0.0/16*. Deve guardar as informações apresentadas na consola, uma vez que irão ser utilizadas mais à frente. De seguida, é necessário executar a próxima sequencia de comandos:

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

De seguida, e de forma a criar uma rede de comunicação entre os diversos *nodes* é necessário executar o comando `sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml`.

Assim, termina-se a configuração do *Master Node*.

### 3.2. Configuração dos Worker Nodes

A partir das outras duas máquinas criadas, seguem-se os seguintes procedimentos para configurar duas máquinas para desempenhar o papel de Worker Node.

Após fazer o login, é necessário fazer `sudo su`.

Neste momento, todos os comandos que iremos escrever irão ser efetuados com permissões de administrador, razão pela qual é necessário ter cuidados redobrados. Posteriormente, é necessário garantir que não existem versões antigas do *Docker*, sendo necessário efetuar o comando `apt remove docker docker-engine docker.io` e de forma a assegurar que existem todos os pacotes necessários para a instalação do *Docker*, efetua-se o comando `apt install ca-certificates software-properties-common gnupg curl apt-transport-https`. Ainda antes de instalar o *Docker*, iremos alterar o nome da máquina para `hostnamectl set-hostname kworker1`, neste caso para a máquina que irá funcionar como *worker 1*, sendo necessário reiniciar para que a alteração seja efetuada. Após a máquina reiniciar, iremos instalar o *Docker*, através do comando `apt-get install docker.io`. De seguida, é necessário ativar o serviço bem como fazer o seu *start*, e para tal são utilizados os comandos `systemctl enable docker.service` e `service docker start`.

Após se encontrar instalado o *Docker*, é necessário proceder à instalação do orquestrador *Kubernetes*. Recomenda-se fazer um *snapshot* neste ponto, e seguidamente fazer *snapshots* com frequência.

Após se encontrar instalado o *Docker*, é necessário proceder à instalação do orquestrador *Kubernetes*. Recomenda-se fazer um *snapshot* neste ponto, e seguidamente fazer *snapshots* com frequência.



Primeiramente é necessário adicionar o repositório do *Kubernetes*, e para isso utiliza-se os comandos:

- `curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt - key add`
- `apt - add - repository "deb http://apt.kubernetes.io/ kubernetes - xenial main"`
- `apt - get update`

Neste momento, já se encontra o repositório do *Kubernetes* na máquina, sendo de seguida necessário desativar o *swap*, com recurso ao comando `swapoff - a`. De seguida, irá ser instalado o *Kubernetes* com o comando `apt - get install kubeadm kubelet kubectl - y`.

Posteriormente, é necessário adicionar o *worker* ao *cluster*, através do comando que se encontra na última linha do output que foi pedido que fosse guardado: `sudo kubeadm join < Enderço IP do Master >:6443 --token < token > -- discovery - token - ca - cert - hash < hash >`. Caso não tenha o respetivo comando, pode obtê-lo através do comando `kubeadm token create --print - join - command`.

Após repetir estes passos para a máquina virtual que irá funcionar como *worker2*, encontra-se implementado um *cluster* com três máquinas.

### 3.3. Proxy

De forma que seja possível contornar a questão da autenticação no orquestrador de *containers Kubernetes*, é possível utilizar um *proxy* que irá redirecionar todos os pedidos à *API*.

Para tal, resolveu-se instalar o servidor web *Nginx* na máquina *Master Node*. Assim, foi necessário executar o comando de instalação do servidor web `sudo apt install nginx`. De seguida é necessário ir ao ficheiro default presente na diretoria `/etc/nginx/site-enabled` e meter como porto de escuta o porto 8080, por exemplo: `listen 8080 default_server`. Feito isto, é necessário correr o comando `kubectl proxy --port = 8081 --address = 0.0.0.0 --accept - hosts = ^ * &`.

Finalizados estes passos, ainda é necessário mapear o porto 8090 no host para o porto 8081 no guest de forma a ser possível comunicar com o *proxy*.

## 4. Aplicação desenvolvida

De forma a interligar o orquestrador *Kubernetes* e os respetivos serviços, com o nosso projeto, implementamos uma aplicação Web, utilizando as frameworks *Laravel* e *Vue.js*, que recorrem às linguagens PHP e *JavaScript* respetivamente, para assegurar a comunicação com as APIs de cada serviço através de pedidos *RESTFull*.

As funcionalidades mandatárias para a aplicação desenvolvida eram:

- Listagem de *nodes* existentes
- Listagem, criação e eliminação de *namespaces*
- Listagem, criação e eliminação de *pods*
- Listagem, criação e eliminação de *deployments*
- Listagem, criação e eliminação de *services*

Além destas funcionalidades também foram implementadas:

- Dashboard com dados estatísticos
- Listagem de *cluster roles*
- Sistema de mudança entre diversos *namespaces*
- Informações sobre portos em uso e resources e volumes de *containers*
- Listagem de *endpoints* e *subsets address*

### 4.1. Componente goFor

Um dos princípios que tomamos por base para a realização deste trabalho laboratorial, passou por juntar a aplicação de gestão do orquestrador *Kubernetes* e a aplicação que comunica com a plataforma *OpenStack*, desenvolvida no trabalho laboratorial 1, numa única aplicação. Assim, seria necessário o utilizador escolher qual dos ambientes deseja utilizar (figura 4), sendo depois direcionado para uma página de login, consoante a opção selecionada. Para tal, foi utilizado o componente *goFor.vue*.

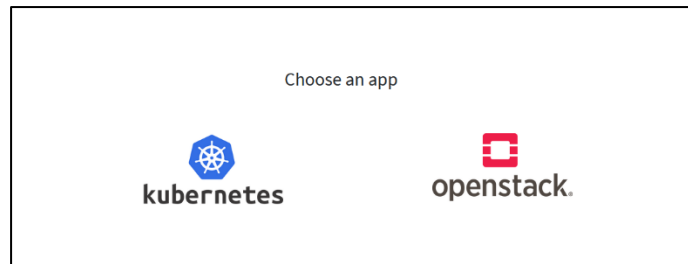


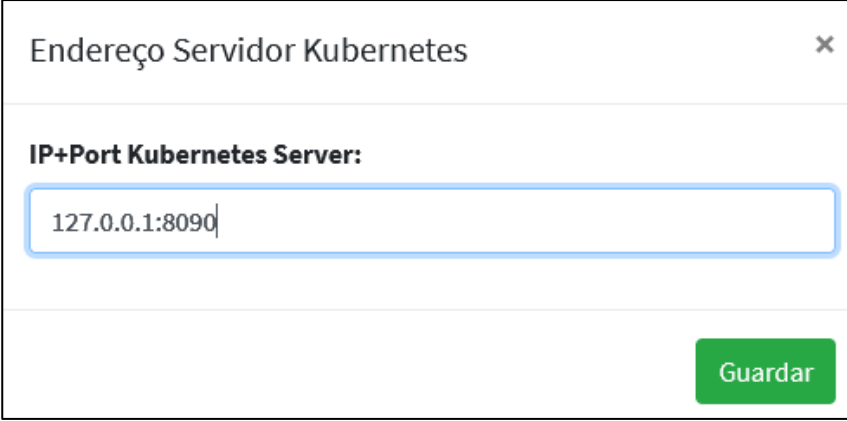
Figura 4 - Escolha da aplicação a utilizar

## 4.2. Componente definirIPKubernetes

De forma a garantir que a aplicação desenvolvida era o mais versátil possível, resolveu-se pedir ao utilizador qual o endereço IP e o porto em que se encontra à escuta o servidor Linux com a instalação do orquestrador *Kubernetes*. Assim, seja qual for o endereço IP e o porto de destino inserido pelo utilizador, desde que válido, a aplicação fica funcional sem que haja necessidade de ajustes programaticamente. Assim, utiliza-se o componente *definirIPKubernetes.vue* para apresentar ao utilizador o formulário de preenchimento.

Na figura 5, encontra-se um exemplo de preenchimento do formulário onde se define qual o endereço IP e o porto em uso. Tendo em conta que se trata de um ambiente de testes e se está a recorrer ao uso de ambientes virtualizados através do *hypervisor Virtual Box*, utilizamos o endereço IP de localhost, uma vez que o servidor com o orquestrador *Kubernetes*, se encontra instalado com uma placa de rede em modo *NAT*.

Uma vez que a placa de rede se encontra em modo *NAT* é necessário proceder ao *Port Forwarding*. Assim, como mostra a figura 6, definiu-se como porto do *host* o porto 8090 e como porto do *guest* o porto 8081, que é o porto em que o *proxy* utilizado se encontra à escuta de pedidos.



The form is titled "Endereço Servidor Kubernetes" and has a close button (X) in the top right corner. It contains a label "IP+Port Kubernetes Server:" followed by a text input field containing "127.0.0.1:8090". At the bottom right, there is a green button labeled "Guardar".

Figura 5 - Definir o endereço IP e o porto a utilizar

IPv4 IPv6						
Name	Protocol	Host IP	Host Port	Guest IP	Guest Port	
Rule 1	TCP	127.0.0.1	2022	10.0.2.7	22	
Rule 2	TCP	127.0.0.1	2023	10.0.2.8	22	
Rule 3	TCP	127.0.0.1	2024	10.0.2.9	22	
Rule 4	TCP	127.0.0.1	8090	10.0.2.7	8081	

Figura 6 - Regras de *Port Forwarding*

### 4.3. Componente loginKubernetes

O componente loginKubernetes.vue apresenta ao utilizador um formulário de login (figura 7), porém, ao contrário do que acontece na plataforma *OpenStack*, onde foi necessário a obtenção de um *Token* para que fosse possível realizar a autenticação, no *Kubernetes* a autenticação pode ser contornada através da implementação de um servidor web *NGINX* que está encarregue de reencaminhar os pedidos entre a API do *Kubernetes* e a aplicação desenvolvida e vice-versa, através do uso de um *proxy*.

Nesse sentido, basta apenas selecionar o botão login para poder ser reencaminhado para o dashboard da aplicação *Kubernetes*, sem que seja preenchido qualquer *username/password*. A implementação de um sistema de login semelhante ao da plataforma OpenStack, através do *Token*, será visto como um trabalho futuro.

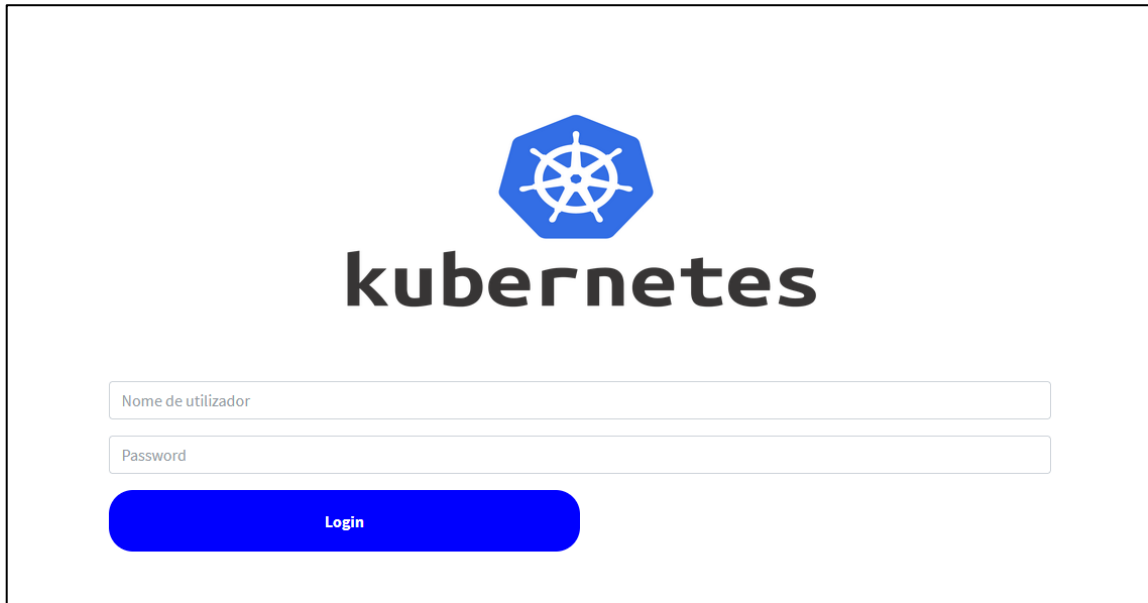


Figura 7 - Formulário de login

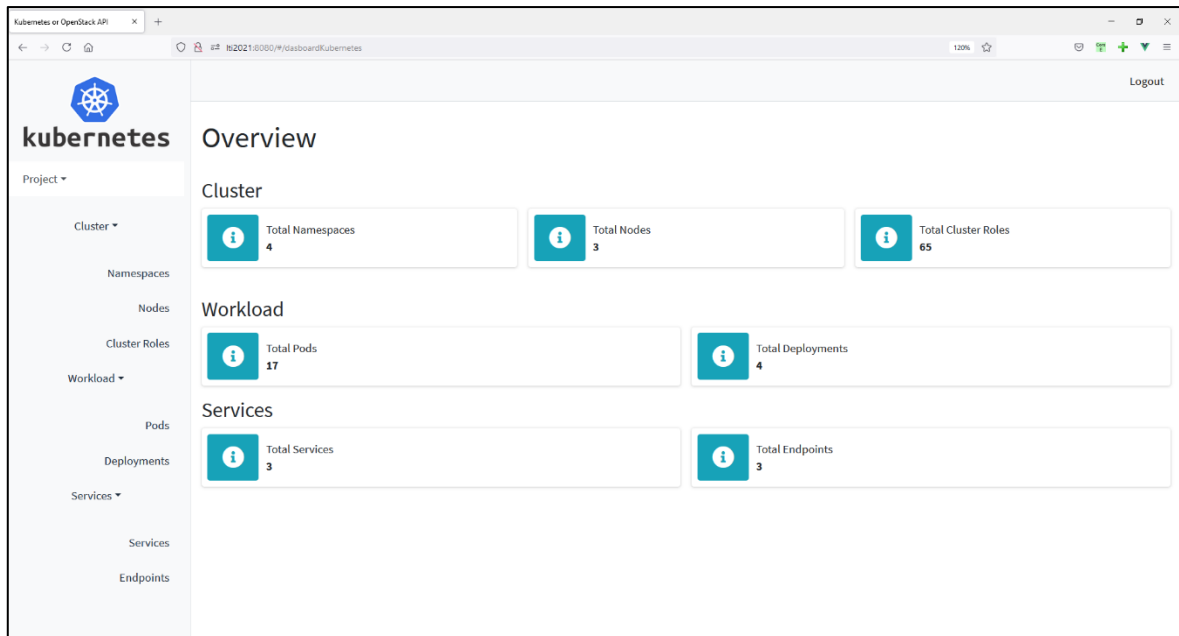
#### 4.4. Componente dashboard

A primeira página que o utilizador irá ter, após iniciar login na aplicação, é um pequeno dashboard onde são apresentadas algumas estatísticas sobre a aplicação *Kubernetes* que está a responder aos pedidos. Assim, e através do componente *dashboard.vue*, é possível visualizar a quantidade de *namespaces*, *nodes* e *cluster roles* existentes no componente *cluster*. Relativamente ao componente *Workload* é possível visualizar a quantidade de *pods* e *deployments* existentes independentemente do *namespace* selecionado. Por fim na parte dos *services*, também é possível visualizar a quantidade de *services* e *endpoints* independentemente do *namespace* selecionado (figura 8).

Neste componente são utilizados os pedidos RESTFull presentes na tabela 1.

Método	URI	Descrição
GET	/api/v1/namespaces	Listar todos os <i>namespaces</i>
GET	/api/v1/nodes	Listar todos os <i>nodes</i>
GET	/apis/rbac.authorization.k8s.io/v1/clusterroles	Listar todos os <i>cluster roles</i>
GET	/api/v1/pods	Listar todos os <i>pods</i>
GET	/apis/apps/v1/deployments	Listar todos os <i>deployments</i>
GET	/api/v1/services	Listar todos os <i>services</i>
GET	/api/v1/endpoints	Listar todos os <i>endpoints</i>

Tabela 1 - Lista de todos pedidos RESTFull utilizados no componente *dashboard.vue*

Figura 8 - Vista do componente *dashboard.vue*

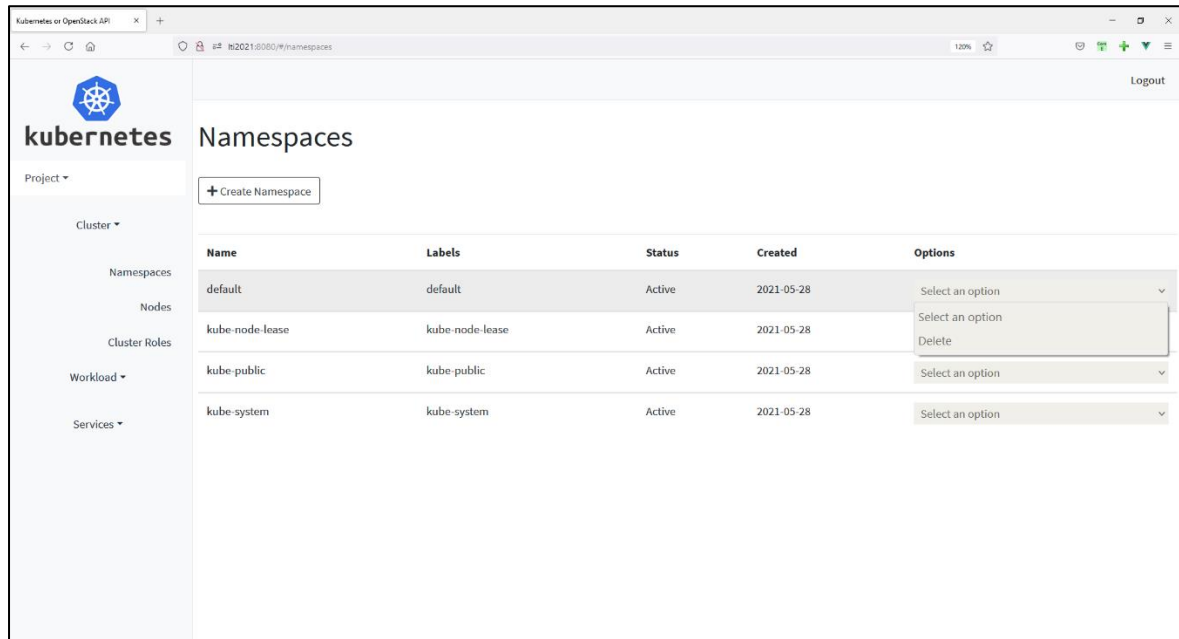
#### 4.5. Componente *namespaces*

No âmbito do orquestrador de containers *Kubernetes*, os *namespaces* representam cluster's virtuais que são criados dentro de um cluster físico. Por defeito, vem quatro *namespaces* iniciais: *default*: é o *namespace* padrão; *kube-system*: é o *namespace* para objetos criados pelo sistema *Kubernetes*; *kube-public*: é o *namespace* reservado, principalmente, para o uso do *cluster*, caso alguns recursos fiquem visíveis em todo o *cluster*; *kube-node-lease*: *namespace* utilizado para objetos associados a cada *node* para melhorar a performance do *node* à medida que o *cluster* vai aumentando.

O componente *namespaces.vue* permite listar todos os *namespaces* existentes no *cluster*, sendo apresentado ao utilizador o nome e *labels* do *namespace*, o seu estado e a data de criação (figura 10) bem como criar (figura 9) ou eliminar *namespaces* existentes.

Neste componente são utilizados os pedidos RESTFull presentes na tabela 2.

Método	URI	Descrição
GET	/api/v1/namespaces	Listar todos os <i>namespaces</i>
DELETE	/api/v1/namespaces/{namespace}	Eliminar um <i>namespace</i> específico
POST	/api/v1/namespaces	Criar um <i>namespace</i>

Tabela 2 - Lista de todos pedidos RESTFull utilizados no componente *namespaces.vue*Figura 10 - Lista de todos os *namespaces*

Create Namespace

Name

Label

✕ Cancel

Create Namespace

Figura 9 - Formulário para criação de um novo *namespace*

## 4.6. Componente *nodes*

No âmbito do orquestrador de containers *Kubernetes*, os *nodes* colocam os *containers* em *pods* para serem executados em *nodes*. Um *node*, pode ser uma máquina virtual ou física, dependendo do *cluster*.

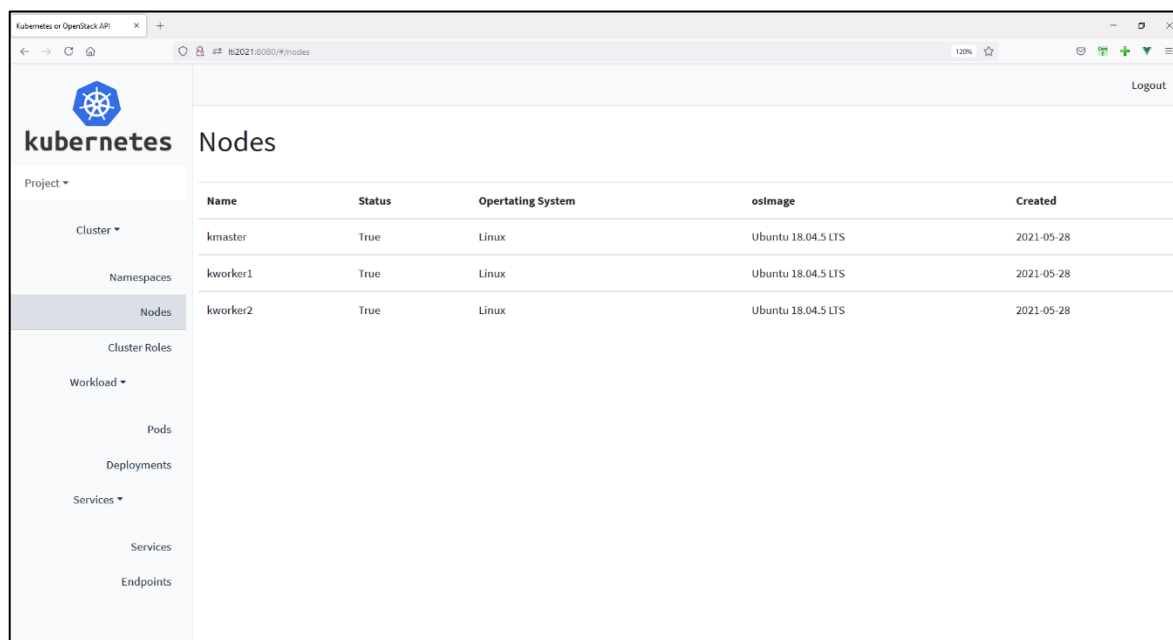


O componente *nodes.vue* é responsável por listar todos os *nodes* existentes no sistema *Kubernetes*. Nessa listagem é possível visualizar o nome do *node*, o seu estado, o sistema operativo que possui bem como a imagem utilizada e a data de criação (figura 11).

Neste componente são utilizados os pedidos RESTFull presentes na tabela 3.

Método	URI	Descrição
GET	/api/v1/nodes	Listar todos os <i>nodes</i>

Tabela 3 - Lista de todos pedidos RESTFull utilizados no componente *nodes.vue*



Name	Status	Operating System	osImage	Created
kmaster	True	Linux	Ubuntu 18.04.5 LTS	2021-05-28
kworker1	True	Linux	Ubuntu 18.04.5 LTS	2021-05-28
kworker2	True	Linux	Ubuntu 18.04.5 LTS	2021-05-28

Figura 11 - Lista de todos os *nodes*

## 4.7. Componente Cluster Roles

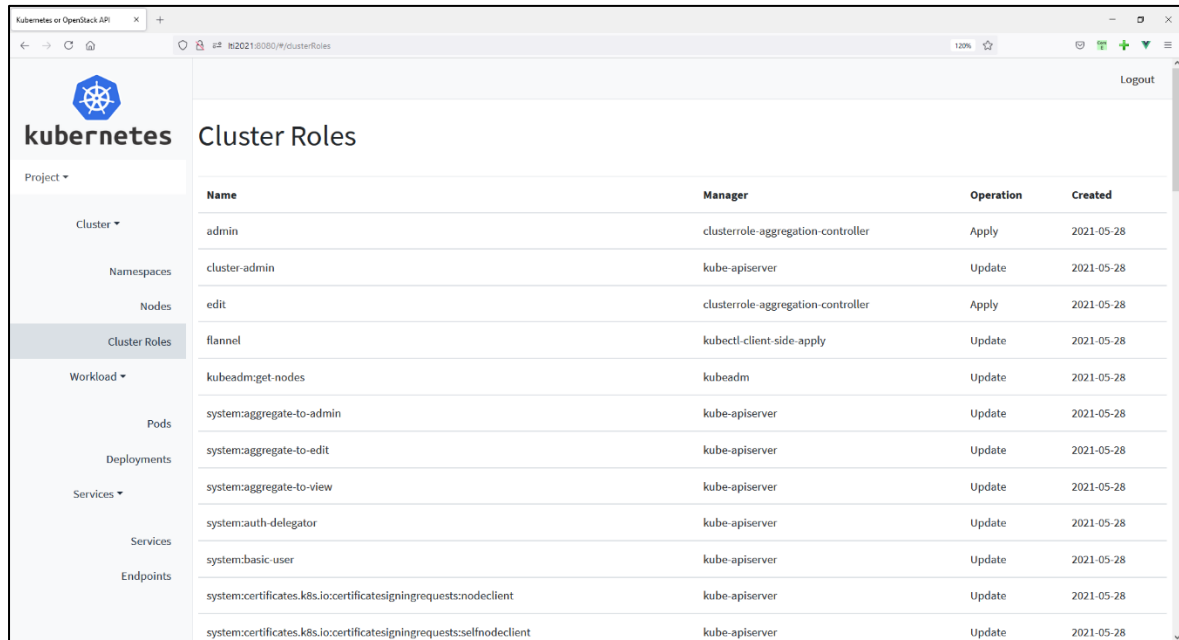
No âmbito do orquestrador de containers *Kubernetes*, o *cluster roles* contém regras que representam um conjunto de permissões aditivas, ou seja, não existe regras com significado de *deny*. Um *cluster role*, pode ser utilizado para definir permissões em recursos com *namespaces* e ser concedido dentro de *namespaces* individuais, em todos os *namespaces* ou num respetivo *cluster*.

No componente *clusterRoles.vue* são apresentados todos os *cluster's roles* existentes, nomeadamente o nome, o *manager* associado, o tipo de operação a que se encontra associado bem como a data de criação (figura 12).

Neste componente são utilizados os pedidos RESTFull presentes na tabela 4.

Método	URI	Descrição
GET	/apis/rbac.authorization.k8s.io/v1/clusterroles	Listar todos os cluster roles

**Tabela 4 - Lista de todos pedidos RESTFull utilizados no componente *clusterRoles.vue***



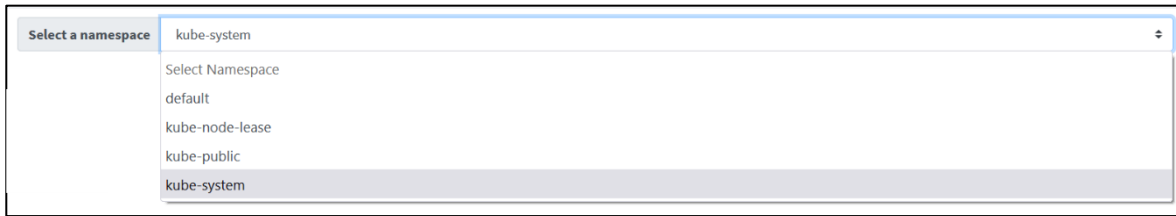
Name	Manager	Operation	Created
admin	clusterrole-aggregation-controller	Apply	2021-05-28
cluster-admin	kube-apiserver	Update	2021-05-28
edit	clusterrole-aggregation-controller	Apply	2021-05-28
flannel	kubectl-client-side-apply	Update	2021-05-28
kubeadm:get-nodes	kubeadm	Update	2021-05-28
system:aggregate-to-admin	kube-apiserver	Update	2021-05-28
system:aggregate-to-edit	kube-apiserver	Update	2021-05-28
system:aggregate-to-view	kube-apiserver	Update	2021-05-28
system:auth-delegator	kube-apiserver	Update	2021-05-28
system:basic-user	kube-apiserver	Update	2021-05-28
system:certificates.k8s.io:certificatesigningrequests:nodesclient	kube-apiserver	Update	2021-05-28
system:certificates.k8s.io:certificatesigningrequests:selfnodesclient	kube-apiserver	Update	2021-05-28

**Figura 12 - Lista de todos os *cluster's roles***

## 4.8. Componente *pods*

No âmbito do orquestrador de containers *Kubernetes*, os *pods* representam um grupo de um ou mais *containers* num único *node*. Todos os *containers* presentes num *pod* partilham o mesmo endereço IP e *hostname*. Cada *pod* isola a rede dos *containers* dos restantes *pods*.

No componente *pods.vue* são apresentados todos os *pods* existentes consoante um determinado *namespace*. Uma vez que existem diferentes *pods* consoante o *namespace* selecionado, foi implementado um componente *select* de forma que sejam apresentados todos os *namespaces* existentes podendo o utilizador selecionar qual é o que pretende (figura 13).



**Figura 13 - Namespaces possíveis para seleção**

Neste componente, existe ainda a possibilidade de listar todos os *pods* existentes, mostrando o nome, o *node* em que está inserido, o endereço IP do host e do *pod*, o seu estado, a classe de *QoS* a que pertence bem como a data da sua criação (figura 14).

Além disso, é possível visualizar quais os portos que estão em uso pelo *pod* (figura 15) bem como informações relativas ao *container* (*resources* e *volumes*) (figura 16). Por fim, existe a possibilidade de eliminar um *pod* existente bem como criar um novo *pod*, através do preenchimento de um formulário em que é necessário o utilizador preencher qual o nome do *pod* (apenas são permitidos nomes em letras minúsculas) e qual o *namespace* a que se pretende associar o *pod* criado (figura 17).

Neste componente são utilizados os pedidos RESTFull presentes na tabela 5.

Name	Node	Host IP	Pod IP	Status	QoS Class	Created	Options
coredns-558bd4d5db-1ct6k	kmaster	10.0.2.7	10.244.0.2	Running	Burstable	2021-05-28	Select an option
coredns-558bd4d5db-vnf56	kmaster	10.0.2.7	10.244.0.3	Running	Burstable	2021-05-28	Select an option
dep-5dbbcf98c-vt6h4	kworker1	10.0.2.8	10.244.1.6	Running	Guaranteed	2021-06-08	Select an option
etcd-kmaster	kmaster	10.0.2.7	10.0.2.7	Running	Burstable	2021-05-28	Select an option
kube-apiserver-kmaster	kmaster	10.0.2.7	10.0.2.7	Running	Burstable	2021-06-08	Select an option
kube-controller-manager-kmaster	kmaster	10.0.2.7	10.0.2.7	Running	Burstable	2021-06-07	Select an option
kube-flannel-ds-7mlp2	kworker2	10.0.2.9	10.0.2.9	Running	Burstable	2021-06-08	Select an option
kube-flannel-ds-7v557	kworker1	10.0.2.8	10.0.2.8	Running	Burstable	2021-06-08	Select an option

**Figura 14 - Lista de todos os pods**

Used Ports		
Name	Port	Protocol
Dns	53	UDP
Dns-tcp	53	TCP
Metrics	9153	TCP
<span>✕ Cancel</span>		

Figura 15 - Portos utilizados por um determinado *pod*

Containers

Resources

Name	Image	CPU Request	Memory Request
coredns	k8s.gcr.io/coredns /coredns:v1.8.0	100m	70Mi

Volumes

Name	Path	Read Only
config-volume	/etc/coredns	True
kube-api-access-zjr25	/var/run/secrets/kubernetes.io /serviceaccount	True

✕ Cancel

Figura 16 - Informações sobre *resources* e volumes de um *pod*

O formulário 'Create Pod' possui um título 'Create Pod' no topo direito. Abaixo dele, há um campo 'Name' com a instrução 'Use only lowercase letters' em vermelho. O campo de entrada para o nome contém o texto 'Name of pod'. Abaixo disso, há um campo 'Namespace' com um menu suspenso que mostra 'kube-system'. No rodapé do formulário, há dois botões: 'Cancel' com um ícone de X e 'Create Pod' em um botão laranja.

Figura 17 - Formulário para a criação de um *pod*

Método	URI	Descrição
GET	/api/v1/namespaces/{namespace}/pods	Listar todos os <i>pods</i>
DELETE	/api/v1/namespaces/{namespace}/pods/{pod}	Eliminar um <i>pod</i> específico
POST	/api/v1/namespaces/{namespace}/pods	Criar um <i>pod</i>

Tabela 5 - Lista de todos pedidos RESTFull utilizados no componente *pods.vue*

## 4.9. Componente *Deployment*

No âmbito do orquestrador de containers *Kubernetes*, os *deployments* controlam quantas cópias do mesmo *pod* devem estar a ser executadas ao mesmo tempo, no cluster.

No componente *deployments.vue* é apresentado ao utilizador uma lista de *deployments*, sendo de destacar o nome, *labels*, *manager*, os *pods* disponíveis, a imagem utilizada, o limite de memória bem como a data de criação do *deployment* (figura 18). Existe ainda a possibilidade de visualizar os portos que se encontram em uso (figura 19). É também possível, eliminar um *deployment* existente, bem como criar um novo, sendo apenas necessário preencher um formulário com o nome do *deployment*, o *namespace* em que se pretende adicionar, a *label* e a imagem que se pretende utilizar (figura 20).

Tal como acontece no componente *pods.vue*, é possível visualizar os diferentes *deployments* de acordo com *namespace* selecionado, assim, foi implementado um componente *select* de forma que sejam apresentados todos os *namespaces* existentes podendo o utilizador selecionar qual é o que pretende.

Neste componente são utilizados os pedidos RESTFull presentes na tabela 6.

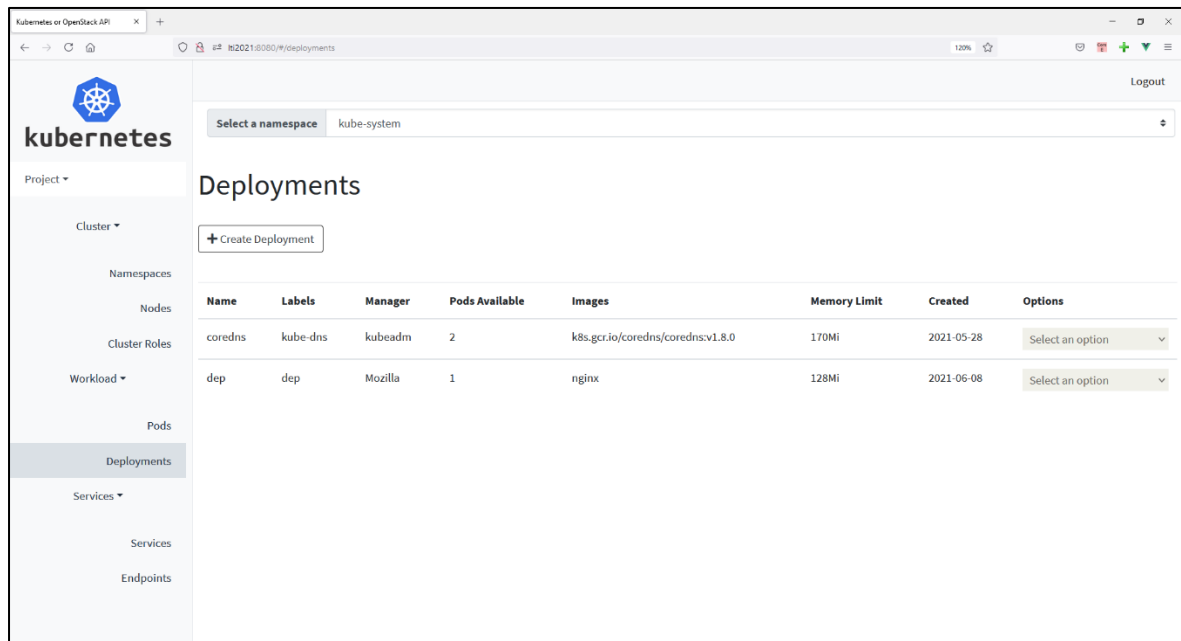


Figura 18 - Lista de todos os *deployments*

Used Ports		
Name	Port	Protocol
Dns	53	UDP
Dns-tcp	53	TCP
Metrics	9153	TCP
Cancel		

Figura 19 - Portos utilizados por um determinado *deployment*

O formulário 'Create Deployment' possui os seguintes campos e elementos:

- Name:** Campo de texto com o placeholder 'Name of deployment'.
- Namespace:** Menu suspenso com 'kube-system' selecionado.
- Label:** Campo de texto com o placeholder 'Label of deployment'.
- Image:** Campo de texto com o placeholder 'Image of deployment'.
- Botões:** 'Cancel' (com ícone de X) e 'Create Deployment' (em laranja).

**Figura 20 - Formulário para a criação de um *deployment***

Método	URI	Descrição
GET	/apis/apps/v1/namespaces/{namespace}/deployments	Listar todos os <i>deployments</i>
DELETE	/apis/apps/v1/namespaces/{namespace}/deployments/{deployment}	Eliminar um <i>deployment</i> específico
POST	/apis/apps/v1/namespaces/{namespace}/deployments	Criar um <i>deployment</i>

**Tabela 6 - Lista de todos pedidos RESTFull utilizados no componente *deployment.vue***

#### 4.10. Componente *Services*

No âmbito do orquestrador de containers *Kubernetes*, os *services* definem um grupo de *pods* e uma política associada a eles. Os *services* funcionam como uma espécie de *proxy* para que, de forma automática, os pedidos que são feitos a um determinado serviço sejam enviados para o *pod* correspondente, independentemente da sua localização no *cluster*.

No componente *services.vue* são apresentados ao utilizador diversos *services* e algumas informações sobre estes como o nome, *manager*, o IP do *cluster*, tipo, *session affinity* e a data de criação (figura 22). Existe, também, a possibilidade de visualizar os portos que estão a ser utilizados por cada *service* (figura 21). Eliminar um *service* ou a possibilidade de criar um novo, sendo necessário preencher o *namespace*, o *deployment* a que pertence, o porto de origem e destino, o tipo de protocolo bem como o nome do protocolo (figura 23) são funcionalidades que também se encontram disponíveis.

Tal como acontece nos componentes *pods.vue* e *deployments.vue*, é possível visualizar os diferentes *services* de acordo com *namespace* selecionado, assim, foi implementado um

componente *select* de forma que sejam apresentados todos os *namespaces* existentes podendo o utilizador seleccionar qual é o que pretende.

Neste componente são utilizados os pedidos RESTFull presentes na tabela 7.

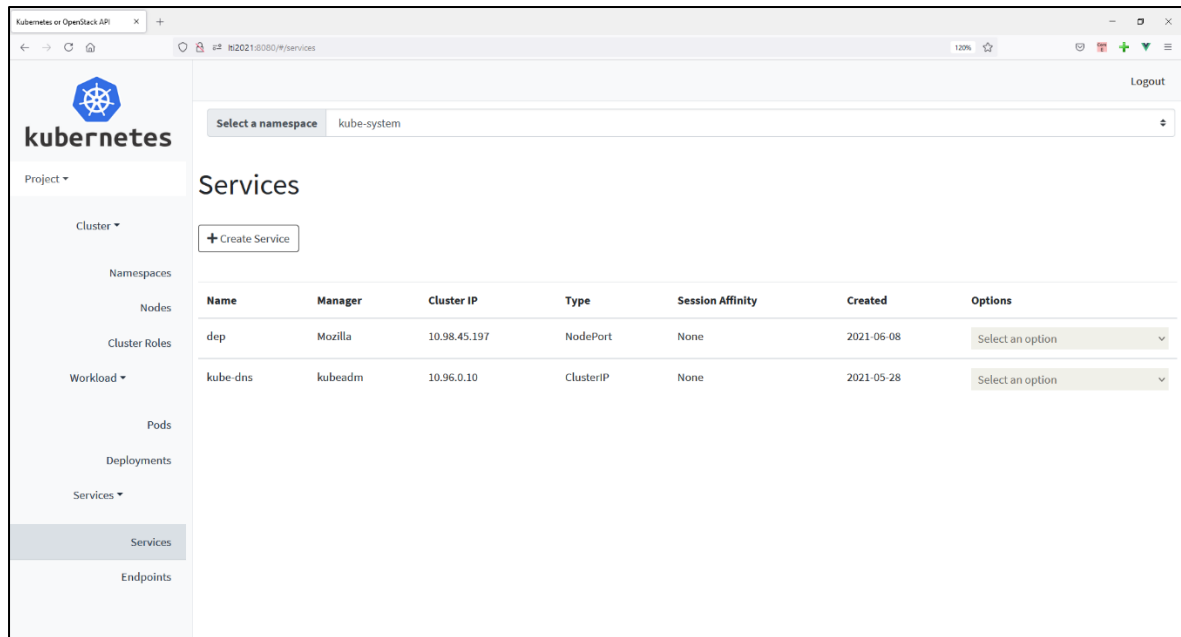


Figura 22 - Lista de todos os *services*

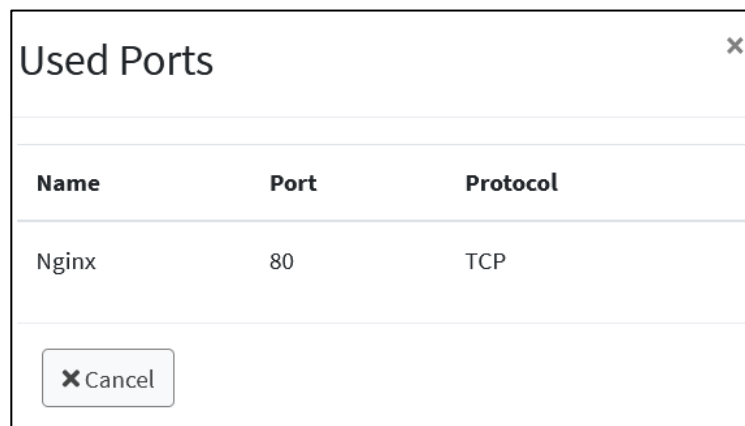


Figura 21 - Portos utilizados por um determinado *service*



O formulário 'Create Service' apresenta os seguintes campos:

- Namespace:** Dropdown menu com 'kube-system' selecionado.
- Deployment:** Dropdown menu com 'Choose a deployment' selecionado.
- Port Source:** Input field com 'Number of port source' e uma seta para cima/descida.
- Port Destination:** Input field com 'Number of port destination' e uma seta para cima/descida.
- Protocol:** Dropdown menu com 'Choose a protocol' selecionado.
- Protocol Name:** Input field com 'Name of protocol'.

Botões de ação: 'Cancel' (com ícone de X) e 'Create Deployment' (em laranja).

Figura 23 - Formulário para a criação de um *service*

Método	URI	Descrição
GET	/api/v1/namespaces/{namespace}/services	Listar todos os <i>services</i>
DELETE	/api/v1/namespaces/{namespace}/services/{service}	Eliminar um <i>service</i> específico
POST	/api/v1/namespaces/{namespace}/services	Criar um <i>service</i>

Tabela 7 - Lista de todos pedidos RESTFull utilizados no componente *services.vue*

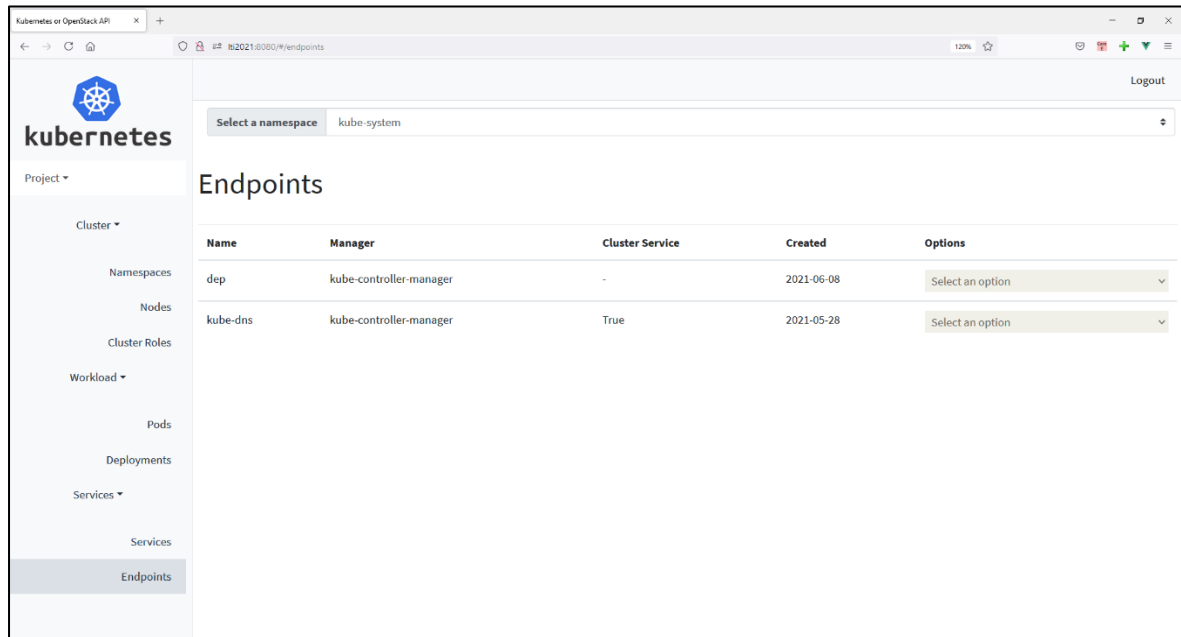
#### 4.11. Componente *endpoints*

No componente *endpoints.vue* são apresentadas informações ao utilizador diversos de *endpoints* existentes, mais especificamente o nome, *manager*, *cluster service* e a data de criação (figura 26). É também possível visualizar os portos que se encontram em uso (figura 25) bem como o *subset address* utilizado (figura 27).

Tal como acontece em outros componentes desenvolvidos, é possível visualizar os diferentes *endpoints* de acordo com *namespace* selecionado, assim, foi implementado um

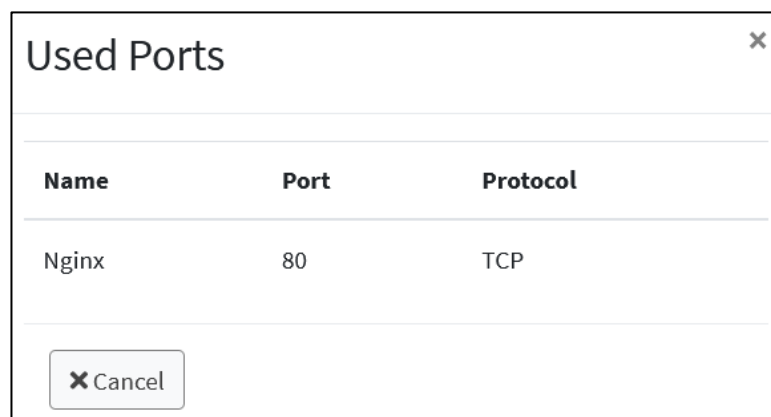
componente *select* de forma que sejam apresentados todos os *namespaces* existentes podendo o utilizador seleccionar qual é o que pretende.

Neste componente são utilizados os pedidos RESTFull presentes na tabela 8.



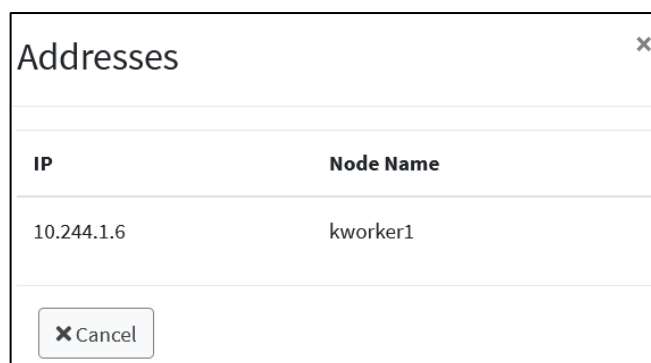
Name	Manager	Cluster Service	Created	Options
dep	kube-controller-manager	-	2021-06-08	Select an option
kube-dns	kube-controller-manager	True	2021-05-28	Select an option

Figura 25- Lista de todos os *endpoints*



Name	Port	Protocol
Nginx	80	TCP

Figura 24 - Portos utilizados por um determinado *endpoint*



IP	Node Name
10.244.1.6	kworker1

Figura 26 - *Subset address* de um determinado *endpoint*

Método	URI	Descrição
GET	/api/v1/namespaces/{namespace}/endpoints	Listar todos os <i>endpoints</i>

Tabela 8 - Lista de todos pedidos *RESTFull* utilizados no componente *endpoint.vue*

## 4.12. Dificuldades encontradas

Durante a realização deste trabalho laboratorial, deparamo-nos com alguns problemas que nos levaram algum tempo a resolver.

Após todo o cenário se encontrar implementado e começando a fazer os primeiros pedidos *RESTFull*, nada era devolvido após o pedido *axios*. Resolveu-se experimentar fazer o mesmo pedido através do browser e aí já existia resposta. Após alguma pesquisa, e tentativas de resolução a única opção viável para resolver o nosso problema foi editar o ficheiro *kube-apiserver.yaml* presente na diretoria */etc/kubernetes/manifests/* e adicionar a entrada presente na figura 28.

```
spec:
  containers:
  - command:
    - kube-apiserver
    - --cors-allowed-origins=http://*
```

Figura 27 - Ficheiro *kube-apiserver.yaml*

Relativamente a problemas que obtivemos com o cabeçalho CORS, resolvemos esses problemas com o uso do browser *Firefox* e com a instalação de duas extensões: *CORS Everywhere* e *Cross Domain – CORS*. Verificamos que, ao utilizarmos apenas a extensão *Cross Domain – CORS*, a plataforma web ficava, aparentemente, funcional, porém quando se pretendia eliminar algo, dava um novo erro, desta vez resolvido com o uso da extensão *CORS Everywhere*.

Após resolvidos estes dois problemas iniciais, e depois de já termos visualizado alguns dados apresentados na plataforma web, deixou de existir resposta por parte do servidor web alojado na máquina onde se encontra instalado o *Kubernetes*. Após recorrermos ao comando *netstat -anp | grep 8081*, em que 8081 significa o porto em que o *proxy*, que estava a reencaminhar os pedidos, se encontrava à escuta, visualizamos que existia um serviço a correr. Assim, chegamos à conclusão que por vezes o *proxy* rebenta, sendo necessário terminar o processo e voltar a ativar. Para facilitar a ativação do *proxy*, foi

desenvolvido um *script* utilizando a linguagem *perl*, para que seja apenas necessário correr o *script* e o *proxy* ficar novamente ativo. Na figura 29, é possível visualizar o *script*, onde se vai procurar o *pid* de um serviço que se encontre à escuta no porto 8081, depois termina-se esse *pid* recorrendo ao comando *kill*, e volta-se a correr o comando onde se ativa o *proxy*.

```
#!/usr/bin/perl
use strict;
use warnings;

my $pid = `netstat -anp | grep :8081 | tr -s ' ' ':' | cut -d: -f7 | cut -d/ -f1`;

system("kill $pid");
system("kubectl proxy --port=8081 --address=0.0.0.0 --accept-hosts=^* &");

exit(0);
```

Figura 28 - Script em *perl* para auxiliar na ativação do *proxy*

### 4.13. URIs utilizados

Ao longo deste relatório já foi possível visualizar os URIs utilizados em cada componente. Nesta secção é apresentada a tabela 9, onde são apresentados todos os URIs necessários para a elaboração deste trabalho laboratorial.

Método	URI	Descrição
DELETE	<code>api/v1/namespaces/{namespace}</code>	Eliminar um <i>namespace</i> específico
DELETE	<code>/api/v1/namespaces/{namespace}/pods/{pod}</code>	Eliminar um <i>pod</i> específico
DELETE	<code>/apis/apps/v1/namespaces/{namespace}/deployments/{deployment}</code>	Eliminar um <i>deployment</i> específico
DELETE	<code>/api/v1/namespaces/{namespace}/services/{service}</code>	Eliminar um <i>service</i> específico
GET	<code>/api/v1/namespaces</code>	Listar todos os <i>namespaces</i>
GET	<code>/api/v1/nodes</code>	Listar todos os <i>nodes</i>
GET	<code>/apis/rbac.authorization.k8s.io/v1/clusterroles</code>	Listar todos os cluster roles
GET	<code>/api/v1/namespaces/{namespace}/pods</code>	Listar todos os <i>pods</i>
GET	<code>/apis/apps/v1/namespaces/{namespace}/deployments</code>	Listar todos os <i>deployments</i>
GET	<code>/api/v1/namespaces/{namespace}/services</code>	Listar todos os <i>services</i>
GET	<code>/api/v1/namespaces/{namespace}/endpoints</code>	Listar todos os <i>endpoints</i>
GET	<code>/api/v1/namespaces</code>	Listar todos os <i>namespaces</i>
GET	<code>/api/v1/nodes</code>	Listar todos os <i>nodes</i>
GET	<code>/apis/rbac.authorization.k8s.io/v1/clusterroles</code>	Listar todos os <i>cluster roles</i>
GET	<code>/api/v1/pods</code>	Listar todos os <i>pods</i>
GET	<code>/apis/apps/v1/deployments</code>	Listar todos os <i>deployments</i>
GET	<code>/api/v1/services</code>	Listar todos os <i>services</i>
GET	<code>/api/v1/endpoints</code>	Listar todos os <i>endpoints</i>
POST	<code>/api/v1/namespaces</code>	Criar um <i>namespace</i>
POST	<code>/api/v1/namespaces/{namespace}/pods</code>	Criar um <i>pod</i>
POST	<code>/apis/apps/v1/namespaces/{namespace}/deployments</code>	Criar um <i>deployment</i>

Tabela 9 - Lista de todos pedidos RESTFull utilizados

## 5. Análise crítica e proposta de melhorias

Ao longo de todo o trabalho laboratorial, tivemos como objetivo implementar o que era pedido no enunciado, sendo que podíamos ter investido mais algum tempo no design da aplicação web.

Num trabalho desta natureza, existe sempre algo que se possa melhorar, sendo que poderia ter sido desenvolvido ferramentas de *searching* e filtros para as tabelas onde se encontram listados os *namespaces*, *Pods*, *cluster roles*, *nodes*, *deployments*, *services* e *endpoints*.

Um outro ponto que pode vir a ser melhorado no futuro, é aprofundar a parte de *config* e *storage*, uma vez que é uma parte que não se encontra disponibilizada na aplicação que desenvolvemos.

Mesmo com as melhorias que podiam ser feitas achámos que o trabalho ficou do nosso agrado, sendo que apresenta as funcionalidades básicas do orquestrador de *containers* *Kubernetes* mesmo sabendo que poderíamos integrar mais ferramentas úteis.

## 6. Conclusão

Terminada a elaboração do trabalho laboratorial, podemos afirmar que os objetivos inicialmente propostos através do enunciado se encontram funcionais, além de termos implementado mais funcionalidades de forma a enriquecer este trabalho.

Reconhecemos que há aspetos que podem ser melhorados, como foi falado anteriormente, mas há que referir que este projeto proporcionou uma grande aprendizagem.

Com a realização deste trabalho, tivemos conhecimento de uma realidade que começamos a tomar conhecimento com a elaboração do trabalho laboratorial 1, e concluímos que o orquestrador de containers *kubernetes* apresenta bastantes vantagens na parte de gestão dos *containers* e o desenvolvimento de uma aplicação deste género acaba por facilitar a manutenção da aplicação *Kubernetes* uma vez que possibilita fazer de forma gráfica o que tinha de ser feito via linha de comandos.

## Bibliografia

- [1] A. T. F. B. Ramon Acedo, “Run Your Kubernetes Cluster on OpenStack in Production,” 16 Março 2021. [Online]. Available: <https://superuser.openstack.org/articles/run-your-kubernetes-cluster-on-openstack-in-production/>.
- [2] “Usando Minikube para criar um cluster,” 19 Abril 2021. [Online]. Available: <https://kubernetes.io/pt-br/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/>.
- [3] J. MSV, 19 Agosto 2016. [Online]. Available: <https://thenewstack.io/taking-kubernetes-api-spin/>.
- [4] “Access Clusters Using the Kubernetes API,” 09 Março 2021. [Online]. Available: <https://kubernetes.io/docs/tasks/administer-cluster/access-cluster-api/>.
- [5] “API OVERVIEW,” 3 Abril 2020. [Online]. Available: <https://v1-18.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.18/>.
- [6] “Welcome!,” 6 Maio 2021. [Online]. Available: <https://minikube.sigs.k8s.io/docs/>.