2. Installing Ansible

2. Installing Ansible

Ansible is installed on a control node, which is the machine that communicates with the managed nodes. The managed nodes are the end devices you want to automate.

2.1 Control node requirements

Ansible can be run from any machine with Python version 2 (2.7+) or Python version 3 (3.5+). This includes but is not limited to Debian, macOS, CentOS, RHEL, or any of the Berkeley Software Distributions.

Note

When implementing a Windows machine as control node Windows Subsystem for Linux must be used, as Ansible is not natively supported in python on Windows.

When choosing a control node, it is important to bear in mind that the Ansible platform greatly benefits from being run near the managed nodes. If you are running Ansible in a cloud environment, consider running a machine within the same environment, instead of using a local workstation.

Warning

Some modules and plugins might have additional requirements. These requirements are listed in the module specific docs.

2.2 Managed Node requirements

A managed node must be capable of functioning as an SSH-server.
Furthermore, the nodes also need python. By default, ansible uses python 2 (version 2.6+) but ansible can also use python 3 (version 3.5+). When using python 3 you need to set the ansible_python_interpreter inventory variable.

If the prospect node has selinux installed it is recommended that libselinux-python is also installed as some command can otherwise be interfered with by selinux.

2.3 Selecting an Ansible version

Which Ansible version to install is based on your particular needs and the environment you want to automate. You can choose any of the following ways to install Ansible:

- Install the latest stable release with an OS package manager
- Install with Python Package manager (pip)
- Install from source to access the development (devel) version to develop or test the latest features.

Note

The development version of Ansible should only be used if you are actively devolving content. The source code of the development version is rapidly changing, which can result in instability.

2.2 Installation

In this section we will explain step by step how to install ansible on several different operation systems.

> **Note**
> If your operating system is not listed, please refer to the Ansible documentation

2.2.1 Installing Ansible on RHEL, CentOS, or Fedora

On Fedora:

```
sudo dnf install ansible
```

On RHEL and CentOS:

```
sudo yum install ansible
```

2.2.2 Ansible on Ubuntu

To configure and install Ansible, you have to run these commands:

```
sudo apt update
sudo apt install software-properties-common
sudo apt-add-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
```

> **Note**
> On older versions of Ubuntu you may want to use apt-get instead of apt.

2.2.3 Installing Ansible on Debian

Installing Ansible from the source makes it difficult to uninstall it. Ansible copies files right into the correct directory, so it is difficult to track which files were created. By building an .deb packages, you can easily uninstall Ansible.
Installing Ansible from a .deb package requires the same packages as installing it from the source. The following commands will install and build Ansible from a Debian package:

```
sudo apt-get update
sudo apt-get install build-essential git python-pip python-dev libffi-
dev libssl-dev asciidoc devscripts  debhelper cdbs
sudo pip install setuptools --upgrade
git clone git://github.com/ansible/ansible.git --recursive
cd ansible
make deb
```

Once the last command is done, you have to locate the built Debian package. This can be done with the following command:

```
find . -name "*.deb" ./deb-build/unstable/ansible_2.2.0-
0.git201607051907.d0ccedc.devel~unstable_all.deb
```

Before you can install this package, you'll need to install some packages that Ansible needs by running this command:

```
sudo apt-get install python-jinja2 python-paramiko sshpass python- markupsafe
```

Once the packages are installed, you can install Ansible from the Debian package you just build:

```
sudo dpkg -i ./deb-build/unstable/ansible_2.2.0-
0.git201607051907.d0ccedc.devel~unstable_all.deb
```

You can now verify that Ansible is installed by checking the version, with *ansible --version*. Because Ansible is installed from a package it can be easily uninstalled, with *apt-get remove ansible*.

2.2.4 Installing Ansible on FreeBSD

Ansible works with different versions of Python, FreeBSD has these different packages for each Python version. To install Python you can use this:

```
sudo pkg install py27-ansible
```

Or:

```
sudo pkg install py36-ansible
```

The specific version of Ansible can be chosen, i.e. *ansible25*.
Older versions of FreeBSD work with this (depends of your choice of package manager):

```
sudo pgk install ansible
```

2.2.5 Installing Ansible on macOS

The preferred way to install Ansible on macOS is with Python Package manager (pip). To install Ansible with pip can be found in *Installing Ansible with pip.* If you are running macOS version 10.12 or older, it is recommended to upgrade to the latest version of pip to connect to the Python Package Index securely.

2.2.6 Installing Ansible with pip

Ansible can be installed with the Python Package manager. If pip isn't installed on your system, you can run the following commands:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python get-pip.py –user
```

Then install Ansible:

Note

Running pip with sudo will make global changes to the system. Since pip does not coordinate with system package managers, it could make changes to system files, leaving the system in an inconsistent or non-functioning state. This is particularly true for macOS. Installing with --user is recommended unless you understand fully the implications of modifying global files on the system.
Please make sure you have the latest version of pip before installing Ansible. If you have an older version of pip installed, you can upgrade with: pip install -U pip

```
pip install --user ansible
```

2.3. Ansible command shell completion

As of Ansible 2.9, shell completion of the Ansible command line utilities is available and provided through an optional dependency called argcomplete. Argcomplete supports bash, and has limited support for zsh and tcsh.
You can install python-argcomplete from EPEL on Red Hat Enterprise based distributions, and or from the standard OS repositories for many other distributions.

2.3.1. Installing argcomplete on RHEL, CentOS, or Fedora

On Fedora:

```
sudo dnf install python-argcomplete
```

On RHEL and CentOS:

```
sudo yum install epel-release
sudo yum install python-argcomplete
```

Installing argcomplete with apt:

```
sudo apt install python-argcomplete
```

Installing argcomplete with pip:

```
python -m pip install argcomplete
```

2.3.2. Configuring argcomplete

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

Globally

Global completion requires bash 4.2.
sudo activate-global-python-argcomplete
This will write a bash completion file to a global location. Use --dest to change the location.

Per command

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
```