# NODEJS

[Required permissions for Linux](https://admhelp.microfocus.com/uftdev/en/2021-24.2/HelpCenter/Content/Installation/LeanFTInstallation.htm)

|  |  |  |
| --- | --- | --- |
| File system permissions to install | Full read and write permissions | * ROOT * /opt(install location) * /var(installer registry file) |
| File system permissions to work | Full write permissions | home directory |

## Manual

Source Code: <https://nodejs.org/dist/v20.14.0/node-v20.14.0.tar.gz>

Prebuilt Binary: <https://nodejs.org/dist/v20.14.0/node-v20.14.0-linux-x64.tar.xz>

**Manual installation**

The approach to follow is to keep it in a way that it is easy to remove and upgrade, the way to do it is to keep all the stuff in one place and add that place to your path, for that One can do

sudo tar -xf node-v20.14.0-linux-x64.tar.xz --directory=/opt/

and add the bin folder to your $PATH variable using the following

echo 'export PATH="/opt/node-v20.14.0-linux-x64/bin/:$PATH"' >> ~/.bashrc && source ~/.bashrc

This is a better approach because you can remove it, and upgrade it easily compared to just dumping all the files in the /bin folder, like many, including my previously written answer ( deleted now ), are telling.

Now on any given day, you can remove the entire folder from /opt and replace it with the version you want or even you can use multiple versions at the same time.

## Using Ubuntu (Node.js 20)

Before you begin, ensure that curl is installed on your system. If curl is not installed, you can install it using the following command:

1. **Download the Node.js setup script:**

curl -fsSL https://deb.nodesource.com/setup\_20.x -o nodesource\_setup.sh

1. **Run the Node.js setup script with sudo:**

sudo -E bash nodesource\_setup.sh

1. **Install Node.js:**

sudo apt-get install -y nodejs

1. **Verify the installation:**

node -v

## NVM

Now you'll need to follow these steps in order to ensure that you've installed Node.js successfully on your system.

**Install Node Version Manager (NVM)**

Install the Node Version Manager (NVM) by using the following command:

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh | bash

When you run this specific command, the curl downloads the NVM installation script from that specific URL. Afterward, bash executes the same script for installing NVM.

**Activate NVM**

Activate the NVM using the command below:

source ~/.bashrc

**Install the latest LTS version of Node**

Install the latest Long Term Support version of Node by using the command below:

nvm install --lts

**Confirm that Node was installed**

Use the command below to check whether the default version is the exact version you just installed:

node -v npm -v

## Using Snap Package Manager

**Step 1: Install Node.js**  
You can use Snap package manager to install Node.js on Linux. Replace “16” with the desired major version number (LTS or Current). The “–classic” argument is required for full system access, which Node.js needs in following command.

sudo snap install node --classic --channel=20

## RPM

1. **Download the Node.js setup script:**

curl -fsSL https://rpm.nodesource.com/setup\_20.x -o nodesource\_setup.sh

1. **Run the Node.js setup script as root:**

bash nodesource\_setup.sh

1. **Install Node.js:**

yum install -y nodejs

1. **Verify the installation:**

node -v

# Ansible

Ansible is an agentless automation tool that you install on a single host (referred to as the control node).

From the control node, Ansible can manage an entire fleet of machines and other devices (referred to as managed nodes) remotely with SSH, Powershell remoting, and numerous other transports, all from a simple command-line interface with no databases or daemons required.

The *managed* node (the machine that Ansible is managing) does not require Ansible to be installed, but requires Python to run Ansible-generated Python code. The managed node also needs a user account that can connect through SSH to the node with an interactive POSIX shell.

Ansible’s community packages are distributed in two ways:

* ansible-core: a minimalist language and runtime package containing a set of [Ansible.Builtin](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/index.html" \l "plugins-in-ansible-builtin).
* ansible: a much larger “batteries included” package, which adds a community-curated selection of [Ansible Collections](https://docs.ansible.com/ansible/latest/collections_guide/index.html#collections) for automating a wide variety of devices.

Choose the package that fits your needs. The following instructions use ansible as a package name, but you can substitute ansible-core if you prefer to start with the minimal package and separately install only the Ansible Collections you require.

The ansible or ansible-core packages may be available in your operating systems package manager, and you are free to install these packages with your preferred method. For more information, see the [Installing Ansible on specific operating systems](https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html#installing-distros) guide. These installation instructions only cover the officially supported means of installing the python packages with pip.

See the [Ansible package release status table](https://docs.ansible.com/ansible/latest/reference_appendices/release_and_maintenance.html#ansible-changelogs) for the ansible-core version included in the package.

On some systems, it may not be possible to install Ansible with pip, due to decisions made by the operating system developers. In such cases, pipx is a widely available alternative.

These instructions will not go over the steps to install pipx; if those instructions are needed, please continue to the [pipx installation instructions](https://pypa.github.io/pipx/installation/) for more information.

[**Installing Ansible**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id12)

Use pipx in your environment to install the full Ansible package:

**$** pipx install --include-deps ansible

You can install the minimal ansible-core package:

**$** pipx install ansible-core

Alternately, you can install a specific version of ansible-core:

**$** pipx install ansible-core**==**2.12.3

[**Upgrading Ansible**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id13)

To upgrade an existing Ansible installation to the latest released version:

**$** pipx upgrade --include-injected ansible

[**Installing Extra Python Dependencies**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id14)

To install additional python dependencies that may be needed, with the example of installing the argcomplete python package as described below:

**$** pipx inject ansible argcomplete

Include the --include-apps option to make apps in the additional python dependency available on your PATH. This allows you to execute commands for those apps from the shell.

**$** pipx inject --include-apps ansible argcomplete

[**Locating Python**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id16)

Locate and remember the path to the Python interpreter you wish to use to run Ansible. The following instructions refer to this Python as python3. For example, if you have determined that you want the Python at /usr/bin/python3.9 to be the one that you will install Ansible under, specify that instead of python3.

[**Ensuring pip is available**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id17)

To verify whether pip is already installed for your preferred Python:

**$** python3 -m pip -V

If all is well, you should see something like the following:

**$** python3 -m pip -V

pip 21.0.1 from /usr/lib/python3.9/site-packages/pip (python 3.9)

If so, pip is available, and you can move on to the [next step](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#pip-install).

If you see an error like No module named pip, you will need to install pip under your chosen Python interpreter before proceeding. This may mean installing an additional OS package (for example, python3-pip), or installing the latest pip directly from the Python Packaging Authority by running the following:

**$** curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

**$** python3 get-pip.py --user

You may need to perform some additional configuration before you are able to run Ansible. See the Python documentation on [installing to the user site](https://packaging.python.org/tutorials/installing-packages/#installing-to-the-user-site) for more information.

[**Installing Ansible**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id18)

Use pip in your selected Python environment to install the full Ansible package for the current user:

**$** python3 -m pip install --user ansible

You can install the minimal ansible-core package for the current user:

**$** python3 -m pip install --user ansible-core

Alternately, you can install a specific version of ansible-core:

**$** python3 -m pip install --user ansible-core**==**2.12.3

You can install the devel branch of ansible-core directly from GitHub with pip:

**$** python3 -m pip install --user https://github.com/ansible/ansible/archive/devel.tar.gz

You can replace devel in the URL mentioned above, with any other branch or tag on GitHub to install older versions of Ansible, tagged alpha or beta versions, and release candidates.

[**Running the devel branch from a clone**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id23)

ansible-core is easy to run from source. You do not need root permissions to use it and there is no software to actually install. No daemons or database setup are required.

1. Clone the ansible-core repository
2. **$** git clone https://github.com/ansible/ansible.git
3. **$** cd ./ansible
4. Setup the Ansible environment
   * Using Bash
   * **$** source ./hacking/env-setup
   * Using Fish
   * **$** source ./hacking/env-setup.fish
   * To suppress spurious warnings/errors, use -q
   * **$** source ./hacking/env-setup -q
5. Install Python dependencies
6. **$** python3 -m pip install --user -r ./requirements.txt
7. Update the devel branch of ansible-core on your local machine

Use pull-with-rebase so any local changes are replayed.

**$** git pull --rebase

[**Confirming your installation**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id24)

You can test that Ansible is installed correctly by checking the version:

**$** ansible --version

The version displayed by this command is for the associated ansible-core package that has been installed.

To check the version of the ansible package that has been installed:

**$** ansible-community --version

[**Adding Ansible command shell completion**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id25)

You can add shell completion of the Ansible command line utilities by installing an optional dependency called argcomplete. It supports bash, and has limited support for zsh and tcsh.

For more information about installation and configuration, see the [argcomplete documentation](https://kislyuk.github.io/argcomplete/).

[**Installing argcomplete**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id26)

If you chose the pipx installation instructions:

**$** pipx inject --include-apps ansible argcomplete

If you chose the pip installation instructions:

**$** python3 -m pip install --user argcomplete

[**Configuring argcomplete**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id27)

There are 2 ways to configure argcomplete to allow shell completion of the Ansible command line utilities: globally or per command.

[**Global configuration**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id28)

Global completion requires bash 4.2.

**$** activate-global-python-argcomplete --user

This will write a bash completion file to a user location. Use --dest to change the location or sudo to set up the completion globally.

[**Per command configuration**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id29)

If you do not have bash 4.2, you must register each script independently.

**$** eval **$(**register-python-argcomplete ansible**)**

**$** eval **$(**register-python-argcomplete ansible-config**)**

**$** eval **$(**register-python-argcomplete ansible-console**)**

**$** eval **$(**register-python-argcomplete ansible-doc**)**

**$** eval **$(**register-python-argcomplete ansible-galaxy**)**

**$** eval **$(**register-python-argcomplete ansible-inventory**)**

**$** eval **$(**register-python-argcomplete ansible-playbook**)**

**$** eval **$(**register-python-argcomplete ansible-pull**)**

**$** eval **$(**register-python-argcomplete ansible-vault**)**

You should place the above commands into your shell’s profile file such as ~/.profile or ~/.bash\_profile.

# YQ

a lightweight and portable command-line YAML, JSON and XML processor. yq uses [jq](https://github.com/stedolan/jq) like syntax but works with yaml files as well as json, xml, properties, csv and tsv. It doesn't yet support everything jq does - but it does support the most common operations and functions, and more is being added continuously.

[**Download the latest binary**](https://github.com/mikefarah/yq/releases/latest)

**wget**

Use wget to download, gzipped pre-compiled binaries:

For instance, VERSION=v4.2.0 and BINARY=yq\_linux\_amd64

**Compressed via tar.gz**

wget https://github.com/mikefarah/yq/releases/download/${VERSION}/${BINARY}.tar.gz -O - |\

tar xz && mv ${BINARY} /usr/bin/yq

**Plain binary**

wget https://github.com/mikefarah/yq/releases/download/${VERSION}/${BINARY} -O /usr/bin/yq &&\

*chmod +x /usr/bin/yq*

**Latest version**

wget https://github.com/mikefarah/yq/releases/latest/download/yq\_linux\_amd64 -O /usr/bin/yq &&\

*chmod +x /usr/bin/yq*

The recommended installation is to download the latest binary from the yq GitHub releases and put it into the user’s ~/bin directory.

Verifying the installation of yq

The command:

*yq --version*

Should print an output like:

*yq (https://github.com/mikefarah/yq/) version 4.25.3*

# JQ

**JQ** is a lightweight and powerful command-line tool designed specifically for working with [JSON data](https://www.scaler.com/topics/javascript/what-is-json/) in Linux environments. It allows users to parse, filter, extract, and manipulate JSON files effortlessly, making it an essential utility for developers, system administrators, and data analysts. The working of the JQ tool is based on a pipeline model, where data is processed through a series of transformations.

It consists of the following components,

* The **JSON Parser** is responsible for reading and parsing JSON data input. It validates and converts the raw JSON input into an internal representation that JQ can work with.
* The **Query Compiler** processes the query or program provided to JQ. This component compiles the query into an intermediate representation known as JQ bytecode.
* The **JQ bytecode** generated by the Query Compiler is executed by the **Virtual Machine**. The VM acts as an interpreter, interpreting and executing the bytecode instructions to perform the desired operations on the JSON data.
* The **Filters and transformations** allow users to extract specific data, filter out unwanted elements, modify values, and perform complex computations on the JSON data.
* Once the desired operations have been performed on the JSON data, the **Output Formatter** takes the processed data and converts it back into a readable format.

The JQ command in Linux can be installed using the recommended [package manager](https://www.scaler.com/topics/cyber-security/package-management-in-linux/) method or directly from the JQ website. Follow step 1 to install the JQ tool in your system through the command line and step 2 to install it from the website,

**1. You can use the given command based on your Linux distro to install the JQ tool**

* **For Debian/Ubuntu-based systems:**
* sudo apt-get update
* sudo apt-get install jq
* **For Fedora/RHEL-based systems:**
* sudo dnf install jq

If your Linux system is openSUSE or Arch distro or you want to use another package manager or want to install the tool on FreeBSD or Solaris, you can find the relative command and instructions through the [Download Page](https://jqlang.github.io/jq/download/) of JQ package.

**2. Follow the following steps to install the JQ tool manually,**

* Visit the official [JQ GitHub repository](https://github.com/jqlang/jq/releases/tag/jq-1.6) using a web browser.
* Download the latest stable release of JQ under the Assets section.
* Once the download is complete, open a terminal and navigate to the directory where the downloaded file is located.
* Extract the downloaded archive using the following command:
* tar xvf jq-<version>.tar.gz

Replace the <version> with your installed version number. The current version is 1.6 and your command will be tar xvf jq-1.6.tar.gz.

* Move into the extracted directory,
* cd jq-<version>
* Configure, compile, and install JQ using the following commands:
* ./configure
* make
* sudo make install

You can also download the binary file directly from the [Download Page](https://jqlang.github.io/jq/download/) of the JQ package and follow the following steps to install the JQ tool from the installed file,

* Go to the downloads folder and xxtract the downloaded file using the following command.
* tar xvf jq-linux64-1.6.tar.gz
* Move into the extracted directory and move the jq binary file to the /usr/local/bin directory to be accessible system-wide.
* sudo mv jq /usr/local/bin/

After completing the steps, to verify the installation, open a new terminal and type,

jq --version