**What is Ansible?**

[Ansible](https://tekslate.com/tutorials/ansible/) is a software tool to deploy an application using ssh without any downtime.It is also used to manage and configure software applications. Python language develops Ansible.

**What are the Advantages of Ansible?**

-Agentless

-Very low overhead

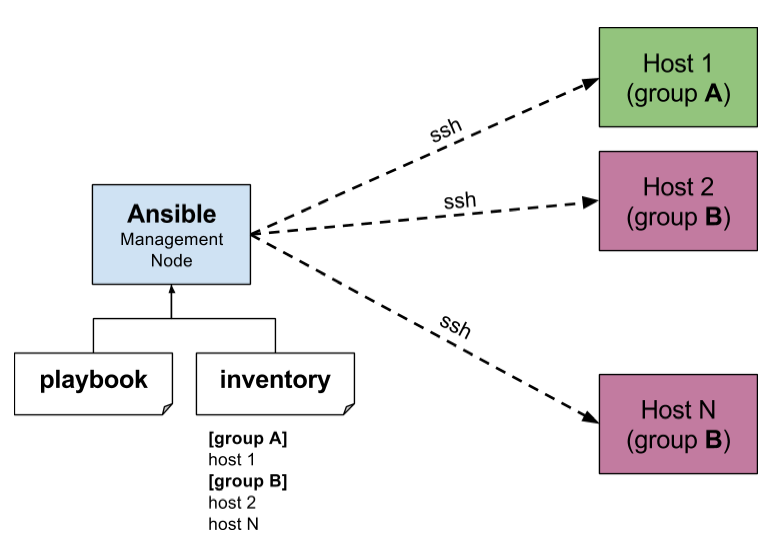
-Good performance

**How Ansible Works?**

There are many similar automation tools available like Puppet, Capistrano, Chef, Salt, Space Walk, etc, but Ansible categorizes into two types of server: controlling machines and nodes.

The controlling machine, where Ansible is installed, and Nodes are managed by this controlling machine over SSH. The location of nodes is specified by controlling the machine through its inventory.

The controlling machine (Ansible) deploys modules to nodes using SSH protocol, and these modules are stored temporarily on remote nodes and communicate with the Ansible machine through a JSON connection over the standard output.



Ansible is agent-less, that means no need for any agent installation on remote nodes, so it means there are no any background daemons or programs are executing for Ansible when it’s not managing any nodes.

Ansible can handle 100’s of nodes from a single system over SSH connection and the entire operation can be handled and executed by one single command ‘ansible’. But, in some cases, where you required to execute multiple commands for a deployment, here we can build playbooks.

Playbooks are a bunch of commands which can perform multiple tasks, and each playbook are in YAML file format.

**What’s the Use of Ansible?**

**Ansible** can be used in IT infrastructure to manage and deploy software applications to remote nodes. For example, let’s say you need to deploy a single software or multiple software to 100’s of nodes by a single command, here ansible comes into picture, with the help of Ansible you can deploy as many as applications to many nodes with one single command, but you must have a little programming knowledge for understanding the ansible scripts.

We’ve compiled a series on [Ansible](https://tekslate.com/ansible-training/" \t "_blank), title ‘**Preparation for the Deployment of your IT Infrastructure with Ansible IT Automation Tool**‘, through parts 1-4 and covers the following topics.

What are the Advantages of Ansible?

– Agent-less

– very overload

– Good Performance.

**Is there a web interface / REST API / etc?**

Yes, Ansible, Inc makes a great product that makes Ansible even more powerful and easy to use. See [Ansible Tower](http://docs.ansible.com/ansible/tower.html).

**How do I submit a change to the documentation?**

Documentation for Ansible is kept in the main project git repository, and complete instructions for contributing can be found in the docs.

**When should I use {{ }}? Also, how to interpolate variables or dynamic variable names**

A steadfast rule is ‘always use {{ }} except when *when:*‘. Conditionals are always run through Jinja2 as to resolve the expression, so *when:* *failed\_when:* and *changed\_when:* are always templated and you should avoid adding *{{}}*.

In most other cases you should always use the brackets, even if previously you could use variables without specifying (like *with\_* clauses), as this made it hard to distinguish between an undefined variable and a string.

Another rule is ‘moustaches don’t stack’. We often see this:

{{ somevar\_{{other\_var}} }}

The above DOES NOT WORK, if you need to use a dynamic variable use the hostvars or vars dictionary as appropriate:

{{ hostvars[inventory\_hostname]['somevar\_' + other\_var] }}

**How to install Ansible**

Installation of Ansible Ubuntu 14.04

The best way to get Ansible for Ubuntu is to add the project’s PPA (personal package archive) to your system.

To do this effectively, we need to install the software-properties-common package, which will give us the ability to work with PPAs easily. (This package was called python-software-properties on older versions of Ubuntu.)

sudo apt-get update

sudo apt-get install software-properties-common

Once the package is installed, we can add the Ansible PPA by typing the following command:

sudo apt-add-repository ppa:ansible/ansible

Press ENTER to accept the PPA addition.

Next, we need to refresh our system’s package index so that it is aware of the packages available in the PPA. Afterwards, we can install the software:

sudo apt-get update

sudo apt-get install ansible

We now have all of the software required to administer our servers through Ansible.

**How do I generate crypted passwords for the user module?**

The mkpasswd utility that is available on most Linux systems is a great option:

mkpasswd --method=sha-512

If this utility is not installed on your system (e.g. you are using OS X) then you can still easily generate these passwords using Python. First, ensure that the [Passlib](https://code.google.com/p/passlib/) password hashing library is installed.

pip install passlib

Once the library is ready, SHA512 password values can then be generated as follows:

python -c "from passlib.hash import sha512\_crypt; import getpass; print sha512\_crypt.encrypt(getpass.getpass())"

Use the integrated [Hashing filters](http://docs.ansible.com/ansible/playbooks_filters.html#hash-filters) to generate a hashed version of a password. You shouldn’t put plaintext passwords in your playbook or host\_vars; instead, use [Vault](http://docs.ansible.com/ansible/playbooks_vault.html) to encrypt sensitive data.

***Desired to gain proficiency on Ansible?***

***Explore the blog post on*** [*Ansible training*](https://tekslate.com/ansible-training/) ***to become a pro in Ansible.***

How do I get ansible to reuse connections, enable Kerberized SSH, or have Ansible pay attention to my local SSH config file?

Switch your default connection type in the configuration file to ‘ssh’, or use ‘-c ssh’ to use Native OpenSSH for connections instead of the Python paramiko library. In Ansible 1.2.1 and later, ‘ssh’ will be used by default if OpenSSH is new enough to support ControlPersist as an option.

Paramiko is great for starting out, but the OpenSSH type offers many advanced options. You will want to run Ansible from a machine new enough to support ControlPersist, if you are using this connection type. You can still manage older clients. If you are using RHEL 6, CentOS 6, SLES 10 or SLES 11, the version of OpenSSH, is still a bit old, so consider managing from a Fedora or openSUSE client even though you are managing older nodes, or just use paramiko.

We keep paramiko as the default as if you are first installing Ansible on an EL box, it offers a better experience for new users.

**What is the best way to make content reusable/redistributable?**

If you have not done so already, read all about “Roles” in the playbooks documentation. This helps you make playbook content self-contained and works well with things like git submodules for sharing content with others.

If some of these plugin types look strange to you, see the API documentation for more details about ways Ansible can be extended.

**How do I see all the inventory vars defined for my host?**

You can see the resulting vars you define in inventory running the following command:

ansible -m debug -a "var=hostvars['hostname']" localhost

**How do I copy files recursively onto a target host?**

The “copy” module has a recursive parameter, though if you want to do something more efficient for many files, look at the “synchronize” module instead, which wraps rsync. See the module index for info on both modules.

**What is Ansible Role?**

Ansible can interact with configured clients from the command line with the ansible command, and how you can automate configuration with playbooks run through the ansible-playbook command.

The first step in creating a role is creating its directory structure. To create the base directory structure, we’re going to use a tool bundled with Ansible called ansible-galaxy:

$ ansible-galaxy init azavea.packer

azavea.packer was created successfully

That command will create an azavea.packer directory with the following structure:

├── README.md

├── defaults

│ └── main.yml

├── files

├── handlers

│ └── main.yml

├── meta

│ └── main.yml

├── tasks

│ └── main.yml

├── templates

└── vars

└── main.yml

**How do I access a variable name programmatically?**

An example may come up where we need to get the ipv4 address of an arbitrary interface, where the interface to be used may be supplied via a role parameter or other input. Variable names can be built by adding strings together, like so:

{{ hostvars[inventory\_hostname]['ansible\_' + which\_interface]['ipv4']['address'] }}

The trick about going through hostvars is necessary because it’s a dictionary of the entire namespace of variables. ‘inventory\_hostname’ is a magic variable that indicates the current host you are looping over in the host loop.

**How do I access shell environment variables?**

If you just need to access existing variables, use the ‘env’ lookup plugin. For example, to access the value of the HOME environment variable on management machine:

---

# ...

vars:

local\_home: "{{ lookup('env','HOME') }}"

If you need to set environment variables, see the Advanced Playbooks section about environments.

Ansible 1.4 will also make remote environment variables available via facts in the ‘ansible\_env’ variable:

{{ ansible\_env.SOME\_VARIABLE }}