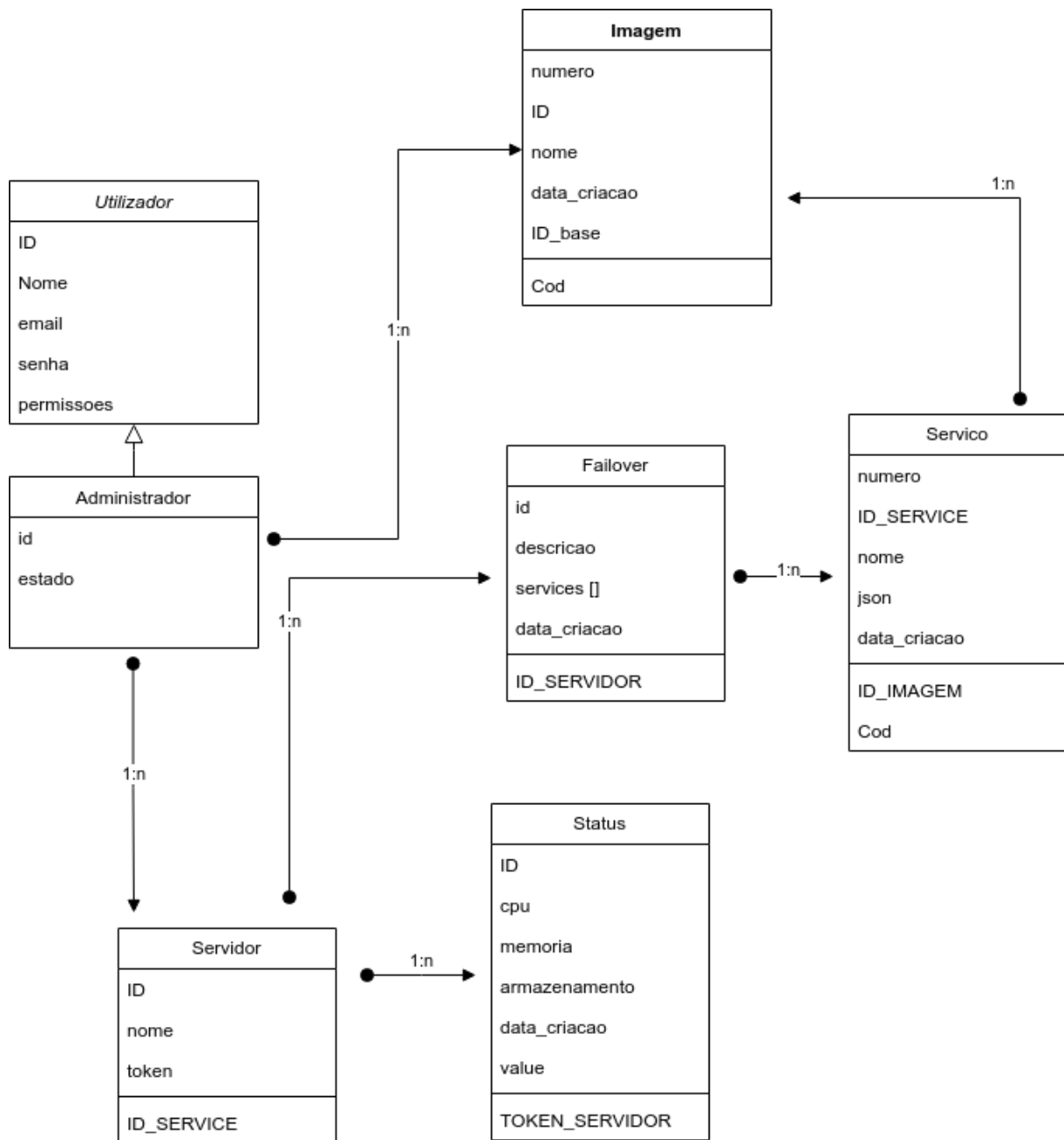


Modelos do sistema

Diagrama de entidades

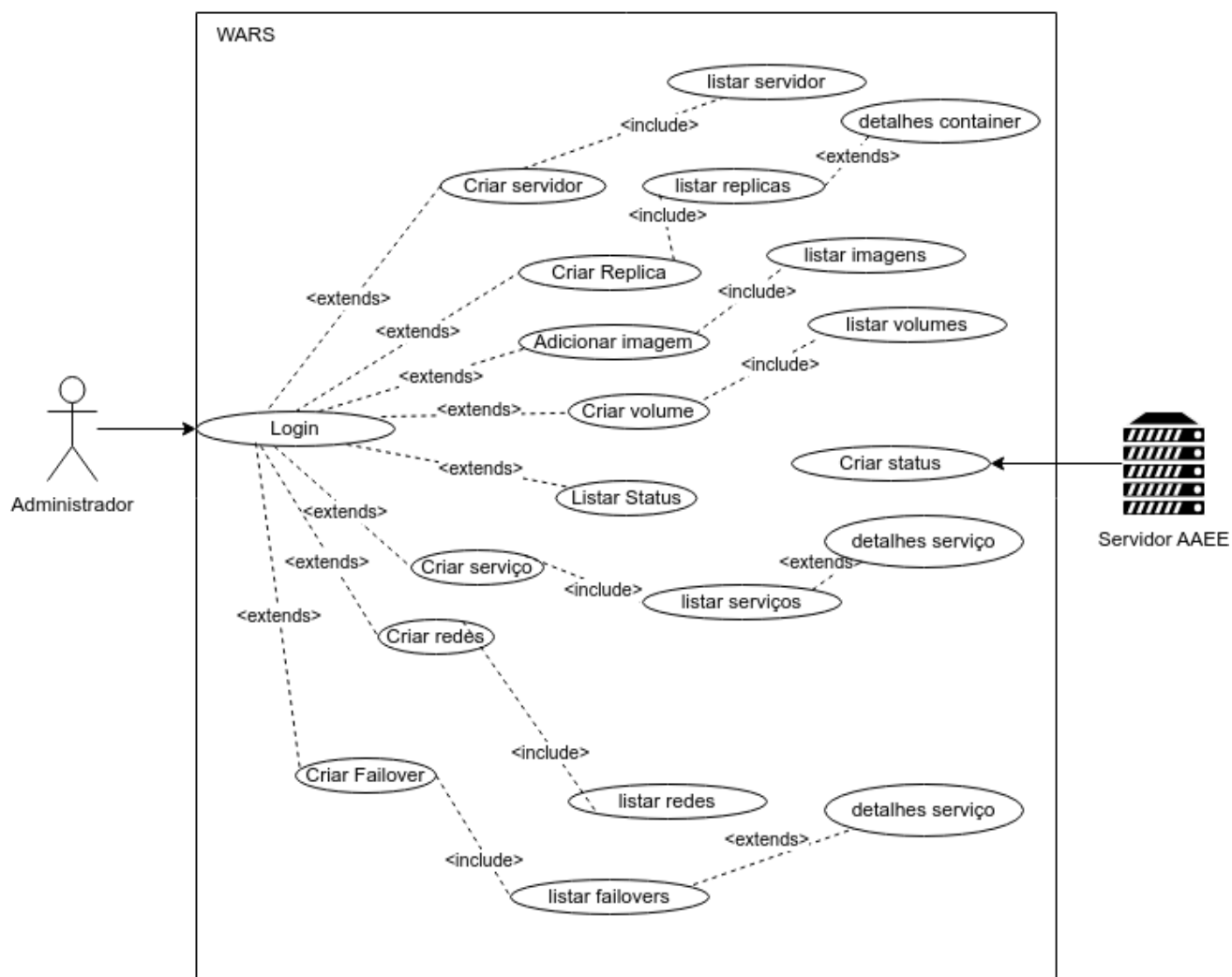
Figura A 1 - Diagrama de entidade relacionamento



Fonte: Autor (2023)

Diagrama de Caso de uso

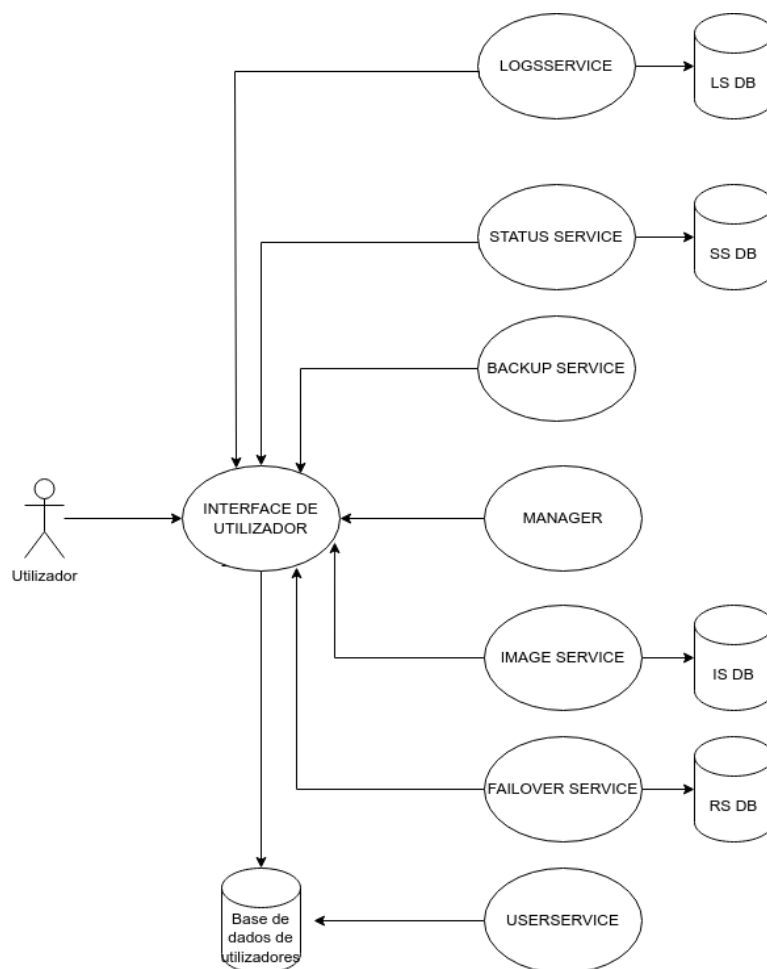
Figura A 2 - Diagrama de caso de uso



Fonte: Autor (2023)

Arquitetura do sistema de replicação

Figura A 3 - Arquitetura de microserviços do sistema



Fonte: Autor (2023)

A figura anterior mostra a arquitetura proposta para o sistema de replicação. Na arquitetura proposta, o utilizar interage com a interface do utilizador que, por sua vez, interage com os microserviços, ou seja, cada funcionalidade da interface do utilizador é mapeada para um microserviço.

Interface de utilizador – Os utilizadores do sistema irão interagir com um sistema por meio de uma aplicação web que será usada através do navegador. É a partir da interface do utilizador que os comandos serão enviados para os microserviços.

Statusservice – será o microserviço responsável pelo monitoramento dos servidores e criação dos servidores monitorados. É por meio dele que será possível saber se um determinado servidor monitorado encontra-se *offline*.

Backupservice – será o microserviço responsável por realizar a cópia de dados de um servidor para um volume. É por meio dele que se serão criados os volumes e containers de sincronização;

Manageservice - será o microserviço que faz a gestão do cluster e de toda infraestrutura, cria os serviços e redes do cluster.

Imageservice - fará a gestão de imagens do *registry*, criará containers de aplicação, guardará containers como imagens, enviará imagens ao *registry*.

Failoverservice - será por meio dele que se irá configurar o *failover*. Este microserviço será responsável por activar os serviços quando a falha no servidor acontecer.

Userservice - este microserviço será responsável pela gestão de utilizadores.

Logsservice – será responsável por guardar todos os eventos do sistema.

Apêndice B – Configuração dos servidores

Configuração do cluster

O cluster vai garantir que a aplicação seja executada em mais de uma máquina. Para o funcionamento do sistema de replicação, cada máquina nó do cluster precisa ter as seguintes configurações mínimas:

- Processador: 2.5GHz x 2;
- Memória: 2 GB (4 GB no nó Master);
- Armazenamento: 28 GB;
- Sistema Operativo: Ubuntu Server 22.04;
- Docker 24.0.2 ou superior.

O cluster deve ser composto por pelo menos 3 máquinas (podem ser virtuais ou físicas) onde uma delas será o nó mestre.

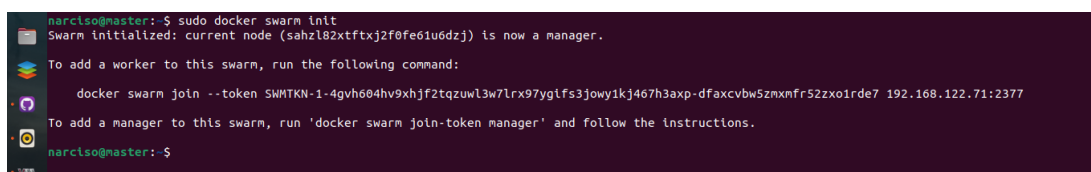
No nó mestre, devem ser feitas as seguintes configurações:

```
mestre$> docker swarm init
```

Com o comando acima, a máquina esta sendo iniciada como nó Mestre.

Resultados:

Figura B 1 - Inicialização do cluster



```
narciso@master:~$ sudo docker swarm init
Swarm initialized: current node (sahzl82xtftxj2f0fe61u6dzj) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-4gvh604hv9xhjf2tqzuwl3w7lrx97ygifs3jowy1kj467h3axp-dfaxcvbw5zmxnfr52zxo1rde7 192.168.122.71:2377

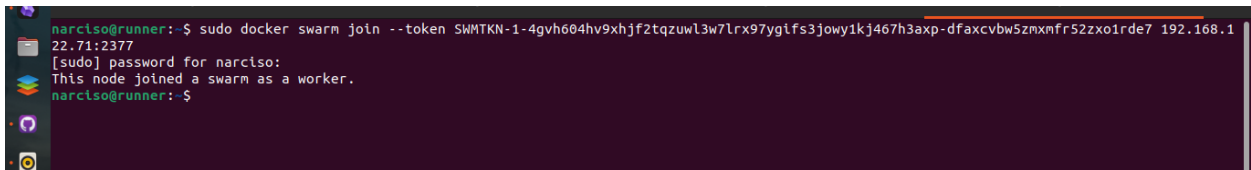
To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

narciso@master:~$
```

Após executar o comando, é retornado um conjunto de informações e, dentre elas existe o comando que deve ser executado nos outros nós do cluster.

Nos outros nós do cluster, deve se executar o comando para a junção no cluster:

Figura B 2 - Inicialização dos nós do cluster



```
narciso@runner:~$ sudo docker swarm join --token SWMTKN-1-4gvh604hv9xhjf2tqzuwl3w7lrx97ygifs3jowy1kj467h3axp-dfaxcvbw5zmxmfr52zxo1rde7 192.168.1.22.71:2377
[sudo] password for narciso:
This node joined a swarm as a worker.
narciso@runner:~$
```

Configuração do Registrador Privado

Para que as imagens sejam executadas no cluster é necessário que elas estejam em um registrador privado por onde os nós do cluster irão buscar a imagem.

O registrador privado é acessado por meio de requisições HTTP, então antes de mais nada, é necessário configurar a criptografia na ligação (HTTPS) e usar autenticação para garantir a segurança do registrador. Para isso será usado o Nginx²³ para servir de proxy do registrador e, os mecanismos de segurança serão configurados nele.

```
Registrador$> mkdir /etc/nginx/certificate
```

```
Registrador$> cd /etc/nginx/certificate
```

```
Registrador$> openssl req -new -newkey rsa:4096 -x509 -sha256 -days 365 -nodes -out nginx-certificate.crt -keyout nginx.key
```

Depois de criar as chaves deve-se adicionar a seguinte configuração no ficheiro “**sudo nano /etc/nginx/sites-available/default**”

```
server {
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;
    ssl_certificate /etc/nginx/certificate/nginx-certificate.crt;
    ssl_certificate_key /etc/nginx/certificate/nginx.key;
    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;
    server_name _;

    location / {
        if ($http_user_agent ~ "^(dockerV1\.(3|4|5(?:?![0-9]-dev))|Go).*$" ) {
            return 404;
        }
        proxy_pass http://localhost:5000;
```

²³ **Nginx** – é um servidor proxy da camada de aplicação.

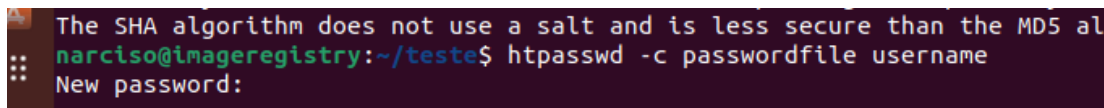
```

proxy_set_header Host      $http_host; # required for docker client's sake
proxy_set_header X-Real-IP  $remote_addr; # pass on real client's IP
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_read_timeout         900;
}
}

```

O passo seguinte é configurar as credencias do registrador. As credencias são configuradas com o `htpasswd`.

Figura B 3 - Criação de Credências



Após configurar as credencias do registrador, é necessário criar um ficheiro com o nome `docker-compose.yml` por onde o registrador privado será criado e serão passadas as credencias recentemente criadas.

```

version: '3'
services:
  registry:
    restart: always
    image: registry:latest
    ports:
      - "5000:5000"
    environment:
      REGISTRY_AUTH: htpasswd
      REGISTRY_AUTH_HTPASSWD_REALM: Registry
      REGISTRY_AUTH_HTPASSWD_PATH: /auth/registry.password
      REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY: /data
    volumes:
      - ./auth:/auth
      - ./data:/data

```

O parâmetro `REGISTRY_AUTH_HTPASSWD_PATH` deve ser substituído pelo caminho do ficheiro de *password* criado anteriormente.

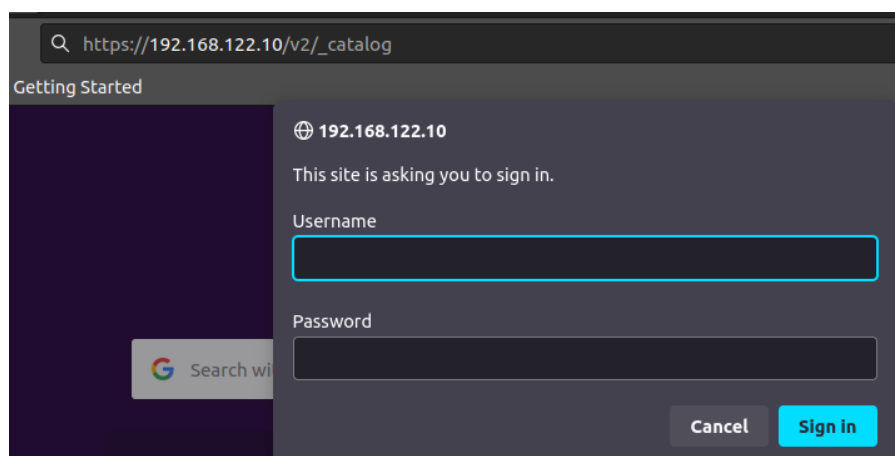
Apos o ficheiro *docker-compose.yml* (`docker-compose up -d`) for executado, será criado o container do registrador:

Figura B 4 - Criação do Container do Registrador

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
8dfsbe01928b	registry:latest	"/entrypoint.sh /etc..."	5 weeks ago	Up 25 minutes	0.0.0.0:5000->5000/tcp, :::5000->5000/tcp	docker-regis

O registrador já foi criado e ele pode ser acedido pelo IP da máquina.

Figura B 5 - Verificando o funcionamento do Registrador



Após instalar o registrador, o comando **sudo docker login <https://192.168.122.10>** deve ser executado em todos nós do cluster de modo a autentica-los no registrador privado. Executando o comando anterior, haverá um erro por causa do certificado digital que no caso é auto-assinado (em um ambiente de produção, o erro não acontecerá com certificados legítimos). Para superar o erro é necessário adicionar a seguinte configuração no ficheiro */etc/docker/daemon.json* (é necessário criar o ficheiro caso ele não existe) presente em todos os nós do cluster:

```
{
  "insecure-registries" : ["https://192.168.122.10"]
}
```

Feito isso, todos os nós do cluster poderão usar as imagens do registrador.

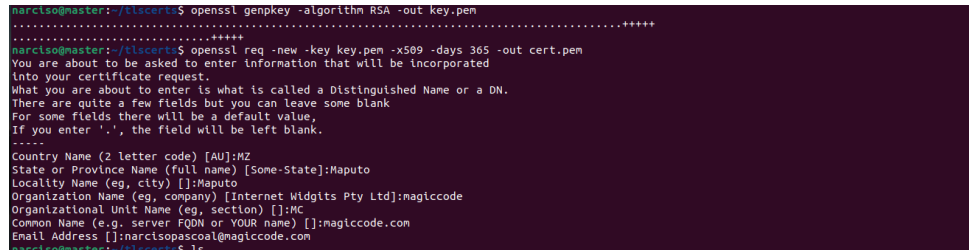
Configurando Docker API

O *docker* API vai possibilitar a manipulação remoto dos containers. A configuração do *docker* API é a base do funcionamento do presente sistema de replicação. Para o funcionamento do sistema de replicação é necessário configurar o *docker* api no nó mestre.

Gerar os certificados para o *docker*:


```
mestre$> mkdir tlscerts
mestre$> cd tlscerts
mestre$> openssl genpkey -algorithm RSA -out key.pem
mestre$> openssl req -new -key key.pem -x509 -days 365 -out cert.pem
```

Figura B 6 - Geração de Chaves



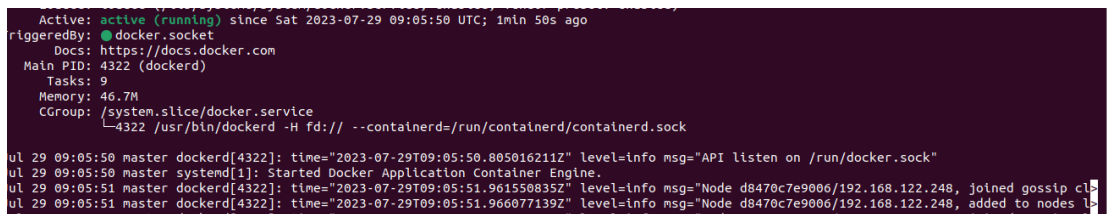
```
narciso@master: /tlscerts$ openssl genpkey -algorithm RSA -out key.pem
.....+++++
narciso@master: /tlscerts$ openssl req -new -key key.pem -x509 -days 365 -out cert.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:MZ
State or Province Name (full name) [Some-State]:Maputo
Locality Name (eg, city) []:Maputo
Organization Name (eg, company) [Internet Widgits Pty Ltd]:magiccode
Organizational Unit Name (eg, section) []:MC
Common Name (e.g. server FQDN or YOUR name) []:magiccode.com
Email Address []:narcisopascal@magiccode.com
-----
```

No ficheiro `/etc/docker/daemon.json` do nó mestre, é necessário acrescentar os seguintes atributos:

```
"hosts": ["tcp://localhost:2376", "unix:///var/run/docker.sock"],
"tls": true,
"tlscert": "/home/narciso/tlscerts/cert.pem",
"tlskey": "/home/narciso/tlscerts/key.pem",
"tlsverify": false,
```

Após isso, usando os comandos **sudo systemctl daemon-reload**, **sudo systemctl restart docker**, **sudo dockerd --debug** deve-se reiniciar o *Docker*.

Figura B 7 - Verificando o estado do *docker*



```
Active: active (running) since Sat 2023-07-29 09:05:50 UTC; 1min 50s ago
TriggeredBy: ● docker.socket
Docs: https://docs.docker.com
Main PID: 4322 (dockerd)
Tasks: 9
Memory: 46.7M
CGroup: /system.slice/docker.service
└─4322 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jul 29 09:05:50 master dockerd[4322]: time="2023-07-29T09:05:50.805016211Z" level=info msg="API listen on /run/docker.sock"
Jul 29 09:05:50 master systemd[1]: Started Docker Application Container Engine.
Jul 29 09:05:51 master dockerd[4322]: time="2023-07-29T09:05:51.961550835Z" level=info msg="Node d8470c7e9006/192.168.122.248, joined gossip cluster"
Jul 29 09:05:51 master dockerd[4322]: time="2023-07-29T09:05:51.966077139Z" level=info msg="Node d8470c7e9006/192.168.122.248, added to nodes list"
```

Quando *docker* reiniciar, será exibida a seguinte informação "API listen on "localhost:2376" e, neste momento deve-se configurar o acesso remoto a API usando *nginx*.

No *nginx*, primeiro é necessário criar as credencias usando **sudo htpasswd -c /etc/nginx/.htpasswd narciso**. Após isso, é necessário configurar o *nginx* para trabalhar com permissões de super-user usando o comando **sudo sed -i 's/user .*/user root;/' /etc/nginx/nginx.conf**. Após isso, deve-se configurar o site no *nginx*.

```
sudo tee /etc/nginx/sites-enabled/docker <<EOF
```

```

upstream docker {
    server unix:/var/run/docker.sock;
}

server {
    listen 7766 default_server ssl;
    ssl_certificate /home/narciso/tlscerts/certificado.crt;
    ssl_certificate_key /home/narciso/tlscerts/chave.key;
    location / {
        proxy_pass http://docker;
        auth_basic_user_file /etc/nginx/.htpasswd;
        auth_basic "Access restricted";
    }
}
EOF

```

Os ficheiros `/home/narciso/tlscerts/certificado.crt` e `/home/narciso/tlscerts/chave.key` são os certificados digitais e, devem ser gerados separadamente.

Depois do passo anterior, é necessário reiniciar o *nginx*.

Figura B 8 - Verificando o funcionamento do Docker API

