comment installer-netbox-irm sur-debian-12

NetBox is an Infrastructure Resource Modelling (IRM) designed for network automation and infrastructure engineering. Initially, it was created by the DigitalOcean team, and now become an open-source project released under the Apache 2 License. NetBox was created in the Python Django Web framework with PostgreSQL as the default database, and the installation of NetBox is quite similar to other Python Django web applications.

In this guide, we'll show you how to install NetBox IRM software on Debian 12 server step-by-step. We'll show you the installation of NetBox with PostgreSQL as the database server and Apache2 web server as a reverse proxy. You'll also secure your NetBox installation with SSl/TLS certificates.

Prerequisites

Before proceeding, ensure you have the following:

- A Debian 12 server.
- A non-root user with administrator privileges.
- A public or local domain name pointed to the server IP address.

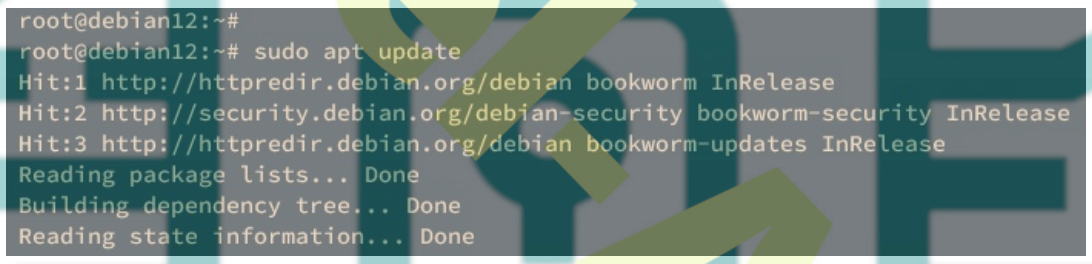
Installing Dependencies

NetBox is a web application based on the Python Django web framework. It can be installed with the PostgreSQL database server and Redis server for cache management.

In the following step, you will install those dependencies that NetBox needs, you will also install the Apache2 web server that will be used as a reverse proxy for your NetBox installation.

To start, execute the following *apt* command to update your Debian repository.

```
sudo apt update
```



Then, install package dependencies for your NetBox IRM installation.

```
sudo apt install apache2 postgresql postgresql-common libpq-dev redis-server git python3 python3-pip python3-venv python3-dev build-essential libxml2-dev libxslt1-dev libffi-dev  libssl-dev zlib1g-dev
```

Type y to proceed with the installation of dependencies such as Apache2 web server, PostgreSQL database server, Redis, Git, Python3 packages, and some additional system libraries.

After dependencies are installed, verify each dependency by executing the command below.

Verify the apache2 service to ensure that the service is enabled and running.

```
sudo systemctl is-enabled apache2
sudo systemctl status apache2
```

If apache2 is running and enabled, you should get an output like the following:

```
root@debian12:~#
root@debian12:~# sudo systemctl is-enabled apache2
enabled
root@debian12:~# sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 18477 (apache2)
      Tasks: 6 (limit: 4642)
     Memory: 21.5M
```

Verify the PostgreSQL service to ensure that the service is running and enabled.

```
sudo systemctl is-enabled postgresql
sudo systemctl status postgresql
```

The PostgreSQL service should be running and enabled like this:

```
root@debian12:~#
root@debian12:~# sudo systemctl is-enabled postgresql
enabled
root@debian12:~# sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; preset: enabled)
     Active: active (exited) since
   Main PID: 23586 (code=exited, status=0/SUCCESS)
        CPU: 3ms
```

Now verify the Redis service to ensure that the service is running and enabled.

```
sudo systemctl is-enabled redis
sudo systemctl status redis
```

The Redis service should be running and enabled like the following

```
root@debian12:~#
root@debian12:~# sudo systemctl is-enabled redis
alias
root@debian12:~# sudo systemctl status redis
● redis-server.service - Advanced key-value store
     Loaded: loaded (/lib/systemd/system/redis-server.service; enabled; preset: enabled)
     Active: active (running) since
       Docs: http://redis.io/documentation,
             man:redis-server(1)
   Main PID: 23902 (redis-server)
     Status: "Ready to accept connections"
      Tasks: 5 (limit: 4642)
     Memory: 7.4M
        CPU: 570ms
```

Lastly, verify the Python version using the command below. The latest version of NetBox IRM supports Python v3.9, 3.10, and 3.11.

```
python3 --version
```

You should see **Python 3.11** is installed on your Debian machine.

```
root@debian12:~#
root@debian12:~# python3 --version
Python 3.11.2
root@debian12:~#
```

Configuring PostgreSQL Server

After installing dependencies, you will create a new PostgreSQL database and user that NetBox will use. To do that, you must log in to the PostgreSQL server via psql command line.

Log in to the PostgreSQL server by executing the command below.

```
sudo -u postgres psql
```

Run the following queries to create a new user **netbox** with password **p4ssw0rd**. Then, create a new database **netboxdb** with the owner **netbox**.

```
CREATE USER netbox LOGIN CREATEDB PASSWORD 'p4ssw0rd';
CREATE DATABASE netboxdb OWNER netbox;
```

```
root@debian12:~#
root@debian12:~# sudo -u postgres psql
could not change directory to "/root": Permission denied
psql (15.3 (Debian 15.3-0+deb12u1))
Type "help" for help.

postgres=# CREATE USER netbox LOGIN CREATEDB PASSWORD 'p4ssw0rd';
CREATE ROLE
postgres=# CREATE DATABASE netboxdb OWNER netbox;
CREATE DATABASE
postgres=#
```

After that, verify the list of users and databases on your PostgreSQL by executing the command below.

```
\l
\du
```

You should see the database **netboxdb** and user **netbox** created on your PostgreSQL server.

```
postgres=#
postgres=# \l
                                                 List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    | ICU Locale | Locale Provider |   Access privileges
-----------+----------+----------+-------------+-------------+------------+-----------------+-----------------------
 netboxdb  | netbox   | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            |
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            | =c/postgres          +
           |          |          |             |             |            |                 | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            | =c/postgres          +
           |          |          |             |             |            |                 | postgres=CTc/postgres
(4 rows)

postgres=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 netbox    | Create DB                                                  | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

Type quit to exit from the PostgreSQL server.

Next, log in to PostgreSQL using the new user **netbox** to the database **netboxdb**. This will ensure that the user **netbox** can connect to the database **netboxdb**.

```
sudo -u postgres psql --username netbox --password --host localhost netboxdb
```

Once connected, verify your connection using the following query.

```
\conninfo
```

In the following output, you should see that you've connected to the database **netboxdb** via user **netbox**.

```
root@debian12:~#
root@debian12:~# sudo -u postgres psql --username netbox --password --host localhost netboxdb

Password:
psql (15.3 (Debian 15.3-0+deb12u1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

netboxdb=> \conninfo
You are connected to database "netboxdb" as user "netbox" on host "localhost" (address "::1") at port "5432".
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
netboxdb=>
netboxdb=> quit
root@debian12:~#
```

Type quit again to exit from your PostgreSQL server.

Configuring Redis Server

With the PostgreSQL database and user created, the next step is to configure your Redis server that will be used as

cache management for NetBox. To do that, you will modify the Redis configuration */etc/redis/redis.conf* and verify your changes via redis-cli.

Open the default Redis configuration */etc/redis/redis.conf* using the following nano editor command.

```
sudo nano /etc/redis/redis.conf
```

Uncomment the option **requirepass** and input your password that will be used to secure your Redis server.

```
requirepass p4ssw0rdNetBox
```

When finished, save and exit the file.

Now run the following *systemctl* command to restart the redis service and apply the changes that you've made.

```
sudo systemctl restart redis
```

To ensure that everything is working, you can verify Redis via *redis-cli*. Access your Redis server using the *redis-cli* command below.

```
redis-cli
```

Authenticate to the Redis server using the following **AUTH** query and be sure to change the password.

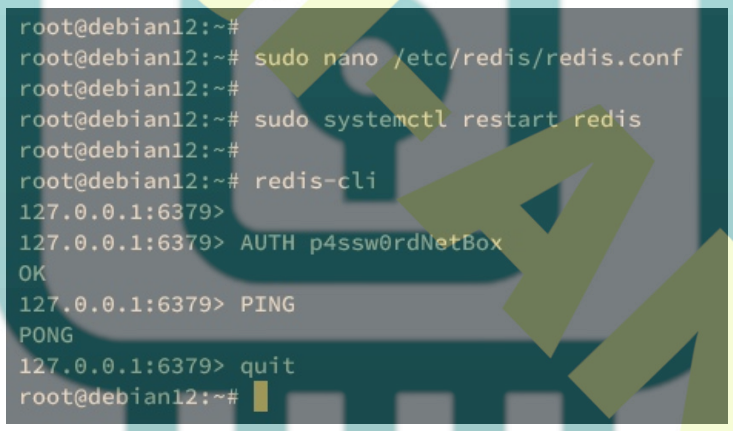
```
AUTH p4ssw0rdNetBox
```

Once authenticated, you should get the output **OK**.

Now run the PING query below to ensure that your connection is successful.

```
PING
```

If successful, you should get the output **PONG** from the Redis server.



Installing NetBox IRM

In the following section, you will download and install NetBox IRM to your system. You will download the NetBox source code via Git, then configure it by modifying the NetBox configuration, adding the database PostgreSQL server and Redis, and then you will also create an administrator user for NetBox.

First, execute the command below to create a new systemd user **netbox** that will be used for running NetBox installation.

```
sudo useradd -r -d /opt/netbox -s /usr/sbin/nologin netbox
```

Download NetBox IRM source code via git and change the ownership of the */opt/netbox* directory to user netbox.

```
cd /opt; sudo git clone -b master --depth 1 https://github.com/netbox-community/netbox.git
sudo chown -R netbox:netbox /opt/netbox
```

```
root@debian12:~#
root@debian12:~# sudo useradd -r -d /opt/netbox -s /usr/sbin/nologin netbox
root@debian12:~#
root@debian12:~# cd /opt; sudo git clone -b master --depth 1 https://github.com/netbox-community/netbox.git
Cloning into 'netbox'...
remote: Enumerating objects: 1642, done.
remote: Counting objects: 100% (1642/1642), done.
remote: Compressing objects: 100% (1433/1433), done.
remote: Total 1642 (delta 323), reused 869 (delta 172), pack-reused 0
Receiving objects: 100% (1642/1642), 7.19 MiB | 739.00 KiB/s, done.
Resolving deltas: 100% (323/323), done.
root@debian12:/opt#
root@debian12:/opt# sudo chown -R netbox:netbox /opt/netbox
root@debian12:/opt#
```

Next, move your working directory to */opt/netbox* and generate the NetBox secret key via the script *generate_secret_key.py*. Be sure to copy the generated secret key that will be used for your NetBox installation.

```
cd /opt/netbox/netbox/netbox
sudo -u netbox python3 ../generate_secret_key.py
```

```
root@debian12:/opt#
root@debian12:/opt# cd /opt/netbox/netbox/netbox
root@debian12:/opt/netbox/netbox/netbox#
root@debian12:/opt/netbox/netbox/netbox# sudo -u netbox python3 ../generate_secret_key.py
ZjYbgz$)j!NnqJcZLR!NB2BCz4(Yyk=o^Xr(1sTIrM)ZyiE%nk
root@debian12:/opt/netbox/netbox/netbox#
root@debian12:/opt/netbox/netbox/netbox#
```

Copy the default configuration *configuration_example.py* to *configuration.py*, then open the new file *configuration.py* using the following nano editor command.

```
sudo -u netbox cp configuration_example.py configuration.py
sudo -u netbox nano configuration.py
```

Within the **ALLOWED_HOSTS** section, add your domain name or your server IP address.

```
ALLOWED_HOSTS = ['netbox.hwdomain.io', '192.168.10.15']
```

Input your PostgreSQL database details to the **DATABASE** section, including the database name, user, password, host, and port.

```
# database configuration
DATABASE = {
    'NAME': 'netboxdb',               # Database name
    'USER': 'netbox',                 # PostgreSQL username
    'PASSWORD': 'p4ssw0rd', # PostgreSQL password
    'HOST': 'localhost',              # Database server
    'PORT': '',                       # Database port (leave blank for default)
    'CONN_MAX_AGE': 300,              # Max database connection age (seconds)
}
```

Within the **REDIS** section, input details of your Redis server to both tasks and caching options.

```
# Redis cache configuration
REDIS = {
    'tasks': {
        'HOST': 'localhost',      # Redis server
        'PORT': 6379,             # Redis port
        'PASSWORD': 'p4ssw0rdNetBox',          # Redis password (optional)
        'DATABASE': 0,            # Database ID
        'SSL': False,             # Use SSL (optional)
    },
    'caching': {
        'HOST': 'localhost',
        'PORT': 6379,
        'PASSWORD': 'p4ssw0rdNetBox',
        'DATABASE': 1,            # Unique ID for second database
        'SSL': False,
    }
}
```

Lastly, input your secret key to the **SECRET_KEY** section.

```
# Secret key
SECRET_KEY = 'ZjYbgz$)j!NnqJcZLR!NB2BCz4(Yyk=o^Xr(1sTIrM)ZyiE%nk'
```

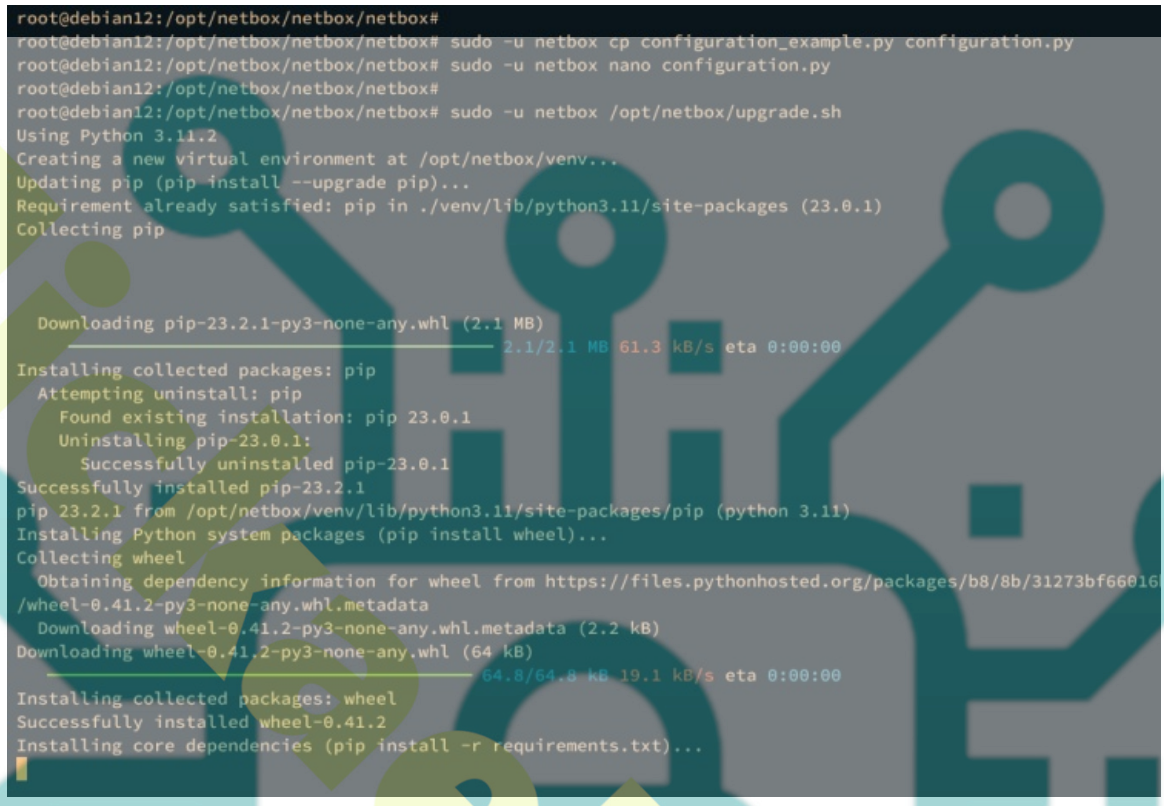
When you're done, save and exit the file.

Next, execute the */opt/netbox/upgrade.sh* script to start your NextBox installation. This will create a new Python virtual environment, install some Python packages and libraries, run database migration to your PostgreSQL server, also

generate static files for NextBox.

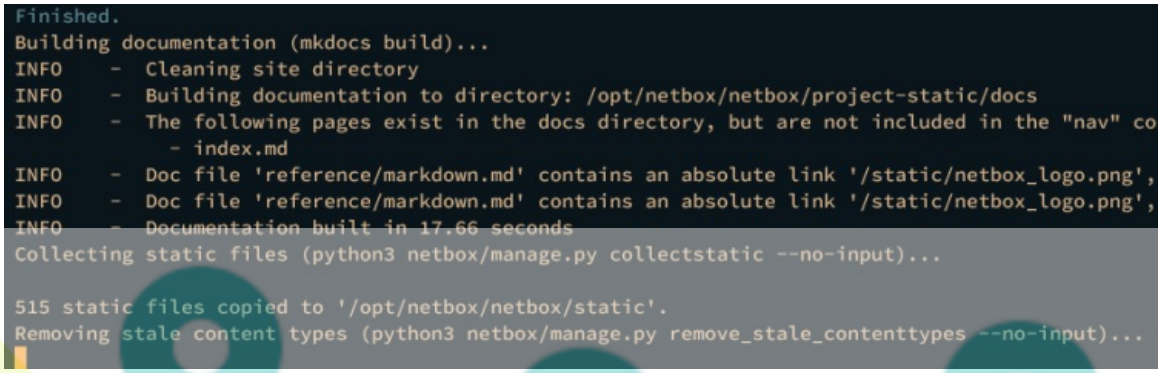
```
sudo -u netbox /opt/netbox/upgrade.sh
```

Below is the similar output you will get during the process.

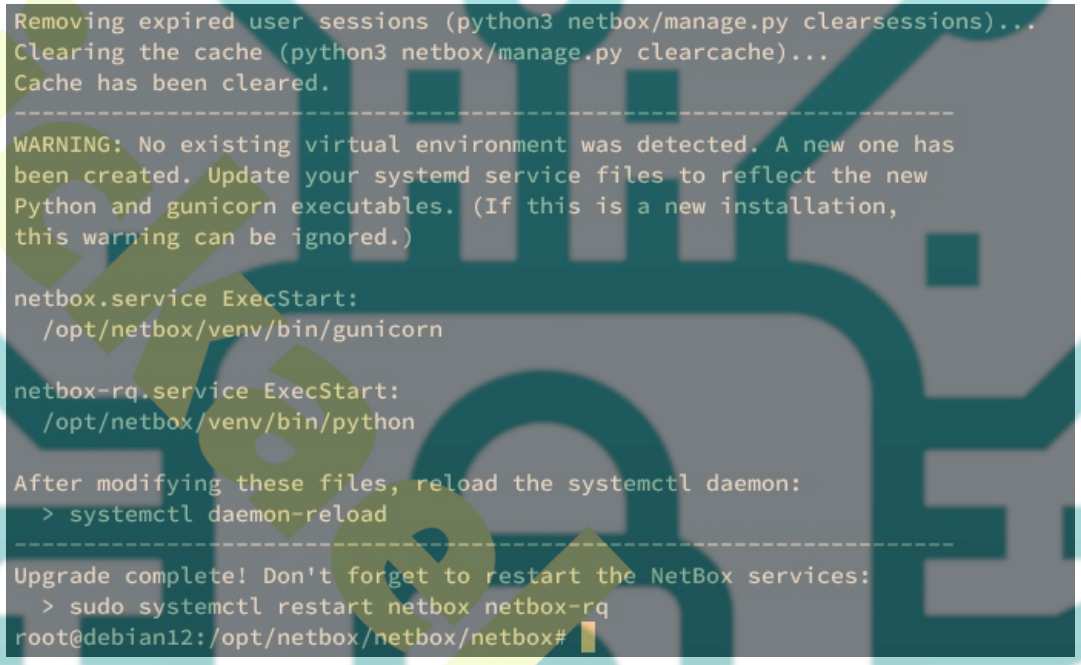


The database migration process.

Generating static files process.



Below is the output when the installation is finished.



After NetBox is configured, you will create an administrator user for NetBox. To do that, log in to the Python virtual environment that is created using the following command.

```
source /opt/netbox/venv/bin/activate
```

Move to the */opt/netbox/netbox* directory and run the *manage.py* script to create a NetBox administrator user.

```
cd /opt/netbox/netbox
python3 manage.py createsuperuser
```

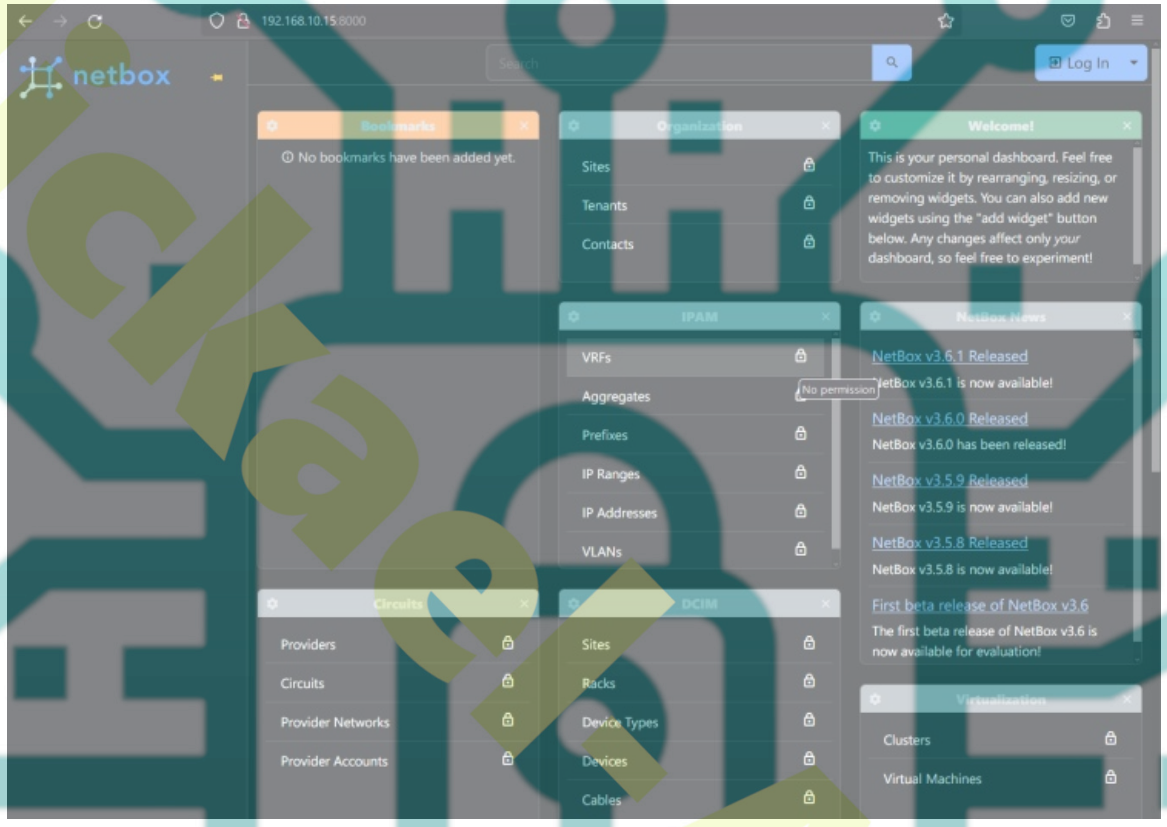
When asked, input your admin email address, username, and password details.

Next, execute the *manage.py* again to verify your NetBox installation. With this, you will run NetBox on your local IP address with port **8000**.

```
python3 manage.py runserver 0.0.0.0:8000 --insecure
```

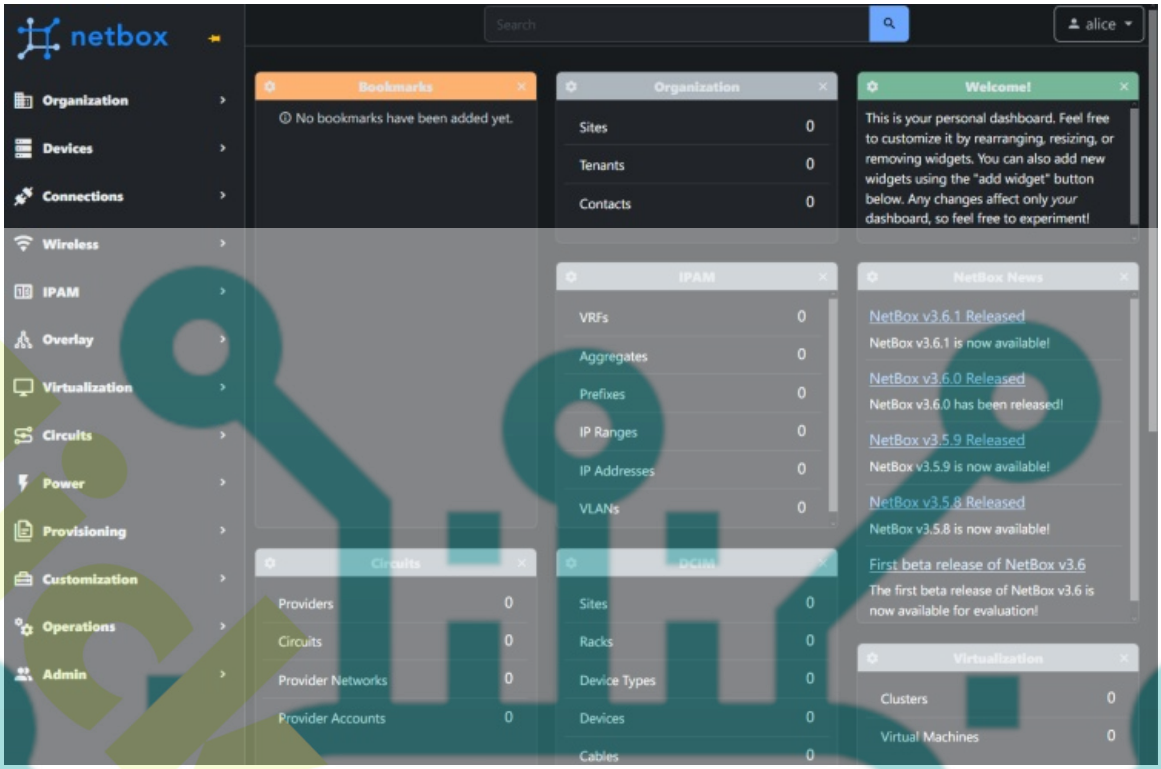
```
(venv) root@debian12:/opt/netbox/netbox#
(venv) root@debian12:/opt/netbox/netbox# python3 manage.py runserver 0.0.0.0:8000 --insecure
Performing system checks...

System check identified no issues (0 silenced).
September 11, 2023 - 13:06:29
Django version 4.2.5, using settings 'netbox.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

Open your web browser and visit your server IP address followed by port 8000, such as http://192.168.10.15:8000/. If your installation is successful, you should get the NetBox IRM index page, and from there, click the **Login** button at the top right.



Input your admin user and password that you've created, then click **Sign In**.

If everything goes well, you should get the NetBox dashboard like the following:



Back to your terminal and press Ctrl+c to terminate the process.

Running NetBox as a Systemd Service

At this point, you've installed NetBox IRM on your Debian machine. To make you easier to manage NetBox, you will be running NetBox as a systemd service, which allows you to control NetBox via the systemctl utility.

Copy the file */opt/netbox/contrib/gunicorn.py* to */opt/netbox/gunicorn.py*, then open the gunicorn.py file using the nano editor command below.

```
sudo -u netbox cp /opt/netbox/contrib/gunicorn.py /opt/netbox/gunicorn.py
sudo -u netbox nano /opt/netbox/gunicorn.py
```

Change the bind option to the following. This will run your NetBox installation in localhost port **8001** via gunicorn.

```
bind = '127.0.0.1:8001'
```

Save and close the file when finished.

Next, copy the systemd service files for NetBox to the */etc/systemd/system/* directory. This will copy the service file *netbox*, *netbor-rq*, and *netbook-housekeeping* to */etc/systemd/system/* directory. Then, reload the systemd manager to apply the new changes on your system.

```
sudo cp -v /opt/netbox/contrib/*.service /etc/systemd/system/
sudo systemctl daemon-reload
```

Now you can start and enable both netbox and netbox-rq service using the *systemctl* command below. After executing the command, your NetBox installation will be running in the background as a systemd service.

```
sudo systemctl start netbox netbox-rq netbox-housekeeping
sudo systemctl enable netbox netbox-rq netbox-housekeeping
```

Lastly, verify both netbox and netbox-rq service using the following command.

```
sudo systemctl status netbox
sudo systemctl status netbox-rq
```

The following output indicates that the netbox service is running and enabled.

```
root@debian12:~#
root@debian12:~# sudo systemctl status netbox
● netbox.service - NetBox WSGI Service
     Loaded: loaded (/etc/systemd/system/netbox.service; enabled; preset: enabled)
     Active: active (running) since
       Docs: https://docs.netbox.dev/
   Main PID: 24554 (gunicorn)
      Tasks: 6 (limit: 4642)
     Memory: 556.7M
        CPU: 11.694s
     CGroup: /system.slice/netbox.service
             ├─24554 /opt/netbox/venv/bin/python3 /opt/netbox/venv/bin/gunicorn --pid /var/
             ├─24557 /opt/netbox/venv/bin/python3 /opt/netbox/venv/bin/gunicorn --pid /var/
             ├─24558 /opt/netbox/venv/bin/python3 /opt/netbox/venv/bin/gunicorn --pid /var/
             ├─24559 /opt/netbox/venv/bin/python3 /opt/netbox/venv/bin/gunicorn --pid /var/
             ├─24560 /opt/netbox/venv/bin/python3 /opt/netbox/venv/bin/gunicorn --pid /var/
             └─24561 /opt/netbox/venv/bin/python3 /opt/netbox/venv/bin/gunicorn --pid /var/
```

The below output confirms that the netbox-rq service is running and enabled.

```
root@debian12:~#
root@debian12:~# sudo systemctl status netbox-rq
● netbox-rq.service - NetBox Request Queue Worker
     Loaded: loaded (/etc/systemd/system/netbox-rq.service; enabled; preset: enabled)
     Active: active (running) since
       Docs: https://docs.netbox.dev/
   Main PID: 24555 (python3)
      Tasks: 3 (limit: 4642)
     Memory: 164.8M
        CPU: 6.552s
     CGroup: /system.slice/netbox-rq.service
             ├─24555 /opt/netbox/venv/bin/python3 /opt/netbox/netbox/manage.py rqworker
             └─24587 /opt/netbox/venv/bin/python3 /opt/netbox/netbox/manage.py rqworker
```

Configuring Apache as a Reverse Proxy

In the following step, you will configure Apache2 as a reverse proxy. Before that, ensure that you have a domain name pointed to your server IP address, you can also use a local domain name.

If you're using a local domain, you can run the following openssl command to generate SSL certificates. If you're using the public domain, you can use Certbot to generate SSL certificates from Letsencrypt.

```
openssl req -x509 -newkey rsa:4096 -sha256 -days 365 \
  -nodes -keyout /etc/ssl/private/netbox.key -out /etc/ssl/certs/netbox.crt -subj "/CN=netbox.hwdomain.io" \
  -addext "subjectAltName=DNS:netbox.hwdomain.io,IP:192.168.10.15"
```

Now copy the Apache virtual host configuration example for NetBox to */etc/apache2/sites-available/netbox.conf*. Then, modify the file */etc/apache2/sites-available/netbox.conf* using the following nano editor command.

```
sudo cp /opt/netbox/contrib/apache.conf /etc/apache2/sites-available/netbox.conf
sudo nano /etc/apache2/sites-available/netbox.conf
```

Change the domain name with your domain and change the path of SSL/TLS certificates with the proper path file.

```
<VirtualHost *:80>
    # CHANGE THIS TO YOUR SERVER'S NAME
    ServerName netbox.hwdomain.io

    ...
</VirtualHost>

<VirtualHost *:443>
    ProxyPreserveHost On

    # CHANGE THIS TO YOUR SERVER'S NAME
    ServerName netbox.hwdomain.io

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/netbox.crt
    SSLCertificateKeyFile /etc/ssl/private/netbox.key

    ...
</VirtualHost>
```

Save and exit the file when finished.

Now run the following command to enable some Apache2 modules that are needed for NetBox.

```
sudo a2enmod ssl proxy proxy_http headers rewrite
```

After that, execute the following command activate the virtual host file netbox.conf and verify your Apache2 configurations to ensure that you've proper syntax.

```
sudo a2ensite netbox.conf
sudo apachectl configtest
```

If you've proper Apache2 syntax, you should get the output **Syntax OK**.

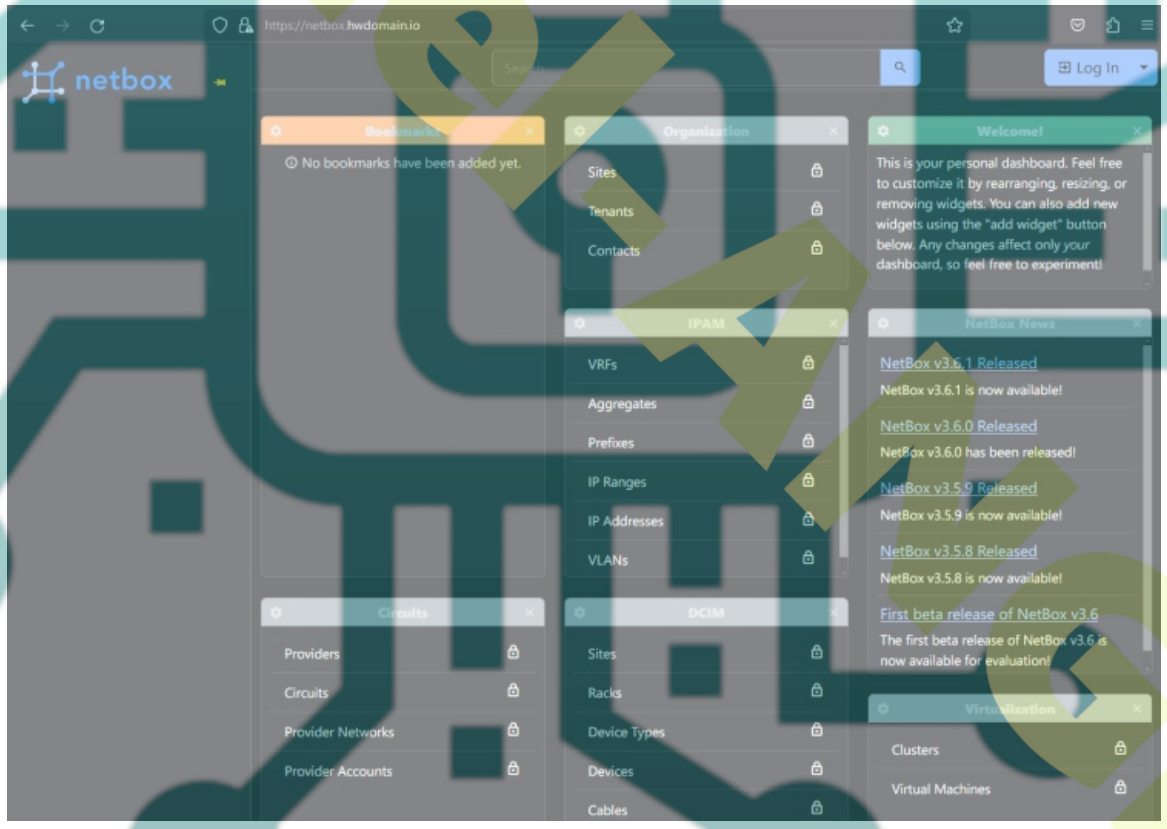
```
root@debian12:~#
root@debian12:~# sudo a2ensite netbox
Enabling site netbox.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@debian12:~# sudo apachectl configtest
AH00558: apache2: Could not reliably determine the serve
s this message
Syntax OK
root@debian12:~# sudo systemctl restart apache2
root@debian12:~#
```

Now run the following command to restart the Apache2 service and apply the changes that you've made.

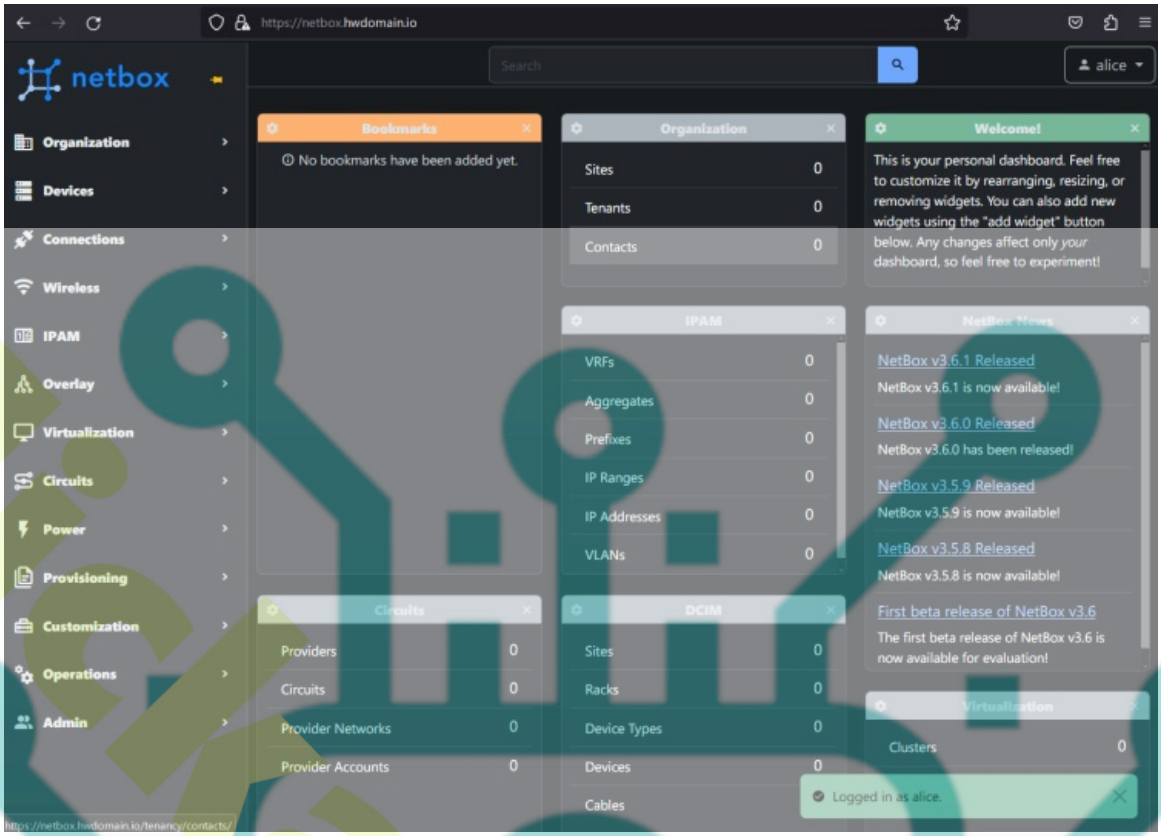
```
sudo systemctl restart apache2
```

Your NetBox installation should be accessible via the domain name.

Open up your web browser and visit the domain name of your NetBox installation, such as https://netbox.hwdomain.io/. If everything goes well, you should see the NetBox index page like the following:



After logging in, you should see the NetBox IRM dashboard running with the domain name under the Apache2 reverse proxy.



Conclusion

In conclusion, you've now installed NetBox IRM on the Debian 12 server with the PostgreSQL database server and Apache2 web server used as a reverse proxy. You've also secured your NetBox installation via SSL/TLS certificates.