**What is Ansible?**

Ansible is an open-source automation tool or platform, used for [cloud provisioning](https://www.ansible.com/provisioning?hsLang=en-us), [configuration management](https://www.ansible.com/configuration-management?hsLang=en-us), [application deployment](https://www.ansible.com/application-deployment?hsLang=en-us), [intra-service orchestration](https://www.ansible.com/orchestration?hsLang=en-us), and many other IT needs.

<https://www.simplilearn.com/what-is-ansible-article>

# What Is Ansible?

[By Shivam Arora](https://www.simplilearn.com/authors/shivam-arora?source=frs_detailsPage)Last updated on Nov 23, 20184084



If you’re in IT, you’re hearing about Ansible more and more lately. [CIO](https://www.cio.com/article/3197670/application-development/why-ansible-has-become-the-devops-darling.html) calls it the DevOps “darling” for software automation, adding that Ansible “has come from nowhere to be the No. 1 choice for software automation in many organizations.” [StackShare](https://stackshare.io/ansible" \t "_blank) lists more than 1,000 companies that use Ansible, including Intel, Evernote, and Hootsuite, and the [Ansible blog](https://www.ansible.com/blog/enterprise-ansible) says Apple and NASA have adopted it as well. So what exactly is Ansible, and why has it gained popularity so quickly? We’ll delve into that here, and we’ll also tell you a bit about why Ansible and Docker work so well together, and how to use Ansible in Docker. You’ll get a glimpse of why the Ansible-Docker combination is so powerful. Let’s get started!

## **What is Ansible?**

Ansible is an open-source automation tool, or platform, used for IT tasks such as configuration management, application deployment, intraservice orchestration and provisioning. Automation is crucial these days, with IT environments that are too complex and often need to scale too quickly for system administrators and developers to keep up if they had to do everything manually. Automation simplifies complex tasks, not just making developers’ jobs more manageable but allowing them to focus attention on other tasks that add value to an organization. In other words, it frees up time and increases efficiency. And Ansible, as noted above, is rapidly rising to the top in the world of automation tools. Let’s look at some of the reasons for Ansible’s popularity.

## **Advantages of Ansible**

* **Free**. Ansible is an open-source tool.
* **Very simple to set up and use**. No special coding skills are necessary to use Ansible’s playbooks (more on playbooks later).
* **Powerful**. Ansible lets you model even highly complex IT workflows.
* **Flexible**. You can orchestrate the entire application environment no matter where it’s deployed. You can also customize it based on your needs.
* **Agentless**. You don’t need to install any other software or firewall ports on the client systems you want to automate. You also don’t have to set up a separate management structure.
* **Efficient**. Because you don’t need to install any extra software, there’s more room for application resources on your server.

## **Ansible’s Features and Capabilities**

**Configuration Management**  
Ansible is designed to be very simple, reliable and consistent for configuration management. If you’re already in IT, you can get up and running with it very quickly. Ansible configurations are simple data descriptions of an infrastructure, and are both readable by humans and parsable by machines. All you need to start managing systems is a password or an SSH (Secure Socket Shell, a network protocol) key. An example of how easy Ansible makes configuration management: If you want to install an updated version of a certain type of software on all the machines in your enterprise, all you have to do is write out all the IP addresses of the nodes (also called remote hosts) and write an Ansible playbook to install it on all the nodes, then run the playbook from your control machine.

## **Application Deployment**

Ansible lets you quickly and easily deploy multitier apps. You won’t need to write custom code to automate your systems; you just list the tasks needed to be done by writing a playbook, and Ansible will figure out how to get your systems to the state you want them to be in. In other words, you won’t have to manually configure the applications on every machine. When you run a playbook from your control machine, Ansible uses SSH to communicate with the remote hosts and run all the commands (tasks).

## **Orchestration**

As the name suggests, orchestration involves bringing different elements into a beautifully run whole operation—similar to the way a musical conductor brings the notes produced by all the different instruments into a cohesive musical work. For example, with application deployment, you need to manage not just the front-end and backend services but the databases, networks, storage and so on. You also need to make sure that all the tasks are handled in the proper order. Ansible uses automated workflows, provisioning and more to make orchestrating tasks easy. And once you’ve defined your infrastructure using the Ansible playbooks, you can use that same orchestration wherever you need to, thanks to the portability of Ansible playbooks.

**Ansible Modules**

## **Debug module:**

The [debug module](https://docs.ansible.com/ansible/debug_module.html) is well known. It is the Ansible equivalent to a print line in Python code that you add now and then to see some variable values when debugging your scripts.

- debug: msg="Creation of directories"

**File Module:** Sets attributes of files, symlinks, and directories, or removes files/symlinks/directories. Many other modules support the same options as the file module - including [copy](https://docs.ansible.com/ansible/2.4/copy_module.html#copy), [template](https://docs.ansible.com/ansible/2.4/template_module.html#template), and [assemble](https://docs.ansible.com/ansible/2.4/assemble_module.html#assemble).

**Example:**

**file:**

**path: "{{ item.path }}"**

**state: directory**

**owner: "{{ item.owner }}"**

**group: "{{ item.group }}"**

**mode: 0755**

**with\_items:**

**- { path: '/opt/AAA/SW4AAA/aaa-cloud-scripts/' , owner: 'aaa', group: 'aaa' }**

**- { path: '/opt/AAA/SW4AAA/aaa-build/' , owner: 'aaa', group: 'aaa' }**

**- { path: '/opt/AAA/SW4AAA/aaa-logs/' , owner: 'aaa', group: 'aaa' }**

**- { path: '/appdata/aaa/aaa\_logs/' , owner: 'aaa', group: 'aaa'}**

**- { path: '/appdata/aaa/ssh\_keys/aaauser' , owner: 'aaa', group: 'aaa'}**

**- { path: '/appdata/aaa/ssh\_keys/aaaperfuser' , owner: 'aaa', group: 'aaa'}**

**- { path: '/appdata/var/' , owner: 'root', group: 'root'}**

**Copy module:** module copies a file from the local or remote machine to a location on the remote machine. Use the [fetch](https://docs.ansible.com/ansible/2.4/fetch_module.html#fetch) module to copy files from remote locations to the local box

- copy:

src: /srv/myfiles/foo.conf

dest: /etc/foo.conf

owner: foo

group: foo

mode: 0644

**shell module:**

* The shell module takes the command name followed by a list of space-delimited arguments.
* It is almost exactly like the [command](https://docs.ansible.com/ansible/latest/modules/command_module.html#command-module) module but runs the command through a shell (/bin/sh) on the remote node.

**Become module:** Ansible allows you to ‘become’ another user, different from the user that logged into the machine (remote user). This is done using existing privilege escalation tools such as sudo, su, pfexec, doas, pbrun, dzdo, ksu, runas, machinectl and others.

- name: Updation of /etc/hosts with OAM and APP nodes

become: true

become\_method: su

become\_user: root

command: /opt/AAA/SW4AAA/aaa-cloud-scripts/aaa\_generic.py --update\_etc\_hosts

**set\_fact Module**

* This module allows setting new variables. Variables are set on a host-by-host basis just like facts discovered by the setup module.
* These variables will be available to subsequent plays during an ansible-playbook run, but will not be saved across executions even if you use a fact cache.

[**https://everythingshouldbevirtual.com/automation/ansible-using-set\_facts-module/**](https://everythingshouldbevirtual.com/automation/ansible-using-set_facts-module/)

**Vars module, set\_fact module**

# get\_url moduel:

* Downloads files from HTTP, HTTPS, or FTP to the remote server. The remote server *must* have direct access to the remote resource.
* get\_url**:**
* url**:** http://example.com/path/file.conf
* dest**:** /etc/foo.conf
* mode**:** '0440'

**Register Module: Ansible registers** are used when you want to capture the output of a task to a variable. You can then use the value of these**registers** for different scenarios like a conditional statement, logging etc.

**Ansible Openstack modules:** <https://docs.ansible.com/ansible/latest/modules/list_of_cloud_modules.html#openstack>

* **Os\_auth module:** Retrieve an auth token from an OpenStack Cloud

Openstacksdk, python >= 2.7 should be present in the host to use this module

**os\_auth:**

**auth\_url: "{{ lookup('env', 'OS\_AUTH\_URL') }}"**

**username: "{{ lookup('env', 'OS\_USERNAME') }}"**

**password: "{{ lookup('env', 'OS\_PASSWORD') }}"**

**project\_name: "{{ lookup('env', 'OS\_PROJECT\_NAME') }}"**

**project\_domain\_name: default**

**user\_domain\_name: default**

# os\_image – Add/Delete images from OpenStack Cloud