

Install ZenML Server Linux



How to install pip install zenml["server"] in linux system



To install `zenml[server]` using `pip` on a Linux system, follow these steps:

1. Ensure Python and pip are installed

- Verify if Python is installed:

```
bash

python3 --version
```

If not installed, install Python:

```
bash

sudo apt update
sudo apt install python3 python3-pip -y
```

- Verify `pip` is installed:

```
bash

pip3 --version
```

If `pip` is missing, install it:

```
bash

sudo apt install python3-pip -y
```

2. Set up a virtual environment (optional but recommended)

Create and activate a Python virtual environment:

```
bash

python3 -m venv zenml-env
```

```
source zenml-env/bin/activate
```

3. Upgrade pip and setuptools

Ensure `pip` and `setuptools` are up-to-date:

```
bash

pip install --upgrade pip setuptools
```

4. Install `zenml[server]`

Use the following command to install `zenml` with server dependencies:

```
bash

pip install "zenml[server]"
```

The quotes around `zenml[server]` ensure proper handling of special characters in some shells.

5. Verify the installation

After installation, check if `zenml` is installed correctly:

```
bash

zenml --version
```

Notes:

- If you face permission issues, consider using a virtual environment or install with `--user`.
- Ensure your system has all required dependencies for `zenml` and its server components, as some might require additional libraries or tools.
- If using a virtual environment, remember to activate it whenever working with `zenml`.



How to freeze requirement file



To freeze a `requirements.txt` file in Python, follow these steps:

1. Activate Your Virtual Environment (if applicable)

If you are using a virtual environment, activate it:

```
bash

source <path-to-your-venv>/bin/activate
```

2. Install Required Packages

Ensure all the packages you want are installed in your environment:

```
bash

pip install <package-name>
```

3. Generate the `requirements.txt` File

Use the `pip freeze` command to list all installed packages and their versions, and save it to a file:

```
bash

pip freeze > requirements.txt
```

This will create a `requirements.txt` file in the current directory.

4. Verify the `requirements.txt` File

Open the `requirements.txt` file to confirm it contains the correct package names and versions:

```
bash

cat requirements.txt
```

5. Customize if Needed

You can manually edit the file to remove unnecessary dependencies or add specific version constraints.

Example of `requirements.txt`:

```
plaintext

numpy==1.24.0
pandas==1.5.3
scikit-learn==1.1.3
```

6. Use the `requirements.txt` File

To recreate the environment in another system, use:

```
bash

pip install -r requirements.txt
```

This ensures the exact same package versions are installed.



How to unhide file in linux



In Linux, hidden files and directories typically have names that start with a dot (`. `). To unhide a file, you can rename it to remove the leading dot. Here are the steps:

1. View Hidden Files

To see hidden files in a directory:

```
bash

ls -a
```

This will display all files, including hidden ones (those starting with `.`).

2. Rename the File to Unhide It

Use the `mv` (move) command to rename the file and remove the leading dot:

```
bash

mv .hiddenfile hiddenfile
```

Replace `.hiddenfile` with the name of your hidden file.

Example:

```
bash

mv .examplefile examplefile
```

3. Verify the File is Unhidden

Check the directory to ensure the file is no longer hidden:

```
bash
```

```
ls
```

Alternative: Unhide Files in a GUI File Manager

If you're using a desktop environment, you can unhide files through the GUI:

1. Open the file manager.
2. Enable "Show Hidden Files" (usually by pressing `Ctrl + H` or through the menu).
3. Rename the file to remove the leading dot.

Notes

- Hidden files are often configuration files or system files, so unhide them only if necessary.
- To hide the file again, rename it by adding a dot at the beginning of its name:

```
bash
```

```
mv filename .filename
```



How to upgrade python 3.8 to python 3.11.9 in linux



Upgrading Python from version 3.8 to 3.11.9 in Linux involves the following steps:

Step 1: Check Current Python Version

Confirm the currently installed Python version:

```
bash
```

```
python3 --version
```

Step 2: Update System Packages

Ensure your system packages are up-to-date:

```
bash

sudo apt update
sudo apt upgrade -y
```

Step 3: Add Deadsnakes PPA (for Ubuntu/Debian-based distros)

The [Deadsnakes PPA](#) provides newer Python versions:

```
bash

sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt update
```

Step 4: Install Python 3.11

Install Python 3.11:

```
bash

sudo apt install python3.11 -y
```

Step 5: Install Required Modules

Install essential modules like `pip`:

```
bash

sudo apt install python3.11-venv python3.11-distutils -y
```

If `pip` is missing, you can install it manually:

```
bash

curl -sS https://bootstrap.pypa.io/get-pip.py | python3.11
```

Step 6: Set Python 3.11 as Default

1. Check available Python versions:

```
bash
```

```
ls /usr/bin/python3*
```

2. Update the alternatives system:

```
bash
```

```
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.8 1
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.11 2
```

3. Choose Python 3.11 as default:

```
bash
```

```
sudo update-alternatives --config python3
```

Select Python 3.11 from the list.

4. Verify the default version:

```
bash
```

```
python3 --version
```

Step 7: Clean Up

Remove older Python versions if they are no longer needed:

```
bash
```

```
sudo apt remove python3.8 -y
sudo apt autoremove -y
```

Step 8: Optional - Reinstall Libraries

Reinstall any libraries or packages you were using:

```
bash
```

```
pip3 install <library-name>
```

For Non-Ubuntu Systems (e.g., CentOS, Fedora)

Download and compile Python from the source:

1. Download the Python source code:

```
bash

wget https://www.python.org/ftp/python/3.11.9/Python-3.11.9.tgz
tar -xvzf Python-3.11.9.tgz
cd Python-3.11.9
```

2. Build and install:

```
bash

./configure --enable-optimizations
make
sudo make altinstall
```

3. Verify installation:

```
bash

python3.11 --version
```

Note: Use `make altinstall` instead of `make install` to avoid overwriting the system Python.

Troubleshooting:

- If the `add-apt-repository` command is missing:

```
bash

sudo apt install software-properties-common -y
```

- Always check dependencies before removing old versions to avoid breaking system tools.