

Generic Software Installation

This guide is provided for your convenience. The guide is provided as-is and may not be updated or maintained by Ultra.cc. Unofficial support may be offered via [Discord](#) only and at the sole discretion of Ultra.cc staff. Use at your own risk and only proceed if you are comfortable troubleshooting.

This guide covers some generic ways to install custom third-party applications on your Ultra.cc service. It is important to remember that these installation procedures are generic and additional steps may be needed for a successful installation. As these procedures are made for custom applications not supported by Ultra.cc, we cannot provide any assistance regarding the installation or issues that may arise once an application has been installed.

If you want us to add an application to the [User Control Panel](#) and officially support it, you can submit a request on our [Feedback site](#), and we will consider it. Please ensure that you do not add duplicate requests and instead add your vote if the application has already been requested.

Installation

Many applications can be installed with a click of a button on the [UCP](#). However, if you want to install an application not included on the UCP, you can do so. As long as you adhere to the [Terms of Service](#) and do not break the [Fair Usage Policy](#), you are free to install pretty much any software on your Ultra.cc service.

While selecting a port for your custom application, select one within the port range assigned to your service; do **NOT** use the default port the application suggests. It is strictly prohibited to use ports outside of your range. More info can be found [here](#).

Important information regarding the installation of custom third-party applications:

- You cannot install an application that requires sudo or root privileges, including:
 - Docker images (look for the *local installation* instructions instead)
 - The placing of files outside your home directory
- Be mindful of application resource usage and IO utilization. See [this guide](#).
- Only use ports assigned to your service. More info can be found [here](#).
- Custom third-party applications are not officially supported by Ultra.cc staff.
- Always read the documentation associated with the software you are installing.

Before proceeding with any of the below installation procedures, you need to connect to your Ultra.cc service via SSH.

- Connect to your Ultra.cc slot via SSH, see guide [here](#).

Unofficial Language Installers

While installing a third-party application, you might encounter some programming language dependencies in the install instructions, such as [Python](#), Rust, Node.js, etc. These programming languages are often not included in a base Linux installation and must be manually installed. This is the case for our servers, as they run on a base Debian distribution.

To make things a bit easier, we offer a selection of unofficial install scripts, which will install the language of your choice on your Ultra service. You can find the complete list of the programming languages that are available [here](#).

Compile from Source

Below you will find generic instructions for how to compile from source. Some applications have specific instructions or required dependencies, so always check the documentation or website of the application you are installing.

- Download the source. Files can be downloaded using various utility tools, such as `wget`, `curl`, `git`, etc.

```
wget https://example-url.com/appname-1.23.tar.gz
```

- Extract the source.

```
tar xvf appname-1.23.tar.gz
```

- Navigate into the extracted directory

```
cd appname-1.23
```

- Configure the application. See app-specific documentation for additional configurations.

```
./configure --prefix="$HOME/bin" && make
```

- Install the software

```
make install
```

- Add the install directory to the path to enable global application execution.

```
echo "PATH=$HOME/.local/bin:$HOME/bin:$PATH" >> ~/.profile && source ~/.profile
```

Pre-built Binaries

Some software is available as pre-compiled binaries and do not have to be built within your Ultra.cc service.

- Download the binary.

```
wget https://example-url.com/appname-1.23.tar.gz
```

- Extract the binary.

```
tar xvzf appname-1.23.tar.gz
```

- Move the binary to a directory within your shell environment PATH.

```
mv appname ~/bin/
```

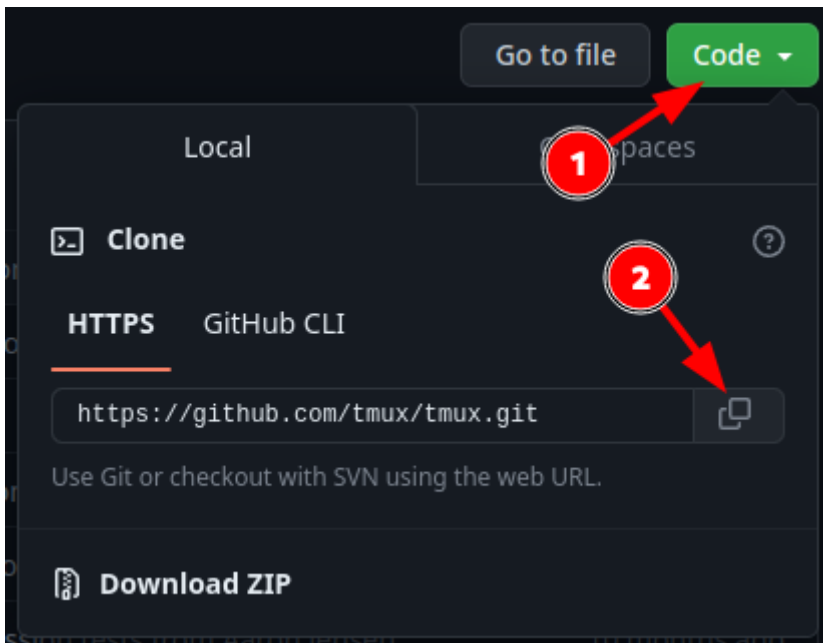
Once the binary has been moved to a directory within your shell environment PATH, you can run the application by executing the filename of the binary. With the name example we are using in this guide, it would look like this:

```
appname
```

Cloning a Repo

Application software can also be installed by cloning a repository. The most popular repository library is [GitHub](#), but there are also others like [GitLab](#) for example.

- While in the GitHub repository, in the top right corner, click the green **Code** button.
- Copy the URL for the repository by clicking the **Copy** button, as shown in the below image.



- Change the current working directory of your Ultra.cc shell to the location where you want to clone the repo. This is usually the root of your home directory.

```
cd
```

- Clone the repo. Do note, `AUTHOR` and `EXAMPLE-REPO` would be replaced with what matches the repo you are cloning.

```
git clone https://github.com/AUTHOR/EXAMPLE-REPO
```

- Next, `cd` into the cloned repo directory. Do note, `EXAMPLE-REPO` would be replaced with what matches the repo you are cloning.

```
cd EXAMPLE-REPO
```

Inside the cloned repo directory, you will find all the files of the repository, and you are free to execute any scripts or binaries that are included (subject to our [Terms of Service](#)).

Python Applications

In this section, we will show you how to install Python and how to install Python packages with `pip`. This is useful as cloned repos occasionally require you to install a Python package. However, before taking further action, ensure you have Python installed on your Ultra.cc service.

- To install Python, see [this guide](#).

Install Python Package

Once you have followed the above guide and successfully installed Python on your Ultra.cc service, you are ready to install Python packages. A complete list of all available Python packages can be found on the [Python Package Index \(PyPI\)](#).

- To install a Python package, execute the following command:

```
pip install <package-name>
```

- Occasionally a Python application requires you to install multiple packages. This is often handled by a `requirements.txt` file and can be installed with the following command:

```
pip install -r requirements.txt
```

- By executing the above command, multiple Python packages will be automatically installed.

Systemd Service

While binaries and scripts can be manually executed or setup as a cron job, you can also set up a systemd service and run your application as a background process. This allows you to have more control and easier management of your custom applications.

- Check the documentation of the application being installed as often guidance will be provided for creating a service.
- To create a systemd service, we need to create a systemd service file. A userland systemd service file is stored at `~/.config/systemd/user`.
- Create the systemd service file with the below command, make sure to replace `SERVICE-NAME` with a name of your choice.

```
touch ~/.config/systemd/user/SERVICE-NAME.service
```

- Open the systemd service file with the nano editor:

```
nano ~/.config/systemd/user/SERVICE-NAME.service
```

- Paste the following into the editor:

```
[Unit]
Description=A description of my custom application
After=network-online.target

[Service]
Type=exec
Restart=on-failure
ExecStart=%h/bin/MY-CUSTOM-APPLICATION
ExecStop=/bin/kill -s QUIT $MAINPID
StandardOutput=file: %h/path/to/logs/my-custom-application.log

[Install]
WantedBy=default.target
```

- The above systemd service file is an example of a basic systemd service file.
 - By editing the command after `ExecStart=`, you can create a systemd service file for a binary, script, etc.
 - For a deeper understanding on how a systemd service file works, see [this guide](#).
- Once you have edited the systemd service file to your liking, press `CTRL+x` and `y` to save and exit, press `ENTER` to confirm.
- Next, whenever a change has been made to a systemd service file, a reload of the systemd daemon is required. To initiate the reload, execute the following command:

Notice the `--user` option. It is always required when interacting with `systemctl` on an Ultra.cc service, as leaving it out requires sudo/root privileges.

```
systemctl --user daemon-reload
```

- Next, to enable the systemd service, execute the following command:
 - Be sure to replace `SERVICE-NAME` with the name you previously selected for your systemd service file.

```
systemctl --user enable --now SERVICE-NAME.service
```

- To check if your systemd service has been successfully enabled, execute the following command:

```
systemctl --user is-enabled SERVICE-NAME.service
```

- To check if your systemd service has been successfully executed, execute the following command:

```
systemctl --user status SERVICE-NAME.service
```

- If successful, you should see an output like this:

```
ultradocs@spica: ~$ systemctl --user status service-name.service
```

- service-name.service - A description of my custom application

```
Loaded: loaded (/home/ultradocs/.config/systemd/user/service-name.service; enabled; vendor preset: enabled)
```

```
Active: active (running) since Sun 2024-01-08 10:07:17 CEST; 2 days ago
```

```
Main PID: 71643 (service)
```

```
CGroup: /user.slice/user-1104.slice/user@1104.service/service-name.service
```

```
└─71643 service: master process /home/ultradocs/bin/service -c
```

```
└─71647 service: worker process
```

- Done!

Enabling HTTPS Encryption

If you have installed a custom application on your Ultra.cc service, and assigned one of your unused ports (see [this guide](#)), your application will be accessible via the HTTP protocol. This means that all traffic will be unencrypted. To secure the traffic of your application, you can enable HTTPS encryption via Nginx.

- Create a new configuration file for the custom application:
 - Be sure to replace `APP-NAME` with a name of your choice.

```
touch ~/.apps/nginx/proxy.d/APP-NAME.conf
```

- Next, open the configuration file with the nano editor:

```
nano ~/.apps/nginx/proxy.d/APP-NAME.conf
```

- It is not unusual for app developers to provide a pre-made Nginx template for their application. You should always search the application documentation or the internet for such a template, and amend the below template accordingly.
- If no template can be found, paste the following as-is into the editor:

```
location /baseurl/ {  
    proxy_pass          http://127.0.0.1:PORT;  
    proxy_http_version  1.1;  
    proxy_set_header    X-Forwarded-Host    $http_host;  
}
```

- Next, in the editor, edit the `/baseurl` and `PORT`.
 - The `/baseurl` is the last part of your custom application URL and can be set to anything that is not already used by another application. For example `https://username.hostname.usbx.me/mycustomapp`.
 - Some applications require the baseurl to be added to the application configuration, so check the documentation.
 - The `PORT` is the 5 digit port you have used to configure your custom application. See [this guide](#) for a complete list of the port range assigned to your Ultra.cc service. It is strictly prohibited to use a port outside of your port range, and doing so can lead to a [Fair Usage Policy](#) violation.
- Next, save and exit the editor by pressing `CTRL+x` and `y`, press `ENTER` to confirm.
- Next, ensure you have configured your application to use the same `/baseurl` as you set in the above Nginx configuration file.
- Restart the webserver from the [UCP](#), or by executing `app-nginx restart`.
- Lastly, check if the application is accessible via the `/baseurl` you have set up. For example, go to `https://username.hostname.usbx.me/mycustomapp` to access the application webUI.

Revision #21

Created 2 September 2023 10:55:18 by varg

Updated 5 May 2024 08:48:27 by varg