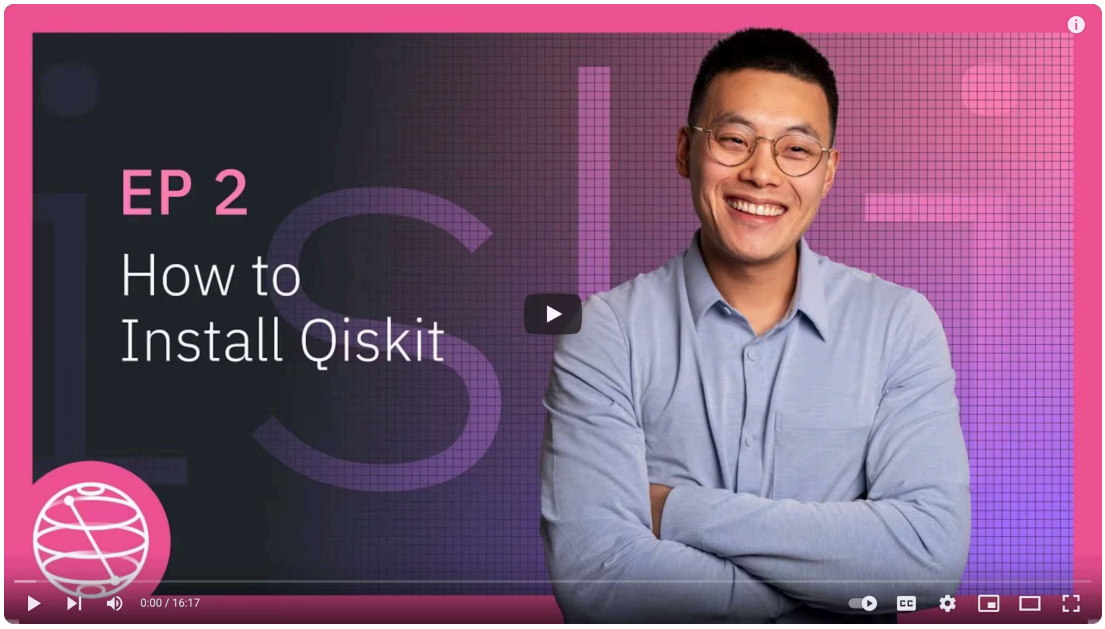
Important

IBM Quantum Platform is moving and this version will be sunset on July 1. To get started on the new platform, read the migration guide.

Install Qiskit

Note

This documentation is relevant to IBM Quantum® Platform Classic. If you need the newer version, go to the new IBM Quantum Platform documentation. ↗



Coding with Qiskit 1.x, Episode 2: How to install Qiskit

Whether you will work locally or in a cloud environment, the first step for all users is to install Qiskit. For those wanting to run on a real quantum processing unit (QPU), your next step is to choose one of two channels in order to access IBM® QPUs: IBM Quantum Platform or IBM Cloud®.

Upgrade from Qiskit 0.x to Qiskit 1.0 and beyond

(If you are installing Qiskit for the first time, skip ahead to the Install and set up section. This notice is relevant only to users who have installed Qiskit previously.)

For those upgrading from version 0.x to 1.0 or later: note that because Qiskit v1.0 uses a new packaging structure, you **cannot** use `pip install -U qiskit` to upgrade from any Qiskit 0.x version to 1.0.

See the Qiskit 1.0 migration guide for details and instructions.

Future updates starting with Qiskit 1.0 will allow for in-place upgrades.

Install the Qiskit SDK and the Qiskit Runtime client

1. Install Python. Check the "Programming Language" section on the Qiskit PyPI project page ↗ to determine which Python versions are supported by the most recent release. For download instructions, see the Python Beginners Guide. ↗

   It is recommended that you use Python virtual environments ↗ to separate Qiskit from other applications.

   > **Note**
   >
   > These instructions use the standard Python distribution from pypi.org ↗. However, you can use other Python distributions, such as Anaconda ↗ or miniconda ↗, along with other dependency management workflows like Poetry ↗.

   If you're new to virtual environments, click here for more information.

   First, navigate to your project directory and create a minimal environment with only Python installed in it.

   macOS | Linux | **Windows**

   ```
   python -m venv .venv
   ```

Next, activate your new environment.

macOS Linux **Windows**

If using PowerShell:

```
.venv\Scripts\Activate.ps1
```

If using Git Bash:

```
source .venv/scripts/activate
```

If using command prompt:

```
.venv\Scripts\activate
```

2. Install pip ↗ if it's not already installed in your environment. Pip is a Python package manager that you use to install Qiskit and other Python packages. Use `pip list` to see what is in your virtual environment. In most Python environments, pip is already installed.
3. Install the Qiskit SDK. If you plan to run jobs on quantum hardware, also install Qiskit Runtime.

```
pip install qiskit
```

```
pip install qiskit-ibm-runtime
```

If you intend to use visualization functionality or Jupyter notebooks, it is recommended to install Qiskit with the extra visualization support (`'qiskit[visualization]'`).

Default zsh

```
pip install qiskit[visualization]
```

1. If you want to run a Jupyter notebook with the Qiskit packages you just installed, you need to install Jupyter in your environment.

```
pip install jupyter
```

Then open your notebook as follows:

```
jupyter notebook <path/to/notebook.ipynb>
```

If you are planning to work locally and use simulators built into Qiskit, then your installation is done. If you want to run jobs on IBM QPUs, next select an access channel and finish your setup.

Stay current with the latest versions

Periodically check the Qiskit release notes and the Qiskit Runtime release notes to see new releases. We recommend frequently updating your requirements for `qiskit` and `qiskit-ibm-runtime` by, for example, changing the versions in `requirements.txt` to the latest versions, then running `pip install -r requirements.txt` or the appropriate command for your dependency management workflow.

Qiskit Code Assistant

Need help? Try asking Qiskit Code Assistant.

```
# Print the version of Qiskit we're using

# Return True if the version of Qiskit is 1.0 or great

# Install Qiskit 1.0.2
```

New to the code assistant? See Qiskit Code Assistant for installation and usage. Note this is an experimental feature and only available to IBM Quantum™ Premium Plan users.

Troubleshooting

"No Module 'qiskit'" error with Jupyter Notebook

If you used `pip install qiskit` and set up your virtual environment in Anaconda, then you may get the `No Module`

`'qiskit'` error when you run a tutorial in Jupyter Notebook. If you have not installed Qiskit or set up your virtual environment, you can follow the installation steps.

The error is caused when trying to import the Qiskit package in an environment where Qiskit is not installed. If you launched Jupyter Notebook from the Anaconda-Navigator, it is possible that Jupyter Notebook is running in the base (root) environment, instead of in your virtual environment. Choose a virtual environment in the Anaconda-Navigator from the **Applications on** dropdown menu. In this menu, you can see all of the virtual environments within Anaconda, and you can select the environment where you have Qiskit installed to launch Jupyter Notebook.

Compilation errors during installation

Qiskit depends on a number of other open-source Python packages, which are automatically installed when doing `pip install qiskit`. Depending on your system's platform and Python version, it is possible that a particular package does not provide pre-built binaries for your system. You can refer to Operating system support for a list of platforms supported by Qiskit, some of which may need an extra compiler. In cases where there are no precompiled binaries available, `pip` will attempt to compile the package from source, which in turn might require some extra dependencies that need to be installed manually.

If the output of `pip install qiskit` contains similar lines to:

```
Failed building wheel for SOME_PACKAGE
...
build/temp.linux-x86_64-3.5/_openssl.c:498:30: fatal error
compilation terminated.
error: command 'x86_64-linux-gnu-gcc' failed with exit stat
```

please check the documentation of the package that failed to install (in the example code, `SOME_PACKAGE`) for information on how to install the libraries needed for compiling from source.

Operating system support

Qiskit strives to support as many operating systems as possible, but due to limitations in available testing resources and operating system availability, not all operating systems can be supported. Operating system support for Qiskit is broken into three tiers with different levels of support for each tier. For platforms outside these, such as FreeBSD or WebAssembly (WASI), Qiskit may still be installable, but it is not tested and you will have to build Qiskit (and likely Qiskit's dependencies) from source.

Additionally, Qiskit only supports the CPython implementation of the Python language. Running with other Python interpreters such as PyPy is not supported.

Qiskit v2.x

In the Qiskit v2.x release series, the supported platforms are:

Tier 1

Tier 1 operating systems are fully tested as part of the development processes to ensure any proposed change will function correctly. Pre-compiled binaries are built, tested, and published to PyPI as part of the release process. Typically, as long as there is a functioning Python environment installed, Qiskit can be installed on these operating systems without needing to install further dependencies.

Tier 1 operating systems:

- Linux x86_64 (distributions compatible with the manylinux 2014 ↗ packaging specification).
- macOS x86_64 (10.12 or later)
- macOS ARM64 (11.0 or newer)
- Windows 64-bit (Windows 10 and later supported)
- Linux AArch64 (distributions compatible with the manylinux 2014 ↗ packaging specification)

Tier 2

There are no Tier 2 operating systems in the Qiskit v2.x release series. Tier 2 operating systems are not tested as part of development process. However, pre-compiled binaries are built, tested, and published to PyPI as part of the release process and

these packages can be expected to be installed with only a functioning Python environment.

Tier 3

Tier 3 operating systems are not tested as part of the development process. Pre-compiled binaries are built and published to PyPI as part of the release process but are not tested. They may not be installable with only a functioning Python environment and might require a C/C++ compiler or additional programs to build dependencies from source as part of the installation process. Support for these operating systems are best effort only.

Tier 3 operating systems:

- Linux ppc64le (distributions compatible with the manylinux 2014 ↗ packaging specification)
- Linux s390x (distributions compatible with the manylinux 2014 ↗ packaging specification)

Starting in Qiskit v2.0.0, only 64-bit platforms are supported and 32-bit platforms are not supported. You will not be able to build from source on 32-bit platforms either, as internally the Qiskit Rust code assumes a 64-bit pointer width.

Qiskit v1.x

In the Qiskit v1.x release series, the supported platforms are:

Tier 1

Tier 1 operating systems are fully tested as part of the development processes to ensure any proposed change will function correctly. Pre-compiled binaries are built, tested, and published to PyPI as part of the release process. Typically, as long as there is a functioning Python environment installed, Qiskit can be installed on these operating systems without needing to install further dependencies.

Tier 1 operating systems:

- Linux x86_64 (distributions compatible with the manylinux 2014 ↗ packaging specification).
- macOS x86_64 (10.12 or later)

- macOS ARM64 (11.0 or newer)
- Windows 64-bit (Windows 10 and later supported)

Tier 2

Tier 2 operating systems are not tested as part of development process. However, pre-compiled binaries are built, tested, and published to PyPI as part of the release process and these packages can be expected to be installed with just a functioning Python environment.

Tier 2 operating systems:

- Linux AArch64 (distributions compatible with the manylinux 2014 ↗ packaging specification)

Tier 3

Tier 3 operating systems are not tested as part of the development process. Pre-compiled binaries are built and published to PyPI as part of the release process but are not tested. They may not be installable with just a functioning Python environment and might require a C/C++ compiler or additional programs to build dependencies from source as part of the installation process. Support for these operating systems are best effort only.

Tier 3 operating systems:

- Linux ppc64le (distributions compatible with the manylinux 2014 ↗ packaging specification)
- Linux s390x (distributions compatible with the manylinux 2014 ↗ packaging specification)
- Linux i686 (distributions compatible with the manylinux 2014 ↗ packaging specification)
- Windows 10 32-bit

Next steps

Recommendations

- Select and set up an IBM Quantum channel.
- Configure Qiskit locally.
- Follow the steps in Hello world to write and run a quantum program.
- Try a tutorial ↗ in IBM Quantum Learning.

Was this page helpful?

Yes No

Report a bug or request content on GitHub ↗.