



Guia de Instalação DefectDojo

Autor: Marcelio Soares Lima

Mentor: Max Eduardo Vizcarra Melgar

Última revisão: Julho de 2025

Sumário


Sumário.....	2
1. Configurações do Ambiente e Pré-requisitos.....	3
1.1 Ambiente utilizado para testes.....	3
1.2 Versões dos componentes.....	4
2. Instalação do Docker.....	4
3. Instalação do DefectDojo.....	7
4. Configurando o DefectDojo para inicializar em conjunto com o Sistema Operacional.....	12
5. Possíveis Problemas que Podem Impedir a Execução do Greenbone.....	14



1. Configurações do Ambiente e Pré-requisitos

Para garantir uma instalação segura, padronizada e compatível com o ambiente proposto, este manual utiliza como base a [instrumentação oficial](#) disponibilizada via GitHub pelos desenvolvedores do DefectDojo. A instalação do Docker será realizada seguindo as instruções publicadas na documentação [oficial do Docker](#).

Esta documentação foi projetada para servir como uma referência padronizada no contexto dos ambientes do PoP (Ponto de Presença da RNP), garantindo uniformidade na instalação da solução de análise de vulnerabilidades em dispositivos que atendam aos requisitos mínimos definidos. Todas as configurações descritas foram pensadas para reduzir complexidade técnica, evitar conflitos comuns de dependências e acelerar o processo de implantação.

 **Nível de dificuldade: intermediário**

Recomenda-se que o executor tenha familiaridade com conceitos de containerização, uso de Docker e Docker Compose, gerenciamento de volumes e leitura de arquivos YAML, controle de permissões e administração de serviços via systemd.

1.1 Ambiente utilizado para testes

A instalação foi realizada em um ambiente controlado com as seguintes características mínimas:

Recurso	Especificação
CPU	4 núcleos físicos
Memória RAM	6 GB
SWAP	8 GB
Armazenamento livre	60 GB
Sistema operacional	Ubuntu 22.04 LTS
Acesso privilegiado	Usuário com <code>sudo</code> habilitado

Atendidos os requisitos descritos, siga para a próxima etapa da instalação.

1.2 Versões dos componentes

Este tutorial foi validado utilizando as seguintes versões dos principais componentes do DefectDojo:

Componente	Versão
Docker	28.3.2
Docker Compose	2.38.2
DefectDojo	2.48.1

2. Instalação do Docker

Passo 1 - Primeiro execute o seguinte comando para remover quaisquer pacotes conflitantes como Docker (Figura 1).

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

```
Reading state information... Done
Package 'docker-compose' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'docker-compose-v2' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'podman-docker' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'containerd' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'runc' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
vagrant@ubuntu2204:~$
```

Figura 1 - Neste caso não há arquivos referentes ao docker instalado e também não possui o docker-engine no repositório.

Passo 2 - Agora execute os comandos para atualizar os repositórios da distro (Figura 2).

```
sudo apt-get update && sudo apt-get upgrade
```

```
Setting up gpg-wks-client (2.2.27-3ubuntu2.4) ...
Setting up gnupg (2.2.27-3ubuntu2.4) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for libc-bin (2.35-0ubuntu3.10) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for install-info (6.8-4build1) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...
systemctl restart gsad.service gvmd.service packagekit.service postgresql@14-main.service systemd-resolved.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
vagrant@ubuntu2204:~$
vagrant@ubuntu2204:~$
```

Figura 2 - Atualizando os componentes da distro.

Passo 3 - Prossiga, instalando os certificados e importando as chaves (Figura 3).

```
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
vagrant@ubuntu2204:~$ sudo apt-get install ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203~22.04.1).
curl is already the newest version (7.81.0-1ubuntu1.20).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
vagrant@ubuntu2204:~$ sudo install -m 0755 -d /etc/apt/keyrings
vagrant@ubuntu2204:~$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
vagrant@ubuntu2204:~$ sudo chmod a+r /etc/apt/keyrings/docker.asc
vagrant@ubuntu2204:~$
```

Figura 3 - Importando os dados do Docker.

Passo 4 - Adicione o repositório ao APT (Figura 4).

```
echo \
  "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable"
| \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

```
vagrant@ubuntu2204:~$ echo \
> "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://downl
d.docker.com/linux/ubuntu \
> $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
> sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
vagrant@ubuntu2204:~$ sudo apt-get update
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Get:2 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [53.5 kB]
Hit:3 https://mirrors.edge.kernel.org/ubuntu jammy InRelease
Hit:4 https://mirrors.edge.kernel.org/ubuntu jammy-updates InRelease
Hit:5 https://mirrors.edge.kernel.org/ubuntu jammy-backports InRelease
Hit:6 https://mirrors.edge.kernel.org/ubuntu jammy-security InRelease
Fetched 102 kB in 2s (65.7 kB/s)
Reading package lists... Done
vagrant@ubuntu2204:~$
```

Figura 4 - Adicionando repositórios do docker.

Passo 5 - Instale o Docker (Figura 5).

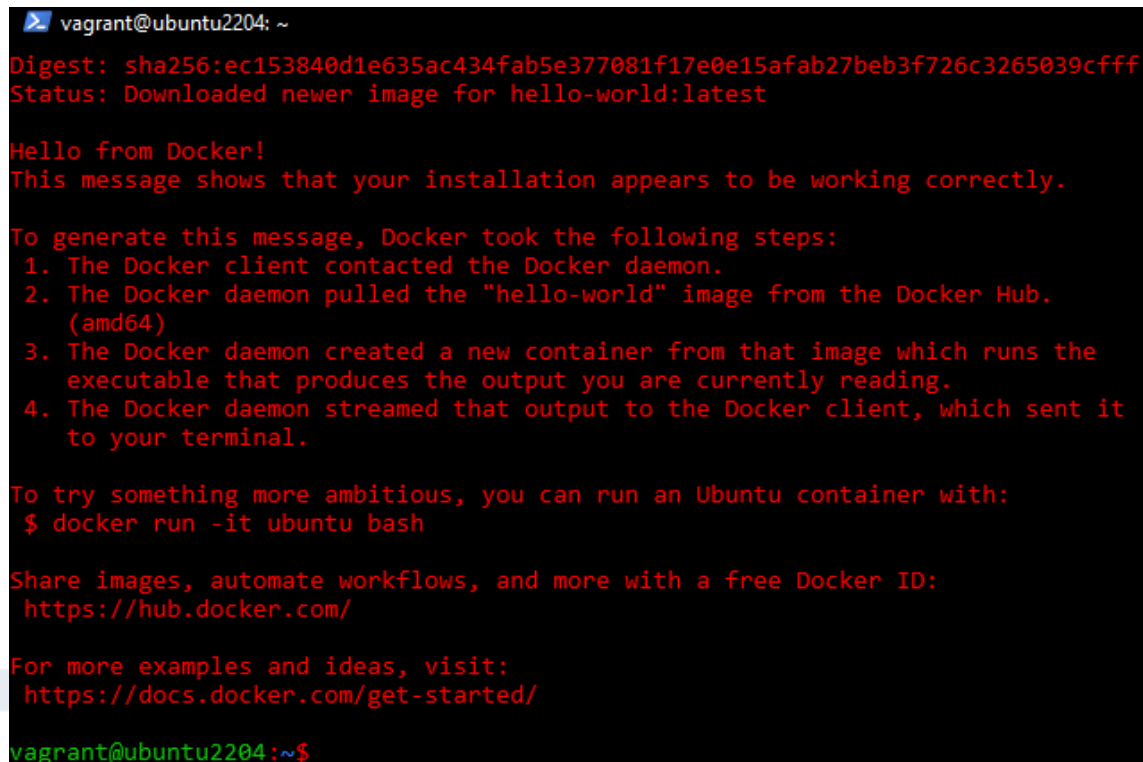
```
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

```
vagrant@ubuntu2204:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plug
in docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras libslirp0 pigz slirp4netns
Suggested packages:
  cgroupfs-mount | cgroup-lite docker-model-plugin
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin libslirp0 pigz slirp4netns
0 upgraded, 9 newly installed, 0 to remove and 0 not upgraded.
Need to get 103 MB of archives.
After this operation, 429 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 https://download.docker.com/linux/ubuntu jammy/stable amd64 containerd.io amd64 1.7.27-1 [30.
5 MB]
Get:2 https://mirrors.edge.kernel.org/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:3 https://mirrors.edge.kernel.org/ubuntu jammy/main amd64 libslirp0 amd64 4.6.1-1build1 [61.5 k
B]
Get:4 https://mirrors.edge.kernel.org/ubuntu jammy/universe amd64 slirp4netns amd64 1.0.1-2 [28.2 k
B]
```

Figura 5 - Instalação do Docker.

Passo 6 - Confira se a instalação está funcional imprimindo um “hello world” (Figura 6).

```
sudo docker run hello-world
```

A terminal window with a dark background and red text. The prompt is 'vagrant@ubuntu2204: ~'. The output shows the Docker client pulling the 'hello-world' image from Docker Hub, creating a container, and running it. The container outputs a message: 'Hello from Docker! This message shows that your installation appears to be working correctly.' It then lists the steps Docker took to generate this message. Finally, it suggests running an Ubuntu container with '\$ docker run -it ubuntu bash' and provides links to Docker Hub and Docker documentation.

```
vagrant@ubuntu2204: ~
Digest: sha256:ec153840d1e635ac434fab5e377081f17e0e15afab27beb3f726c3265039cfff
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

vagrant@ubuntu2204:~$
```

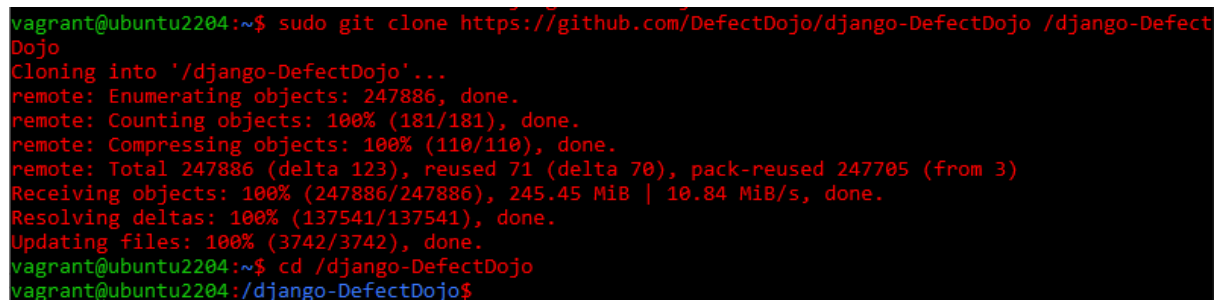
Figura 6 - Imprimindo “Hello World” no Docker.

Passo 7 - Pronto, agora podemos prosseguir com a instalação do DefectDojo.

3. Instalação do DefectDojo.

Passo 1 - Faça um clone do projeto para a VM (Figura 7).

```
sudo git clone https://github.com/DefectDojo/django-DefectDojo
/django-DefectDojo
cd /django-DefectDojo
```

A terminal window with a dark background and red text. The prompt is 'vagrant@ubuntu2204:~\$'. The command 'sudo git clone https://github.com/DefectDojo/django-DefectDojo /django-DefectDojo' is entered. The output shows the cloning process, including enumerating objects, counting objects, compressing objects, and receiving objects. The final prompt is 'vagrant@ubuntu2204:~/django-DefectDojo\$'.

```
vagrant@ubuntu2204:~$ sudo git clone https://github.com/DefectDojo/django-DefectDojo /django-DefectDojo
Cloning into '/django-DefectDojo'...
remote: Enumerating objects: 247886, done.
remote: Counting objects: 100% (181/181), done.
remote: Compressing objects: 100% (110/110), done.
remote: Total 247886 (delta 123), reused 71 (delta 70), pack-reused 247705 (from 3)
Receiving objects: 100% (247886/247886), 245.45 MiB | 10.84 MiB/s, done.
Resolving deltas: 100% (137541/137541), done.
Updating files: 100% (3742/3742), done.
vagrant@ubuntu2204:~$ cd /django-DefectDojo
vagrant@ubuntu2204:~/django-DefectDojo$
```

Figura 7 - Clonando o projeto do DefectDojo para a VM.

Passo 2 - Cheque se o “Toolkit” instalado é compatível (Figura 8).

```
sudo ./docker/docker-compose-check.sh
```

```
vagrant@ubuntu2204:/django-DefectDojo$ sudo ./docker/docker-compose-check.sh
Checking docker compose version
Supported docker compose version
```

Figura 8 - Verificando a compatibilidade com o docker compose.

Passo 3 - Compile as “imagens” do Docker (Figura 9).

```
sudo docker compose build
```

```
=> => exporting layers 99.5s
=> => writing image sha256:062351dce7cc2661915bae2c985edc08c4f655dea26708806de33663a49a3fe8 0.1s
=> => naming to docker.io/defectdojo/defectdojo-django:latest 0.1s
=> [nginx collectstatic 3/7] COPY components/ ./components/ 0.7s
=> [nginx collectstatic 4/7] RUN cd components && yarn 31.6s
=> [nginx collectstatic 5/7] COPY manage.py ./ 0.5s
=> [nginx collectstatic 6/7] COPY dojo/ ./dojo/ 5.9s
=> [nginx collectstatic 7/7] RUN env DD_SECRET_KEY='.' python3 manage.py collectstatic --n 50.2s
=> [uwsgi] resolving provenance for metadata file 2.9s
=> [nginx stage-3 2/5] COPY --from=collectstatic /app/static/ /usr/share/nginx/html/static 15.9s
=> [nginx stage-3 3/5] COPY wsgi_params nginx/nginx.conf nginx/nginx_TLS.conf /etc/nginx/ 0.6s
=> [nginx stage-3 4/5] COPY docker/entrypoint-nginx.sh / 0.2s
=> [nginx stage-3 5/5] RUN apk add --no-cache openssl && chmod -R g=u /var/cache/nginx 3.9s
=> [nginx] exporting to image 3.1s
=> => exporting layers 2.9s
=> => writing image sha256:c86a8237c5783295d8392b8d16bc8abd3414087d47d0735feb675ec2d4f5766d 0.0s
=> => naming to docker.io/defectdojo/defectdojo-nginx:latest 0.0s
=> [nginx] resolving provenance for metadata file 0.1s
[+] Building 2/2
  nginx Built 0.0s
  uwsgi Built 0.0s
vagrant@ubuntu2204:/django-DefectDojo$
```

Figura 9 - “Building Docker Images”.

Passo 4 - Configure para que a página web do DefectDojo seja acessada via HTTPS.

```
ln -s docker-compose.override.https.yml docker-compose.override.yml
```


Passo 5 - Execute a aplicação (Figura 10).

```
sudo docker compose up -d
```

```
vagrant@ubuntu2204:/django-DefectDojo$ sudo docker compose up -d
+ Running 20/20
redis Pulled 11.5s
  f18232174bc9 Already exists 0.0s
  775aad11a9b4 Pull complete 3.4s
  2dc49684c57a Pull complete 4.2s
  db5812f7c6e1 Pull complete 4.4s
  161e5dca6428 Pull complete 8.4s
  4d5b2da7c87f Pull complete 8.4s
  4f4fb700ef54 Pull complete 8.5s
  5ef94da019ec Pull complete 8.5s
postgres Pulled 28.0s
  fe07684b16b8 Pull complete 1.6s
  8fdb66080d86 Pull complete 1.7s
  fa4fc0c3be6f Pull complete 1.7s
  0796d800157f Pull complete 1.8s
  beb76af926b7 Pull complete 19.7s
  ec2b26b9d4c9 Pull complete 25.1s
  e0dc4151d8ff Pull complete 25.1s
  490b710f5dec Pull complete 25.1s
  72bd8efbdd17 Pull complete 25.1s
  b6579d264f5b Pull complete 25.2s
```

Figura 10 - Executando o DefectDojo.

Passo 6 - Você pode obter a senha do administrador através do comando abaixo (Figura 11). O inicializador pode demorar até 3 minutos para executar.

```
sudo docker compose logs initializer | grep "Admin password:"
```

```
vagrant@ubuntu2204:/django-DefectDojo$ sudo docker compose logs initializer | grep "Admin password:"
initializer-1 | Admin password: M2F4CQWZ04ShlFeiNzw1YN
vagrant@ubuntu2204:/django-DefectDojo$
```

Figura 11 - Senha do administrador impresso no terminal.

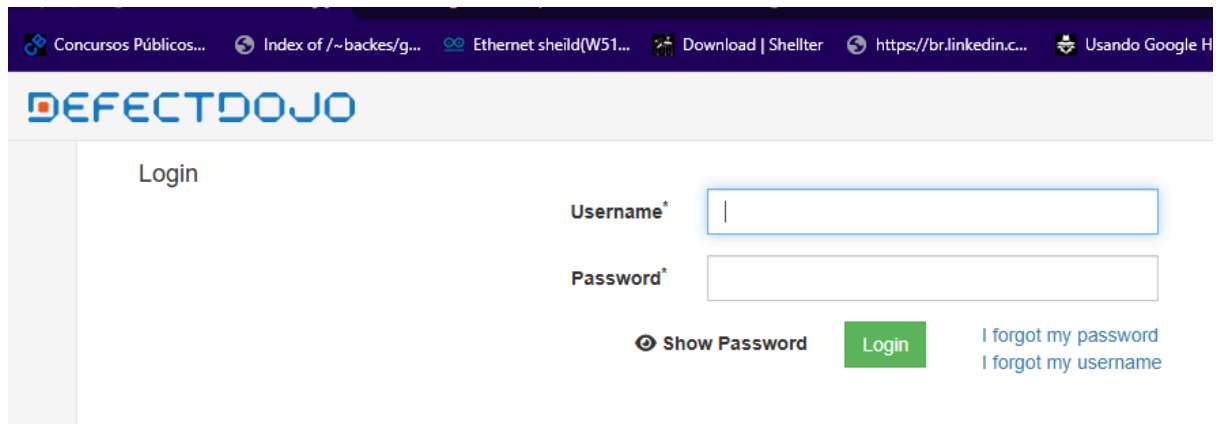
Passo 7 - Verifique o endereço de IP da VM configurada utilizando o seguinte comando.

```
sudo ip address show
```

```
vagrant@ubuntu2204:~$ sudo ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:8c:69:41 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    inet 10.0.2.15/24 metric 100 brd 10.0.2.255 scope global dynamic eth0
        valid_lft 86131sec preferred_lft 86131sec
    inet6 fe80::a00:27ff:fe8c:6941/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:64:b0:cd brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    inet 192.168.1.199/24 brd 192.168.1.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe64:b0cd/64 scope link
        valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:df:9f:3a brd ff:ff:ff:ff:ff:ff
    altname enp0s9
```

Figura 12 - Imprimindo os IPs configurados.

Passo 8 - Agora a partir do navegador web na máquina host, acesse a o endereço de IP da VM na porta 8443 (Exemplo:[<https://192.168.1.1:8443>]) (Figura 13).



The screenshot shows the DefectDojo login page. At the top, there's a navigation bar with the DefectDojo logo. Below it, the word "Login" is centered. To the right, there are two input fields: "Username*" and "Password*". Below the password field, there's a "Show Password" link with an eye icon, a green "Login" button, and two links: "I forgot my password" and "I forgot my username".

Figura 13 - Interface de login do DefectDojo.

Passo 9 - Agora insira as credenciais do administrador adquiridas no passo anterior, e acesse o dashboard do DefectDojo.

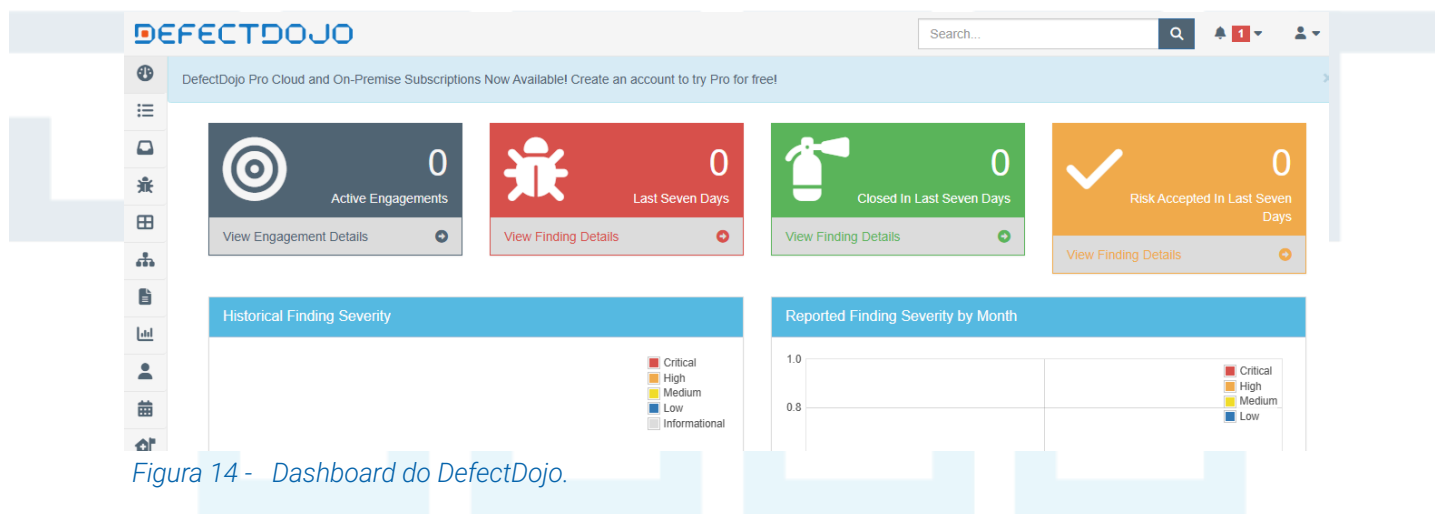


Figura 14 - Dashboard do DefectDojo.

Passo 10 - Realize a troca da senha clicando no ícone de usuário → Admin → Change Password.

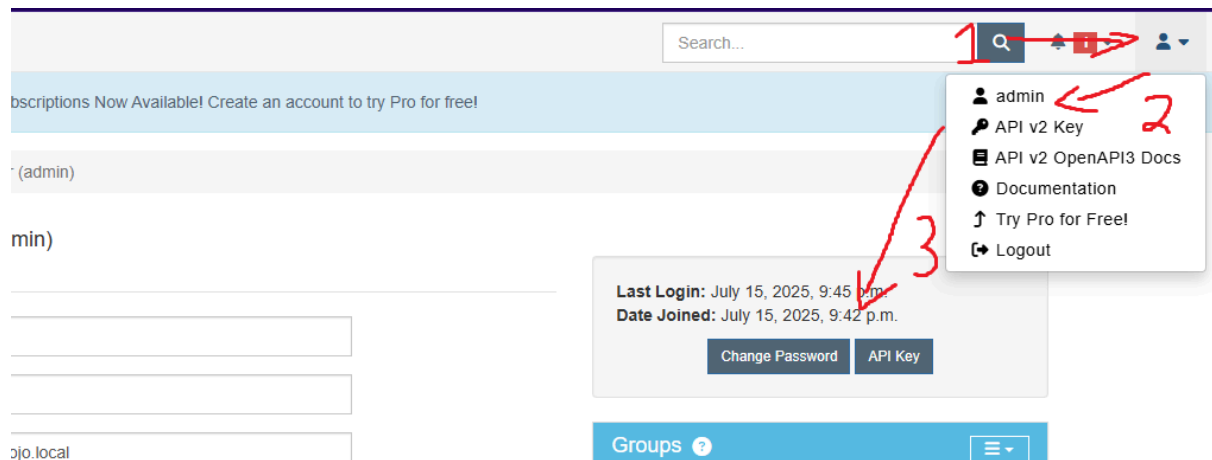


Figura 15 - Acessando a troca de senha do usuário administrador.

Passo 11 - Coloque a senha de sua preferência e pronto, agora o DefectDojo está funcional e configurado.



4. Configurando o DefectDojo para inicializar em conjunto com o Sistema Operacional.

Agora será configurado para que o DefectDojo seja gerenciado pelo systemd, permitindo que o serviço seja iniciado automaticamente junto com o sistema operacional. Além disso, em caso de falhas leves que interrompam sua execução, o próprio systemd será capaz de reinicializar o serviço automaticamente, mantendo-o sempre disponível.

Para aplicar essa configuração, será necessário integrar os comandos de execução do Docker a um unit file do systemd, utilizando como base o comando “docker compose”.

Passo 1 - Primeiro pare qualquer instância em execução do DefectDojo (Figura 16).

```
sudo docker compose down
```

```
vagrant@ubuntu2204:/django-DefectDojo$ sudo docker compose down
[+] Running 8/8
  Container django-defectdojo-celeryworker-1   Removed
  Container django-defectdojo-nginx-1         Remov...
  Container django-defectdojo-celerybeat-1     Removed
  Container django-defectdojo-uwsgi-1          Remov...
  Container django-defectdojo-initializer-1    Removed
  Container django-defectdojo-redis-1         Remov...
  Container django-defectdojo-postgres-1      Re...
  Network django-defectdojo_default           Removed
vagrant@ubuntu2204:/django-DefectDojo$
```

Figura 16 - Desligando a instância do DefectDojo.

Passo 2 - Retorne ao diretório home ou em alguma pasta com permissões do seu usuário, copie e cole o código abaixo para gerar as configurações de serviços. (Figura 17).

```
cd
cat << EOF > defectdojo.service
[Unit]
Description=DefectDojo Service
Requires=docker.service
After=docker.service

[Service]
WorkingDirectory=/django-DefectDojo
ExecStart=/usr/bin/docker compose up
ExecStop=/usr/bin/docker compose down
TimeoutStartSec=0

[Install]
WantedBy=multi-user.target
EOF
```

```
vagrant@ubuntu2204:~$ cat << EOF > defectdojo.service
> [Unit]
> Description=DefectDojo Service
> Requires=docker.service
> After=docker.service
>
>
> [Service]
> WorkingDirectory=/django-DefectDojo
> ExecStart=/usr/bin/docker compose up
> ExecStop=/usr/bin/docker compose down
> TimeoutStartSec=0
>
>
> [Install]
> WantedBy=multi-user.target
> EOF
vagrant@ubuntu2204:~$
```

Figura 17 - Criando um novo arquivo defectdojo.service.

Passo 3 - Agora copie o arquivo para a pasta de serviços do systemd.

```
sudo cp defectdojo.service /etc/systemd/system/
```

Passo 4 - Recarregue o Systemd e habilite o serviço do DefectDojo (Figura 18).

```
sudo systemctl daemon-reload
sudo systemctl enable defectdojo.service
```

```
vagrant@ubuntu2204:~$ sudo cp defectdojo.service /etc/systemd/system/
vagrant@ubuntu2204:~$ sudo systemctl daemon-reload
vagrant@ubuntu2204:~$ sudo systemctl enable defectdojo.service
Created symlink /etc/systemd/system/multi-user.target.wants/defectdojo.service → /etc/systemd/system/defectdojo.service.
vagrant@ubuntu2204:~$
```

Figura 18 - Reiniciando o Daemon e habilitando o serviço do DefectDojo.

Passo 5 - Inicie o serviço e verifique se ele está ativado e funcionando normalmente (Figura 19).

```
sudo systemctl start defectdojo.service
sudo systemctl status defectdojo.service
```

```
● defectdojo.service - DefectDojo Service
   Loaded: loaded (/etc/systemd/system/defectdojo.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2025-07-15 22:06:37 UTC; 69ms ago
     Main PID: 21964 (docker)
        Tasks: 14 (limit: 2807)
       Memory: 14.7M
          CPU: 49ms
      CGroup: /system.slice/defectdojo.service
              └─21964 /usr/bin/docker compose up
                  └─21977 /usr/libexec/docker/cli-plugins/docker-compose docker-cli-plugin-metadata
vagrant@ubuntu2204:~$
```

Figura 19 - Verificando se o DefectDojo está ativo.

Passo 6 - Verifique novamente se consegue acessar a página web através do endereço de IP da VM na porta 8443.

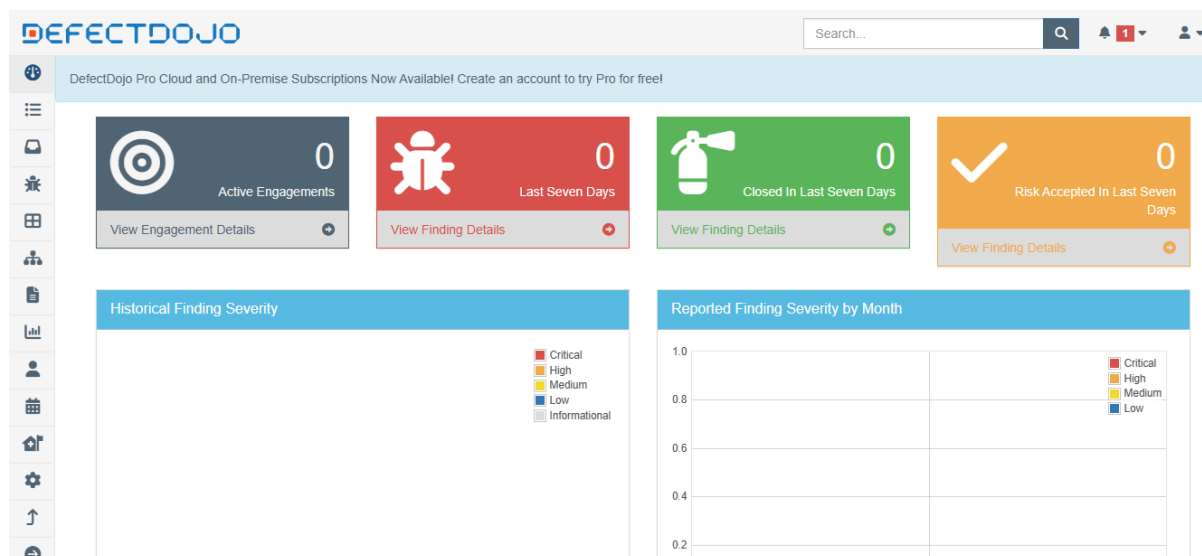


Figura 20 - Dashboard do DefectDojo.

Passo 7 - Pronto, agora o DefectDojo será gerenciado pelo systemd e será inicializado junto com o sistema operacional.

5. Possíveis Problemas que Podem Impedir a Execução do Greenbone

Durante a instalação ou operação do DefectDojo, alguns fatores podem bloquear seu funcionamento. Entre os mais comuns estão:

- **Falhas na inicialização dos containers:** Ocorrências por versões incompatíveis do Docker/Docker Compose ou ausência de permissões adequadas para build e execução.
- **Configuração incorrecta de variáveis de ambiente:** Dados como DD_ADMIN_USER, DD_ADMIN_PASSWORD e outras variáveis críticas precisam estar definidos corretamente no .env ou docker-compose.yml.
- **Serviços dependentes não estão funcionando:** PostgreSQL, Redis ou Celery podem não ter iniciado corretamente ou estar fora de sincronia, afetando o backend do sistema.
- **Conflito de portas ou bloqueios de rede:** A porta 8443 deve estar disponível e liberada no firewall local, pois é utilizada para acesso à interface web do DefectDojo via HTTPS. Se já estiver em uso ou bloqueada, o serviço não será acessível.