Installing 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.

Control node requirements

For your control node (the machine that runs Ansible), you can use nearly any UNIX-like machine with Python 3.8 or newer installed. This includes Red Hat, Debian, Ubuntu, macOS, BSDs, and Windows under a Windows Subsystem for Linux (WSL) distribution. Windows without WSL is not natively supported as a control node; see Matt Davis' blog post for more information.

Selecting an Ansible package and version to install

Ansible's community packages are distributed in two ways: a minimalist language and runtime package called `ansible-core`, and a much larger "batteries included" package called `ansible`, which adds a community-curated selection of :ref:`Ansible Collections <collections>` for automating a wide variety of devices. Choose the package that fits your needs; The following instructions use `ansible`, but you can substitute `ansible-core` if you prefer to start with a more 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. These installation instructions only cover the officially supported means of installing the python package with `pip`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\installation_guide\[ansible-devel][docs][docsite][rst][installation_guide]intro_installation.rst, line 27); *backlink***
>
> Unknown interpreted text role "ref".

Installing and upgrading Ansible

Locating Python

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've determined that you want the Python at `/usr/bin/python3.9` to be the one that you'll install Ansible under, specify that instead of `python3`.

Ensuring `pip` is available

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 :ref:`next step <pip_install>`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\installation_guide\[ansible-devel][docs][docsite][rst][installation_guide]intro_installation.rst, line 53); *backlink***
>
> Unknown interpreted text role "ref".

If you see an error like `No module named pip`, you'll 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 for more information.

Installing Ansible

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

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

Alternately, you can install a specific version of `ansible-core` in this Python environment:

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

Upgrading Ansible

To upgrade an existing Ansible installation in this Python environment to the latest released version, simply add `--upgrade` to the command above:

```
$ python3 -m pip install --upgrade --user ansible
```

Confirming your installation

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:

```
$ python3 -m pip show ansible
```

Installing for development

If you are testing new features, fixing bugs, or otherwise working with the development team on changes to the core code, you can install and run the source from GitHub.

> **Note**
>
> You should only install and run the `devel` branch if you are modifying `ansible-core` or trying out features under development. This is a rapidly changing source of code and can become unstable at any point.

For more information on getting involved in the Ansible project, see the :ref:`ansible_community_guide`. For more information on creating Ansible modules and Collections, see the :ref:`developer_guide`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\installation_guide\[ansible-devel][docs][docsite][rst][installation_guide]intro_installation.rst, line 122); *backlink***
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\installation_guide\[ansible-devel][docs][docsite][rst][installation_guide]intro_installation.rst, line 122); *backlink***
>
> Unknown interpreted text role "ref".

Installing `devel` from GitHub with `pip`

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

`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

   ```
   $ git clone https://github.com/ansible/ansible.git
   $ cd ./ansible
   ```

2. 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
     ```

3. Install Python dependencies

   ```
   $ python3 -m pip install --user -r ./requirements.txt
   ```

4. Update the `devel` branch of `ansible-core` on your local machine

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

   ```
   $ git pull --rebase
   ```

Adding Ansible command shell completion

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

For more information about installation and configuration, see the argcomplete documentation.

Installing `argcomplete`

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

Configuring `argcomplete`

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

Global configuration

Global completion requires bash 4.2.

```
$ activate-global-python-argcomplete
```

This will write a bash completion file to a global location. Use `--dest` to change the location.

Per command configuration

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 shells profile file such as `~/.profile` or `~/.bash_profile`.

Using `argcomplete` with zsh or tcsh

See the argcomplete documentation.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\installation_guide\[ansible-devel][docs][docsite][rst][installation_guide]intro_installation.rst, line 242)

Unknown directive type "seealso".

```
.. seealso::

   :ref:`intro_adhoc`
       Examples of basic commands
   :ref:`working_with_playbooks`
       Learning ansible's configuration management language
   :ref:`installation_faqs`
       Ansible Installation related to FAQs
   `Mailing List <https://groups.google.com/group/ansible-project>`_
       Questions? Help? Ideas?  Stop by the list on Google Groups
   :ref:`communication_irc`
       How to join Ansible chat channels
```