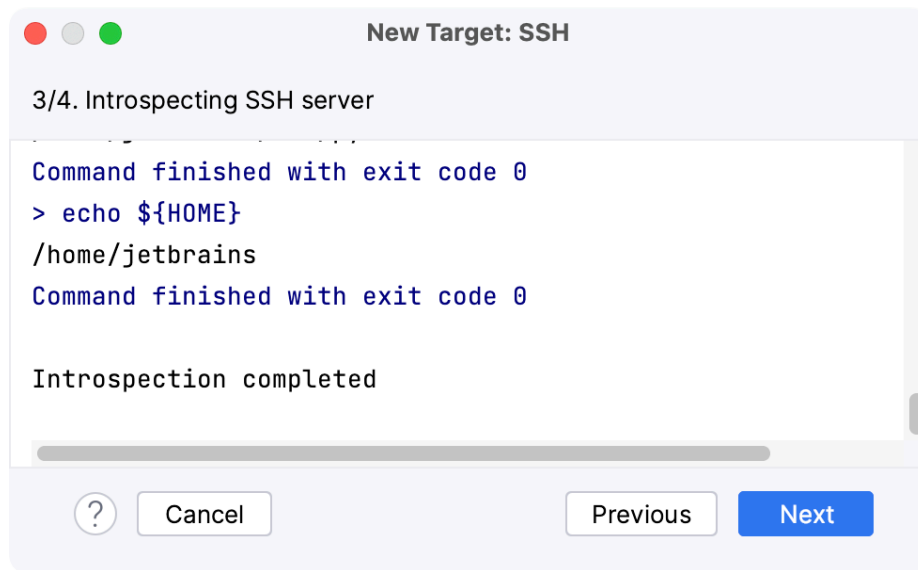


Select **Password** or **Key pair (OpenSSH or PuTTY)** and enter your password or passphrase. If **Key pair (OpenSSH or PuTTY)** is selected, specify:

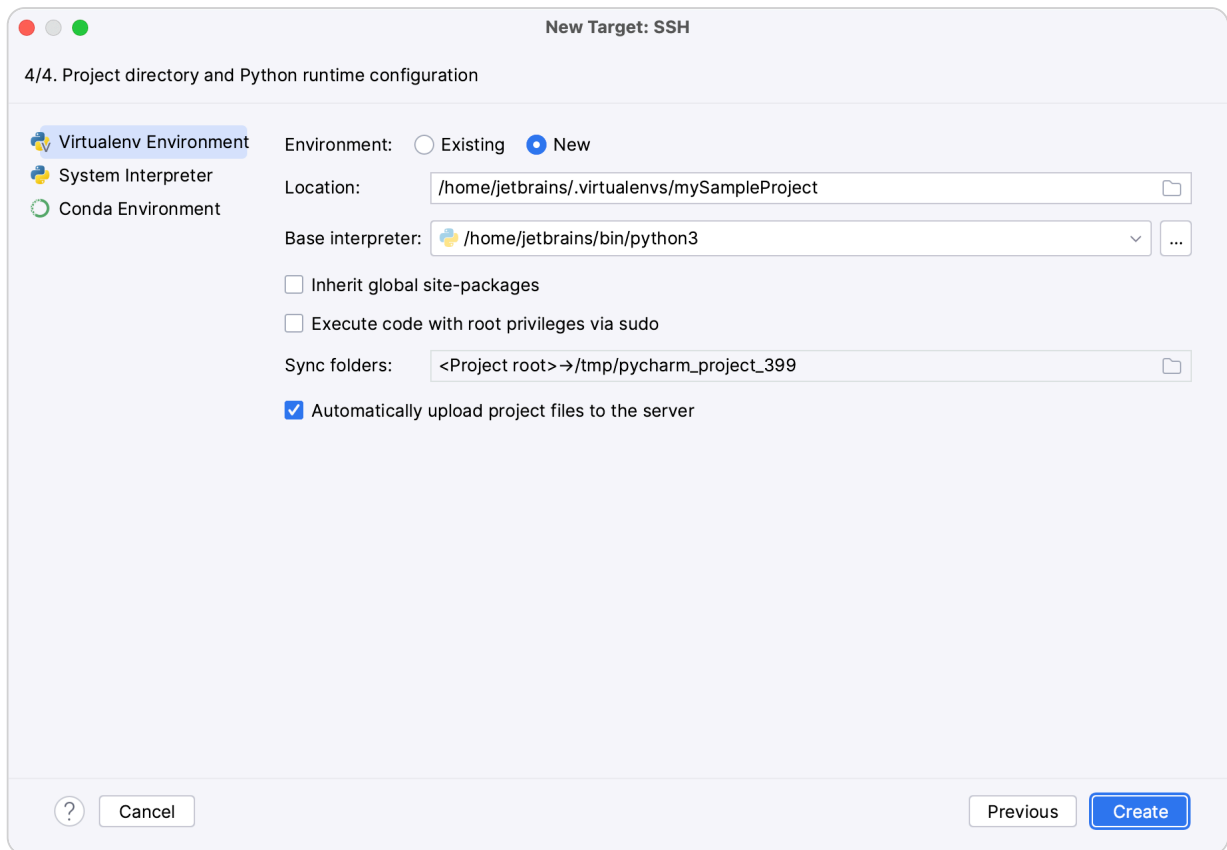
- **Private key:** location of the file with a private key
- **Passphrase:** similar to a password, it serves to encrypt the private key [↗].

Click **Next** to proceed.

6. Wait until PyCharm completes the introspection of the SSH server.



7. In the next dialog, select a type of Python environment to configure on the SSH server.



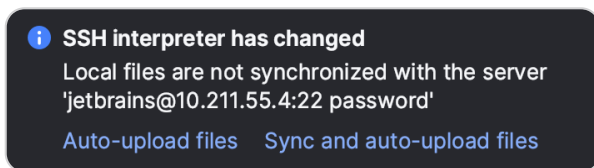
You can create a new virtual environment or conda environment, select an existing one, or use a system interpreter.

- Select the **Inherit global site-packages** checkbox if you want all packages installed in the global Python on your machine to be added to the virtual environment you're going to create. This checkbox corresponds to the `--system-site-packages` option of the [virtualenv](#) [↗] tool.
- If you need to execute your Python code on the SSH server as a sudo user, enable the **Execute code with root privileges via sudo** checkbox.
- You can configure the path mappings between your local project and the server. To do that, click the **Browse** icon in the **Sync folders** field and enter the path to the local project folder and the path to the folder on the remote server.

Click **Create** to complete adding the interpreter.

Synchronizing project files when switching SSH interpreters

When you change the project interpreter and select an SSH interpreter, you might need to synchronize the local content with the target server. Mind a notification balloon in the lower-right corner:



You can choose to enable the automatic uploading of files to the server:

- Click **Auto-upload files** to start uploading on the next save.
- Click **Sync and auto-upload files** to immediately sync the files and upload them on every save in future.

To configure additional settings, select **Tools | Deployment | Options** from the main menu.




Professional

Create SSH configurations

Last modified: 23 February 2024

Available only in PyCharm Professional: [download ↗](#) to try or [compare editions ↗](#)

In PyCharm, you can save the remote server SSH connection parameters as a dedicated SSH configuration. The created configuration can be then used for configuring remote interpreters, [connecting to SFTP deployment servers](#), and [launching SSH sessions](#).

1. In the **Settings** dialog ( Cmd ,) , go to **Tools | SSH Configurations**.
2. In the left-hand pane that lists all the existing SSH configurations, click +.
3. Use the **Visible only for this project** checkbox to enable reuse of this server access configuration in other projects.
 - Select the checkbox to restrict the use of the SSH configuration to the current project. Such SSH configuration cannot be reused outside the current project. It does not appear in the list of available configurations in other projects.

The SSH configurations are stored in the `.idea` directory together with the project, which allows sharing them between team members [through a VCS](#).
 - When the checkbox is cleared, the SSH configuration is visible in all PyCharm projects. Its settings can be reused across several projects.
4. In the **Host**, **User name**, and **Port** fields, specify the connection parameters. **Local port** is used only with the Database Tools and SQL plugin to [establish a connection to a remote database](#).

5. Choose the way to authenticate to the server. Do one of the following:

- **Password:** Access the host with a password. To save the password in PyCharm, select the **Save password** checkbox.
- **Key pair (OpenSSH or PuTTY):** Use [SSH authentication](#) ↗ with a key pair. To apply this authentication method, you must have a private key on the client machine and a public key on the remote server. PyCharm supports private keys that are generated with the [OpenSSH](#) ↗ utility.

Specify the path to the file where your **private key** is stored and type the passphrase (if any) in the corresponding fields. To have PyCharm remember the passphrase, select the **Save passphrase** checkbox.

- **OpenSSH config and authentication agent:** Use a credentials helper application that manages your SSH keys, such as [ssh-agent](#) ↗.

For more information about working with SSH keys, refer to the [Generating a new SSH key and adding it to the ssh-agent](#) ↗ tutorial.

If you select the OpenSSH config options, PyCharm parses OpenSSH directives recorded in SSH config file: `/etc/ssh/ssh_config` > and `~/.ssh/config` on Linux and macOS, or `C:\Users\<username>\.ssh\config` on Windows.

PyCharm supports a limited set of [OpenSSH directives](#).

6. Click the **Test Connection** button to make sure that the settings are correct and PyCharm can connect to the target server.



OpenSSH logs

SSH connections in PyCharm run via [OpenSSH](#) ↗, which maintains comprehensive logs both on the client and on the server. The exact location depends on your operating system. For example, in Linux distributions that are based on Fedora, you should be able to see the logs by running `journalctl -u ssh`.

Supported OpenSSH directives

- [AliveInterval ↗](#)
- [ClearAllForwardings ↗](#)
- [Compression ↗](#)
- [ConnectTimeout ↗](#)
- [ForwardAgent ↗](#)
- [ForwardX11 ↗](#)
- [ForwardX11Trusted ↗](#)
- [GSSAPIAuthentication ↗](#)
- [HashKnownHosts ↗](#)
- [Host ↗](#)
- [Hostname ↗](#)
- [IdentitiesOnly ↗](#)
- [IdentityAgent ↗](#)
- [LocalForward ↗](#)
- [PreferredAuthentications ↗](#)
- [ProxyCommand ↗](#)
- [RemoteForward ↗](#)
- [ServerAliveCountMax ↗](#)

- [StrictHostKeyChecking ↗](#)
- [UserKnownHostsFile ↗](#)
- [XAuthLocation ↗](#)



Console

Last modified: 12 March 2024

File | Settings | Build, Execution, Deployment | Console for Windows and Linux

PyCharm | Settings | Build, Execution, Deployment | Console for macOS

⌘ Cmd , ⚙

Use this page to define console options for the Python console.

Console common options

Item	Description
Always show debug console	If this checkbox is selected, the debug console will be shown by default in the Debug view.
Use IPython if available	<p>When the checkbox is selected (by default): If IPython is installed, then IPython console will be launched.</p> <p>If the checkbox is not selected, then, even with the installed IPython, a Python console will be launched.</p>
Show console variables by default	This checkbox is selected by default to show variables in a console. Deselect it if you do not need to preview variables while working in console.
Use existing console for "Run with Python console"	If you work in a console and don't want to open a separate Python console when running your Python script, select this checkbox. Your script will be launched in the currently opened console. By default, this option is disabled.
Command queue for Python Console	Select this checkbox to be able to <u>manage the command execution queue</u> in the Python Console.

Item	Description
Code completion	<p>Use this drop-down list to select the type of code completion available in the Python console and the Debug Console.</p> <p>Runtime code completion can provide more suggestions by executing functions and methods as you type in the console. To prevent the side effects, use static code completion.</p> <p>By default, Static is selected.</p>