comment installer-pydio-on-debian-12

Pydio Cells is a self-hosted Document Sharing and Collaboration platform. It also gives you full control of your document-sharing environment. The Pydio Cells is a fast performance, can handle huge file transfer sizes, and provides advanced workflow automation.

In the following guide, I will walk you through the installation of Pydio Cells as a file-sharing and collaboration tool on a Debian 12 server. You will install Pydio Cells with the MariaDB database server and Apache2 reverse proxy. You'll also secure the installation with SSL/TLS certificates that you will generate via Certbot and Letsencrypt.

Prerequisites

Before moving on, gathers the following:

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

Installing Dependencies

The Pydio Cells is an open-source file sharing, management, and collaboration. Before installing it, you must install dependencies such as MariaDB for the database server and Apache2 for the reverse proxy. You will also install Certbot for generating SSL/TLS certificates to secure your installation.

First, refresh your Debian package index using the following *apt update* command.

```
sudo apt update
```

Now install dependencies via the *apt install* command below. You will install the MariaDB server that will be used as the database for Pydio Cells, the Apache2 web server as a reverse proxy, and Certbot for generating SSL/TLS certificates from Letsencrypt.

```
sudo apt install mariadb-server apache2 certbot wget
```

Type y to confirm the installation and press ENTER.

Once dependencies are installed, verify the apache2 service using the following *systemctl* command. Ensure that the apache2 service is enabled and running.

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

The following output indicates that apache2 is enabled and running.

```
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: 5530 (apache2)
      Tasks: 55 (limit: 4642)
     Memory: 12.9M
        CPU: 46ms
     CGroup: /system.slice/apache2.service
             ├─5530 /usr/sbin/apache2 -k start
             ├─5531 /usr/sbin/apache2 -k start
             └─5532 /usr/sbin/apache2 -k start
```

Lastly, verify the mariadb service via the following *systemctl* command.

```
sudo systemctl is-enabled mariadb
sudo systemctl status mariadb
```

The output should be similar, which indicates the mariadb service is running and enabled.

```
root@debian12:~#
root@debian12:~# sudo systemctl is-enabled mariadb
enabled
root@debian12:~# sudo systemctl status mariadb
● mariadb.service - MariaDB 10.11.3 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; preset: enabled)
     Active: active (running) since
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
   Main PID: 7189 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 14 (limit: 4642)
     Memory: 168.5M
        CPU: 662ms
     CGroup: /system.slice/mariadb.service
             └─7189 /usr/sbin/mariadbd
```

Configuring MariaDB Server

In the following step, you will be securing your MariaDB server installation via the mariadb-secure-installation utility. Then, you will create a new database and user for Pydio Cells.

Execute the *mariadb-secure-installation* command below to start configuring the MariaDB server.

```
sudo mariadb-secure-installation
```

The setup process will require you to input Y to confirm the new settings or n for no. Below are some of the MariaDB server configurations you will be asked for:

- Switch local authentication to unix_socket? Input n.
- Set up the new MariaDB root password? Input y to confirm, then type the new password for your MariaDB server deployment.
- Remove anonymous user? Input y to confirm.
- Remove the default database test from the deployment?. Input y to confirm.
- Disallow MariaDB root login from remote connections? Input y to confirm.
- Reload table privileges and apply the changes? Input y and press ENTER.

After the MariaDB is secured, you will create a new database and user for the Pydio Cells installation. To do that, you must log in to the MariaDB server.

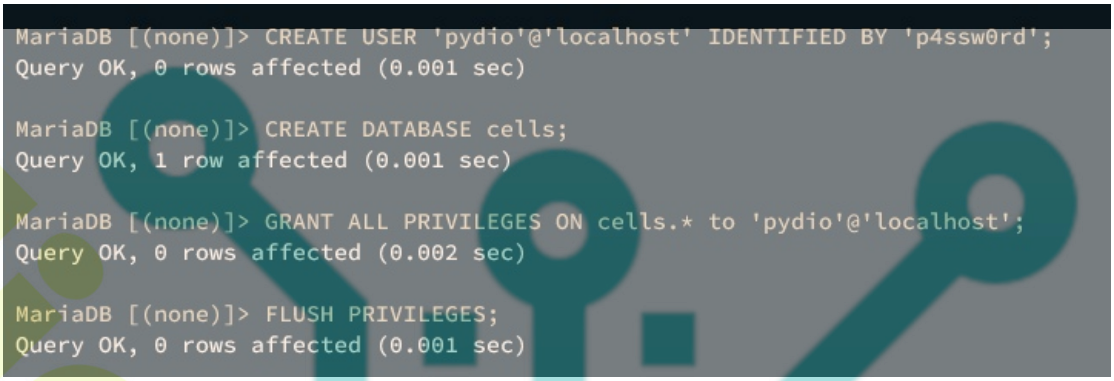
Execute the following mariadb command to log in to the MariaDB server. Input your MariaDB root password when prompted.

```
sudo mariadb -u root -p
```

Once logged in, run the following queries to create a new database **cells**, a user **pydio** with the password **p4ssw0rd**.

Then, allow user **pydio** to access the database **cells**.

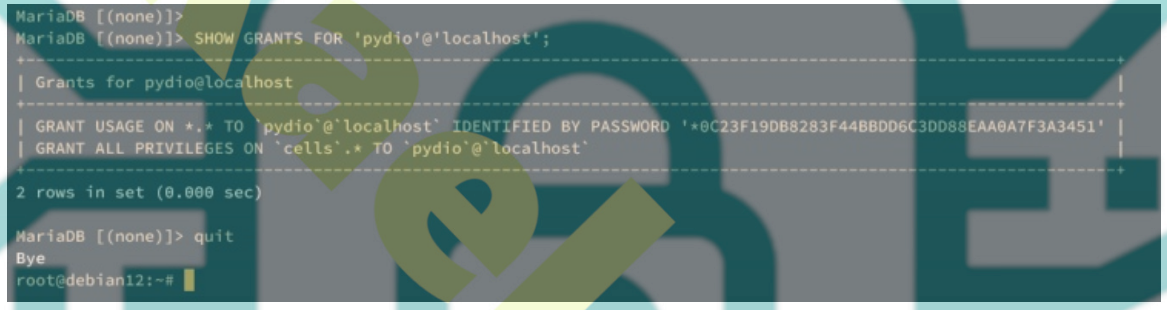
```
CREATE DATABASE cells;
CREATE USER 'pydio'@'localhost' IDENTIFIED BY 'p4ssw0rd';
GRANT ALL PRIVILEGES ON cells.* to 'pydio'@'localhost';
FLUSH PRIVILEGES;
```



Next, run the following query to ensure the user **pydio** can access the database **cells**.

```
SHOW GRANTS FOR 'pydio'@'localhost';
```

The following output shows you the user **pydio** has privileges to access and manage the database **cells**.



Type **quit** to exit from the MariaDB server.

Installing Pydio Cells

After configuring the MariaDB server, you will install the Pydio Cells via static binary file. And before that, you must prepare your system by creating a new dedicated user, setting up a data directory, and creating some system environment variables that are needed by Pydio Cells.

Setting Up User and Environment Variables

First, create a new user **pydio** using the following command.

```
sudo useradd -m -s /bin/bash pydio
```

Now create a new data directory */var/cells* for your Pydio Cells installation and change the ownership to the user **pydio**.

```
sudo mkdir -p /opt/pydio/bin /var/cells
sudo chown -R pydio: /opt/pydio /var/cells
```

Next, run the following command to create new environment variables configuration */etc/profile.d/cells-env.sh* and change the permission to *0755* to make it executable. The environment variable **CELLS_WORKING_DIR** for the data directory, **CELLS_BIND** to determine which IP address and port Pydio Cells will be running, and the **CELLS_EXTERNAL** is the domain name of your Pydio Cells installation.

```
sudo tee -a /etc/profile.d/cells-env.sh << EOF
export CELLS_WORKING_DIR=/var/cells
export CELLS_BIND=https://127.0.0.1:8080
export CELLS_EXTERNAL=https://cells.hwdomain.io
EOF
sudo chmod 0755 /etc/profile.d/cells-env.sh
```

```
root@debian12:~#
root@debian12:~# sudo useradd -m -s /bin/bash pydio
root@debian12:~#
root@debian12:~# sudo mkdir -p /opt/pydio/bin /var/cells
root@debian12:~# sudo chown -R pydio: /opt/pydio /var/cells
root@debian12:~#
root@debian12:~# sudo tee -a /etc/profile.d/cells-env.sh << EOF
export CELLS_WORKING_DIR=/var/cells
export CELLS_BIND=https://localhost:8080
export CELLS_EXTERNAL=https://cells.hwdomain.io
EOF
export CELLS_WORKING_DIR=/var/cells
export CELLS_BIND=https://localhost:8080
export CELLS_EXTERNAL=https://cells.hwdomain.io
root@debian12:~#
root@debian12:~# sudo chmod 0755 /etc/profile.d/cells-env.sh
root@debian12:~#
```

Now log in as the user pydio and verify environment variables *CELLS_WORKING_DIR, CELLS_BIND, and CELLS_EXTERNAL*.

```
su - pydio
```

```
echo $CELLS_WORKING_DIR
echo $CELLS_BIND
echo $CELLS_EXTERNAL
```

If successful, you should see each environment variable will be matched with the file */etc/profile.d/cells-env.sh*.

```
root@debian12:~#
root@debian12:~# su - pydio
pydio@debian12:~$
pydio@debian12:~$ echo $CELLS_WORKING_DIR
/var/cells
pydio@debian12:~$ echo $CELLS_BIND
https://localhost:8080
pydio@debian12:~$
pydio@debian12:~$ echo $CELLS_EXTERNAL
https://cells.hwdomain.io
pydio@debian12:~$
pydio@debian12:~$
```

Downloading and Installing Pydio Cells

Execute the following command to download the binary static file of Pydio Cells to */opt/pydio/bin/cells*.

```
export distribId=cells
wget -O /opt/pydio/bin/cells https://download.pydio.com/latest/${distribId}/release/{latest}/linux-amd64/${distribId}
```

Once the Pydio Cells are downloaded, make it executable using the following command. Then, type exit to log out from user **pydio**.

```
chmod a+x /opt/pydio/bin/cells
exit
```

Now run the following command to allow cells to bind in the privileged ports. Then, create a symlink for the */opt/pydio/bin/cells* command to */usr/local/bin/cells*.

```
sudo setcap 'cap_net_bind_service=+ep' /opt/pydio/bin/cells
sudo ln -s /opt/pydio/bin/cells /usr/local/bin/cells
```

Now log in again as a pydio user and check the binary file of cells. Then, verify your current cells version.

```
su - pydio
```

```
which cells
cells version
```

You should see the cells binary file is located at /usr/local/bin/cells and the cells version that was installed is **4.2.5**.

```
pydio@debian12:~$
pydio@debian12:~$ which cells
/usr/local/bin/cells
pydio@debian12:~$
pydio@debian12:~$ cells version
Pydio Cells Home Edition
 Version:       4.2.5
 Built:         17 Jul 23 09:04 +0000
 Git commit:    0b98e8a5e7976e77964fab784b123ac55971aa04
 OS/Arch:       linux/amd64
 Go version:    go1.19.11
pydio@debian12:~$
```

Configuring Pydio Cells

With the Pydio Cells binary file installed, you will start configuring it, which can be done via CLI (command-line interface) or web browser. As for this case, you will configure Pydio Cells from the command-line terminal, you will set up the database, create the admin user, then will create a new systemd service file to run Pydia Cells in the background.

Run the *cells* command below to start configuring the Pydio Cells installation. The parameter *--cli* allow you to configure Pydio Cells from your terminal with an interactive environment.

```
cells configure --cli
```

Below some configurations that you will be asked for:

- For the database configuration, select via TCP and input the database host as localhost, port with default 3306, database name cells, the user pydio, and the password.
- Input n when prompted about the MongoDB configuration for high-availability Cells installation.
- Input the new admin user and password for your Pydio Cells installation.
- For the storage configuration, select the option /var/cells/data.

When the configuration process is finished, you should get an output Installation Finished like the following:

```
pydio@debian12:~$
pydio@debian12:~$ cells configure --cli

Welcome to Pydio Cells Home Edition installation
Pydio Cells Home Edition (v4.2.5) will be configured to run on this machine.
Make sure to prepare access and credentials to a MySQL 5.6+ (or MariaDB equivalent) server.
Pick your installation mode when you are ready.


## Database Connection
✓ TCP
Database Hostname: localhost
Database Port: 3306
Database Name: cells
Database User: pydio
Database Password (leave empty if not needed): ********
✓ Successfully connected to the database
✗ Do you wish to configure a MongoDB connection (better for scalability and required for clustering deployment):
... Starting installation now
... Created main database

## Administrative User Configuration
Admin Login: admin
Admin Password: **********
✓ Confirm Password: **********

## Default storage location
✓ It's ok for me, use default location

## Applying configuration
... Starting installation now
... Generating secrets
... Created main database
... Created default datasources
Adding admin credentials to config, to be inserted at next start
... Creation of logs directory
✓ Configuration done

## Software is ready to run!
Cells will be accessible through the following URLs:
https://cells.hwdomain.io, https://127.0.0.1:8080
Edit these URLs by running 'cells configure sites' command.

Now use 'cells start' to start the server.


✓ Installation Finished
```

Now that you've configured Pydio Cells, the next step you will set up cells to run in the background as a systemd

service. This makes you easier to manage cells via the systemctl command utility.

Use the following nano editor command to create a new systemd service file */etc/systemd/system/cells.service*.

```
sudo nano /etc/systemd/system/cells.service
```

Insert the following configuration and be sure to change some environment variables *CELLS_WORKING_DIR, CELLS_BIND, and CELLS_EXTERNAL* within the below configuration.

```
[Unit]
Description=Pydio Cells
Documentation=https://pydio.com
Wants=network-online.target
After=network-online.target
AssertFileIsExecutable=/opt/pydio/bin/cells

[Service]
User=pydio
Group=pydio
PermissionsStartOnly=true
AmbientCapabilities=CAP_NET_BIND_SERVICE
ExecStart=/opt/pydio/bin/cells start
Restart=on-failure
StandardOutput=journal
StandardError=inherit
LimitNOFILE=65536
TimeoutStopSec=5
KillSignal=INT
SendSIGKILL=yes
SuccessExitStatus=0
WorkingDirectory=/home/pydio

# Add environment variables
Environment=CELLS_WORKING_DIR=/var/cells
Environment=CELLS_BIND=https://127.0.0.1:8080
Environment=CELLS_EXTERNAL=https://cells.hwdomain.io

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

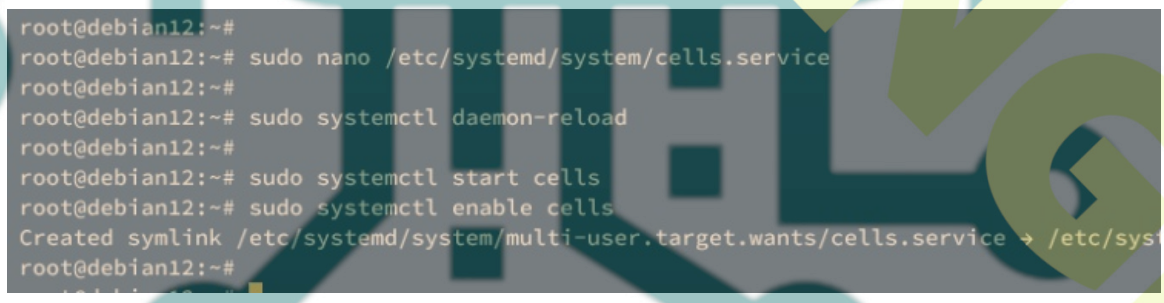
When finished, save the file and exit the editor.

Now run the following *systemctl* command to reload the systemd manager and apply the new systemd service.

```
sudo systemctl daemon-reload
```

Start and enable the cells service using the *systemctl* command below. This command will add the cells service to start automatically at system boot.

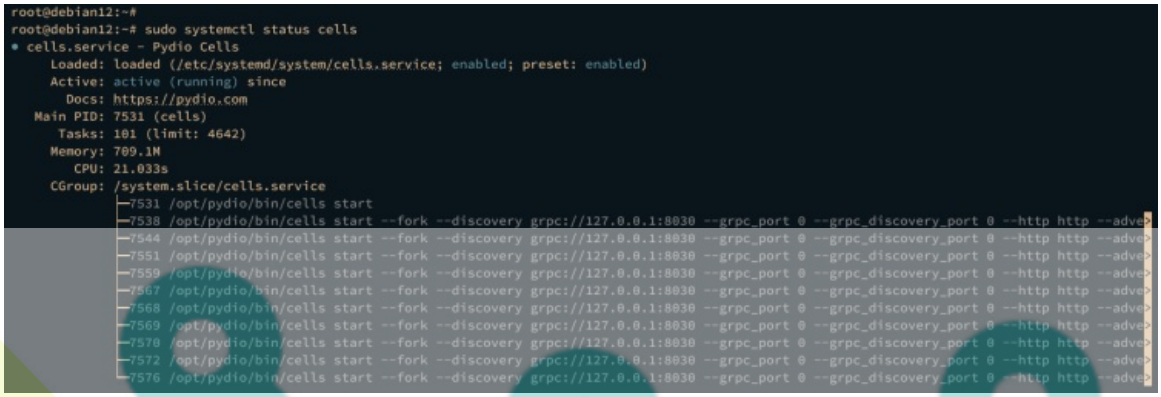
```
sudo systemctl start cells
sudo systemctl enable cells
```



Verify the cells service status using the command below. The Pydio Cells should be running on 127.0.0.1 with port **8080**, as defined within the **CELLS_BIND** environment variable.

```
sudo systemctl status cells
```

If running, you should get an output such as **active (running)**.



Configuring Apache2 as a Reverse Proxy

At this point, the Pydio Cells are up and running in the background on localhost with default port xxx. And this step, you will be configuring Apache2 as a reverse proxy for your Pydio Cells application. Also, you will generate new SSL/TLS certificates for your domain name, so be sure that you've prepared the domain name and pointed to the server IP address.

First, execute the *a2enmod* command below to enable some Apache2 extensions that will be used as a reverse proxy.

```
sudo a2enmod rewrite proxy proxy_http proxy_wstunnel http2 proxy_http2
```

Then run the following command to create a new directory */var/www/html/cells/public_html* and change the ownership to *www-data* user. This directory will be used for the verification when generating Letsencrypt certificates.

```
sudo mkdir -p /var/www/html/cells/public_html
sudo chown -R www-data:www-data /var/www/html/cells/public_html
```

After that, run the *certbot* command below to generate new SSL/TLS certificates for your Pydio Cells domain name. Be sure to change the email address and the domain name with your information.

```
sudo certbot certonly --agree-tos --email user@email.com --no-eff-email --webroot -w /var/www/html/cells/public_html -
d cells.hwdomain.io
```

When the process is complete, your SSL/TLS certificates will be available at the */etc/letsencrypt/live/domain.com* directory.

Next, create a new Apache2 virtual host configuration */etc/apache2/sites-available/cells.conf* using the following nano editor command.

```
sudo nano /etc/apache2/sites-available/cells.conf
```

Insert the following configuration and be sure to change the domain name and the path of SSL/TLS certificates with your information.

```
<VirtualHost *:80>
    ServerName cells.hwdomain.io

    RewriteEngine On
    RewriteCond %{HTTPS} off
    RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}

    RewriteCond %{SERVER_NAME} =cells.hwdomain.io
    RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>

<VirtualHost *:443>
    ServerName cells.hwdomain.io
    AllowEncodedSlashes On
    RewriteEngine On

    # be aware of this
    # Allow reverse proxy via self-signed certificates
    SSLProxyEngine On
    SSLProxyVerify none
    SSLProxyCheckPeerCN off
    SSLProxyCheckPeerName off
    SSLProxyCheckPeerExpire off

    ## The order of the directives matters.
    # If Cells is not running with https, consider using ws instead of wss
    ProxyPassMatch "/ws/(.*)" wss://localhost:8080/ws/$1 nocanon
```

```
    ## This rewrite condition is required if using Cells-Sync
    # RewriteCond %{HTTP:Content-Type} =application/grpc [NC]
    # RewriteRule /(.*) h2://localhost:8080/$1 [P,L]

    ProxyPass "/" "https://127.0.0.1:8080/"
    ProxyPassReverse "/" "https://127.0.0.1:8080/"

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    SSLCertificateFile /etc/letsencrypt/live/cells.hwdomain.io/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/cells.hwdomain.io/privkey.pem
    #Include /etc/letsencrypt/options-ssl-apache.conf
</VirtualHost>
```

Save the file and exit the editor when finished.

Now run the command below to activate the virtual host file cells.conf and verify Apache syntax for error. If you have proper Apache2 syntax, you should get an output **Syntax OK**.

```
sudo a2ensite cells.conf
sudo apachectl configtest
```

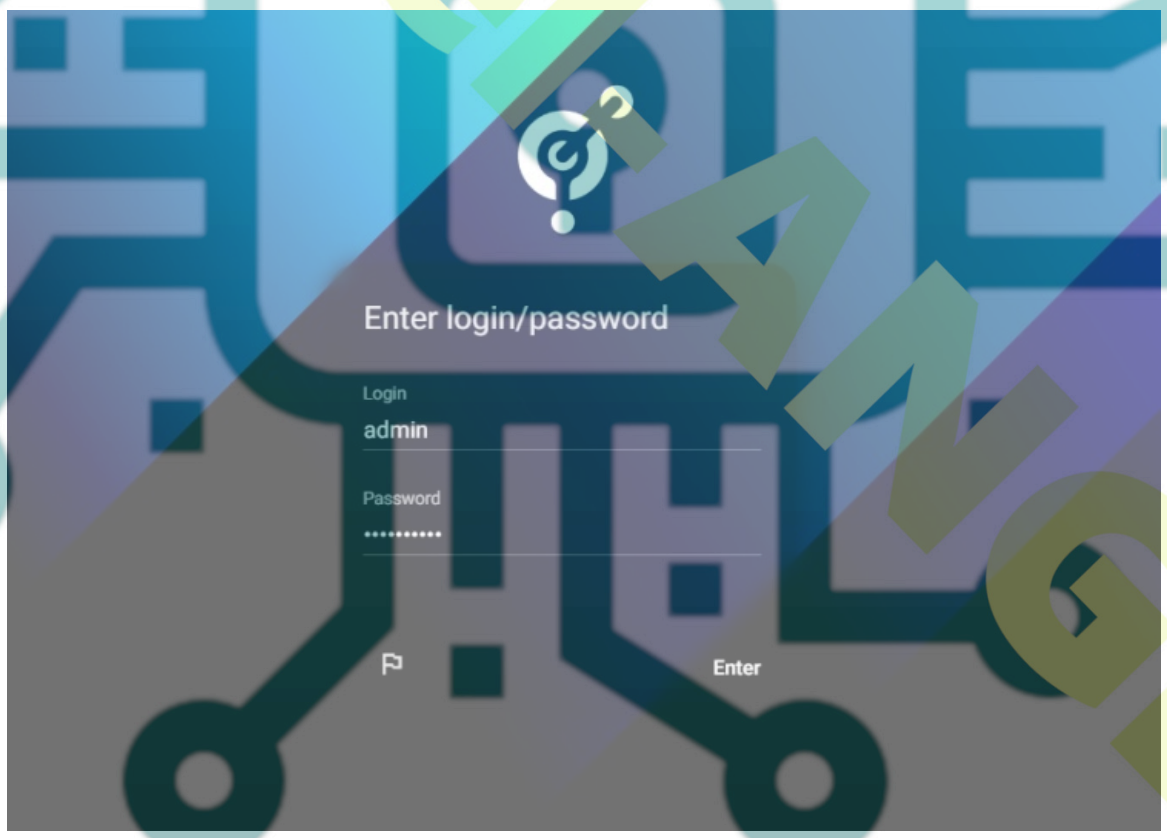
Lastly, run the following systemctl command to restart the apache2 service and apply the changes. With this, your Pydio Cells should be accessible via a secure HTTPS connection of Apache2 reverse proxy.

```
sudo systemctl restart apache2
```

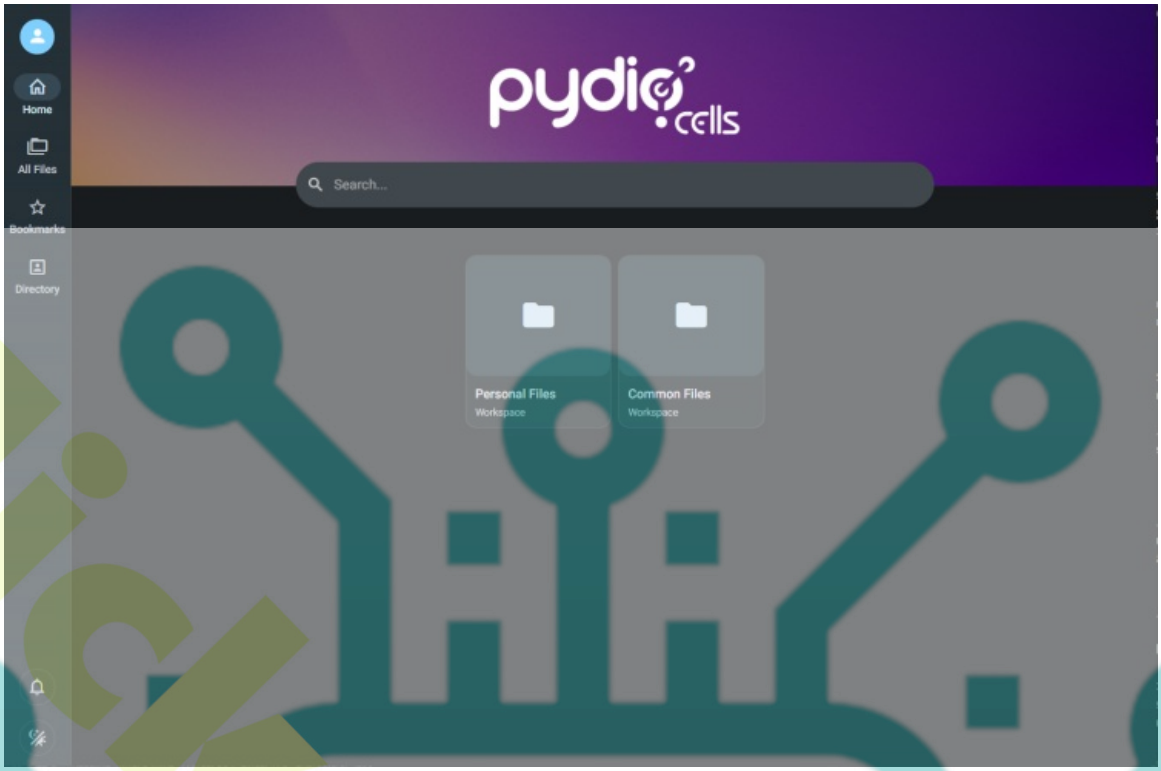
Accessing Pydio Cells Installation

Launch your web browser and visit the domain name of the Pydio Cells installation, such as https://cells.hwdomain.io/. If everything goes well, you should be redirected to the Pydio Cells login page.

Input the admin user and password that you've created during the configuration process, then click **Enter**.



If successful, you should see the Pydio Cells user dashboard like this:



Click on the **Personal Files** workspace and you should get the Pydio Cells file manager. Click the New button and upload a new file to ensure that you can upload files to Pydio Cells.

Conclusion

Following this guide, you've installed Pydio Cells on the Debian 12 server. You've installed Pydio Cells with MariaDB database server and Apache2 reverse proxy, and on top of that, you've also secured your Pydio Cells installation with SSL/TLS certificates generated from Letsencrypt. From here, you can now use Pydio Cells for your document and file management, collaboration, and sharing.
