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

**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**

**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.

**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 study about its 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.

[Learn DevOps Now](https://www.edureka.co/devops/)

**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.

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**

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

• 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/).