**Ansible Interview Questions**

Contents

[Q1) What is Ansible? 6](#_Toc76504727)

[Q2) Ansible Playbooks vs Roles 6](#_Toc76504728)

[Q3) What are the advantages of using Ansible? 6](#_Toc76504729)

[Q4) Compare Ansible VS Puppet 6](#_Toc76504730)

[Q5) How Ansible Works? 7](#_Toc76504731)

[Q6) What’s the Use of Ansible. 7](#_Toc76504732)

[Q7) Explain Ansible architecture? 8](#_Toc76504733)

[Q7A) Define some easy to use modules in Ansible? 10](#_Toc76504734)

[Q8) What is CI/CD? And how Ansible is related to it? 11](#_Toc76504735)

[Q9) Can you create reusable content with Ansible? 12](#_Toc76504736)

[Q10) Is Ansible a Configuration management tool? 12](#_Toc76504737)

[Q11) What are the differences between the variable name and environment variables? 12](#_Toc76504738)

[Q12) How to create an empty file with Ansible? 13](#_Toc76504739)

[Q13) How will you set the environment variable or any path for a task or entire playbook? 13](#_Toc76504740)

[Q18) So how does Ansible work? Please explain in detail? 13](#_Toc76504741)

[Q19) What does Ansible offer? 13](#_Toc76504742)

[Q20) Can we manage Windows Nano Server using Ansible? 14](#_Toc76504743)

[Q21) Do we have any Web Interface/ Rest API etc for Ansible. 14](#_Toc76504744)

[Q22) What is Ansible Tower? 14](#_Toc76504745)

[Q23) What are the features of the Ansible Tower? 14](#_Toc76504746)

[Q24) How do change the documentation and submit it? 14](#_Toc76504747)

[Q37) What is idempotency? 14](#_Toc76504748)

[Q38) Can you create encrypted files with Ansible? 14](#_Toc76504749)

[Q39) What is the difference between a playbook and a play? 15](#_Toc76504750)

[Q31) How can you access a list of Ansible\_Variables? 15](#_Toc76504751)

[Q36) Can you keep data secret in the playbook? 15](#_Toc76504752)

[Q43) What are handlers? 15](#_Toc76504753)

[Q44) How will you upgrade Ansible? 15](#_Toc76504754)

[Q45) Ansible vs Chef? 16](#_Toc76504755)

[Q49) Can you explain how to handle various machines requiring different user accounts or ports to log in? 16](#_Toc76504756)

[Q50) Do you know what language Ansible is written in? 16](#_Toc76504757)

[Q51) Please explain what is Red Hat Ansible? 17](#_Toc76504758)

[Q52) Is Ansible is an open-source tool? 17](#_Toc76504759)

[Q53) Why you have to learn Ansible? 17](#_Toc76504760)

[Q54) What are Ansible server requirements? 17](#_Toc76504761)

[Q55)  How to install Ansible on CentOS? 17](#_Toc76504762)

[Q56) How can you connect to other devices within Ansible? 18](#_Toc76504763)

[Q57) Can you build your own modules with Ansible? 18](#_Toc76504764)

[Q58) How can you find information in Ansible? 18](#_Toc76504765)

[Q59) What does Fact mean in Ansible? 18](#_Toc76504766)

[Q66) Explain in detail ad-hoc commands? 19](#_Toc76504767)

[Ansible Interview Questions for Experienced 21](#_Toc76504768)

[Q25) How do you access Shell Environment Variables? 21](#_Toc76504769)

[Q26) How can you speed up management inside EC2? 21](#_Toc76504770)

[Q27) Is it possible to increase the Ansible reboot module to more than 600 seconds? 21](#_Toc76504771)

[Q28) How can you use docker modules in Ansible? 21](#_Toc76504772)

[Q29) Explain how you will copy files recursively onto a target host? 21](#_Toc76504773)

[Q30) How can you disable cowsay? 22](#_Toc76504774)

[Q32) How can you see all the variables specific to my host? 22](#_Toc76504775)

[Q33) How do you access a variable name programmatically? 22](#_Toc76504776)

[Q34) How to configure a jump host for accessing servers that have no direct access? 22](#_Toc76504777)

[Q35) Explain how you can generate encrypted passwords for the user module? 23](#_Toc76504778)

[Q40) How will you get access to the ansible host when I delegate a task? 23](#_Toc76504779)

[Q41) Explain the Ansible Tag's usage? 23](#_Toc76504780)

[Q42) How can you filter out tasks in tags? 23](#_Toc76504781)

[Q46) Why don’t you ship in X format? 23](#_Toc76504782)

[Q47) What is Ansible can do? 24](#_Toc76504783)

[Q48) Please define what is Ansible Galaxy? 24](#_Toc76504784)

[Q60) What is ask\_pass in ansible? 24](#_Toc76504785)

[Q61) Explain What is ask\_sudo\_pass 24](#_Toc76504786)

[Q62) Explain what is ask\_vault\_pass? 24](#_Toc76504787)

[Q63) Explain Callback\_plugin in Ansible? 25](#_Toc76504788)

[Q64) Explain Module utilities in Ansible? 25](#_Toc76504789)

[Q65) Where is the unit testing is available in Ansible? 25](#_Toc76504790)

[How can I set the PATH or any other environment variable for a task or entire play? 25](#_Toc76504791)

[How do I handle different machines needing different user accounts or ports to log in with? 25](#_Toc76504792)

[How do I get ansible to reuse connections, enable Kerberized SSH, or have Ansible pay attention to my local SSH config file? 26](#_Toc76504793)

[How do I configure a jump host to access servers that I have no direct access to? 26](#_Toc76504794)

[How do I get Ansible to notice a dead target in a timely manner? 27](#_Toc76504795)

[How do I speed up run of ansible for servers from cloud providers (EC2, openstack,.. )? 27](#_Toc76504796)

[How do I handle not having a Python interpreter at /usr/bin/python on a remote machine? 27](#_Toc76504797)

[How do I handle the package dependencies required by Ansible package dependencies during Ansible installation ? 28](#_Toc76504798)

[Common Platform Issues 28](#_Toc76504799)

[What customer platforms does Red Hat support? 28](#_Toc76504800)

[Running in a virtualenv 29](#_Toc76504801)

[Running on BSD 29](#_Toc76504802)

[Running on Solaris 29](#_Toc76504803)

[Running on z/OS 30](#_Toc76504804)

[Running under fakeroot 31](#_Toc76504805)

[What is the best way to make content reusable/redistributable? 31](#_Toc76504806)

[Where does the configuration file live and what can I configure in it? 31](#_Toc76504807)

[How do I disable cowsay? 31](#_Toc76504808)

[How do I see a list of all of the ansible\_ variables? 31](#_Toc76504809)

[How do I see all the inventory variables defined for my host? 32](#_Toc76504810)

[How do I see all the variables specific to my host? 32](#_Toc76504811)

[How do I loop over a list of hosts in a group, inside of a template? 32](#_Toc76504812)

[How do I access a variable name programmatically? 32](#_Toc76504813)

[How do I access a group variable? 33](#_Toc76504814)

[How do I access a variable of the first host in a group? 33](#_Toc76504815)

[How do I copy files recursively onto a target host? 34](#_Toc76504816)

[How do I access shell environment variables? 34](#_Toc76504817)

[How do I generate encrypted passwords for the user module? 34](#_Toc76504818)

[Ansible allows dot notation and array notation for variables. Which notation should I use? 35](#_Toc76504819)

[When is it unsafe to bulk-set task arguments from a variable? 35](#_Toc76504820)

[Can I get training on Ansible? 36](#_Toc76504821)

[Is there a web interface / REST API / GUI? 36](#_Toc76504822)

[How do I keep secret data in my playbook? 36](#_Toc76504823)

[When should I use {{ }}? Also, how to interpolate variables or dynamic variable names 37](#_Toc76504824)

[Why don’t you ship ansible in wheel format (or other packaging format) ? 37](#_Toc76504825)

[How do I get the original ansible\_host when I delegate a task? 37](#_Toc76504826)

[How do I fix ‘protocol error: filename does not match request’ when fetching a file? 37](#_Toc76504827)

[Does Ansible support multiple factor authentication 2FA/MFA/biometrics/finterprint/usbkey/OTP/… 38](#_Toc76504828)

[How do I submit a change to the documentation? 39](#_Toc76504829)

Idempotency

### Q1) What is Ansible?

Ansible is developed in [Python language](https://mindmajix.com/python-tutorial). It is a software tool. It is useful while deploying any application using ssh without any downtime. Using this tool one can manage and configure software applications very easily.

### Q2) Ansible Playbooks vs Roles

|  |  |
| --- | --- |
| **Roles** | **Playbooks** |
| Roles are reusable subsets of a play. | Playbooks contain Plays. |
| A set of tasks for accomplishing a certain role. | Mapps among hosts and roles. |
| Example: common, webservers. | Example: site.yml, fooservers.yml, webservers.yml. |

### Q3) What are the advantages of using Ansible?

The main three advantages of using this tool are,i.e. Ansible

1. Agentless
2. Very low overhead
3. Good performance
4. Configuration scripts will be written in YAML which is easy to write and understand

### Q4) Compare Ansible VS Puppet

|  |  |
| --- | --- |
| **Ansible** | **Puppet** |
| Simplest Technology | Complex Technology |
| Written in YAML language | Written in Ruby language |
| Automated workflow for Continuous Delivery | Visualization and reporting |
| Agent-less install and deploy | Easy install |
| No support for Windows | Support for all major OS’s |
| GUI -work under progress | Good GUI |
| CLI accepts commands in almost any language | Must learn the Puppet DSL |

### Q5) How Ansible Works?

There are many similar automation tools available like [Puppet](https://mindmajix.com/puppet-tutorial#puppet), Capistrano, Chef, Salt, Space Walk, etc, but Ansible categorizes into two types of servers: 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 the controlling 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, which means no need for any agent installation on remote nodes, so it means there are no 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 an 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.

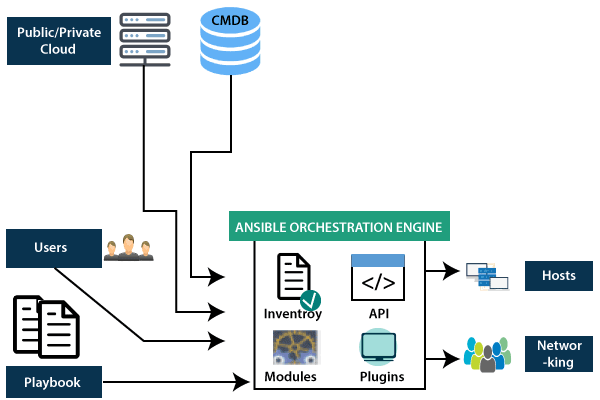
### Q6) What’s the Use of Ansible.

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 the picture, with the help of Ansible you can deploy as many applications as possible to many nodes with one single command, but you must have a little programming knowledge for understanding the ansible scripts.

### Q7) Explain Ansible architecture?

Ansible automation engine is the main component of Ansible, which interacts directly with the configuration management database, cloud services, and various users who write playbooks to execute it.

The below figure depicts the Ansible architecture:



The following are the components of the Ansible Automation engine:

* **Modules:** Ansible works effectively by connecting nodes and pushing out scripts called "Ansible modules". It helps to manage packages, system resources, files, libraries, etc.

**Command**

**Shell**

**Service**

**Yum**

**Apt**

**Apk**

**Copy**

**Cron**

**-** name**:** Copy file with owner and permissions

copy**:**

src**:** /srv/myfiles/foo.conf

dest**:** /etc/foo.conf

owner**:** foo

group**:** foo

mode**:** '0644'

**-** name**:** Copy file with owner and permission, using symbolic representation

copy**:**

src**:** /srv/myfiles/foo.conf

dest**:** /etc/foo.conf

owner**:** foo

group**:** foo

mode**:** u=rw,g=r,o=r

**-** name**:** Ensure a job that runs at 2 and 5 exists. Creates an entry like "0 5,2 \* \* ls -alh > /dev/null"

cron**:**

name**:** "checkdirs"

minute**:** "0"

hour**:** "5,2"

job**:** "ls-alh>/dev/null"

* **Inventories:** These are the lists of nodes or hosts containing their databases, servers, IP addresses, etc.

/etc/ansible/hosts

mail.example.com

[webservers]

foo.example.com

bar.example.com

[dbservers]

one.example.com

two.example.com

three.example.com

* **APIs:** These used for commuting public or private cloud services.
* **Plugins:** Plugins augment Ansible's core functionality. Also offers extensions and options for the core features of Ansible - transforming data, connecting to inventory, logging output, and more.
* **Playbooks:** Describes the tasks that need to be executed. They are simple code files written in YAML format and can be used to declare configurations, automating tasks, etc.
* **Hosts:** Hosts are node systems that are automated by Ansible on any machine like Linux, RedHat, Windows, etc.
* **Networking:** Ansible can be used to automate multiple networks and services. It uses a secure and simple automation framework for IT operations and development.
* **Cloud:** A system of remote servers that allows you to store, manage, and process data, rather than a local server.
* **CMDB:** It is a type of repository which acts as a data warehouse for IT installations.

### Q7A) Define some easy to use modules in Ansible?

#### Ping Module

[Ping](https://gist.github.com/slathia15/450ecc43059ed73c75a74688cbdba35e) is used when we want to check whether the connection with our hosts defined in the inventory file is established or not.

ansible test-servers -m ping -u ec2-user

**ping**changes to **pong** if an SSH connection is established.

#### Setup Module

The setup module is used when we want to see the information of all the hosts, their configuration, and detailed information.

ansible test-servers -m setup -u ec2-user

#### Copy Module

The [copy module](https://gist.github.com/slathia15/65c164727aa572d1f913c9fcc8d63db3) is often used in writing playbooks when we want to copy a file from a remote server to destination nodes.

For example, suppose we want to copy a file from a remote server to all destination machines.

ansible test-servers -m copy -a 'src=/home/knoldus/Personal/blogs/blog3.txt dest=/tmp' -u ec2-user

#### Yum Module

We use the [Yum](https://gist.github.com/slathia15/d228b0c2d638a45ce98555190680afb2) module to install a service.

ansible test-servers -m yum -a 'name=httpd state=present' -become -u ec2-user

Apache2 will be installed on our machines.

The key point to note here is that we have to use -become, which is new in version 2.6; before, we had to use -s.

#### Shell Module

When we want to run UNIX commands then we use shell module

ansible test-servers -m shell -a 'ls -la' -u ec2-user

This will display all the files present in our machine with their permissions.

#### Service Module

When we want to ensure the state of a service that is service is running we use the service module.

ansible test-servers -m service -a 'name=httpd state=started' -become -u ec2-user

Apache2 is up on my machine.

#### Debug Module

To print a msg on hosts we use Debug module.

ansible test-servers -m debug -a 'msg=Hello' -u ec2-user

Hello, a message is printed on my machine.

#### Template Module

The Template module is used to copy a configuration file from the local system to the host server. It is the same as the copy module, but it dynamically binds group variables defined by us.

#### Include Module

When we want to include another playbook in our playbook, then we use the [Include module](https://gist.github.com/slathia15/685fa46cbc6c85af5029617dfd7fbdb9).

#### User Module

To add a particular user to our module we can use User module. [Here](https://gist.github.com/slathia15/a91dc8f67c31afc0978703674181ec40), we have added a user named Sachin to our module.

### Q8) What is CI/CD? And how Ansible is related to it?

CI/CD is one of the best software development practices to implement and develop code effectively. CI stands for Continuous Integration, and CD stands for continuous delivery. Continuous Integration is a collection of practices that drive developers to implement and check-in code to version control repositories. Continuous delivery picks up where continuous Integration ends. This process builds software in such a way that software will be released into production at any given time.

Ansible is an excellent tool for CI/CD processes, which provide a stable infrastructure to a provision target environment and then deploy the application to it.

### Q9) Can you create reusable content with Ansible?

Yes, Ansible has the concept of roles that helps to create reusable content. To create a role, you need to follow Ansible's conventions of structuring directories and naming files.

For example, for ***webserver*** role below will be the directory structure:

|  |  |
| --- | --- |
| Path/filename | Descriptions |
| /playbook/roles/webserver/defaults/main.yml | defaults like port no 8080 etc |
| /playbook/roles/webserver/vars/main.yml | for defining variables |
| /playbook/roles/webserver/handlers/main.yml | For handlers |
| /playbook/roles/webserver/files/main.yml | File to be transferred to remote host |
| /playbook/roles/webserver/meta/main.yml | File that establish roles dependency like Author Name, supported platform, dependency. |
| /playbook/master.yml | Have target information like  Inventory file, user, connection etc. |

### Q10) Is Ansible a Configuration management tool?

Configuration management is the practice to handle updates and manage the consistency of a product's performance over a particular period of time. Ansible is an open-source IT Configuration Management tool, which automates a wide variety of challenges in complex multi-tier IT application environments.

### Q11) What are the differences between the variable name and environment variables?

|  |  |
| --- | --- |
| **Variable Names** | **Environment Variables** |
| By adding strings, we can build variable names | By accessing existing variables, we can access environment variables |
| Supports adding more strings | The advanced playbooks section sets the environment variables. |
| Use the IPV4 address for variable names. | Use {{ansible\_env.SOME\_VARIABLE}} for remote environment variables |

### Q12) How to create an empty file with Ansible?

To create an empty file, Ansible uses a file module. For this, we need to set up two parameters.

1. **Path**- This place represents the location where the file gets created, either the relative or an absolute path. Also, the name of the file includes here.
2. **State**- For creating a new file, this parameter should be set to touch.

### Q13) How will you set the environment variable or any path for a task or entire playbook?

To set the environment variables, we use the environment keyword. We'll use it at the task or other levels in the play:

environment:

PATH: "{{ ansible\_env.PATH }}:/thingy/bin"

SOME: value

### Q18) So how does Ansible work? Please explain in detail?

Within the market, they are many automation tools like Puppet, Capistrano, Chef, Salt, Space Walk, etc.

* When it comes to Ansible, this tool is categorized into two types of servers:  
      1. Controlling machines  
      2. Nodes
* Ansible is an agentless tool so it doesn’t require any mandatory installations on remote nodes. So, there are no background programs that are executed while it is managing any nodes.
* Ansible is able to handle a lot of nodes from a single system over an SSH connection.
* Playbooks are defined as a bunch of commands where they are capable of performing multiple tasks and they are in YAML file format.

### Q19) What does Ansible offer?

 Ansible  offers:

* Security and Compliance policy integration
* Automated workflow for Continuous Delivery
* Simplified orchestration
* App deployment
* Configuration management
* Streamlined provisioning

### Q20) Can we manage Windows Nano Server using Ansible?

No, it is not possible to manage Windows Nano Server using Ansible as it doesn't have full access to the .Net framework, which is primarily used by internal components and modules.

### Q21) Do we have any Web Interface/ Rest API etc for Ansible.

Yes, Ansible Inc makes a great efficient tool. It is easy to use.

### Q22) What is Ansible Tower?

Ansible is classified as a web-based solution which makes Ansible very easy to use. It is considered to be or acts like a hub for all of your automation tasks. The tower is free for usage till 10 nodes.

### Q23) What are the features of the Ansible Tower?

Features of the Ansible Tower are:

* Ansible Dashboard
* Real-time job status updates
* Multi-playbook workflows
* Who Ran What Job When
* Scale capacity with tower clusters
* Integrated notifications
* Schedule ansible jobs
* Manage and track inventory
* Remote command execution
* REST API & Tower CLI Tool

### Q24) How do change the documentation and submit it?

Usually, the documentation is kept in the main project folder in the git repository. For complete instructions on this can be available in docs.

### Q37) What is idempotency?

Idempotence is an essential feature of Ansible, which helps you to execute one or more tasks on a server as many times as needed, but without changing the result beyond the initial application.

### Q38) Can you create encrypted files with Ansible?

Yes, using the 'ansible-vault create' command, we can create encrypted files

$ ansible-vault create filename.yaml

### Q39) What is the difference between a playbook and a play?

A playbook is a list of plays. A play is a set of tasks and roles that run on one or more managed hosts. Play includes one or more tasks.

### Q31) How can you access a list of Ansible\_Variables?

By default, Ansible gathers facts under machines under management. Further, these facts are accessed in Playbooks and in templates. One of the best ways to view a list of all the facts that are available in a machine, then need to run the setup module in the ad-hoc way:

Ansible- m setup hostname

Once this statement is executed, it will print out a dictionary of all the facts that are available for that particular host. This is the best way to access the list of Ansible\_variables.

### Q36) Can you keep data secret in the playbook?

Yes. If any task that you want to keep secret in the playbook when using -v (verbose) mode, the following playbook attribute will be helpful:

- name: secret task

shell: /usr/bin/do\_something --value={{ secret\_value }}

no\_log: True

It hides sensitive information from others and provides the verbose output.

### Q43) What are handlers?

In Ansible, handlers are just like normal tasks in a playbook but run when tasks include the notify directive and also indicate that it changed something. It runs only once after all the tasks executed in a particular play. It automatically loads through roles/<role\_name>/handlers/main.yaml.

They are used to trigger the status of a service, such as restarting or stopping a service.

### Q44) How will you upgrade Ansible?

Using the command "sudo pip install ansible==<version-number>", we can easily upgrade Ansible.

### Q45) Ansible vs Chef?

|  |  |
| --- | --- |
| **Ansible** | **Chef** |
| Ansible is easier to set up and provides faster performance | Compared to Ansible, Chef is not very easy to set up |
| Ansible uses YAML (Python) for managing configurations | Chef uses DSL (Ruby) for managing configurations |
| Highly scalable | Highly scalable |
| Ansible charges annually $10,000 | Chef Automate charges an annual fee of $13700 |

### Q49) Can you explain how to handle various machines requiring different user accounts or ports to log in?

Just by setting inventories in the inventory file, we can handle various machines requiring different user accounts or ports to log in.

For example, the following hosts have different ports and usernames:

[webservers]

asdf.example.com ansible\_port=5000 ansible\_user=alice

jkl.example.com ansible\_port=5001 ansible\_user=bob

You can specify the connection type to be used by:

[testcluster]

localhost ansible\_connection=local

/path/to/chroot1 ansible\_connection=chroot

foo.example.com ansible\_connection=paramiko

File them in a group\_vars/<group-name> file.

### Q50) Do you know what language Ansible is written in?

Ansible is written in Python and PowerShell

### Q51) Please explain what is Red Hat Ansible?

Ansible and Ansible Tower by Red Hat, both are an end to end complete automation platforms which are capable of providing the following features or functionalities:

* Provisioning
* Deploying applications
* Orchestrating workflows
* Manage IT systems
* Configuration of IT systems
* Networks
* Applications

All of these activities are dealt with by Ansible where it can help the business to solve real-time business problems.

### Q52) Is Ansible is an open-source tool?

Yes, Ansible is an open-source tool that is a powerful automation software tool that one can use.

### Q53) Why you have to learn Ansible?

Ansible is more a tool for servers but does it have anything for networking. If you closely look into it, there is some support available in the market for networking devices. Using this tool, it will give you an overall view of your environment and also knowledge of how it works when it comes to network automation.

It is one of those tools where it is considered to be good to explore a new tool.

### Q54) What are Ansible server requirements?

You need to have a virtual machine with Linux installed, which has Python 2.6 version or higher.

### Q55)  How to install Ansible on CentOS?

**Step 1:** Update your Control Node

yum update

**Step 2:** Install the EPEL Repository

yum install epel-release

**Step 3:** Install Ansible

yum install Ansible

### Q56) How can you connect to other devices within Ansible?

Once, Ansible is installed and the basic setup has been completed, an inventory is created. This would be the base and one can start testing ansible. To connect to a different device then you have to use “Ping module”. This can be used as a simple connection test.

Ansible - m ping all

### Q57) Can you build your own modules with Ansible?

Yes, we can create or own modules within Ansible.

It is an open-source tool that primarily works on Python. If you are good at programming in Python you can start creating your own modules in few hours from scratch and you don't need to have any prior knowledge of the same.

### Q58) How can you find information in Ansible?

After completing the basic setup, one has to make sure to find out the module called the “setup” module. Using this setup module, you will be able to find out a lot of information.

### Q59) What does Fact mean in Ansible?

The term “Facts” is commonly used in an Ansible environment. They are described in the playbook areas where it displays known and discovered variables about the system.  Facts are used to implement conditionals executions and also used for getting ad-hoc information of information.

You can see all the facts via:

$ ansible all- m setup

So if you want to extract only a certain part of the information then you can use the “setup” module where you will have an option to filter out the output and just get hold of the fact that you are in need of.

### Q66) Explain in detail ad-hoc commands?

Well, ad-hoc commands are nothing but a command which is used to do something quickly and it is more sort of one-time use. Unlike, the playbook is used for a repeated action which is something that is very useful in the Ansible environment. But there might be scenarios where we want to use ad-hoc commands which can simply do the required activity and it is a nonrepetitive activity.

## Ansible Interview Questions for Experienced

### Q25) How do you access Shell Environment Variables?

If you are just looking to access the existing variables, then you can use the “env” lookup plugin.

For example:

Accessing the value of Home environment variable on management machine:

local\_home:”{{lookup(‘env’,’HOME’)}}”

### Q26) How can you speed up management inside EC2?

It is not advised to manage a group of EC2 machines from your laptop. The best way is to connect to a management node inside Ec2 first and then execute Ansible from there.

### Q27) Is it possible to increase the Ansible reboot module to more than 600 seconds?

Yes, it is possible to increase the Ansible reboot module to specific values using the below syntax:

- name: Reboot a Linux system

reboot:

reboot\_timeout: 1000

### Q28) How can you use docker modules in Ansible?

Docker modules require docker-py installed on the host running Ansible.

$ pip install 'docker-py>=1.7.0'

The docker\_service module also requires docker-compose

$ pip install 'docker-compose>=1.7.0'

### Q29) Explain how you will copy files recursively onto a target host?

The copy file in Ansible has a recursive parameter. If you have to copy files for a large number of files, then the synchronizing module is the best choice for it.

- synchronize:

src: /first/absolute/path

dest: /second/absolute/path

delegate\_to: "{{ inventory\_hostname }}"

### Q30) How can you disable cowsay?

If cowsay is installed then executing your playbooks within Ansible is very smooth.

Even if you think that you want to work in a professional cow free environment, then you will have two options:

1. Uninstall cowsay
2. Setting up value for the environment variable, like below

Export ANSIBLE\_NOCOWS=1

### Q32) How can you see all the variables specific to my host?

To see all the host-specific variables, that include all facts and other resources are:

Ansible - m debug- a “var=hostvars[‘hostname’]” localhost

### Q33) How do you access a variable name programmatically?

By adding strings together, the variables names are built programmatically like below format:

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

'inventory\_hostname' is a variable that represents the present host you are looping over.

### Q34) How to configure a jump host for accessing servers that have no direct access?

We should set a ProxyCommand in the ansible\_ssh\_common\_args inventory variable. For connecting to the relevant host, arguments defined in this variable are added to scp/ssh/sftp command line.

For example,

[gatewayed]

foo ansible\_host=192.0.2.1

bar ansible\_host=192.0.2.2

With the following contents, create the group\_vars/gatewayed.yml

ansible\_ssh\_common\_args: '-o ProxyCommand="ssh -W %h:%p -q user@gateway.example.com"'

When connecting to any hosts in the group gatewayed, Ansible will append these arguments to the command line.

### Q35) Explain how you can generate encrypted passwords for the user module?

Ansible ad-hoc command is the easiest option:

ansible all -i localhost, -m debug -a "msg={{ 'mypassword' | password\_hash('sha512', 'mysecret') }}"

The mkpasswd utility available on the Linux systems is also the best option:

mkpasswd --method=sha-512

### Q40) How will you get access to the ansible host when I delegate a task?

We can access it through host variables and even works for all the overridden variables like ansible\_port, ansible\_user, etc.

original\_host: "{{ hostvars[inventory\_hostname]['ansible\_host'] }}"

### Q41) Explain the Ansible Tag's usage?

A tag is an attribute that sets the ansible structure(plays, tasks, roles). When there's an extensive playbook needed, it's more useful to run just a part of it as opposed to the entire thing. That's where tags usage is required.

### Q42) How can you filter out tasks in tags?

* Use –tags or –skip-tags options on the command line
* Use the TAGS\_RUN and TAGS\_SKIP options, If you're in Ansible configuration settings.

### Q46) Why don’t you ship in X format?

They are several reasons for not shipping in X format. In general, it caters to maintainability. Within the market, they are tons of different ways to ship software and it is very tedious to support all of them.

### Q47) What is Ansible can do?

Ansible can do the following for us:

1. Configuration management
2. Application deployment
3. Task automation
4. IT orchestration

### Q48) Please define what is Ansible Galaxy?

Ansible Galaxy refers to the website Galaxy where the users will be able to share all the roles to a CLI ( Command Line Interface) where the installation, creation, and managing of roles happen

### Q60) What is ask\_pass in ansible?

The ask\_pass is controlled in Ansible Playbook.

This controls whether ansible-playbook prompts a password by default. Usually, the default behavior is no:

It is always set to ask\_pass=True

If you are using SSH keys for authentication purposes, then you really don’t have to change this setting at all.

### Q61) Explain What is ask\_sudo\_pass

This control is very similar to ask\_pass

The ask\_sudo\_pass controls the Ansible Playbook to prompt a Sudo password. Usually, the default behavior is no:

ask\_sudo\_pass= True

One has to make sure and change this setting where the sudo passwords are enabled most of the time.

### Q62) Explain what is ask\_vault\_pass?

Using this control we can determine whether Ansible Playbook should prompt a password for the vault password by default. As usual, the default behavior is no

ask\_vault\_pass= True

### Q63) Explain Callback\_plugin in Ansible?

Callbacks are explained as a piece of code in ansible environments where to get is used to call a specific event and permit the notifications.

This is more sort of a developer-related feature and allows low-level extensions around ansible so that they can be loaded from different locations without any problem.

### Q64) Explain Module utilities in Ansible?

Ansible provides a wide variety of module utilities that help the developers while developing their own modules. The basic.py is a module that provides the main entry point for accessing the Ansible library and using those as basics one can start off working.

### Q65) Where is the unit testing is available in Ansible?

Unit tests for all the modules are available in .test/units/modules. Firstly you have to set up your testing environment

## How can I set the PATH or any other environment variable for a task or entire play?

Setting environment variables can be done with the environment keyword. It can be used at the task or other levels in the play.

shell**:**

cmd**:** date

environment**:**

LANG=fr\_FR.UTF-8

hosts**:** servers

environment**:**

PATH**:** "{{ansible\_env.PATH}}:/thingy/bin"

SOME**:** value

**Note**

starting in 2.0.1 the setup task from gather\_facts also inherits the environment directive from the play, you might need to use the |default filter to avoid errors if setting this at play level.

## How do I handle different machines needing different user accounts or ports to log in with?

Setting inventory variables in the inventory file is the easiest way.

For instance, suppose these hosts have different usernames and ports:

**[webservers]**

asdf.example.com ansible\_port**=**5000 ansible\_user=alice

jkl.example.com ansible\_port**=**5001 ansible\_user=bob

You can also dictate the connection type to be used, if you want:

**[testcluster]**

localhost ansible\_connection**=**local

/path/to/chroot1 ansible\_connection**=**chroot

foo.example.com ansible\_connection**=**paramiko

You may also wish to keep these in group variables instead, or file them in a group\_vars/<groupname> file. See the rest of the documentation for more information about how to organize variables.

## 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 these enterprise operating systems, it offers a better experience for new users.

## How do I configure a jump host to access servers that I have no direct access to?

You can set a ProxyCommand in the ansible\_ssh\_common\_args inventory variable. Any arguments specified in this variable are added to the sftp/scp/ssh command line when connecting to the relevant host(s). Consider the following inventory group:

**[gatewayed]**

foo ansible\_host**=**192.0.2.1

bar ansible\_host**=**192.0.2.2

You can create group\_vars/gatewayed.yml with the following contents:

ansible\_ssh\_common\_args**:** '-oProxyCommand="ssh-W%h:%p-quser@gateway.example.com"'

Ansible will append these arguments to the command line when trying to connect to any hosts in the group gatewayed. (These arguments are used in addition to any ssh\_args from ansible.cfg, so you do not need to repeat global ControlPersist settings in ansible\_ssh\_common\_args.)

Note that ssh -W is available only with OpenSSH 5.4 or later. With older versions, it’s necessary to execute nc %h:%p or some equivalent command on the bastion host.

With earlier versions of Ansible, it was necessary to configure a suitable ProxyCommand for one or more hosts in ~/.ssh/config, or globally by setting ssh\_args in ansible.cfg.

## How do I get Ansible to notice a dead target in a timely manner?

You can add -o ServerAliveInterval=NumberOfSeconds in ssh\_args from ansible.cfg. Without this option, SSH and therefore Ansible will wait until the TCP connection times out. Another solution is to add ServerAliveInterval into your global SSH configuration. A good value for ServerAliveInterval is up to you to decide; keep in mind that ServerAliveCountMax=3 is the SSH default so any value you set will be tripled before terminating the SSH session.

## How do I speed up run of ansible for servers from cloud providers (EC2, openstack,.. )?

Don’t try to manage a fleet of machines of a cloud provider from your laptop. Rather connect to a management node inside this cloud provider first and run Ansible from there.

## How do I handle not having a Python interpreter at /usr/bin/python on a remote machine?

While you can write Ansible modules in any language, most Ansible modules are written in Python, including the ones central to letting Ansible work.

By default, Ansible assumes it can find a **/usr/bin/python** on your remote system that is either Python2, version 2.6 or higher or Python3, 3.5 or higher.

Setting the inventory variable ansible\_python\_interpreter on any host will tell Ansible to auto-replace the Python interpreter with that value instead. Thus, you can point to any Python you want on the system if **/usr/bin/python** on your system does not point to a compatible Python interpreter.

Some platforms may only have Python 3 installed by default. If it is not installed as **/usr/bin/python**, you will need to configure the path to the interpreter via ansible\_python\_interpreter. Although most core modules will work with Python 3, there may be some special purpose ones which do not or you may encounter a bug in an edge case. As a temporary workaround you can install Python 2 on the managed host and configure Ansible to use that Python via ansible\_python\_interpreter. If there’s no mention in the module’s documentation that the module requires Python 2, you can also report a bug on our [bug tracker](https://github.com/ansible/ansible/issues) so that the incompatibility can be fixed in a future release.

Do not replace the shebang lines of your python modules. Ansible will do this for you automatically at deploy time.

Also, this works for ANY interpreter, for example ruby: ansible\_ruby\_interpreter, perl: ansible\_perl\_interpreter, and so on, so you can use this for custom modules written in any scripting language and control the interpreter location.

Keep in mind that if you put env in your module shebang line (#!/usr/bin/env <other>), this facility will be ignored so you will be at the mercy of the remote $PATH.

## How do I handle the package dependencies required by Ansible package dependencies during Ansible installation ?

While installing Ansible, sometimes you may encounter errors such as No package ‘libffi’ found or fatal error: Python.h: No such file or directory These errors are generally caused by the missing packages, which are dependencies of the packages required by Ansible. For example, libffi package is dependency of pynacl and paramiko (Ansible -> paramiko -> pynacl -> libffi).

In order to solve these kinds of dependency issues, you might need to install required packages using the OS native package managers, such as yum, dnf, or apt, or as mentioned in the package installation guide.

Refer to the documentation of the respective package for such dependencies and their installation methods.

## Common Platform Issues

### What customer platforms does Red Hat support?

A number of them! For a definitive list please see this [Knowledge Base article](https://access.redhat.com/articles/3168091).

### Running in a virtualenv

You can install Ansible into a virtualenv on the controller quite simply:

$ virtualenv ansible

$ source ./ansible/bin/activate

$ pip install ansible

If you want to run under Python 3 instead of Python 2 you may want to change that slightly:

$ virtualenv -p python3 ansible

$ source ./ansible/bin/activate

$ pip install ansible

If you need to use any libraries which are not available via pip (for instance, SELinux Python bindings on systems such as Red Hat Enterprise Linux or Fedora that have SELinux enabled), then you need to install them into the virtualenv. There are two methods:

* When you create the virtualenv, specify --system-site-packages to make use of any libraries installed in the system’s Python:
* $ virtualenv ansible --system-site-packages
* Copy those files in manually from the system. For instance, for SELinux bindings you might do:
* $ virtualenv ansible --system-site-packages
* $ cp -r -v /usr/lib64/python3.\*/site-packages/selinux/ ./py3-ansible/lib64/python3.\*/site-packages/
* $ cp -v /usr/lib64/python3.\*/site-packages/\*selinux\*.so ./py3-ansible/lib64/python3.\*/site-packages/

### Running on BSD

**See also**

[Ansible and BSD](https://docs.ansible.com/ansible/latest/user_guide/intro_bsd.html#working-with-bsd)

### Running on Solaris

By default, Solaris 10 and earlier run a non-POSIX shell which does not correctly expand the default tmp directory Ansible uses ( ~/.ansible/tmp). If you see module failures on Solaris machines, this is likely the problem. There are several workarounds:

* You can set remote\_tmp to a path that will expand correctly with the shell you are using (see the plugin documentation for [C shell](https://docs.ansible.com/ansible/2.9/plugins/shell/csh.html#csh-shell), [fish shell](https://docs.ansible.com/ansible/2.9/plugins/shell/fish.html#fish-shell), and [Powershell](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/powershell_shell.html#powershell-shell)). For example, in the ansible config file you can set:
* remote\_tmp=$HOME/.ansible/tmp

In Ansible 2.5 and later, you can also set it per-host in inventory like this:

solaris1 ansible\_remote\_tmp=$HOME/.ansible/tmp

* You can set [ansible\_shell\_executable](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#ansible-shell-executable) to the path to a POSIX compatible shell. For instance, many Solaris hosts have a POSIX shell located at /usr/xpg4/bin/sh so you can set this in inventory like so:
* solaris1 ansible\_shell\_executable=/usr/xpg4/bin/sh

(bash, ksh, and zsh should also be POSIX compatible if you have any of those installed).

### Running on z/OS

There are a few common errors that one might run into when trying to execute Ansible on z/OS as a target.

* Version 2.7.6 of python for z/OS will not work with Ansible because it represents strings internally as EBCDIC.

To get around this limitation, download and install a later version of [python for z/OS](https://www.rocketsoftware.com/zos-open-source) (2.7.13 or 3.6.1) that represents strings internally as ASCII. Version 2.7.13 is verified to work.

* When pipelining = False in /etc/ansible/ansible.cfg then Ansible modules are transferred in binary mode via sftp however execution of python fails with

**Error**

SyntaxError: Non-UTF-8 code starting with ‘\x83’ in file /a/user1/.ansible/tmp/ansible-tmp-1548232945.35-274513842609025/AnsiballZ\_stat.py on line 1, but no encoding declared; see <https://python.org/dev/peps/pep-0263/> for details

To fix it set pipelining = True in /etc/ansible/ansible.cfg.

* Python interpret cannot be found in default location /usr/bin/python on target host.

**Error**

/usr/bin/python: EDC5129I No such file or directory

To fix this set the path to the python installation in your inventory like so:

zos1 ansible\_python\_interpreter=/usr/lpp/python/python-2017-04-12-py27/python27/bin/python

* Start of python fails with The module libpython2.7.so was not found.

**Error**

EE3501S The module libpython2.7.so was not found.

On z/OS, you must execute python from gnu bash. If gnu bash is installed at /usr/lpp/bash, you can fix this in your inventory by specifying an ansible\_shell\_executable:

zos1 ansible\_shell\_executable=/usr/lpp/bash/bin/bash

### Running under fakeroot

Some issues arise as fakeroot does not create a full nor POSIX compliant system by default. It is known that it will not correctly expand the default tmp directory Ansible uses (~/.ansible/tmp). If you see module failures, this is likely the problem. The simple workaround is to set remote\_tmp to a path that will expand correctly (see documentation of the shell plugin you are using for specifics).

For example, in the ansible config file (or via environment variable) you can set:

remote\_tmp=$HOME/.ansible/tmp

## 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.

## Where does the configuration file live and what can I configure in it?

See [Configuring Ansible](https://docs.ansible.com/ansible/latest/installation_guide/intro_configuration.html#intro-configuration).

## How do I disable cowsay?

If cowsay is installed, Ansible takes it upon itself to make your day happier when running playbooks. If you decide that you would like to work in a professional cow-free environment, you can either uninstall cowsay, set nocows=1 in ansible.cfg, or set the [**ANSIBLE\_NOCOWS**](https://docs.ansible.com/ansible/latest/reference_appendices/config.html#envvar-ANSIBLE_NOCOWS) environment variable:

export ANSIBLE\_NOCOWS=1

## How do I see a list of all of the ansible\_ variables?

Ansible by default gathers “facts” about the machines under management, and these facts can be accessed in playbooks and in templates. To see a list of all of the facts that are available about a machine, you can run the setup module as an ad hoc action:

ansible -m setup hostname

This will print out a dictionary of all of the facts that are available for that particular host. You might want to pipe the output to a pager.This does NOT include inventory variables or internal ‘magic’ variables. See the next question if you need more than just ‘facts’.

## How do I see all the inventory variables defined for my host?

By running the following command, you can see inventory variables for a host:

ansible-inventory --list --yaml

## How do I see all the variables specific to my host?

To see all host specific variables, which might include facts and other sources:

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

Unless you are using a fact cache, you normally need to use a play that gathers facts first, for facts included in the task above.

## How do I loop over a list of hosts in a group, inside of a template?

A pretty common pattern is to iterate over a list of hosts inside of a host group, perhaps to populate a template configuration file with a list of servers. To do this, you can just access the “$groups” dictionary in your template, like this:

{% **for** **host** **in** **groups[**'db\_servers'**]** %}

{{ **host** }}

{% **endfor** %}

If you need to access facts about these hosts, for instance, the IP address of each hostname, you need to make sure that the facts have been populated. For example, make sure you have a play that talks to db\_servers:

**-** hosts**:** db\_servers

tasks**:**

**-** debug**:** msg="doesn't matter what you do, just that they were talked to previously."

Then you can use the facts inside your template, like this:

{% **for** **host** **in** **groups[**'db\_servers'**]** %}

{{ **hostvars[host][**'ansible\_eth0'**][**'ipv4'**][**'address'**]** }}

{% **endfor** %}

## 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 using “~”, 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.

In the example above, if your interface names have dashes, you must replace them with underscores:

{{ **hostvars[inventory\_hostname][**'ansible\_' **~** **which\_interface** **|** **replace(**'\_'**,** '-'**)** **][**'ipv4'**][**'address'**]** }}

Also see [dynamic\_variables](https://docs.ansible.com/ansible/latest/reference_appendices/faq.html#dynamic-variables).

## How do I access a group variable?

Technically, you don’t, Ansible does not really use groups directly. Groups are labels for host selection and a way to bulk assign variables, they are not a first class entity, Ansible only cares about Hosts and Tasks.

That said, you could just access the variable by selecting a host that is part of that group, see [first\_host\_in\_a\_group](https://docs.ansible.com/ansible/latest/reference_appendices/faq.html#first-host-in-a-group) below for an example.

## How do I access a variable of the first host in a group?

What happens if we want the ip address of the first webserver in the webservers group? Well, we can do that too. Note that if we are using dynamic inventory, which host is the ‘first’ may not be consistent, so you wouldn’t want to do this unless your inventory is static and predictable. (If you are using [Red Hat Ansible Tower](https://docs.ansible.com/ansible/latest/reference_appendices/tower.html#ansible-tower), it will use database order, so this isn’t a problem even if you are using cloud based inventory scripts).

Anyway, here’s the trick:

{{ **hostvars[groups[**'webservers'**][**0**]][**'ansible\_eth0'**][**'ipv4'**][**'address'**]** }}

Notice how we’re pulling out the hostname of the first machine of the webservers group. If you are doing this in a template, you could use the Jinja2 ‘#set’ directive to simplify this, or in a playbook, you could also use set\_fact:

**-** set\_fact**:** headnode={{ **groups[**'webservers'**][**0**]** }}

**-** debug**:** msg={{ **hostvars[headnode].ansible\_eth0.ipv4.address** }}

Notice how we interchanged the bracket syntax for dots – that can be done anywhere.

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

The copy module has a recursive parameter. However, take a look at the synchronize module if you want to do something more efficient for a large number of files. The synchronize module wraps rsync. See the module index for info on both of these modules.

## How do I access shell environment variables?

**On controller machine :** Access existing variables from controller use the env lookup plugin. For example, to access the value of the HOME environment variable on the management machine:

**---**

*# ...*

vars**:**

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

**On target machines :** Environment variables are available via facts in the ansible\_env variable:

{{ **ansible\_env.HOME** }}

If you need to set environment variables for TASK execution, see [Setting the remote environment](https://docs.ansible.com/ansible/latest/user_guide/playbooks_environment.html#playbooks-environment) in the [Advanced Playbooks](https://docs.ansible.com/ansible/latest/user_guide/playbooks_special_topics.html#playbooks-special-topics) section. There are several ways to set environment variables on your target machines. You can use the [template](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/template_module.html#template-module), [replace](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/replace_module.html#replace-module), or [lineinfile](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/lineinfile_module.html#lineinfile-module) modules to introduce environment variables into files. The exact files to edit vary depending on your OS and distribution and local configuration.

## How do I generate encrypted passwords for the user module?

Ansible ad hoc command is the easiest option:

ansible all -i localhost, -m debug -a "msg={{ 'mypassword' | password\_hash('sha512', 'mysecretsalt') }}"

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

mkpasswd --method=sha-512

If this utility is not installed on your system (for example, you are using macOS) then you can still easily generate these passwords using Python. First, ensure that the [Passlib](https://foss.heptapod.net/python-libs/passlib/-/wikis/home) 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.using(rounds=5000).hash(getpass.getpass()))"

Use the integrated [Encrypting and checksumming strings and passwords](https://docs.ansible.com/ansible/latest/user_guide/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 [Using encrypted variables and files](https://docs.ansible.com/ansible/latest/user_guide/vault.html#playbooks-vault) to encrypt sensitive data.

In OpenBSD, a similar option is available in the base system called encrypt (1)

## Ansible allows dot notation and array notation for variables. Which notation should I use?

The dot notation comes from Jinja and works fine for variables without special characters. If your variable contains dots (.), colons (:), or dashes (-), if a key begins and ends with two underscores, or if a key uses any of the known public attributes, it is safer to use the array notation. See [Using Variables](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#playbooks-variables) for a list of the known public attributes.

item[0]['checksum:md5']

item['section']['2.1']

item['region']['Mid-Atlantic']

It is {{ **temperature[**'Celsius'**][**'-3'**]** }} outside.

Also array notation allows for dynamic variable composition, see [dynamic\_variables](https://docs.ansible.com/ansible/latest/reference_appendices/faq.html#dynamic-variables).

Another problem with ‘dot notation’ is that some keys can cause problems because they collide with attributes and methods of python dictionaries.

item.update # this breaks if item is a dictionary, as 'update()' is a python method for dictionaries

item['update'] # this works

## When is it unsafe to bulk-set task arguments from a variable?

You can set all of a task’s arguments from a dictionary-typed variable. This technique can be useful in some dynamic execution scenarios. However, it introduces a security risk. We do not recommend it, so Ansible issues a warning when you do something like this:

*#...*

vars**:**

usermod\_args**:**

name**:** testuser

state**:** present

update\_password**:** always

tasks**:**

**-** user**:** '{{ **usermod\_args** }}'

This particular example is safe. However, constructing tasks like this is risky because the parameters and values passed to usermod\_args could be overwritten by malicious values in the host facts on a compromised target machine. To mitigate this risk:

* set bulk variables at a level of precedence greater than host facts in the order of precedence found in [Variable precedence: Where should I put a variable?](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#ansible-variable-precedence) (the example above is safe because play vars take precedence over facts)
* disable the [INJECT\_FACTS\_AS\_VARS](https://docs.ansible.com/ansible/latest/reference_appendices/config.html#inject-facts-as-vars) configuration setting to prevent fact values from colliding with variables (this will also disable the original warning)

## Can I get training on Ansible?

Yes! See our [services page](https://www.ansible.com/products/consulting) for information on our services and training offerings. Email [info@ansible.com](mailto:info%40ansible.com) for further details.

We also offer free web-based training classes on a regular basis. See our [webinar page](https://www.ansible.com/resources/webinars-training) for more info on upcoming webinars.

## Is there a web interface / REST API / GUI?

Yes! Ansible, Inc makes a great product that makes Ansible even more powerful and easy to use. See [Red Hat Ansible Tower](https://docs.ansible.com/ansible/latest/reference_appendices/tower.html#ansible-tower).

## How do I keep secret data in my playbook?

If you would like to keep secret data in your Ansible content and still share it publicly or keep things in source control, see [Using encrypted variables and files](https://docs.ansible.com/ansible/latest/user_guide/vault.html#playbooks-vault).

If you have a task that you don’t want to show the results or command given to it when using -v (verbose) mode, the following task or playbook attribute can be useful:

**-** name**:** secret task

shell**:** /usr/bin/do\_something --value={{ **secret\_value** }}

no\_log**:** True

This can be used to keep verbose output but hide sensitive information from others who would otherwise like to be able to see the output.

The no\_log attribute can also apply to an entire play:

**-** hosts**:** all

no\_log**:** True

Though this will make the play somewhat difficult to debug. It’s recommended that this be applied to single tasks only, once a playbook is completed. Note that the use of the no\_log attribute does not prevent data from being shown when debugging Ansible itself via the [**ANSIBLE\_DEBUG**](https://docs.ansible.com/ansible/latest/reference_appendices/config.html#envvar-ANSIBLE_DEBUG) environment variable.

## 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 loop or 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 as you expect, if you need to use a dynamic variable use the following as appropriate:

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

For ‘non host vars’ you can use the [vars lookup](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/vars_lookup.html#vars-lookup) plugin:

{{ **lookup(**'vars'**,** 'somevar\_' **~** **other\_var)** }}

## Why don’t you ship ansible in wheel format (or other packaging format) ?

In most cases it has to do with maintainability. There are many ways to ship software and we do not have the resources to release Ansible on every platform. In some cases there are technical issues. For example, our dependencies are not present on Python Wheels.

## How do I get the original ansible\_host when I delegate a task?

As the documentation states, connection variables are taken from the delegate\_to host so ansible\_host is overwritten, but you can still access the original via hostvars:

original\_host**:** "{{ **hostvars[inventory\_hostname][**'ansible\_host'**]** }}"

This works for all overridden connection variables, like ansible\_user, ansible\_port, and so on.

## How do I fix ‘protocol error: filename does not match request’ when fetching a file?

Since release 7.9p1 of OpenSSH there is a [bug](https://bugzilla.mindrot.org/show_bug.cgi?id=2966) in the SCP client that can trigger this error on the Ansible controller when using SCP as the file transfer mechanism:

failed to transfer file to /tmp/ansible/file.txt\r\nprotocol error**:** filename does not match request

In these releases, SCP tries to validate that the path of the file to fetch matches the requested path. The validation fails if the remote filename requires quotes to escape spaces or non-ascii characters in its path. To avoid this error:

* **Use SFTP instead of SCP by setting scp\_if\_ssh to smart (which tries SFTP first) or to False. You can do this in one of four ways:**
  + Rely on the default setting, which is **smart** - this works if **scp\_if\_ssh** is not explicitly set anywhere
  + Set a [host variable](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#host-variables) or [group variable](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#group-variables) in inventory: **ansible\_scp\_if\_ssh: False**
  + Set an environment variable on your control node: **export ANSIBLE\_SCP\_IF\_SSH=False**
  + Pass an environment variable when you run Ansible: **ANSIBLE\_SCP\_IF\_SSH=smart ansible-playbook**
  + Modify your **ansible.cfg** file: add **scp\_if\_ssh=False** to the **[ssh\_connection]** section
* **If you must use SCP, set the -T arg to tell the SCP client to ignore path validation. You can do this in one of three ways:**
  + Set a [host variable](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#host-variables) or [group variable](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#group-variables): **ansible\_scp\_extra\_args=-T**,
  + Export or pass an environment variable: **ANSIBLE\_SCP\_EXTRA\_ARGS=-T**
  + Modify your **ansible.cfg** file: add **scp\_extra\_args=-T** to the **[ssh\_connection]** section

**Note**

If you see an invalid argument error when using -T, then your SCP client is not performing filename validation and will not trigger this error.

## Does Ansible support multiple factor authentication 2FA/MFA/biometrics/finterprint/usbkey/OTP/…

No, Ansible is designed to execute multiple tasks against multiple targets, minimizing user interaction. As with most automation tools, it is not compatible with interactive security systems designed to handle human interaction. Most of these systems require a secondary prompt per target, which prevents scaling to thousands of targets. They also tend to have very short expiration periods so it requires frequent reauthorization, also an issue with many hosts and/or a long set of tasks.

In such environments we recommend securing around Ansible’s execution but still allowing it to use an ‘automation user’ that does not require such measures. This is something that Tower/AWX excels at by allowing administrators to set up RBAC access to inventory, along with managing credentials and job execution.

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