# What Is Ansible? – Configuration Management And Automation With Ansible

## ****What Is Ansible?****

This is the first blog of my Ansible tutorial series on “What Is Ansible”. I hope you will enjoy reading it.

Ansible is an open source IT Configuration Management, Deployment & Orchestration tool. It aims to provide large productivity gains to a wide variety of automation challenges. This tool is very simple to use yet powerful enough to automate complex multi-tier IT application environments.

On this blog you will be learning:

* What is Ansible?
* [Why do we need Ansible](https://www.edureka.co/blog/what-is-ansible/#why_do_we_need_ansible)?
* [Advantages of using Ansible](https://www.edureka.co/blog/what-is-ansible/#advantages_of_using_ansible)
* [What Ansible can do](https://www.edureka.co/blog/what-is-ansible/#what_ansible_can_do)?
* [Ansible Architecture](https://www.edureka.co/blog/what-is-ansible/#ansible_architecture)
* [Ansible in DevOps](https://www.edureka.co/blog/what-is-ansible/#ansible_in_devops)
* [Real-Life usage of Ansible by NASA](https://www.edureka.co/blog/what-is-ansible/#nasa_case_study)
* Some [Ansible terms](https://www.edureka.co/blog/what-is-ansible/#ansible_terms), to help you understand Ansible better.

You can go through this What is Ansible video lecture where our [**Ansible Training**](https://www.edureka.co/devops) expert is discussing each & every nuance of the technology.

## ****What is Ansible | DevOps Training | Edureka****



## ****Why Do We Need Ansible?****

Well before I tell you what is Ansible, it is of utmost importance to understand the problems that were faced before Ansible.

Let us take a little flashback to the beginning of networked computing when deploying and managing servers reliably and efficiently has been a challenge. Previously, system administrators managed servers by hand, installing software, changing configurations, and administering services on individual servers.  
As data centers grew, and hosted applications became more complex, administrators realized they couldn’t scale their manual systems management as fast as the applications they were enabling. It also hampered the velocity of the work of the developers since the development team was agile and releasing software frequently, but IT operations were spending more time configuring the systems. That’s why server provisioning and configuration management tools came to flourish.

Consider the tedious routine of administering a server fleet. We always need to keep updating, pushing changes, copying files on them etc. These tasks make things very complicated and time consuming.

But let me tell you that there is a solution to the above stated problem. The solution is –**Ansible.**

But before I go ahead to explain you all about Ansible, let me get you familiarized with few Ansible terminologies:

## ****Ansible Terms:****

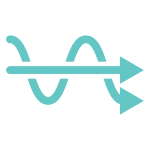
* **Controller Machine**: The machine where Ansible is installed, responsible for running the provisioning on the servers you are managing.
* **Inventory**: An initialization file that contains information about the servers you are managing.
* **Playbook**: The entry point for Ansible provisioning, where the automation is defined through tasks using YAML format.
* **Task**: A block that defines a single procedure to be executed, e.g. Install a package.
* **Module**: A module typically abstracts a system task, like dealing with packages or creating and changing files. Ansible has a multitude of built-in modules, but you can also create custom ones.
* **Role**: A pre-defined way for organizing playbooks and other files in order to facilitate sharing and reusing portions of a provisioning.
* **Play**: A provisioning executed from start to finish is called a play. In simple words, execution of a playbook is called a play.
* **Facts**: Global variables containing information about the system, like network interfaces or operating system.
* **Handlers**: Used to trigger service status changes, like restarting or stopping a service.

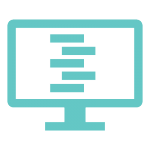
Ansible is a helpful tool that allows you to create groups of machines, describe how these machines should be configured or what actions should be taken on them. Ansible issues all commands from a central location to perform these tasks.

No other client software is installed on the node machines. It uses SSH to connect to the nodes. Ansible only needs to be installed on the control machine (the machine from which you will be running commands) which can even be your laptop. It is a simple solution to a complicated problem.

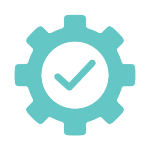
I am not boasting off when I say that Ansible has filled up all the holes in Configuration Management and IT Orchestration world. You will know it too, when you take a look at the benefits of Ansible mentioned below:

## ****Advantages Of Using Ansible****

  
**Simple:**Ansible uses a simple syntax written in YAML called**playbooks**. YAML is a human-readable data serialization language. It is extraordinarily simple. So, no special coding skills are required and even people in your IT organization, who do not know what is Ansible can likely read a playbook and understand what is happening. Ansible always executes tasks in order. It is simple to install too (Don’t believe me? Check out my [***Ansible Installation***](https://www.edureka.co/blog/install-ansible/) blog). Altogether the simplicity ensures that you can get started quickly.

  
**Agentless:** Finally, Ansible is completely agentless. There are no agents/software or additional firewall ports that you need to install on the  client systems or hosts which you want to automate. You do not have to separately set up a management infrastructure which includes managing your entire systems, network and storage. Ansible further reduces the effort required for your team to start automating right away.

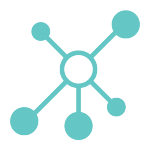
**Powerful & Flexible:** Ansible has powerful features that can enable you to model even the most complex IT workflows. In this aspect, Ansible’s batteries included approach (This philosophy means that something is self-sufficient, comes out-of-the-box ready to use, with everything that is needed) can manage the infrastructure, networks, operating systems and services that you are already using, as Ansible provides you with hundreds of modules to manage them. Together Ansible’s capabilities allow you to orchestrate the entire application environment regardless of where it is deployed.

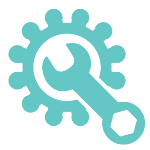
****

**Efficient:** No extra software on your servers means more resources for your applications. Also, since Ansible modules work via JSON, Ansible is extensible with modules written in a programming language you already know. Ansible introduces modules as basic building blocks for your software. So, you can even customize it as per your needs. For e.g. If you have an existing message sending module which sends messages in plain-text, and you want to send images too, you can add image sending features on top of it.

## ****What Ansible Can Do?****

Ansible is usually grouped along with other Configuration Management tools like Puppet, Chef, SaltStack etc. Well, let me tell you, Ansible is not just limited to Configuration Management. It can be used in many different ways too. I have mentioned some of them below:

  
**Provisioning:** Your apps have to live somewhere. If you’re PXE (Preboot eXecution Environment) booting and kick starting bare-metal servers or Virtual Machines, or creating virtual or cloud instances from templates, Ansible & Ansible Tower helps to streamline this process. For example, if I want to test the debug version of an application that is built with Visual C++, I ought to meet some prerequisite requirements like having Visual C++ library DLLs (msvcr100d.dll). I will also need Visual Studio installed in your computer. This is when Ansible makes sure that the required packages are downloaded and installed in order to provision my application.

  
**Configuration Management:** It establishes and maintains consistency of the product performance by recording and updating detailed information which describes an enterprise’s hardware and software. Such information typically includes the versions and updates that have been applied to installed software packages and the locations and network addresses of hardware devices. For e.g. If you want to install the new version of Tomcat on all of the machines present in your enterprise, it is not feasible for you to manually go and update each and every machine. You can install Tomcat in one go on all of your machines with Ansible playbooks and inventory written in the most simple way. All you have to  do is list out the IP addresses of your nodes in the inventory and write a playbook to install Tomcat. Run the playbook from your control machine & it will be installed on all your nodes.

[[](https://www.edureka.co/devops-certification-training)](https://www.edureka.co/devops-certification-training" \t "_blank)

### [DevOps Certification Training](https://www.edureka.co/devops-certification-training" \t "_blank)

* *[Instructor-led Sessions](https://www.edureka.co/devops-certification-training" \t "_blank)*
* *[Real-life Case Studies](https://www.edureka.co/devops-certification-training" \t "_blank)*
* *[Assignments](https://www.edureka.co/devops-certification-training" \t "_blank)*
* *[Lifetime Access](https://www.edureka.co/devops-certification-training" \t "_blank)*

[Explore Curriculum](https://www.edureka.co/devops-certification-training" \t "_blank)

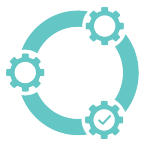
  
**Application Deployment:** When you define your application with Ansible, and manage the deployment with Ansible Tower, teams are able to effectively manage the entire application life cycle from development to production. For example, let’s say I want to deploy the Default Servlet Engine. There are a number of steps that needs to be undergone to deploy the engine.

* Move a .war application from dropins directory to apps directory
* Add server.xml file
* Navigate to the webpage to see your application.

But why worry about performing these steps one by one when we have a tool like Ansible. All you need to do is list these tasks in your Ansible playbook and sit back watching Ansible executing these tasks in order.

****

**Security and Compliance:** When you define your security policy in Ansible, scanning and remediation of site-wide security policy can be integrated into other automated processes. And it’ll be integral in everything that is deployed. It means that, you need to configure your security details once in your control machine and it will be embedded in all other nodes automatically. Moreover, all the credentials (admin users id’s & passwords) that are stored within Ansible are not retrievable in plain-text by any user.

  
**Orchestration:** Configurations alone don’t define your environment. You need to define how multiple configurations interact and ensure the disparate pieces can be managed as a whole. Out of complexity and chaos, Ansible brings order. Ansible provides Orchestration in the sense of aligning the business request with the applications, data, and infrastructure. It defines the policies and service levels through automated workflows, provisioning, and change management. This creates an application-aligned infrastructure that can be scaled up or down based on the needs of each application.

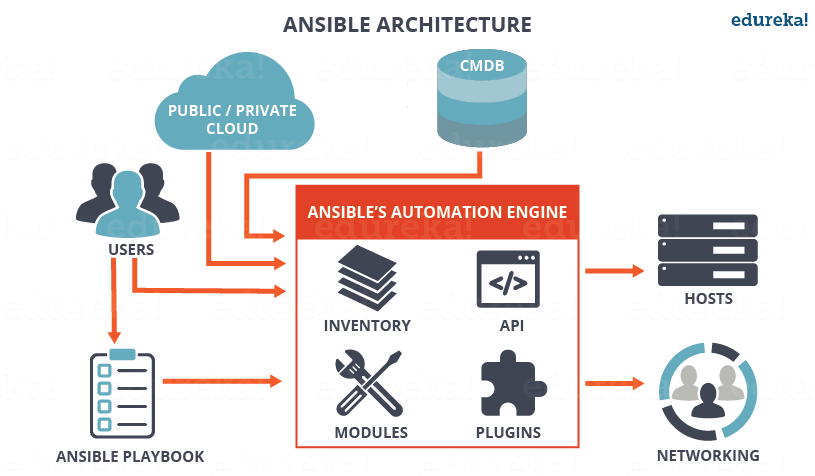
For example, Consider the situation where I want to deploy a new website in place of my existing one. For that, we will remove the existing website, and deploy our new website, and restart  the load balancer or the web cluster if needed. Now, if we just did something like this, users would notice downtime because we have not removed live traffic going to these machines via the load balancer. So, we need some type of pre-task, where we tell the load balancer to put this web server into maintenance mode, so that we can temporarily disable traffic from going to it, as it gets upgraded. Let’s say, I added a block up here, that says a pre-task will be to disable web node in the load balancer.

So, this is our pre-task, where we disable traffic, then down here, we upgrade the node using these various tasks. Finally, we need some type of post-task, which will enable traffic to this web node again, by taking it out of maintenance mode. These tasks can be written in Ansible playbooks and hence it helps to orchestrate the environment.

You will understand the working of Ansible better when you get a clear picture of its (Ansible’s) architecture.

## ****What is Ansible & its Architecture?****

Ansible architecture is fairly straightforward. Refer to the diagram below to understand the Ansible architecture:



As you can see, in the diagram above, the Ansible automation engine has a direct interaction with the users who write playbooks to execute the Ansible Automation engine. It also interacts with cloud services and Configuration Management Database (CMDB).

The Ansible Automation engine consists of:

* **Inventories:** Ansible inventories are lists of hosts (nodes) along with their IP addresses, servers, databases etc. which needs to be managed. Ansible then takes action via a transport – SSH for UNIX, Linux or Networking devices and WinRM for Windows system.
* **APIs:** APIs in Ansible are used as transport for Cloud services, public or private.
* **Modules:** Modules are executed directly on remote hosts through playbooks. The modules can control system resources, like services, packages, or files (anything really), or execute system commands. Modules do it by acting on system files, installing packages or making API calls to the service network. There are over 450 Ansible-provided modules that automate nearly every part of your environment. For e.g.
  + Cloud Modules like cloudformation which creates or deletes an AWS cloud formation stack;
  + Database modules like mssql\_db which removes MYSQL databases from remote hosts.
* **Plugins:** Plugins allows to execute Ansible tasks as a job build step. Plugins are pieces of code that augment Ansible’s core functionality. Ansible ships with a number of handy plugins, and you can easily write your own. For example,
  + Action plugins are front ends to modules and can execute tasks on the controller before calling the modules themselves.
  + Cache plugins are used to keep a cache of ‘facts’ to avoid costly fact-gathering operations.
  + Callback plugins enable you to hook into Ansible events for display or logging purposes.

There are a few more components in Ansible Architecture which are explained below:

**Networking**: Ansible can also be used to automate different networks. Ansible uses the same simple, powerful, and the agentless automation framework IT operations and development are already using. It uses a data model (a playbook or role) that is separate from the Ansible automation engine that easily spans different network hardware.

**Hosts**: The hosts in the Ansible architecture are just node systems which are getting automated by Ansible. It can be any kind of machine – Windows, Linux, RedHat etc.

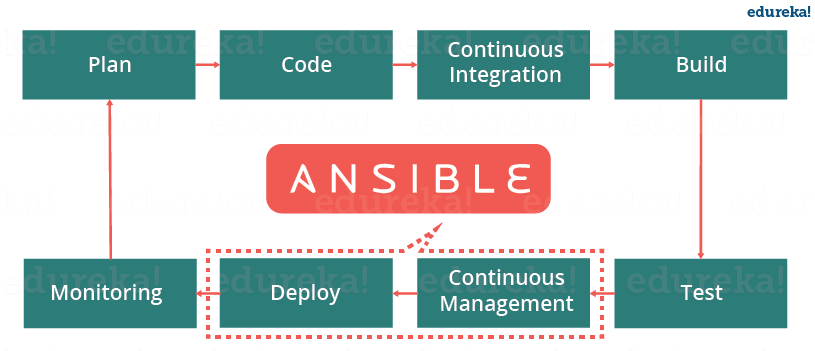
**Playbooks:** Playbooks are simple files written in YAML format which describes the tasks to be executed by Ansible. Playbooks can declare configurations, but they can also orchestrate the steps of any manual ordered process, even if it contains jump statements. They can launch tasks synchronously or asynchronously.

**CMDB** : It is a repository that acts as a data warehouse for IT installations. It holds data relating to a collection of IT assets (commonly referred to as configuration items (CI)), as well as to describe relationships between such assets.

**Cloud:** It is a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server. You can launch your resources and instances on cloud and connect to your servers.

## ****What is Ansible in DevOps?****

In DevOps, as we know development and operations work is integrated. This integration is very important for modern test-driven application design. Hence, Ansible integrates this by providing a stable environment to both development and operations resulting in smooth orchestration. Refer to the image below to see how Ansible fits into DevOps:



Let us discuss now how Ansible manages the entire DevOps infrastructure. When developers begin to think of infrastructure as part of their application i.e as Infrastructure as code (**IaC**), stability and performance become normative. Infrastructure as Code is the process of managing and provisioning computing infrastructure (processes, bare-metal servers, virtual servers, etc.) and their configuration through machine-processable definition files, rather than physical hardware configuration or the use of interactive configuration tools. This is where Ansible automation plays a major role and stands out among its peers.

In DevOps, Sysadmins work tightly with developers, development velocity is improved, and more time is spent doing activities like performance tuning, experimenting, and getting things done, and less time is spent fixing problems. Refer to the diagram below to understand how the tasks of sysadmins and other users are simplified by Ansible.

### DevOps Training

### [DEVOPS CERTIFICATION TRAINING](https://www.edureka.co/devops-certification-training" \o "DevOps Certification Training" \t "_blank)

### [DevOps Certification Training](https://www.edureka.co/devops-certification-training" \o "DevOps Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/devops-certification-training" \o "DevOps Certification Training" \t "_blank)*

**[5](https://www.edureka.co/devops-certification-training" \o "DevOps Certification Training" \t "_blank)**[(87450)](https://www.edureka.co/devops-certification-training" \o "DevOps Certification Training" \t "_blank)

### [AWS CERTIFIED DEVOPS ENGINEER TRAINING](https://www.edureka.co/aws-certified-devops-training" \o "AWS Certified DevOps Engineer Training " \t "_blank)

### [AWS Certified DevOps Engineer Training](https://www.edureka.co/aws-certified-devops-training" \o "AWS Certified DevOps Engineer Training " \t "_blank)

*[Reviews](https://www.edureka.co/aws-certified-devops-training" \o "AWS Certified DevOps Engineer Training " \t "_blank)*

**[5](https://www.edureka.co/aws-certified-devops-training" \o "AWS Certified DevOps Engineer Training " \t "_blank)**[(3627)](https://www.edureka.co/aws-certified-devops-training" \o "AWS Certified DevOps Engineer Training " \t "_blank)

### [KUBERNETES CERTIFICATION TRAINING COURSE](https://www.edureka.co/kubernetes-certification" \o "Kubernetes Certification Training Course" \t "_blank)

### [Kubernetes Certification Training Course](https://www.edureka.co/kubernetes-certification" \o "Kubernetes Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/kubernetes-certification" \o "Kubernetes Certification Training Course" \t "_blank)*

**[5](https://www.edureka.co/kubernetes-certification" \o "Kubernetes Certification Training Course" \t "_blank)**[(6361)](https://www.edureka.co/kubernetes-certification" \o "Kubernetes Certification Training Course" \t "_blank)

### [DOCKER CERTIFIED ASSOCIATE TRAINING](https://www.edureka.co/docker-training" \o "Docker Certified Associate Training" \t "_blank)

### [Docker Certified Associate Training](https://www.edureka.co/docker-training" \o "Docker Certified Associate Training" \t "_blank)

*[Reviews](https://www.edureka.co/docker-training" \o "Docker Certified Associate Training" \t "_blank)*

**[5](https://www.edureka.co/docker-training" \o "Docker Certified Associate Training" \t "_blank)**[(5458)](https://www.edureka.co/docker-training" \o "Docker Certified Associate Training" \t "_blank)

### [GIT AND GITHUB](https://www.edureka.co/git-github-sp" \o "Git and GitHub" \t "_blank)

### [Git and GitHub](https://www.edureka.co/git-github-sp" \o "Git and GitHub" \t "_blank)

*[Reviews](https://www.edureka.co/git-github-sp" \o "Git and GitHub" \t "_blank)*

**[5](https://www.edureka.co/git-github-sp" \o "Git and GitHub" \t "_blank)**[(3480)](https://www.edureka.co/git-github-sp" \o "Git and GitHub" \t "_blank)

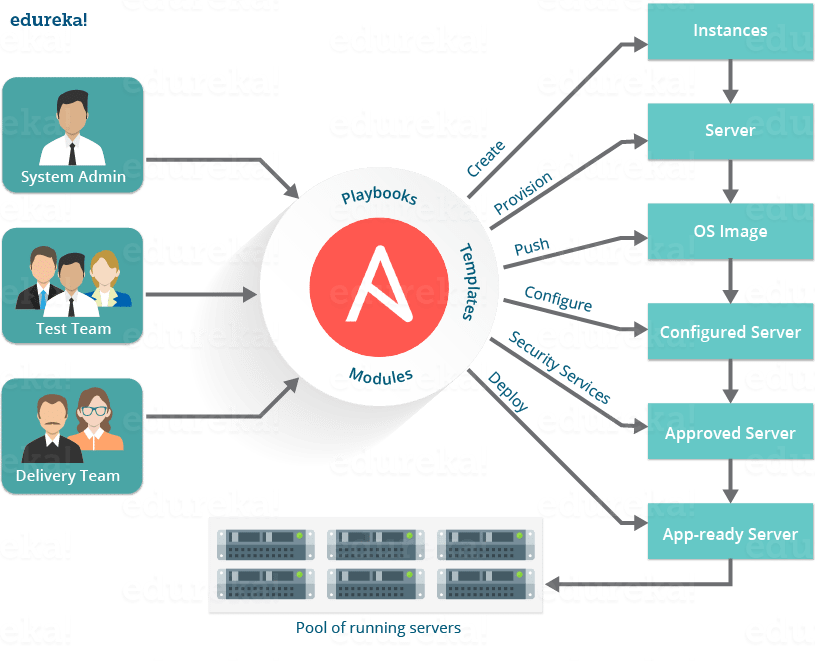
### [CONTINUOUS INTEGRATION WITH JENKINS CERTIFICATION TRAINING](https://www.edureka.co/jenkins" \o "Continuous Integration with Jenkins Certification Training" \t "_blank)

### [Continuous Integration with Jenkins Certification Training](https://www.edureka.co/jenkins" \o "Continuous Integration with Jenkins Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/jenkins" \o "Continuous Integration with Jenkins Certification Training" \t "_blank)*

**[5](https://www.edureka.co/jenkins" \o "Continuous Integration with Jenkins Certification Training" \t "_blank)**[(7872)](https://www.edureka.co/jenkins" \o "Continuous Integration with Jenkins Certification Training" \t "_blank)

Next



At this point you know how beneficial using Ansible is. So, now let us see a real life example of how NASA has benefited through Ansible.

## ****Ansible Case Study – A Real Life Usage by NASA****NASA - What Is Ansible - Edureka

Let us consider the business challenge that was faced by NASA.

NASA needed to move 65 applications from a traditional hardware based data center to a cloud-based environment for better agility and cost savings. The rapid timeline resulted in many applications being migrated ‘as it is’ to a cloud environment. This created an environment which spanned multiple virtual private clouds (VPCs) and AWS accounts that could not be managed easily. Even simple things, like ensuring every system administrator had access to every server, or simple security patching, were extremely cumbersome.

The solution was to leverage Ansible Tower to manage and schedule the cloud environment.

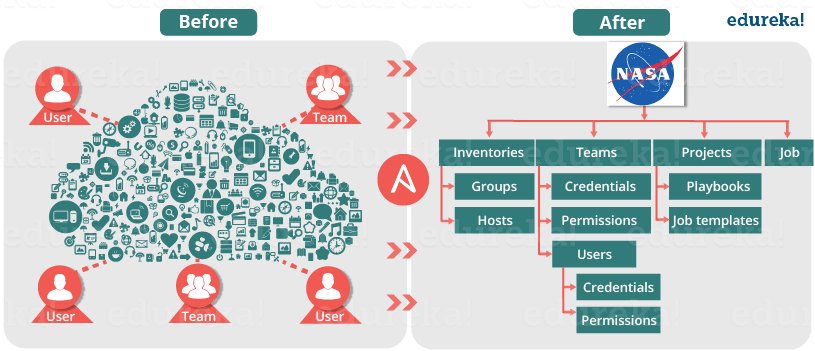
Hence, to solve the problems that NASA had with lack of centralized management and a diverse environment, they evaluated multiple solutions and decided on an implementation of Ansible Tower. NASA is now leveraging Ansible Tower to manage their environment in a very organized and scheduled way.

**How NASA is using Ansible:**

Ansible Tower provided with a dashboard which provided the status summary of all hosts and jobs which allowed NASA to group all contents and manage access permissions across different departments. It also helped to split up the organization by associating content and control permission for groups as well.

Ansible Tower is a web-based interface for managing Ansible. One of the top items in Ansible users’ wishlists was an easy-to-use UI for managing quick deployments and monitoring one’s configurations. Ansible management came up with Ansible Tower in response.

Further, Ansible divided the tasks among teams by assigning various roles. It managed the clean up of old job history, activity streams, data marked for deletion and system tracking info. Refer to the diagram below to understand how Ansible has simplified the work of NASA.



As a result, NASA has achieved the following efficiencies:

• NASA web app servers are being patched routinely and automatically through Ansible Tower with a very simple 10-line Ansible playbook.

• Ansible is also being used to re-mediate security issues and was leveraged to re-mediate OpenSSL issues. This not only saved time but allowed to quickly re-mediate a very daunting security issue.

• Every single week, both the full and mobile versions of www.nasa.gov are updated via Ansible, generally only taking about 5 minutes to do.

• OS level user accounts for mission critical staff are continually checked and created if missing. Now, everyone who needs access has access, even if that means adding or removing a user almost instantly from all servers.

• NASA has also integrated Ansible facts into their CMDB, CloudAware, for better management visibility of entire AWS inventory. As a result, it became possible to organize the inventory of AWS resources in a very granular way that was not possible before.

• Ansible is also used to ensure that the environment is compliant with necessary Federal security standards as outlined by FedRAMP and other regulatory requirements.

**Results:**

As a result of implementing Ansible, NASA is better equipped to manage its AWS environment. Ansible allowed NASA to provide better operations and security to its clients. It has also increased efficiency as a team.

If we see by the numbers:

• Updating**nasa.gov** went from over 1 hour to under 5 minutes

• Security Patching updates went from a multi-day process to 45 minutes

[[](https://www.edureka.co/devops-certification-training)](https://www.edureka.co/devops-certification-training" \t "_blank)

### [DevOps Certification Training](https://www.edureka.co/devops-certification-training" \t "_blank)

[Weekday / Weekend BatchesSee Batch Details](https://www.edureka.co/devops-certification-training" \t "_blank)

• Achieving near real-time RAM and disk monitoring (accomplished without agents)

• Provisioning OS Accounts across entire environment in under 10 minutes

• Baselining standard AMIs (Amazon Machine Image) went from 1 hour of manual configuration to becoming an invisible and seamless background process

• Application stacks set up time reduced from 1-2 hours to under 10 minutes per stack.

I hope you have enjoyed this blog and learned what is Ansible. Now let us learn how to use Ansible using Adhoc commands and playbooks in my next blog on [***Ansible Tutorial***](https://www.edureka.co/blog/ansible-tutorial/).

**Summary**: Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows.  ***Source***: [*Wikipedia*](https://en.wikipedia.org/wiki/Ansible_(software)).

# Ansible Tutorial – Learn To Write Ansible Playbooks

 Last updated on May 22,201957.8K Views



[**Reshma Ahmed**](https://www.edureka.co/blog/author/reshma-ahmed/)Reshma is a tech-savvy professional working as a Research Analyst at Edureka. She is a DevOps evangelist, a Cloud enthusiast, a Big Data Hadoop...

* [**5 Comments**](javascript:void(0))
* [**Bookmark**](https://www.edureka.co/blog/ansible-tutorial/)

**3 / 8 Blog from Ansible**

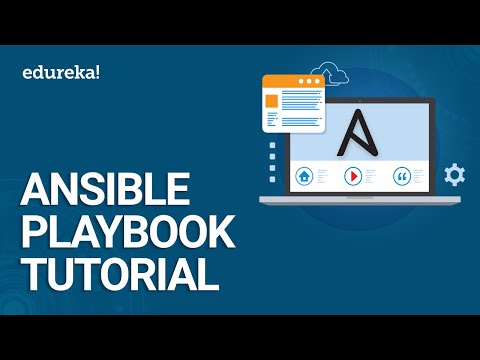
## ****Ansible Tutorial****

I hope you went through my previous blog to learn [***what is Ansible***](https://www.edureka.co/blog/what-is-ansible/) and the most used terminologies of Ansible. In case you haven’t, please do check it out so that you can get a better understanding of this Ansible Tutorial. You should also know that Ansible makes up a crucial part of [***DevOps Certification***](https://www.edureka.co/devops) as a tool for configuration management, deployment and orchestration.

Let me give you an overview of this ‘Ansible Tutorial’:

* You will learn to [write playbooks](https://www.edureka.co/blog/ansible-tutorial/#ansible_playbook)
* You will learn about different [modules in Ansible](https://www.edureka.co/blog/ansible-tutorial/#ansible_modules)
* You will learn to write [Adhoc commands](https://www.edureka.co/blog/ansible-tutorial/" \l "adhoc_commands)
* [Hands on](https://www.edureka.co/blog/ansible-tutorial/#hands_on) with Ansible

## ****Ansible Playbook Tutorial | DevOps Training | Edureka****



## ****Ansible Tutorial – Writing Ansible Playbooks****

Playbooks in Ansible are written in YAML format. It is a human-readable data serialization language. It is commonly used for configuration files. It  can also be used in many applications where data is being stored.

For Ansible, nearly every YAML file starts with a list. Each item in the list is a list of key/value pairs, commonly called a “hash” or a “dictionary”. So, we need to know how to write lists and dictionaries in YAML.

All members of a list are lines beginning at the same indentation level starting with a “- ” (dash and space). More complicated data structures are possible, such as lists of dictionaries or mixed dictionaries whose values are lists or a mix of both.

 e.g. For a list of departments in edureka:

departments:

- marketing

- sales

- solutions

- content writing

- support

- product

Now let me give you an example of a dictionary:

-USA

-continent: North America

-capital: Washington DC

-population: 319 million

## ****Hosts And Users:****

For each play in a playbook, you get to choose which machines in your infrastructure to target and which remote user to complete the tasks. To include hosts in Ansible inventory, we will be using the IP addresses of the host machines.

Generally the hosts are a list one or more groups or host patterns, separated by colons. The remote user is just the name of the user account.

## ****Variables:****

Ansible uses variables which are defined previously to enable more flexibility in playbooks and roles. They can be used to loop through a set of given values, access various information like the host name of a system and replace certain strings in templates with specific values.

Ansible already defines a rich set of variables, individual for each system. Whenever Ansible will run on a system, all facts and information about the system are gathered and set as variables.

But there is a rule for naming variables. Variable names should be letters, numbers, and underscores. Variables should always start with a letter. E.g. wamp\_21, port5 is valid variable names, whereas 01\_port, \_server are invalid.

## ****Tasks:****

Tasks allow you to break up bits of configuration policy into smaller files. Task includes pull from other files. Tasks in Ansible go with pretty much the English meaning of it.

E.g: Install <package\_name>, update <software\_name> etc.

## ****Handlers:****

Handlers are just like regular tasks in an Ansible playbook, but are only run if the Task contains a notify directive and also indicates that it changed something. For example, if a config file is changed, then the task referencing the config file may notify a service restart handler.

Let me give you an example of a playbook which will start the Apache httpd server program:

---

- hosts: webservers

vars:

http\_port: 80

max\_clients: 200

remote\_user: root

tasks:

- name: ensure apache is at the latest version

yum: name=httpd state=latest

- name: write the apache config file

template: src=/srv/httpd.j2 dest=/etc/httpd.conf

notify:

- restart apache

- name: ensure apache is running (and enable it at boot)

service: name=httpd state=started enabled=yes

handlers:

- name: restart apache

service: name=httpd state=restarted

I hope the example will relate you to all the description of the playbook components that I have mentioned above. If it is still not clear to you, don’t worry all your doubts will be clear in the later part of this blog.

This is all about playbooks. The playbooks which will be written by you. But Ansible provides you with a wide range of modules as well, which you can use.

## ****Ansible Tutorial – Modules****

Modules in Ansible are idempotent. From a RESTful service standpoint, for an operation (or service call) to be idempotent, clients can make that same call repeatedly while producing the same result. In other words, making multiple identical requests has the same effect as making a single request.

There are different types of modules in Ansible

* Core modules
* extras modules

## ****Core Modules****

These are modules that the core Ansible team maintains and will always ship with Ansible itself. They will also receive slightly higher priority for all requests than those in the “extras” repos.

[[](https://www.edureka.co/devops-certification-training)](https://www.edureka.co/devops-certification-training" \t "_blank)

### [DevOps Certification Training](https://www.edureka.co/devops-certification-training" \t "_blank)

* *[Instructor-led Sessions](https://www.edureka.co/devops-certification-training" \t "_blank)*
* *[Real-life Case Studies](https://www.edureka.co/devops-certification-training" \t "_blank)*
* *[Assignments](https://www.edureka.co/devops-certification-training" \t "_blank)*
* *[Lifetime Access](https://www.edureka.co/devops-certification-training" \t "_blank)*

[Explore Curriculum](https://www.edureka.co/devops-certification-training" \t "_blank)

The source of these modules is hosted by Ansible on GitHub in the Ansible-modules-core.

## ****Extras Modules****

These modules are currently shipped with Ansible, but might be shipped separately in the future. They are also mostly maintained by the Ansible community. Non-core modules are still fully usable, but may receive slightly lower response rates for issues and pull requests.

Popular “extras” modules may be promoted to core modules over time.

The source for these modules is hosted by Ansible on GitHub in the Ansible-modules-extras.

E.g: The one of the extras module in Remote Management Modules is ipmi\_power module, which is a power manger for the remote machines. It requires python 2.6 or later and pyghmi to run.

You can use this module by writing an adhoc command like the one I have written below:

ipmi\_power : name ="test.domain.com" user="localhost" password="xyz" state="on"

## ****Ansible Tutorial – Return Values****

Ansible modules normally return a data structure that can be registered into a variable, or seen directly when output by the Ansible program. Each module can optionally document its own unique return values.

Some examples of return values are:

* changed: returns with a boolean value whenever the task makes any change.
* failed: returns a boolean value, if the task is failed
* msg: it returns a string with a generic message relayed to the user.

## ****Ansible Tutorial – AdHoc Commands****

Adhoc commands are simple one line command to perform some action. Running modules with Ansible commands are adhoc commands.

E.g:

ansible host -m netscaler -a "nsc\_host=nsc.example.com user=apiuser password=apipass"

The above adhoc command uses the netscaler module to disable the server. There are hundreds of modules available in Ansible from where you can refer to and write adhoc commands.

Well, enough with all the theoretical explanations, let me explain you Ansible with some hands on.

## ****Ansible Tutorial – Hands On****

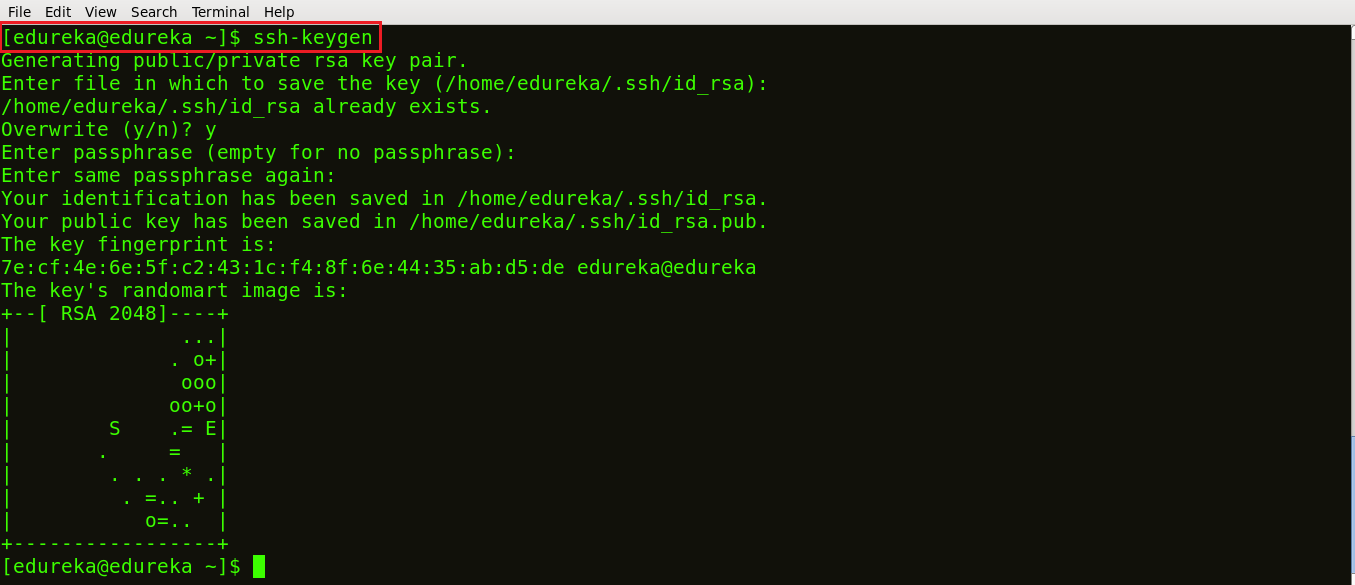
I am going to write a playbook to install Nginx on my node/host machine.

Let’s begin :)

**Step 1:** Connect to your hosts using SSH. For that, you need to generate a public SSH key.

Use the command below:

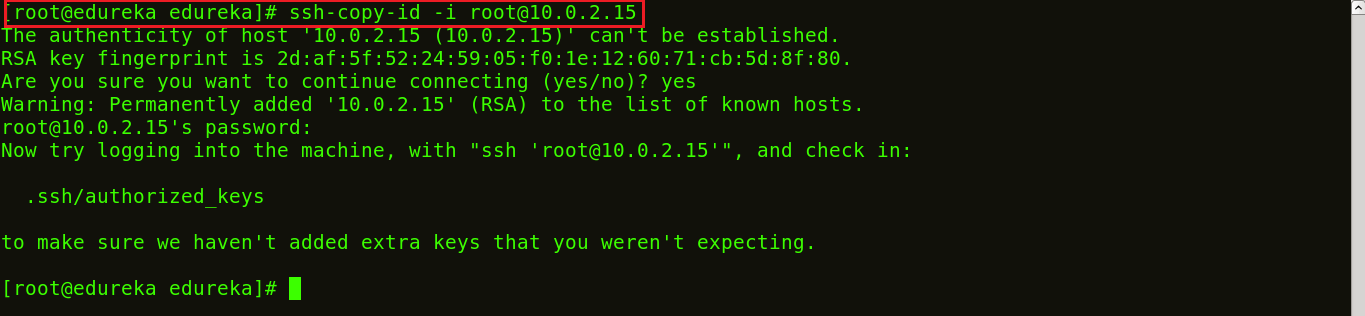
**ssh-keygen**



As you can see in the snapshot above, the command **ssh-keygen** generated a public SSH key.

**Step 2:** Your next task is to copy the public SSH key on your hosts. In order to do that, use the command below:

**ssh-copy-id -i root@<IP address of your host>**



### DevOps Training

### [DEVOPS CERTIFICATION TRAINING](https://www.edureka.co/devops-certification-training" \o "DevOps Certification Training" \t "_blank)

### [DevOps Certification Training](https://www.edureka.co/devops-certification-training" \o "DevOps Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/devops-certification-training" \o "DevOps Certification Training" \t "_blank)*

**[5](https://www.edureka.co/devops-certification-training" \o "DevOps Certification Training" \t "_blank)**[(87450)](https://www.edureka.co/devops-certification-training" \o "DevOps Certification Training" \t "_blank)

### [AWS CERTIFIED DEVOPS ENGINEER TRAINING](https://www.edureka.co/aws-certified-devops-training" \o "AWS Certified DevOps Engineer Training " \t "_blank)

### [AWS Certified DevOps Engineer Training](https://www.edureka.co/aws-certified-devops-training" \o "AWS Certified DevOps Engineer Training " \t "_blank)

*[Reviews](https://www.edureka.co/aws-certified-devops-training" \o "AWS Certified DevOps Engineer Training " \t "_blank)*

**[5](https://www.edureka.co/aws-certified-devops-training" \o "AWS Certified DevOps Engineer Training " \t "_blank)**[(3627)](https://www.edureka.co/aws-certified-devops-training" \o "AWS Certified DevOps Engineer Training " \t "_blank)

### [KUBERNETES CERTIFICATION TRAINING COURSE](https://www.edureka.co/kubernetes-certification" \o "Kubernetes Certification Training Course" \t "_blank)

### [Kubernetes Certification Training Course](https://www.edureka.co/kubernetes-certification" \o "Kubernetes Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/kubernetes-certification" \o "Kubernetes Certification Training Course" \t "_blank)*

**[5](https://www.edureka.co/kubernetes-certification" \o "Kubernetes Certification Training Course" \t "_blank)**[(6361)](https://www.edureka.co/kubernetes-certification" \o "Kubernetes Certification Training Course" \t "_blank)

### [DOCKER CERTIFIED ASSOCIATE TRAINING](https://www.edureka.co/docker-training" \o "Docker Certified Associate Training" \t "_blank)

### [Docker Certified Associate Training](https://www.edureka.co/docker-training" \o "Docker Certified Associate Training" \t "_blank)

*[Reviews](https://www.edureka.co/docker-training" \o "Docker Certified Associate Training" \t "_blank)*

**[5](https://www.edureka.co/docker-training" \o "Docker Certified Associate Training" \t "_blank)**[(5458)](https://www.edureka.co/docker-training" \o "Docker Certified Associate Training" \t "_blank)

### [GIT AND GITHUB](https://www.edureka.co/git-github-sp" \o "Git and GitHub" \t "_blank)

### [Git and GitHub](https://www.edureka.co/git-github-sp" \o "Git and GitHub" \t "_blank)

*[Reviews](https://www.edureka.co/git-github-sp" \o "Git and GitHub" \t "_blank)*

**[5](https://www.edureka.co/git-github-sp" \o "Git and GitHub" \t "_blank)**[(3480)](https://www.edureka.co/git-github-sp" \o "Git and GitHub" \t "_blank)

### [CONTINUOUS INTEGRATION WITH JENKINS CERTIFICATION TRAINING](https://www.edureka.co/jenkins" \o "Continuous Integration with Jenkins Certification Training" \t "_blank)

### [Continuous Integration with Jenkins Certification Training](https://www.edureka.co/jenkins" \o "Continuous Integration with Jenkins Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/jenkins" \o "Continuous Integration with Jenkins Certification Training" \t "_blank)*

**[5](https://www.edureka.co/jenkins" \o "Continuous Integration with Jenkins Certification Training" \t "_blank)**[(7872)](https://www.edureka.co/jenkins" \o "Continuous Integration with Jenkins Certification Training" \t "_blank)

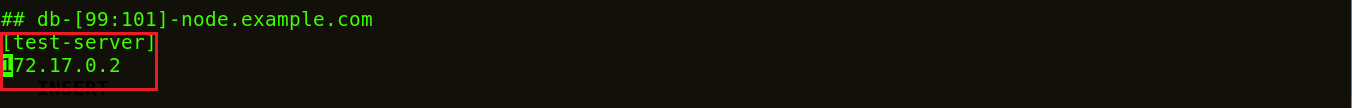
Next

The snapshot above shows the SSH key being copied to the hosts.

**Step 3:** List the IP addresses of your hosts/nodes in your inventory.

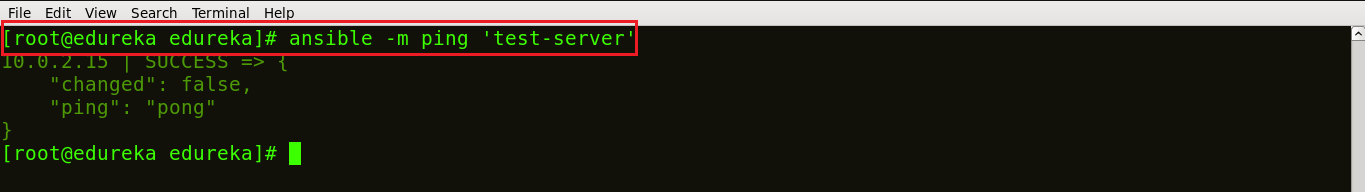
Use the following command:

**vi /etc/ansible/hosts**



This will open a vi editor where you can list down the IP addresses of your hosts. This is now your inventory.

**Step 4:** Let’s ping to ensure a connection has been established.

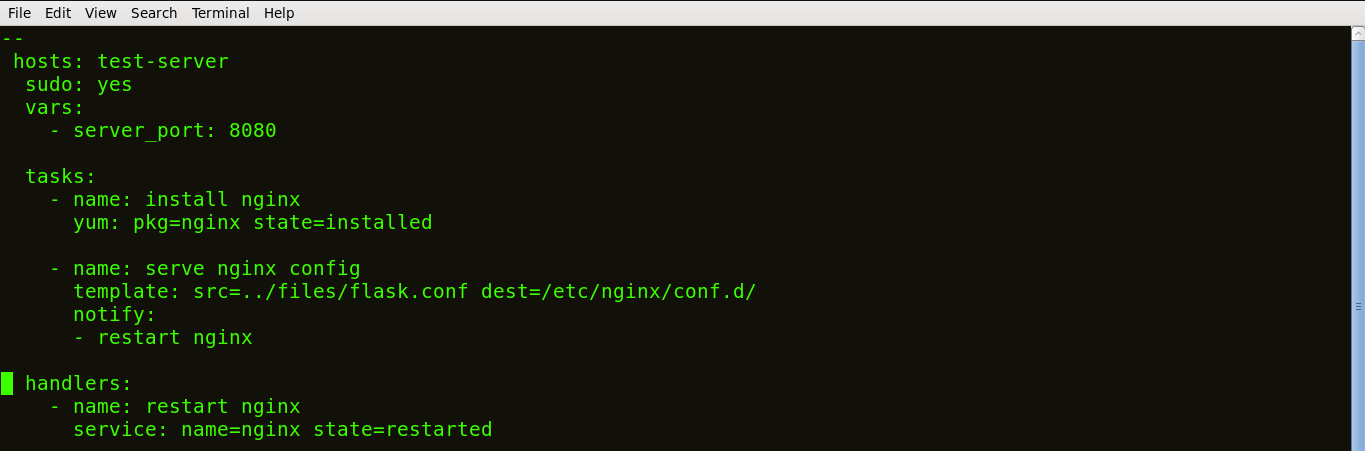


The snapshot above confirms that connection has been made between your control machine and host.

**Step 5:** Let us now write a playbook to install Nginx on the host machine. You can write your playbook in the vi editor. For that, simply create your playbook, using the command:

**vi <name of your file>.yml**

The below snapshot shows my playbook to install Nginx written in YAML format.



The tasks of a playbook are defined in YAML as a list of dictionaries and executed from top to bottom. If we have several hosts, then each task is tried for each host before moving on to the next one. Each task is defined as a dictionary that can have several keys, such as “name” or “sudo” which signify the name of the task and whether it requires sudo privileges.

A variable server\_port is set that listens on TCP port **8080** for incoming requests.

Here, the first task is to get the necessary package for installation of Nginx and then install it. Internally, Ansible will check if the directory exists and create it if it’s not, otherwise it will do nothing.

The next task is to configure Nginx. In Nginx, contexts contain configuration details.

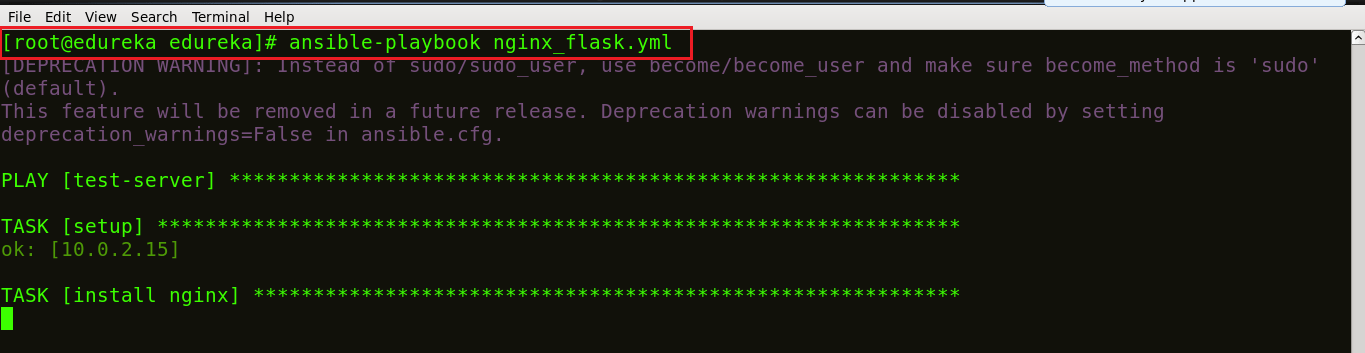
Here, the template is a file you can deploy on hosts. However, template files also include some reference variables which are pulled from variables defined as part of an Ansible playbook or facts gathered from the hosts. Facts containing the configuration details are being pulled from a source directory and being copied to a destination directory.

Handlers here define the action to be performed only upon notification of tasks or state changes. In this playbook, we defined, notify: restart Nginx handler which will restart Nginx once the files and templates are copied to hosts.

Now, save the file and exit.

**Step 6:** Now let’s run this playbook, using the command below:

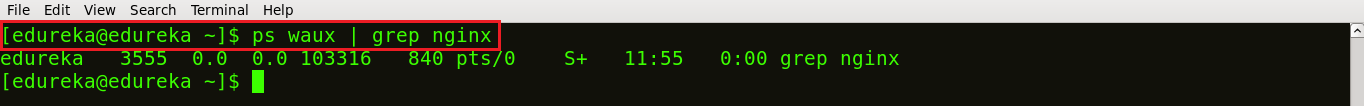
**ansible-playbook <name of your file>.yml**



We can see in the screenshot above that our task is getting executed; Nginx being installed.

**Step 7:** Let’s check if Nginx is installed on my host machine. Use the command below:

**ps waux | grep nginx**



You can see in the screenshot above, that different process ids 3555 and 103316 are running which ensures that Nginx is running on your host machines.

**ANSIBLE PROJECTS & PLAYBOOK COLLECTIONS**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SAMPLE PLAYBOOKS

1)

[root@controller playbooks]# cat httpd.yaml

---

- name: This sets up an httpd webserver

hosts: ansibleclient01.local

tasks:

- name: Install apache packages

yum:

name: httpd

state: present

- name: ensure httpd is running

service:

name: httpd

state: started

- name: Open port 80 for http access

firewalld:

service: http

permanent: true

state: enabled

- name: Restart the firewalld service to load in the firewall changes

service:

name: firewalld

state: restarted

...

2)

Playbook to create a directory

---

- hosts: all

tasks:

- name: ansible create directory example

file:

path: /tmp/devops\_directory

state: directory

output

======

:~$ ls -lrt /tmp

drwxr-xr-x 2 root root 4096 Oct 4 07:19 devops\_directory

3)

Create a directory and change permission

---

- hosts: all

tasks:

- name: ansible create directory with mode setting example

file:

path: /tmp/devops\_directory

state: directory

mode: "u=rw,g=wx,o=rwx"

4)

Create multiple directories using with\_items

- hosts: all

tasks:

- name: ansible create multiple directory example

file:

path: "{{ item }}"

state: directory

with\_items:

- '/tmp/devops\_system1'

- '/tmp/devops\_system2'

- '/tmp/devops\_system3'

output

======

mdtutorials2@system01:~$ ls -lrt /tmp

total 16

drwxr-xr-x 2 root root 4096 Oct 4 09:59 devops\_system1

drwxr-xr-x 2 root root 4096 Oct 4 09:59 devops\_system2

drwxr-xr-x 2 root root 4096 Oct 4 09:59 devops\_system3

5)

But what if you need to set the permission differently for each directory while using with\_items. In the following task, I am independently setting the modes for each directory.

- hosts: all

tasks:

- name: ansible create directory with\_items example

file:

path: "{{ item.dest }}"

mode: "{{item.mode}}"

state: directory

with\_items:

- { dest: '/tmp/devops\_system1', mode: '0777'}

- { dest: '/tmp/devops\_system2', mode: '0707'}

- { dest: '/tmp/devops\_system3', mode: '0575'}

output

======

mdtutorials2@system01:~$ ls -lrt /tmp/

total 16

drwxrwxrwx 2 root root 4096 Oct 4 09:59 devops\_system1

drwx---rwx 2 root root 4096 Oct 4 09:59 devops\_system2

dr-xrwxr-x 2 root root 4096 Oct 4 09:59 devops\_system3

6)

**Creating an empty file in Ansible**

You can Create an empty file using the file module. You just need to set two parameters.

**Path** – This is the location where you want the file to be created. It can be either a relative path or an absolute path. You should also include the name of the file being created here.

**State** – To create a new file this parameter should be set to ‘touch’. It works like the touch operation in Linux.

In the following task, I am creating a new file called ‘devops\_server’ in the location set against the ‘path’ parameter. I am giving the absolute path. Please see the default permissions and owner in the output.

- hosts: all

tasks:

- name: Ansible create file if it doesn't exist example

file:

path: "/Users/mdtutorials2/Documents/Ansible/devops\_server.txt"

state: touch

7)

**Creating a new file with content**

One way is to create an empty file using the above method and then using the **blockinfile** or **lineinfile** module to add content to it.

But an easier method is to use the Ansible copy module to create a new file with content inside.

The copy module is commonly used to copy a source file to a destination file. But there is a parameter ‘content’ which can be used to create a file with the content.

Note: If the file already exists then the module would check for the contents of that file. And if the contents are same then nothing happens. But if the content is different then the file will be overwritten. So do not use this on an existing file unless necessary.

The following task will create a new file ‘remote\_server.txt’ if the file is not present.

- hosts: all

tasks:

- name: Ansible create file with content example

copy:

dest: "/Users/mdtutorials2/Documents/Ansible/remote\_server.txt"

content: |

dog1

tiger

output

------

-rw-r--r-- 1 root wheel 11 Feb 6 08:37 remote\_server.txt

bash-3.2# cat remote\_server.txt

dog1

tiger

8)

**Ansible Playbook to patch Debian and RedHat based servers**

---

# This playbook patches dirty cow

- hosts: all

gather\_facts: true

tasks:

- name: Grouping!

group\_by:

key: "{{ ansible\_os\_family }}"

- hosts: Debian

gather\_facts: false

serial: 1

become: yes

max\_fail\_percentage: 0

tasks:

- name: New linux-image-generic installation

apt:

name: linux-image-generic

state: latest

update\_cache: yes

cache\_valid\_time: 600

register: upgraded

- name: reboot

shell: shutdown -r now

when: upgraded.changed

register: reboot

- name: Wait for the server to come back

wait\_for:

host: "{{ ansible\_ssh\_host | default(inventory\_hostname) }}"

delay: 30

state: started

search\_regex: OpenSSH

port: 22

become: false

when: reboot.changed

delegate\_to: localhost

- hosts: RedHat

gather\_facts: false

serial: 1

become: yes

max\_fail\_percentage: 0

tasks:

- name: Yum update the kernel

package:

name: kernel

state: latest

register: upgraded

- name: reboot

shell: shutdown -r now

when: upgraded.changed

register: reboot

- name: Wait for the server to come back

wait\_for:

host: "{{ ansible\_ssh\_host | default(inventory\_hostname) }}"

delay: 30

state: started

search\_regex: OpenSSH

port: 22

become: false

when: reboot.changed

delegate\_to: localhost

9)

---

- name: Network Getting Started First Playbook Extended

connection: network\_cli

gather\_facts: false

hosts: all

tasks:

- name: Get config for VyOS devices

vyos\_facts:

gather\_subset: all

- name: Display the config

debug:

msg: "The hostname is {{ ansible\_net\_hostname }} and the OS is {{ ansible\_net\_version }}"

- name: Update the hostname

vyos\_config:

backup: yes

lines:

- set system host-name vyos-changed

- name: Get changed config for VyOS devices

vyos\_facts:

gather\_subset: all

- name: Display the changed config

debug:

msg: "The hostname is {{ ansible\_net\_hostname }} and the OS is {{ ansible\_net\_version }}"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\*\*Ansible Project 1\*\***

You have been tasked with putting together a presentation to demonstrate how Ansible may be used to install software on remote hosts automatically. Before the demo, you will need to configure your test systems. You have been provided two hosts called control and workstation. You will need to configure the ansible user on workstation to have sudo access without a password to the automated software installed via Ansible. You must also configure the control host as your Ansible control server by installing Ansible on it as well as configuring the Ansible user with a pre-shared key to login to the workstation host as the ansible user.

Once the basic configuration is complete, you will need to create a simple inventory in /home/ansible/inventory on the control server containing the workstation host. Afterward, you will write a simple playbook in /home/ansbile/git-setup.yml on the control host that installs git on the workstation host. You will need to make sure the playbook works by running it from the control server.

Summary tasks list:

Install Ansible on the control host.

Create an ansible user on both the control host and workstation host.

Configure a pre-shared key for Ansible that allows the user to log in from control to workstation without a password.

Configure the Ansible user on the workstation host so that Ansible may sudo without a password.

Create a simple inventory in /home/ansible/inventory consisting of only the workstation host.

Write and execute an Ansible playbook in /home/ansible/git-setup.yml on the control node that installs git on the workstation host.

**SOLUTION:**

1. Install Ansible on the Control Node

Run the following commands on the control host:

sudo yum install epel-release

sudo yum install ansible

2. Create an Ansible user on both the control host and manage node

On each host, run the noted commands below. Make sure you set a password you can remember (you will need it later).

Assuming you are logged in as cloud\_user:

sudo useradd ansible

sudo passwd ansible

3. Configure a pre-shared key for Ansible that allows the user to log in from `control` to `workstation` without a password.

Assuming you are logged into control as cloud\_user, run the following commands providing the appropirate passwords when prompted and default options otherwise:

sudo -i -u ansible (provide cloud\_user a sudo password)

ssh-keygen (accept default options by pressing enter )

ssh-copy-id workstation (provide ansible user a password)

logout

4. Configure the Ansible user on the workstation host so that Ansible may sudo without a password.

Log into the workstation host as cloud\_user and run the following commands:

sudo visudo

Add text at the end of the file that is opened:

ansible ALL=(ALL) NOPASSWD: ALL

Save file:

(:wq in vim)

5. Create a simple inventory in `/home/ansible/inventory` consisting of only the `workstation` host.

On the control host as the ansible user run the following commands:

vim /home/ansible/inventory (note: you may use any text editor with which you are comfortable)

Add the text "workstation" to the file and save using (:wq in vim).

6. Write an Ansible playbook in `/home/ansible/git-setup.yml` on the control node that installs `git` on `workstation` then execute the playbook.

On the control host as the ansible user run the following commands:

vim /home/ansible/git-setup.yml (You may use any text editor with which your are comfortable.)

Add the following text to the file:

--- # install git on target host

- hosts: workstation

become: yes

tasks:

- name: install git

yum:

name: git

state: latest

Save (:wq in vim) and quit the text editor.

Run ansible-playbook -i /home/ansible/inventory /home/ansible/git-setup.yml.

**\*\*Ansible Project 2\*\***

**Additional Information and Resources**

Your CIO has greenlit a proof of concept for Ansible in your environment. You are to set up an Ansible control node in a test environment and verify basic functionality. You have three demo hosts, one to be the control node (control1), and two to serve as managed nodes (node1 and node2). You must complete the following steps:

1. Install Ansible on the control node.

2. Configure the ansible user on the control node for ssh shared key access to managed nodes.

Note: do not use a passphrase for the key pair.

3. Create a simple Ansible inventory on the control node in /home/ansible/inventory containing node1 and node2.

4. Configure sudo access for Ansible on node1 and node2 so that Ansible may use sudo for any command with no password prompt.

5. Verify each managed node can be accessed by Ansible from the control node using the ping module. Redirect the output of a successful command to /home/ansible/output.

Important Notes:

The user ansible is already present on all servers for your convenience.

The ansible user has the same password as the cloud\_user.

/etc/hosts entries are present on control1 for the managed nodes.

SOLUTIONS:

1) Install Ansible on the control node.

To install Ansible on the control node, run sudo yum install ansible.

2) Configure the `ansible` user on the control node for ssh shared key access to managed nodes. Do not use a passphrase for the key pair.

i. To create a keypair for the ansible user on the control host, run the following:

sudo su - ansible

ssh-keygen (accept all defaults: press enter for each prompt)

ii. Copy the public key to both node1 and node2.

iii. As the ansible user on the control host:

ssh-copy-id node1 (accept the host key if prompted, authenticate as ansible user)

ssh-copy-id node2 (accept the host key if prompted, authenticate as ansible user)

3) Create a simple Ansible inventory on the control node in `/home/ansible/inventory` containing `node1` and `node2`.

i. On the control host:

sudo su - ansible (if not already ansible user)

touch /home/ansible/inventory

echo "node1" >> /home/ansible/inventory

echo "node2" >> /home/ansible/inventory

4) Configure sudo access for Ansible on `node1` and `node2` such that Ansible may use sudo for any command with no password prompt.

\* Log in to node1 as cloud\_user and edit the sudoers file to contain appropriate access for the ansible user:

ssh cloud\_user@node1

sudo visudo

Add the following line to the file and save:

ansible ALL=(ALL) NOPASSWD: ALL

Repeate these steps for node2.

5) Verify each managed node is able to be accessed by Ansible from the control node using the `ping` module. Redirect the output of a successful command to `/home/ansible/output`.

i. To verify each node, run the following as the ansible user from the control host:

ansible -i /home/ansible/inventory node1 -m ping

ansible -i /home/ansible/inventory node2 -m ping

ii. To redirect output of a successful command to /home/ansible/output:

ansible -i /home/ansible/inventory node1 -m ping > /home/ansible/output

**\*\*\*Ansible Project 3\*\*\***

Additional Information and Resources

Some consultants have been employed to perform audits on a number of systems in your company's environment. You must create the user accounts noted in /home/ansible/userlist.txt and set up the provided public keys for their accounts. The security team has built a jump host for the consultants to access production systems and provided the full key-pair to you so you may set up and test the connection. All hosts in dbsystems will need the provided public key installed so the consultants may use key-pair authentication to access the systems. Also, you must ensure the auditd service is enabled and running on all systems.

To summarize, you must do the following:

1. Create the user accounts noted in /home/ansible/userlist.txt.

2. Copy the authorized\_keys file for each user to the correct location so the new accounts can log in with ssh key authentication.

3. Ensure auditd is enabled and running on all systems.

Important notes:

\* For your convenience, Ansible is already on the control node. If you connect to the server by clicking on the Public IP address in your browser, make sure to change to the ansible user with the su - ansible command.

\* The user ansible is present on all servers with appropriate shared keys for access to managed servers from the control node. Make sure to use this user to complete the commands.

\* The ansible user has the same password as cloud\_user.

\* The default Ansible inventory has been configured for you with the appropriate hosts and groups.

/etc/hosts entries are present on control1 for the managed servers.

SOLUTIONS:

1. Create the User Accounts Noted in `/home/ansible/userlist.txt`

ansible dbsystems -b -m user -a "name=consultant"

ansible dbsystems -b -m user -a "name=supervisor"

2. Place Key Files in the Correct Location, `/home/$USER/.ssh/authorized\_keys`, on Hosts in `dbsystems`

\* ansible dbsystems -b -m file -a "path=/home/consultant/.ssh state=directory owner=consultant group=consultant mode=0755"

\* ansible dbsystems -b -m copy -a "src=/home/ansible/keys/consultant/authorized\_keys dest=/home/consultant/.ssh/authorized\_keys mode=0600 owner=consultant group=consultant"

\* ansible dbsystems -b -m file -a "path=/home/supervisor/.ssh state=directory owner=supervisor group=supervisor mode=0755"

\* ansible dbsystems -b -m copy -a "src=/home/ansible/keys/supervisor/authorized\_keys dest=/home/supervisor/.ssh/authorized\_keys mode=0600 owner=supervisor group=supervisor"

3. Ensure `auditd` Is Enabled and Running on All Hosts

\* ansible all -b -m service -a "name=auditd state=started enabled=yes"

**\*\*Ansible Project 4\*\***

Additional Information and Resources

Your company has been increasing the deployment of small broacher-style websites for clients. The head of IT has decided that each client should have their own web servers for better client isolation and has tasked you with creating concept automation to quickly deploy web-nodes with simple static website content.

You have been provided an ansible control node and 2 test lab servers (node1 and node2) that have been preconfigured with the ansible user and key.

You must create an ansible inventory in /home/ansible/inventory containing a host group named web. The web group should contain node1 and node2.

Furthermore, you must design an Ansible playbook that will execute the following tasks on your configured inventory: install httpd, start and enable the httpd service, and install a simple website provided on a repo server. Create the playbook in /home/ansible/web.yml. The simple website may be accessed from http://repo.example.com/website.tgz.

**Summary tasks list:**

1. Create an inventory in /home/ansible/inventory containing a host group named web. The web group should contain node1 and node2.

2. Create a playbook in /home/ansible/web.yml.

3. Configure the playbook to install httpd on the web group.

4. Configure the playbook to start and enable the httpd service on the web group.

Configure the playbook to retrieve the website from http://repo.example.com/website.tgz on each server in the web group.

5. Configure the playbook to unarchive the website into /var/www/html on all servers in the web group.

6. Execute the playbook you created using the inventory you created to verify your work.

Important notes:

For your convenience, Ansible has been installed on the control node.

The user ansible is present on all servers with appropriate shared keys for access to managed servers from the control node.

The ansible user has the same password as cloud\_user.

/etc/hosts entries have been made on control1 for the managed servers.

Learning Objectives

1) Create an Inventory in `/home/ansible/inventory `That Contains a Host Group Named `web`. The `web` Group Should Contain `node1` and `node2`

echo "[web]" >> /home/ansible/inventory

echo "node1" >> /home/ansible/inventory

echo "node2" >> /home/ansible/inventory

check\_circle

Create a Playbook in `/home/ansible/web.yml`

echo "---" >> /home/ansible/web.yml

2) Configure the Playbook to Install `httpd` on the `web` Group

Using a text editor, such as vim, edit /home/ansible/web.yml to contain the following text block below the line containing "---":

- hosts: web

become: yes

tasks:

- name: install httpd

yum: name=httpd state=latest

check\_circle

Configure the Playbook to Start and Enable the `httpd` Service on the `web` Group

Using a text editor such as vim, edit /home/ansible/web.yml to contain the following task block after the "install httpd task":

- name: start and enable httpd

service: name=httpd state=started enabled=yes

3) Configure the Playbook to Retrieve the Website from \*http://repo.example.com/website.tgz\* on Each Server in the `web` Group

Using a text editor such as vim, edit /home/ansible/web.yml to contain the following task block after the "start and enable httpd" task:

- name: retrieve website from repo

get\_url: url=http://repo.example.com/website.tgz dest=/tmp/website.tgz

4) Configure the Playbook to Unarchive the Website into `/var/www/html` on All Servers in the `web` Group

Using a text editor such as vim, edit /home/ansible/web.yml to contain the following task block after the "retrieve website from repo" task:

- name: install website

unarchive: remote\_src=yes src=/tmp/website.tgz dest=/var/www/html/

5) Verify the Work by Executing the Playbook Using the Inventory

ansible-playbook -i /home/ansible/inventory /home/ansible/web.yml

**\*\*Ansible Project 6\*\***

Additional Information and Resources

Your supervisor has asked you to find a way to automate and audit basic system configuration for new servers in your environment. Given that Ansible is already configured at a basic level in your environment, the simplest solution would be to just write a playbook for bootstrapping your new hosts. Create a playbook called /home/ansible/bootstrap.yml to fulfill the following boot strap requirements:

**All servers:**

Edit /etc/ansible/hosts to include the following entry:

ansible.xyzcorp.com 169.168.0.1

Install elinks

Create the user xyzcorp\_audit

Copy the files /home/ansible/motd and /home/ansible/issue to /etc/

**Network servers:**

Install nmap-ncat

Create the user xyzcorp\_network

**SysAdmin servers:**

Copy /home/ansible/scripts.tgz from the control node to /mnt/storage

The Ansible control node has been configured for you and each testing server has already been configured for use with Ansible. The default inventory has been configured to include a the groups network and sysadmin. Each group includes a sample host.

**SOLUTIONS:**

1) Create a basic playbook for all servers in the Ansible inventory.

Create a basic playbook in /home/ansible/bootstrap.yml that completes the noted task required for all servers in the Ansible inventory.

Create the file /home/ansible/bootstrap.yml and add the following content:

- hosts: all

become: yes

tasks:

- name: edit host file

lineinfile:

path: /etc/ansible/hosts

line: "169.168.0.1 ansible.dhope.com"

- name: install elinks

package:

name: elinks

state: latest

- name: create audit user

user:

name: dhope\_audit

state: present

- name: update motd

copy:

src: /home/ansible/motd

dest: /etc/motd

- name: update issue

copy:

src: /home/ansible/issue

dest: /etc/issue

2) Add a section to the playbook for the Network servers in the Ansible inventory.

Add a section to the playbook in /home/ansible/bootstrap.yml that completes the noted task required for the Network servers in the Ansible inventory.

Create a play in the file /home/ansible/bootstrap.yml similar to the following:

- hosts: network

become: yes

tasks:

- name: install netcat

yum:

name: nmap-ncat

state: latest

- name: create network user

user:

name: dhope\_network

state: present

3) Add a section to the playbook for the SysAdmin servers in the Ansible inventory.

Add a section to the playbook in /home/ansible/bootstrap.yml that completes the noted task required for the s SysAdmin servers in the Ansible inventory.

Create a play in the file /home/ansible/bootstrap.yml similar to the following:

- hosts: sysadmin

become: yes

tasks:

- name: copy tarball

copy:

src: /home/ansible/scripts.tgz

dest: /mnt/storage/

4) Execute playbook to verify your playbook works correctly.

Execute playbook /home/ansible/bootstrap.yml to verify your playbook works correctly.

Run ansible-playbook /home/ansible/bootstrap.yml from the control node.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Build Jenkins job from Ansible**

https://www.youtube.com/watch?v=Fx49hGhAnVk

Launch Docker Container using Ansible

https://www.youtube.com/watch?v=lqEnooL0BkI

https://www.youtube.com/channel/UCSgr-7zbMjng7ykqampEBGQ

https://www.youtube.com/watch?v=gEX1HbM4KSM