

```
// atualiza os pacotes
sudo apt update
```

```
// instala Docker e Docker Compose
sudo apt install -y docker.io docker-compose
```

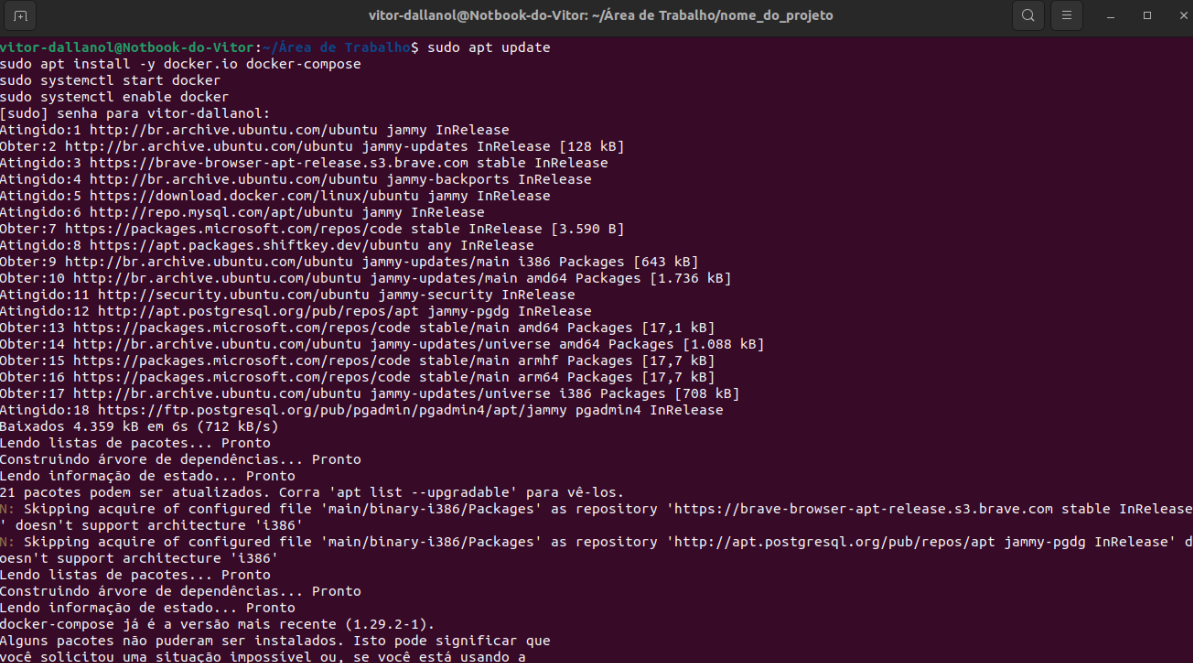
```
// inicia o serviço Docker
sudo systemctl start docker
```

```
// habilita o docker para iniciar automaticamente para não ter que dar o comando de cima
toda vez que reiniciar o computador
sudo systemctl enable docker
```

Copie e cole no terminal:

```
sudo apt update
sudo apt install -y docker.io docker-compose
sudo systemctl start docker
sudo systemctl enable docker
```

(após executado pedirá sua senha)



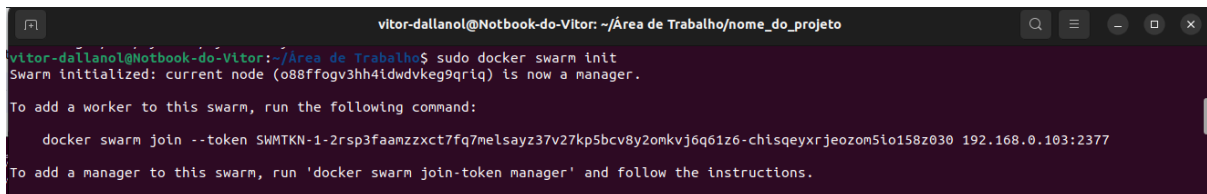
```
vitor-dallanol@Notbook-do-Vitor: ~/Área de Trabalho/nome_do_projeto
vitor-dallanol@Notbook-do-Vitor:~/Área de Trabalho$ sudo apt update
sudo apt install -y docker.io docker-compose
sudo systemctl start docker
sudo systemctl enable docker
[sudo] senha para vitor-dallanol:
Atingido:1 http://br.archive.ubuntu.com/ubuntu jammy InRelease
Obter:2 http://br.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Atingido:3 https://brave-browser-apt-release.s3.brave.com stable InRelease
Atingido:4 http://br.archive.ubuntu.com/ubuntu jammy-backports InRelease
Atingido:5 https://download.docker.com/linux/ubuntu jammy InRelease
Atingido:6 http://repo.mysql.com/apt/ubuntu jammy InRelease
Obter:7 https://packages.microsoft.com/repos/code stable InRelease [3.590 B]
Atingido:8 https://apt.packages.shiftkey.dev/ubuntu any InRelease
Obter:9 http://br.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [643 kB]
Obter:10 http://br.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1.736 kB]
Atingido:11 http://security.ubuntu.com/ubuntu jammy-security InRelease
Atingido:12 http://apt.postgresql.org/pub/repos/apt jammy-pgdg InRelease
Obter:13 https://packages.microsoft.com/repos/code stable/main amd64 Packages [17,1 kB]
Obter:14 http://br.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1.088 kB]
Obter:15 https://packages.microsoft.com/repos/code stable/main armhf Packages [17,7 kB]
Obter:16 https://packages.microsoft.com/repos/code stable/main arm64 Packages [17,7 kB]
Obter:17 http://br.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [708 kB]
Atingido:18 https://ftp.postgresql.org/pub/pgadmin/pgadmin4/apt/jammy pgadmin4 InRelease
Baixados 4.359 kB em 6s (712 kB/s)
Lendo listas de pacotes... Pronto
Construindo árvore de dependências... Pronto
Lendo informação de estado... Pronto
21 pacotes podem ser atualizados. Corra 'apt list --upgradable' para vê-los.
W: Skipping acquire of configured file 'main/binary-i386/Packages' as repository 'https://brave-browser-apt-release.s3.brave.com stable InRelease' doesn't support architecture 'i386'
W: Skipping acquire of configured file 'main/binary-i386/Packages' as repository 'http://apt.postgresql.org/pub/repos/apt jammy-pgdg InRelease' doesn't support architecture 'i386'
Lendo listas de pacotes... Pronto
Construindo árvore de dependências... Pronto
Lendo informação de estado... Pronto
docker-compose já é a versão mais recente (1.29.2-1).
Alguns pacotes não puderam ser instalados. Isto pode significar que
você solicitou uma situação impossível ou, se você está usando a
```

após isso:

```
// inicia o docker swarm
sudo docker swarm init
```

Copie e cole no terminal:

```
sudo docker swarm init
```

A terminal window with a dark background and light green text. The title bar reads 'vitor-dallanol@Notbook-do-Vitor: ~/Área de Trabalho/nome_do_projeto'. The terminal shows the command 'sudo docker swarm init' and its output: 'Swarm initialized: current node (o88ffogv3hh4tdwdvkeg9qriq) is now a manager.' It then provides instructions on how to add workers and managers to the swarm, including a long token string.

```
vitor-dallanol@Notbook-do-Vitor: ~/Área de Trabalho/nome_do_projeto
vitor-dallanol@Notbook-do-Vitor:~/Área de Trabalho$ sudo docker swarm init
Swarm initialized: current node (o88ffogv3hh4tdwdvkeg9qriq) is now a manager.

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

    docker swarm join --token SWMTKN-1-2rsp3faamzxc7f7q7melsayz37v27kp5bcv8y2omkvj6q6iz6-chisqeyxrjeozom5io158z030 192.168.0.103:2377

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

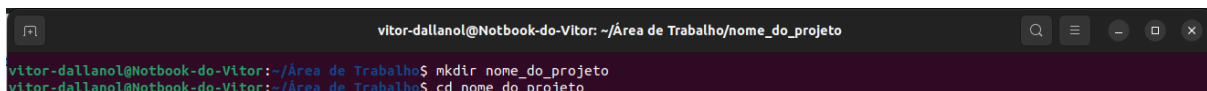
após isso:

```
// cria uma pasta com o nome: 'nome_do_projeto'
mkdir nome_do_projeto
```

```
// entra na pasta criada
cd nome_do_projeto
```

Copie e cole no terminal:

```
mkdir nome_do_projeto
cd nome_do_projeto
```

A terminal window with a dark background and light green text. The title bar reads 'vitor-dallanol@Notbook-do-Vitor: ~/Área de Trabalho/nome_do_projeto'. The terminal shows the commands 'mkdir nome_do_projeto' and 'cd nome_do_projeto' being executed successfully.

```
vitor-dallanol@Notbook-do-Vitor: ~/Área de Trabalho/nome_do_projeto
vitor-dallanol@Notbook-do-Vitor:~/Área de Trabalho$ mkdir nome_do_projeto
vitor-dallanol@Notbook-do-Vitor:~/Área de Trabalho$ cd nome_do_projeto
```

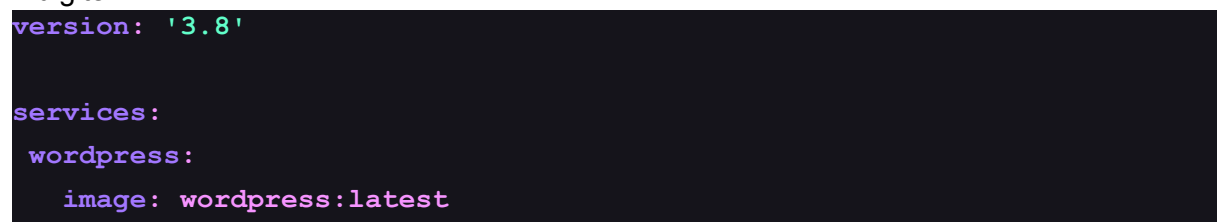
após isso:

```
// cria e abre o arquivo docker-compose para edição
nano docker-compose.yml
```

Copie e cole no terminal:

```
nano docker-compose.yml
```

// digite:

A code block with a dark background and light green text. It shows the configuration for a Docker Compose file, specifically setting the version to '3.8' and defining a service named 'wordpress' with the image 'wordpress:latest'.

```
version: '3.8'

services:
  wordpress:
    image: wordpress:latest
```

```
ports:
  - '8080:80'
environment:
  WORDPRESS_DB_HOST: db:3306
  WORDPRESS_DB_USER: wordpress
  WORDPRESS_DB_PASSWORD: wordpress_password
  WORDPRESS_DB_NAME: wordpress
volumes:
  - wordpress_data:/var/www/html
deploy:
  replicas: 1
  restart_policy:
    condition: on-failure

db:
  image: mysql:5.7
  environment:
    MYSQL_DATABASE: wordpress
    MYSQL_USER: wordpress
    MYSQL_PASSWORD: wordpress_password
    MYSQL_ROOT_PASSWORD: root_password
  volumes:
    - db_data:/var/lib/mysql
  deploy:
    replicas: 1
    restart_policy:
      condition: on-failure

redis:
  image: redis:latest
  ports:
    - '6379:6379'
  volumes:
    - redis_data:/data
  deploy:
    replicas: 1
    restart_policy:
      condition: on-failure

prometheus:
  image: prom/prometheus:latest
  ports:
    - '9090:9090'
```

```

volumes:
  - ./prometheus.yml:/etc/prometheus/prometheus.yml
deploy:
  replicas: 1
  restart_policy:
    condition: on-failure

grafana:
  image: grafana/grafana:latest
  ports:
    - '3000:3000'
  environment:
    - GF_SECURITY_ADMIN_PASSWORD=admin
  volumes:
    - grafana_data:/var/lib/grafana
  deploy:
    replicas: 1
    restart_policy:
      condition: on-failure

volumes:
  wordpress_data:
  db_data:
  redis_data:
  grafana_data:

```

// após digitado

// utilize CTRL + O e depois Enter para salvar

// e CTRL + X para sair

/* ATENÇÃO A INDENTAÇÃO DESSE CÓDIGO DEVE SEGUIR A INDENTAÇÃO DA IMAGEM ABAIXO. CASO CONTRARIO O CÓDIGO PODERÁ NÃO FUNCIONAR */

```
version: '3.8'

services:
  wordpress:
    image: wordpress:latest
    ports:
      - '8080:80'
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress_password
      WORDPRESS_DB_NAME: wordpress
    volumes:
      - wordpress_data:/var/www/html
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

  db:
    image: mysql:5.7
    environment:
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress_password
      MYSQL_ROOT_PASSWORD: root_password
    volumes:
      - db_data:/var/lib/mysql
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

  redis:
    image: redis:latest
    ports:
      - '6379:6379'
    volumes:
      - redis_data:/data
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

  prometheus:
    image: prom/prometheus:latest
    ports:
      - '9090:9090'
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

  grafana:
    image: grafana/grafana:latest
    ports:
      - '3000:3000'
    environment:
      GF_SECURITY_ADMIN_PASSWORD=admin
    volumes:
      - grafana_data:/var/lib/grafana
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

volumes:
  wordpress_data:
  db_data:
  redis_data:
  grafana_data:
```

```

vitor-dallanol@Notbook-do-Vitor: ~/Área de Trabalho/nome_do_projeto
vitor-dallanol@Notbook-do-Vitor:~/Área de Trabalho/nome_do_projeto$ nano docker-compose.yml

```

após isso:

```
// cria e abre o arquivo prometheus para edição
nano prometheus.yml
```

Copie e cole no terminal:
nano prometheus.yml

// digite:

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'wordpress'
    static_configs:
      - targets: ['wordpress:80']

  - job_name: 'mysql'
    static_configs:
      - targets: ['db:3306']

  - job_name: 'redis'
    static_configs:
      - targets: ['redis:6379']

  - job_name: 'docker'
    metrics_path: /metric
    static_configs:
      - targets: ['docker:9323']
```

```
// após digitado
// utilize CTRL + O e depois Enter para salvar
// e CTRL + X para sair
```

/* ATENÇÃO A INDENTAÇÃO DESSE CÓDIGO DEVE SEGUIR A INDENTAÇÃO DA IMAGEM ABAIXO. CASO CONTRARIO O CÓDIGO PODERÁ NÃO FUNCIONAR */

```
global:
  scrape_interval: 15s

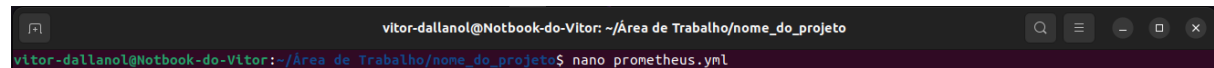
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'wordpress'
    static_configs:
      - targets: ['wordpress:80']

  - job_name: 'mysql'
    static_configs:
      - targets: ['db:3306']

  - job_name: 'redis'
    static_configs:
      - targets: ['redis:6379']

  - job_name: 'docker'
    metrics_path: /metric
    static_configs:
      - targets: ['docker:9323']
```



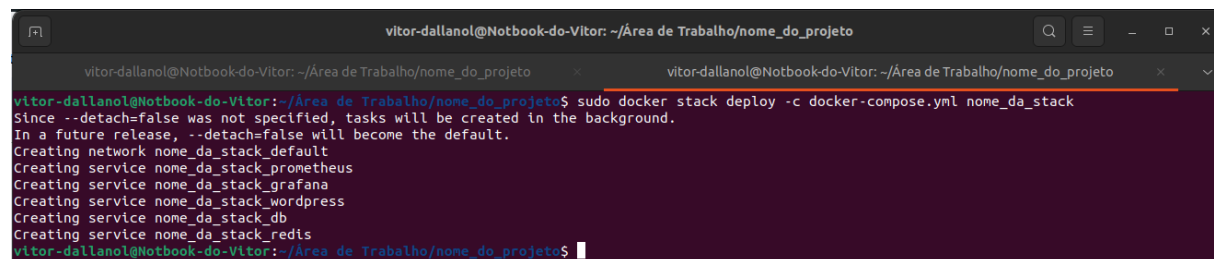
A terminal window with the title bar "vitor-dallanol@Notbook-do-Vitor: ~/Área de Trabalho/nome_do_projeto". The command "nano prometheus.yml" is entered at the prompt.

// utilizado para dar o deploy no docker swarm do arquivo docker-compose com o nome da stack: 'nome_da_stack'

sudo docker stack deploy -c docker-compose.yml nome_da_stack

Copie e cole no terminal:

sudo docker stack deploy -c docker-compose.yml nome_da_stack



A terminal window with the title bar "vitor-dallanol@Notbook-do-Vitor: ~/Área de Trabalho/nome_do_projeto". The command "sudo docker stack deploy -c docker-compose.yml nome_da_stack" is entered. The output shows the creation of a network and services for the stack.

```
vitor-dallanol@Notbook-do-Vitor: ~/Área de Trabalho/nome_do_projeto$ sudo docker stack deploy -c docker-compose.yml nome_da_stack
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network nome_da_stack_default
Creating service nome_da_stack_prometheus
Creating service nome_da_stack_grafana
Creating service nome_da_stack_wordpress
Creating service nome_da_stack_db
Creating service nome_da_stack_redis
vitor-dallanol@Notbook-do-Vitor: ~/Área de Trabalho/nome_do_projeto$
```

// Utilize o comando docker service ls para verificar os status dos serviços.

Copie e cole no terminal:

docker service ls

```
vitor-dallanol@Notbook-do-Vitor:~/Área de Trabalho/nome_do_projeto$ docker service ls
ID                NAME                MODE                REPLICAS        IMAGE                PORTS
o22e1l35kqe1     nome_da_stack_db    replicated          1/1             mysql:5.7           *:3000->3000/tcp
fdc1ivur145i     nome_da_stack_grafana replicated          1/1             grafana/grafana:latest *:3000->3000/tcp
6uwt0g1gckco     nome_da_stack_prometheus replicated          1/1             prom/prometheus:latest *:9090->9090/tcp
nr649bq0ugng     nome_da_stack_redis  replicated          1/1             redis:latest         *:6379->6379/tcp
wg50wkz7wji2     nome_da_stack_wordpress replicated          1/1             wordpress:latest      *:8080->80/tcp
```

// Caso alguma esteja 0/1 nas replicas verifique se você tem todas as imagens utilizadas no sistema

// utilize o comando docker images

```
vitor-dallanol@Notbook-do-Vitor:~/Área de Trabalho/nome_do_projeto$ docker images
REPOSITORY        TAG                IMAGE ID           CREATED           SIZE
grafana/grafana   latest            97bf66938f26      17 hours ago     445MB
wordpress         latest            73ca9d5ac9d3      18 hours ago     685MB
prom/prometheus   latest            b74abbcc4eac      6 days ago       271MB
redis             latest            aceb1262c1ea      4 weeks ago      117MB
ubuntu           latest            des2d803b224      2 months ago     76.2MB
maven            latest            c38802599f18      6 months ago     510MB
mysql             5.7              5107333e00a8      6 months ago     501MB
```

caso necessário utilize o comando para cada um faltante:

// comando utilizado para puxar imagens

prometheus:

docker pull prom/prometheus

grafana:

docker pull grafana/grafana

redis:

docker pull redis

mysql:

docker pull mysql:5.7

wordpress:

docker pull wordpress

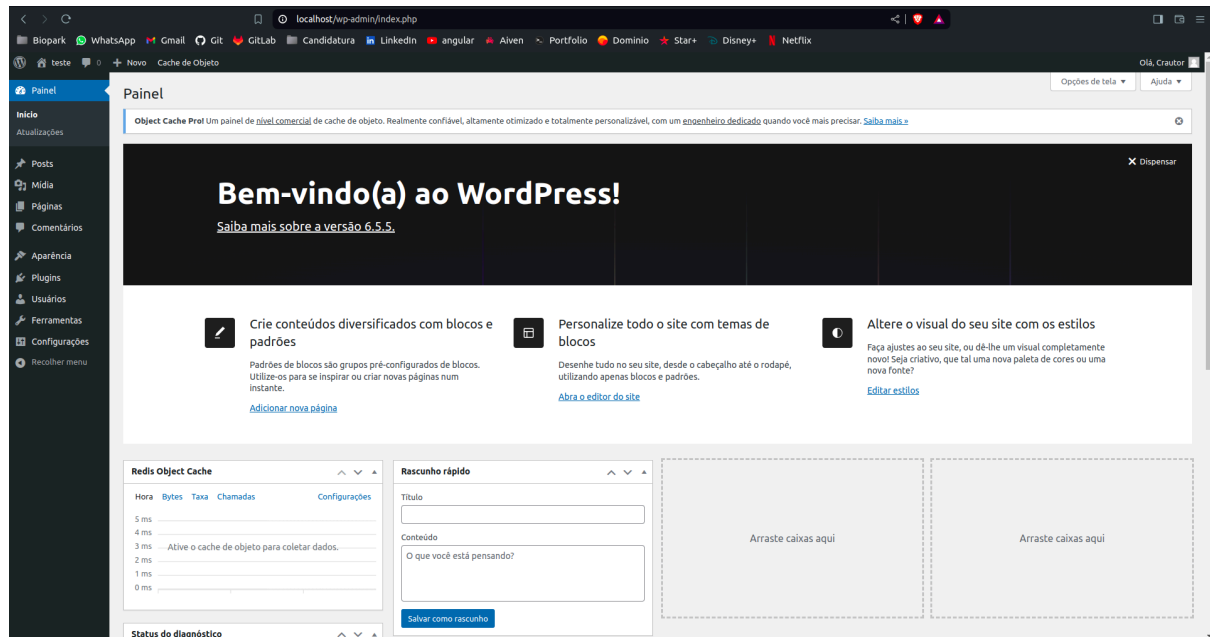
Acesse:

<http://localhost:80>

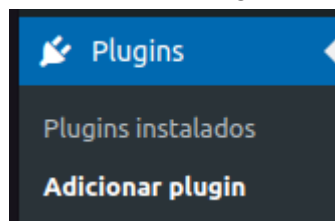
e siga os passos necessários para terminar a instalação

(é bem didático só preencher tudo)

após a instalação completa você terá acesso a essa pagina:

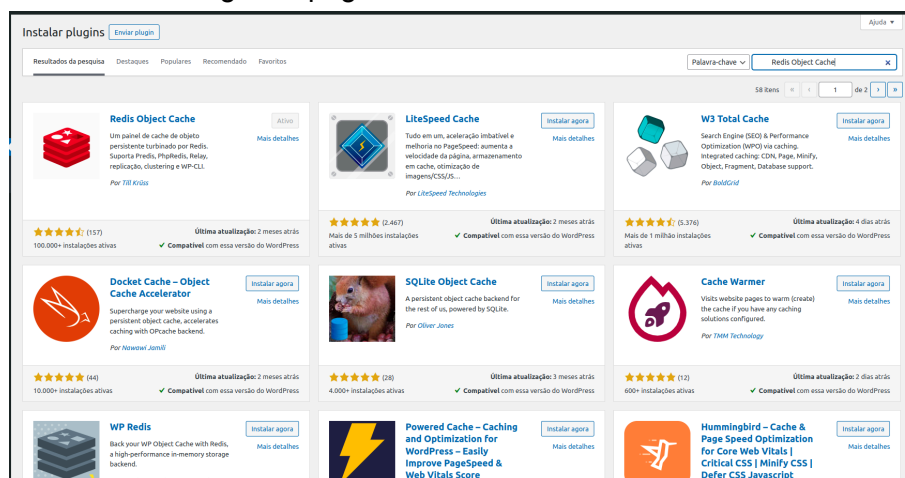


abra a aba de plugins e vá em adicionar plugin:



procure por **Redis Object Cache**

e você verá a seguinte pagina:



clique em instalar e você terá a seguinte visão:



Redis Object Cache

Um painel de cache de objeto persistente turbinado por Redis. Suporta Predis, PhpRedis, Relay, replicação, clustering e WP-CLI.

Por Till Krüss

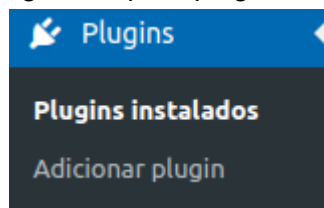
Ativo

Mais detalhes

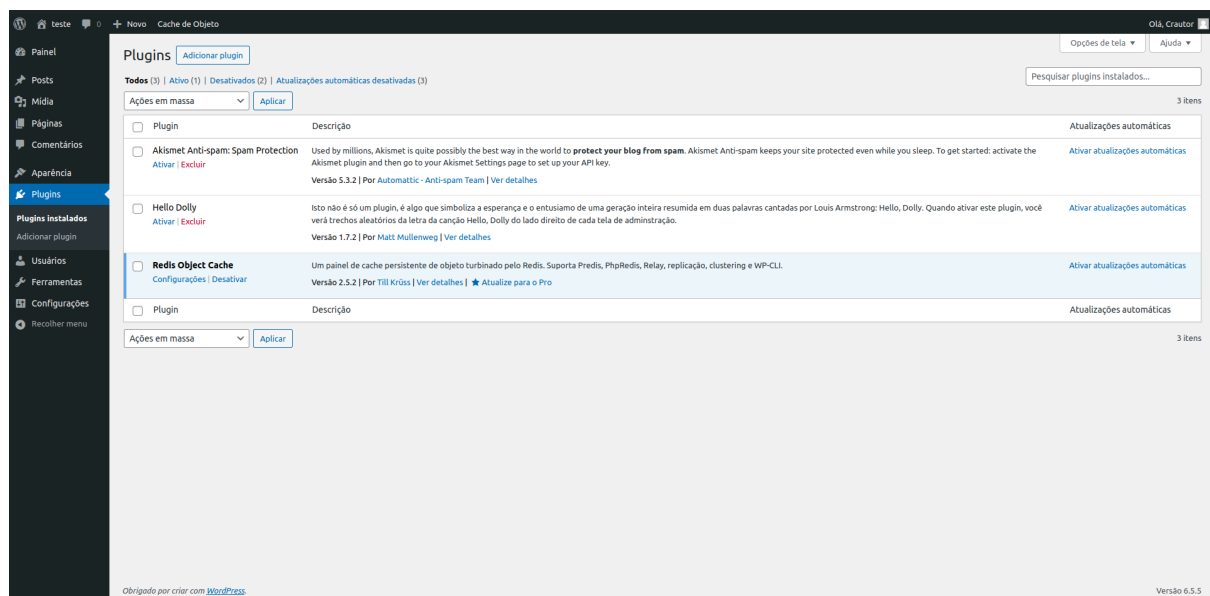
★★★★☆ (157)
100.000+ instalações ativas

Última atualização: 2 meses atrás
✓ **Compatível** com essa versão do WordPress

agora vá para plugins instalados:



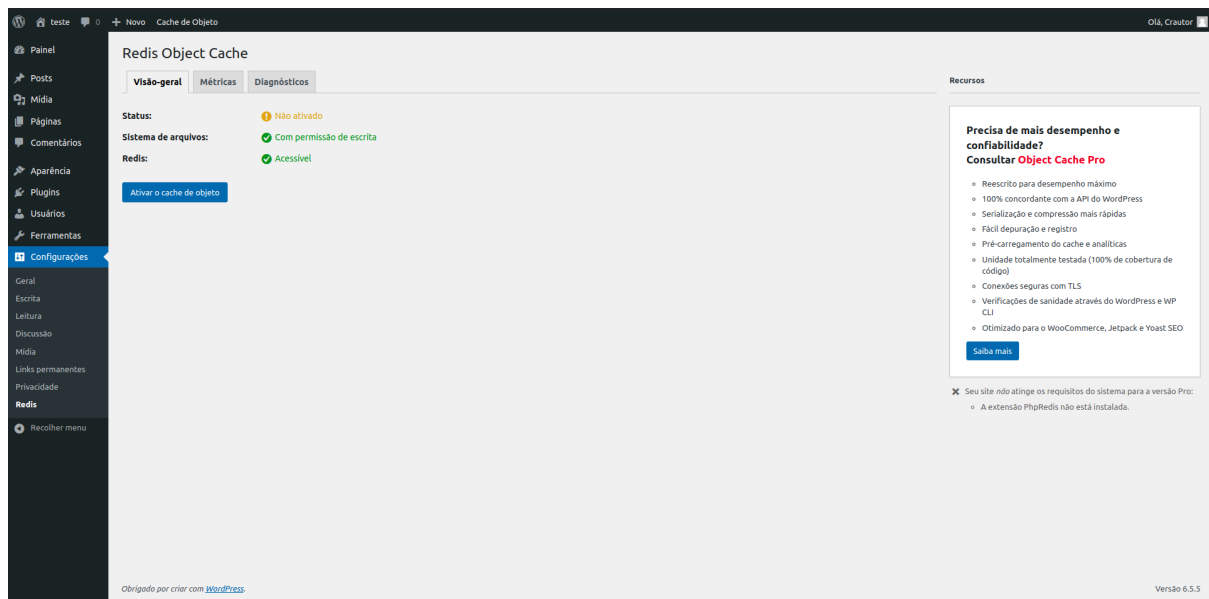
e terá essa visão:



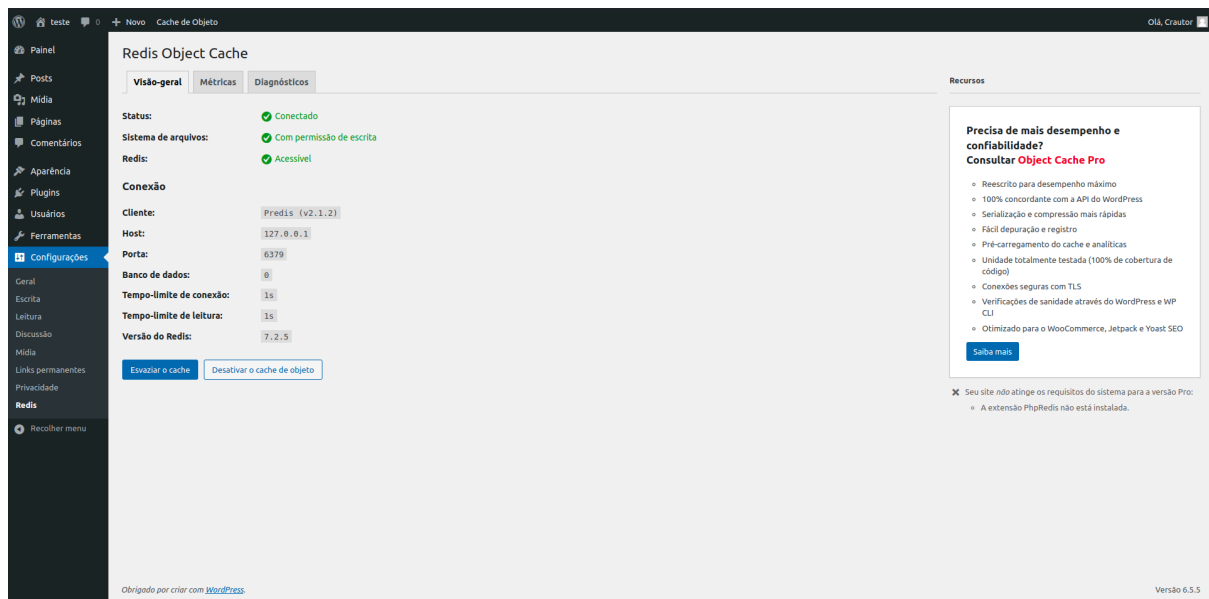
The screenshot shows the WordPress admin interface with the 'Plugins' menu selected. The 'Plugins instalados' section is active, displaying a list of installed plugins. The 'Redis Object Cache' plugin is highlighted, showing its description, version (2.5.2), and author (Till Krüss). The interface includes a sidebar with navigation links, a top bar with user information, and a main content area with a search bar and a table of installed plugins.

Plugin	Descrição	Atualizações automáticas
<input type="checkbox"/> Akismet Anti-spam: Spam Protection	Used by millions, Akismet is quite possibly the best way in the world to protect your blog from spam . Akismet Anti-spam keeps your site protected even while you sleep. To get started: activate the Akismet plugin and then go to your Akismet Settings page to set up your API key. Versão 5.3.2 Por Automattic - Anti-spam Team Ver detalhes	Ativar atualizações automáticas
<input type="checkbox"/> Hello Dolly	Isto não é só um plugin, é algo que simboliza a esperança e o entusiasmo de uma geração inteira resumida em duas palavras cantadas por Louis Armstrong: Hello, Dolly. Quando ativar este plugin, você verá trechos aleatórios da letra da canção Hello, Dolly do lado direito de cada tela de administração. Versão 1.7.2 Por Matt Mullenweg Ver detalhes	Ativar atualizações automáticas
<input checked="" type="checkbox"/> Redis Object Cache	Um painel de cache persistente de objeto turbinado pelo Redis. Suporta Predis, PhpRedis, Relay, replicação, clustering e WP-CLI. Versão 2.5.2 Por Till Krüss Ver detalhes Atualize para o Pro	Ativar atualizações automáticas
<input type="checkbox"/> Plugin	Descrição	Atualizações automáticas

acesse configurar no **Redis Object Cache** e verá:



clique em **ATIVAR O CACHE DE OBJETO** e verá:



links:

WordPress: <http://localhost:8080> ou <http://localhost:80>

Prometheus: <http://localhost:9090>

Grafana: <http://localhost:3000>

Usuário: admin

Senha: admin