**Implementation of Ansible over AWS**

by

Yaksh Rajeshkumar Thakkar

Submitted in partial fulfilment of the requirements for the degree of

MASTER OF ENGINEERING

Major Subject: Internetworking

at

DALHOUSIE UNIVERSITY

Halifax, Nova Scotia   
December 2022

© Copyright by *Yaksh Thakkar*, 2022

**Dalhousie University**

**Faculty of Engineering**

**Internetworking**

The undersigned hereby certify that they have read and award a pass in INWK 6800 for the course/seminar entitled "Implementation of Ansible over AWS" by *Yaksh Rajeshkumar Thakkar* in partial fulfilment of the requirements for the degree of Master of Engineering.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Maen Artimy

**DALHOUSIE UNIVERSITY**

**INTERNETWORKING PROGRAM**

**AUTHORITY TO DISTRIBUTE REPORT**

Title:

Implementation of Ansible over AWS

The Internetworking Program may make available or authorise others to make available individual photo/microfilm or soft copies of this report without restrictions after December 2022.

The author attests that permission has been obtained for the use of any copyrighted material appearing in this report (other than brief excerpts requiring only proper acknowledgement in scholarly writing) and that all such use is clearly acknowledged.

Full Name of Author: Yaksh Rajeshkumar Thakkar

Signature of Author: Yaksh Rajeshkumar Thakkar

Date: 28/11/2022

Table of Contents

LIST OF FIGURES v

ACKNOWLEDGEMENTS vi

EXECUTIVE SUMMARY vii

1 INTRODUCTION 1

1.1 Outline 2

2 AWS 3

2.1 Cloud Computing 3

2.1.1 Types of Cloud computing 3

2.2 What is AWS? 4

2.3 Types of AWS Services: 4

2.3.1 Computing Services 5

2.3.2 Networking Services 5

2.3.3 Storage Services 5

2.3.4 Application Services 6

2.4 Types of EC2 Instances 6

2.5 Summary 7

3 Ansible 8

3.1 Use of Ansible 9

3.2 Working of Ansible 10

3.3 Ansible Architecture: 11

3.4 Features of Ansible 13

3.5 Summary 16

4 Implementation 17

4.1 Creating an EC2 instance on AWS 17

4.2 Install ansible on Master 20

4.3 Install python on Slave 21

4.4 Configure SSH access to Ansible Host 21

4.5 Setting up ansible host and testing connection 22

4.6 Run a Playbook to install nginx server 24

5 Conclusion 25

REFERENCES 26

LIST OF FIGURES

Figure 3.1 Features of Ansible 5

Figure 4.1 Launching an Instance 7

Figure 4.2 Selecting OS 7

Figure 4.3 Creating Key pair 8

Figure 4.4 Creating/Selecting security group 11

Figure 4.5 Verify ansible installation 12

Figure 4.6 SSH from Master to Slave 13

Figure 4.7 Creating Group. 14

Figure 4.8 Pinging through Ansible 17

Figure 4.9 Creating Playbook 24

Figure 4.10 Executing Playbook 24

ACKNOWLEDGEMENTS

I want to thank Dalhousie University for giving me the opportunity to talk about my implementation of Ansible over AWS study. I also want to express my gratitude to Dr. Maen Artimy, Professor of Engineering Mathematics and Internetworking, who served as my seminar guide and instructor, for his support and counsel throughout the course, which helped me to successfully complete my seminar project and achieve the desired results.

In addition, I want to thank my friends for helping me learn about Fortinet products. I also want to thank my lecturers, especially Dr. Maen Artimy, who taught network design in such detail and the labs which contained Ansible which encouraged me to choose this topic for the seminar.

I also want to thank my parents for providing me with a platform so I could succeed and build on the experiences that have prepared me for challenges. I appreciate the assistance and guidance I've received.

EXECUTIVE SUMMARY

Companies can access anything from programmes to storage from just a cloud service provider, renting access to them instead of owning their local computing infrastructure including data centres hence Cloud computing has boomed so much nowadays. One of the major cloud computing platforms available right now is AWS.

You may use Ansible, an open-source programme, to control your AWS installations. Using automation playbooks, you may use it to build, publish, and monitor applications and services. With the help of these playbooks, you may describe configurations only once and deliver them uniformly across environments.

The assurance of secure automation is a further advantage of utilising Ansible. In cloud systems, configuration errors are a serious vulnerability, but automation may help you make sure that only authorised configurations are deployed. You don't want every member of your team to have automatic access to whatever they please, though.

In this Implementation, I have first created a couple of EC2 instances and naming them as master and slave. Initially, I have installed ansible on master and python on remote slave. After that I configured master to do keyless SSH to the slave. Once the keyless SSH was successful, I pinged the Slave through Ansible by creating groups, and it was successful. In the end, wrote a playbook to install nginx server on the slave remotely.

The implementation is to show that suppose we want to install multiple application on multiple instances then instead of going to each instances manually and installing the application, It is very efficient to install it through Ansible remotely from a single instance.

# INTRODUCTION

There is a subtle increase in the cloud computing nowadays. The on-demand, pay-as-you-go deployment of IT resources through the Internet is known as cloud computing. You can use technological services, such as computing power, storage, and databases on an as-needed basis from a cloud provider like Amazon Web Services (AWS) rather than purchasing, operating, and maintaining physical data centres and servers.

The most complete and widely used cloud platform in the world, Amazon Web Services (AWS), provides over 200+ fully functional services from data centres across the world. Millions of clients use AWS to save costs, increase agility, and accelerate innovation, including the largest corporations, most successful governmental organisations, and the fastest-growing start-ups.

A software programme called Ansible offers straightforward yet effective automation to support computers on various platforms. It is primarily designed for IT professionals, who utilise it for almost everything a systems administrator performs on a weekly or daily basis, including application deployment, upgrades on machines and servers, automating cloud infrastructure, managing configuration, intra-service orchestration, and more. Ansible is simple to deploy because it doesn't require agent software and doesn't require any additional security architecture.

Ansible does a great job in automation over AWS. Suppose we need to install an application on all the instances that are running right now then it is a big hassle to go to each instance and install the application manually. We can create a playbook with the instruction to install the application and it does the stuff of going to each instances automatically. Hence Ansible makes cloud computing very efficient.

## Outline

I have divided the content of my reports into 6 parts. Each chapter consists of following topics:

**Chapter 1-Introduction:** An overview of cloud computing, AWS, Ansible and benefit of Ansible over AWS.

**Chapter 2-AWS:** This chapter deals with the architecture of AWS and the services available inside AWS and its types

**Chapter 3-Ansible:** This chapter deals with the architecture of Ansible, it benefits and how to use it

**Chapter 4-Implementation:** step by step implementation of Ansible over AWS from scratch

**Chapter 5-Conclusion:** Conclusion consists of the things I learned from this topic and how Ansible integrates with AWS so well.

# AWS

In this chapter, we will discuss the concepts of AWS, what is cloud computing and its types, types of AWS services available and its use and types of EC2 instances.

## Cloud Computing

As we know that now a days, cloud computing has increased a lot because it reduces IT costs, increasing collaboration, more secure and flexible to use. [1]

### Types of Cloud computing

Basically, there are three types of cloud:

* **Public Cloud**: This is a type of cloud which is managed by a third party and the services needed to be used in this type of cloud is managed by this third party which are then become available to public through internet.
* **Private Cloud**: This is a type of cloud which is created for an organization and the services needed on this cloud are selected by the organization themselves according to their need.
* **Hybrid Cloud**: Hybrid cloud is the mixture of public and private. The decision of keeping the cloud public or private stays on the type of the parameters like the information that needs to be secured, certificates used in the cloud etc.

## What is AWS?

There are various cloud computing services that are available, and these services put together in a single platform forms AWS which provide ease, reliability, and very economical solution to the users hence AWS is so much in the market and used by various organizations right now for their cloud computing solutions.

An important term to get used to in AWS is an instance. What is an instance? It is a virtual server that can be launched on the AWS console which can be configured according to the type of OS, applications and services need on the server. [2]

## Types of AWS Services:

Services are the most important part of the AWS which make it so strong and on top in the market. There are more than 200 types of services offered by AWS and each of them has their own role which has been created to satisfy the needs of a user.

* **AWS Auto-Scaling**: It is a type of AWS service which automatically provides the resources to the instance and adjust the resources as needed by the instance for it be used as smooth as possible and as economical as possible. [3]
* **Elastic load Balancing**: This service performs the task of load balancer. Suppose the traffic on an instance increases suddenly then due to this service it creates another instance automatically to reduce the traffic on a single instance and when the traffic decreases such that another instance is not needed now then it kills the instances that are not needed now thereby limiting the use of the resources. [4]

AWS Services can be divided based on the type of the work they do:

### Computing Services

* **EC2 Instance:** EC2 stands for elastic compute cloud which means that it is used to create a virtual server which is elastic. Elastic here means that it can adjust the resources needed on the server according to the need of the user to run their applications. [5]
* **Lambda:** This service works as a function in the coding language i.e., whenever a function is called it runs a bunch of code written in the function for an assigned work similarly this service runs code when an event happens to manage resources in an instance or to create an instance.
* **Elastic Beanstalk:** This service is used for container management as it deploys and handles those containers to create a platform to interact with web applications.

### Networking Services [2]

* **VPC:** VPC stands for virtual private cloud. As from the name we can say that it is used to create a cloud that has its resources private. Typically, VPC is used by an organization who wants to use the AWS architecture to create their own private network.
* **Direct Connect:** This is type of services which establishes direct connection between host and AWS virtual server without internet connection.
* **Route 53:** This service deals with the port 53 hence is named as route 53. It is used as DNS service for the domain which runs on port 53.

### Storage Services

* **S3:** S3 stands for simple storage service. It provides the resource to store data on the cloud and is scalable, secure, and reliable.
* **EFS:** EFS stands for elastic file storage which can be common between various EC2 instances.
* **DMS:** DMS stands from data migration service which is used to transfer whole databases on the cloud and is used to convert a database. It is easy and secure.

### Application Services

* **API Gateway:** This service is API management service which secures, manages, keeps track, and creates API.
* **SES:** Simple email service is used for the interaction through email inside AWS.
* **Elastic Transcoder:** This service is used for the low-cost conversion of media files over the cloud.

## Types of EC2 Instances [5]

* General Instance: It is used to create an instance in which the resources are used in a balance to make the instance smooth and economical.
* Computer Instance: It is used to create an instance with more computing power hence more resources are used on the processor.
* Memory Instance: It is used to create an instance with more RAM for the applications which require more RAM usage.
* Storage Instance: It is used to create an instance with more storage for the applications that need more storage.
* GPU Instance: It is used to create an instance with more graphical resources for the applications that require more graphical rendering.

## Summary

In this chapter, we learnt the types of services according to the work they do. We also saw the types of instances that can be created. We saw the definition of cloud computing and its types.

# Ansible

Ansible is a tool to provide automation over the cloud which can be used between multiple platforms and free to use by the users and can make use of “infrastructure as code” approach to provide the automation needed

Since the beginning, Ansible has grown vastly to provide proper automation results to its users in various industries. It is a dominant automation solution which is both open-source and has created a standard in IT market. Ansible works on various platforms which are similar to Unix and manages architecture like Microsoft and Unix. Ansible was created by Michael DeHaan and was took over by RedHat later in 2015.

Ansible is used to execute commands on a remote SSH or windows remote management connection which require no one in between thereby creating it agentless. Ansible is used to create time efficient solution which automates the work and increases consistency, scalability, and reliability of the infrastructure

Automation creates a task easy to handle which is complex to manage hence user can make use of that time to do something fruitful for their organization. This means that production is increased in less time. Ansible provides quick configuration of whole network with no expertise in programming. Some common tasks performed by Ansible are installing software on hosts, creating daily tasks automatic, increase security and sharing these automation task to the whole network. [6]

## Use of Ansible

* DevOps: Ansible creates the execution of an application automatic for the production hence makes DevOps easy to use. Ansible is among the most used DevOps tool right now to automate, orchestration, configure and manage the IT architecture. Ansible’s main upper hand is that it can adapt and scale according to the needs of the demand. The benefits of using Ansible for DevOps are quick acceleration of feedback loop, dependable installations, co-ordination of IT architecture and fast deployments.
* Docker Container Management: Docker is used to create and implement containers on server which is a powerhouse framework. Docker container is almost similar to virtual server which are lightweight. Ansible has various built-in modules which can be used to create scripts for automation easier. Ansible make use of YAML language as playbook to achieve that task.
* Automation: The main task of Ansible is automation. Automation of any deployment is the initial step in automating the whole service of the application. Ansible has capacity to automating gigantic IT architecture, virtual server, and cloud platforms. Also, it can configure multiple systems, databases, networks, storage devices and firewall automatically.
* Managing Configuration: Ansible is used to automatically configure devices from a remote server. Anyone with IT experience can set it up in no time. Ansible configuration is a description of basic infrastructure data that computers can read and process for humans. Only a password or SSH (Secure Socket Shell) key is required to start monitoring a machine.
* To install application using Ansible: Applications with several tiers may be executed quickly and easily thanks to Ansible. To manage processes, one does not need to write code; instead, one may define the required measures in a playbook, and Ansible will figure out how to get the processes to the desired result. In other terms, it won't be necessary to manually configure software on each computer. When a module is started from a control device, Ansible uses SSH to communicate with external networks and carry out all instructions. [6] [7]

## Working of Ansible

Your networks are communicated with by Ansible, which also sends them tiny programmes known as modules. These modules are used in systems created to be asset designs for the functionality in the desired condition to complete automated tasks. Ansible executes these modules and removes them once they are finished. You would have to rely on ad hoc methods and scripting to execute tasks if modules weren't available. The main node in charge of supervising the Playbook's execution is Ansible's management node.

Before running the components and downloading the software on the host workstations, the management node establishes an SSH connection. It removes the modules after they have been deployed. So that's how Ansible functions. Ansible scripts are written in Python and connect to remote machines using SSH as stated in the configuration file. Ansible is agentless, which means that the nodes it commands don't need to have any software installed.

To decide which devices, you want to control, Ansible uses inventory data. Although it comes with a standard configuration file, users can modify it to handle the specific servers they desire. Ansible uses the SSH protocol to establish connections to servers and perform tasks. Ansible connects to the network connection and disperses the modules needed for playbook or command execution. Ansible employs YAML templates that are readable by humans to make it possible to automate routine tasks without having to learn a difficult programming language. [6]

Every program code that can output JSON, such as Ruby, Python, and shell, can be used to create Ansible modules. The creation of Windows automation modules uses PowerShell. It also keeps track of the history of your application, making it easy to update or repeal to an earlier standard. It offers the following benefits: [7]

* Ansible is a freely available and accessible software tool for all.
* The installation and use of Ansible's playbooks don't require coding knowledge; they are simple to set up and employ.
* Even for the most complicated IT processes are modellable with Ansible.
* Whatever the location of the application's deployment, the entire app environment may be orchestrated.
* You don't need to install any special software or access any firewall ports, and it doesn't have an agent on the client computers you want to run.
* Servers have more room for app resources because no additional software is required, which increases efficiency.

## Ansible Architecture:

* **Modules**: All local connection sites or remote hosts that Ansible connects to receive modules from the command computer. They are predefined commands that remote hosts immediately carry out. Playbooks execute modules that control files, packages, and applications. All components for distributing changes or carrying out the appropriate action are run by Ansible, and after they are finished, they are removed. There are approximately 450 modules in Ansible for common tasks. Whenever a playbook is launched, multitudes of built-in modules in Ansible are executed. A playbook has pieces, which encompass a variety of activities and modules [8]
* **Playbooks**: Ansible playbooks are instructions for specific tasks. Your workflow is dictated by playbooks since they run functions in the sequence they are created. They are plain text documents made in the human-readable data serialisation language known as YAML. They are the foundation about whatever makes Ansible so appealing since they rapidly communicate the activities one should complete without forcing the user to know specific terms. They could specify settings, coordinate the steps of any manually planned task, and carry out tasks simultaneously or sequentially. [9]
* **Plugins**: Small pieces of code known as plugins increase a website's capabilities. There are several of these included with Ansible, but one can also make their own. In this scenario, a certain class of module is a plugin. The plugins are executed until a module is executed on the nodes. Plugins are run on the principal control unit for logging purposes. Call-back plugins are useful since they let you connect to different Ansible events for reporting and presentation. Cache plugins are utilized to reduce the costs associated with fact-finding procedures. As opposed to calling the modules directly, action plugins are among front modules that operate on the controller system. [10]
* **Inventory:** All the endpoints or servers that need to be maintained are listed in the inventory, along with respective IP addresses, server information, databases, and other metadata. This detailed data is given in a single, simple-to-read file that contains a list of all computers (control and nodes) being used with Ansible. You can give parameters to numerous hosts using a straightforward text-based syntax after registering the inventory.
* **API:** Application programming interfaces, sometimes known as APIs, can be used to improve Ansible's connectivity options. This has callbacks and other features in addition to using SSH for communication. Applications for both public and private clouds can be accessed through the Ansible APIs.
* **Hosts and Networking:** In the Ansible design, hosts are any computer such as Linux, RedHat, Windows as well as the nodal systems that Ansible administers. It uses a data schema that is specific to the Ansible automated systems and can easily move between different hardware platforms.
* **Cloud:** A cloud, either private or public, is a network of remote servers that can be used to gather, arrange, and process data. Those applications are hosted online rather than storing information on a local machine. You can start working remotely after just deploying the resources in the cloud and instances and connecting them to the databases.
* **Configuration management database (CMDB):** A database acting as a data store for IT systems is known as a CMDB. Users can automatically convert the output of Ansible's data-collection functionality into a static HTML page displaying by installing the Ansible-CMDB code.

## Features of Ansible

Diagram

Description automatically generated

Figure3.1 Features of Ansible [7]

* **Agentless Functioning:** Because Ansible is a cloud - based system, the computers it commands don't get new software installed. By doing so, potential flaws and online threats are removed while also reducing system resources. All master-agent communications are handled by Ansible. Either conventional SSH or indeed the Para Miko package, a Python configuration of SSH2, are used to accomplish this. The tool may be controlled without the deployment of agents on distant networks, which reduces cost of maintenance and performance degradation.

Windows Remote Management and SSH allow for agentless Ansible usage. This causes distant workstations to "self-destruct" rather than execute Unix commands by sending them tiny, automatically produced modules. These modules provide JSON data while defining desired states and structured processes.

Since the way activities are carried out, this highly effective technique may reuse connections and uses very little network traffic. This approach eliminates all issues associated with agent-based systems and increases resource efficiency for both users and the central administration server. For instance, Ansible eliminates the problem of "who will manage the management." Two further characteristics that increase the attractiveness of this setup are non-root accessibility and a reduced attack surface.

* **Python Support:** Ansible can be used in a wide range of ways with an API perspective. By using Ansible Python API, you can manage nodes, extend Ansible to react to various Python events, create plugins, and integrate inventory information from numerous data sources. The learning curve for Ansible, which is built in Python, is not too steep.

Ansible offers a simple installation procedure that doesn't call for any software installation, hosts, or client daemons. By design, it runs in parallel and uses SSH to manage nodes. In contrast to competing solutions created in languages like Ruby, Ansible was built in Python. Because most Linux PCs already have Python libraries preinstalled, configuring, and using it is simpler. Python is more likely to be well-known among programmers and system administrators than Ruby. Additionally, users can create Ansible modules in any language to expand the tool's functionality as long as the information is in JSON format.

* **SSH as Security:** Aneffective password-free network authentication system. So, it's up for you to give the user this key. Instead of using a separate client agent, Ansible connects to clients via SSH and pushes modules to them, which are then executed directly on the client end. The Ansible server receives the outcome once more. SSH is a protocol that Ansible employs to connect to remote computers. Similar to SSH, Ansible establishes connections to remote machines utilizing native OpenSSH with your existing username.

SSH makes the entire setup process simpler. Inventory files store client data including such hostnames, IP addresses, and SSH ports. Once produced and filled out, an inventory file may be used by Ansible. You can confirm that using the same username and SSH, you can communicate with every inventory node. Additionally, if required, one can add the open SSH key to those systems' authorized keys file. Ansible uses SSH to run commands against local Linux systems. You must configure Ansible's SSH server so that it may automatically ignore password requests in order to utilize it.

* **Ease to setup:** The setup feature organizes playbooks, jobs, inventories, and parameter files by functions, using labels at the action and task level for more control and specificity. There are other options for structuring Ansible data, despite the fact that this is a reliable and flexible method. Ansible automates tasks and does not require human deployment, making it easy to execute jobs. In addition to other things, it may organize more complicated activities like continuous deployment and zero downtime when implementing new features.

Continuous deployment is the term used to describe the process of deploying software to a server following continuous integration and continual deployment. To use Ansible from the command - line interface to configure it, try installing the Ansible package solely on a single system. There is no requirement to launch any daemons or create a database. It takes two machines. While the second machine is known to as a "Node" and acts as a controller node, the first machine is called the "Server" and functions as a managed node.

## Summary

In this chapter, we saw the working, use and feature of the Ansible. We went through the architecture and important terms of Ansible.

# Implementation

In this section, we will do step by step implementation of Ansible over AWS. We will first start from creating an EC2 instance thereby running a playbook to install nginx server on the remote host.

## Creating an EC2 instance on AWS

For our implementation, we will require 2 EC2 instances and we will name them as Master and Slave. Master is the instance in which ansible will be installed and Slave is the host. [11]

Firstly, to create an instance we will have to create an account on AWS and then select EC2 as the service then select launch instance

Graphical user interface, text, application, email

Description automatically generated

**Figure 4.1 Launching an instance**

Afterwards, Selecting the type of instance OS we want on the server:

I am taking ubuntu as OS for the implementation

Graphical user interface, application

Description automatically generated

Figure 4.2 Selecting OS

We will have to create a key pair in order to login to the instance that we have created.

Graphical user interface, text, application

Description automatically generated

Figure 4.3 Creating key pair

We have to select or create a security group that we need on the instance

Graphical user interface, text, application, email

Description automatically generated

Figure 4.4 Creating/selecting security group

We will rename the 2 instances as master and slave. After the instance starts running, we will login to the instances from CMD using the key value pair that we created. We will use the following command to read the key by the terminal:

**chmod 400 keyaws.pem**

After the above step is done on the terminal, we will have to connect with the VM. We can do so by connecting with the following command:

**ssh -i keyaws.pem** [**ubuntu@44.201.117.3**](mailto:ubuntu@44.201.117.3)

Ubuntu is the username and 44.201.117.3 is the IP Address provided by the AWS console to connect with the instance. We can see that keyaws is used in order to connect with the instance.

-i is the identity file in which key is used to connect with the instance. Essentially, it instructs the SSH protocol to seek for the key file required for authentication on the target server. If you utilise key authentication and were given a credential, you should enter it here.

## Install ansible on Master

These are command needed to be run on Master to install ansible:

**sudo apt-get update**

**sudo apt install software-properties-common**

**sudo apt-add-repository ppa:ansible/ansible**

**sudo apt update**

**sudo apt install ansible**

After ansible is installed correcting when we run ansible –version command, we will be able to see the version installed [11]

Graphical user interface, text

Description automatically generated

Figure 4.5 Verify ansible installation

## Install python on Slave

These are the command needed to be run on Slave as Slave needs to have python on their instance: [11]

**sudo apt-get update**

**sudo apt-get install python3**

## Configure SSH access to Ansible Host

On Master, we will run following code to generate a key for the slave: [11]

**ssh key-gen**

It will generate a key similar to this:

ssh-rsa  ubuntu@ip-172-31-86-236

We have to copy this key and paste it in .ssh/authorized\_keys file of slave so that SSH connection can be established between master and slave. After the key has been generated and added into the authorized key file of slave, we can SSH from master to slave using this command:

ssh ubuntu@<slave instance ipaddress>

Text

Description automatically generated

Figure 4.6 SSH from Master to Slave

## Setting up ansible host and testing connection

We will first create a file name host and add following configuration inside the file. First, we will run this command on Master: [11]

**sudo nano /etc/ansible/hosts**

We will add the IP Address of the slave inside this file:

Text

Description automatically generated

Figure 4.7 Creating group

After we have created the group inside in the hosts file, we can ping the server with hosts mentioned in the group created.

**ansible -m ping all**

Text

Description automatically generated

Figure 4.8 Pinging through Ansible

This is the output of the ping command that we did. Here our ansible connection has been established. We can now create a playbook to run a set of tasks that are needed to be executed on the slave.

## Run a Playbook to install nginx server

Now we will run a playbook to install a nginx server on Slave machine. We will create a playbook named as playbook.yml on the desktop. [12]

Text

Description automatically generated

Figure 4.9 Creating Playbook

As we can see that first, I am applying the playbook on all hosts. Become: true means that we are giving the sudo rights to the playbook. We are implementing a task of installing nginx server on the slave machine.

After we created the playbook, we have to run the playbook with the following code:

**ansible-playbook playbook.yml**

It will give the following output once it is successful:

Text

Description automatically generated with medium confidence

Figure 4.10 Executing playbook

It took me couple of tries to create a perfect syntax of the playbook and it is very important. You can see that in the last output nginx is installed on the slave.

Here our implementation is complete

# Conclusion

In conclusion, we learnt in the paper how to implement Ansible over AWS. The building blocks of AWS, basic of cloud computing and types of cloud computing available and types of services used in the AWS according to the type of work they do.

In Ansible, we learnt the use of Ansible, how to use it and benefits of using Ansible. We showed the step by step implementation of Ansible over AWS. We started from showing how to create an EC2 instance, then installed ansible on master and python on Slave, also did keyless SSH connection between Master and Slave, created production group and added the ip address of slave in that and when the ansible was able to ping the remote host (Slave). We created a playbook to install nginx server on the Slave.

Overall, we learnt the automation of installing an application through playbook instead of going to the instance and installing the application manually.

REFERENCES

|  |  |
| --- | --- |
| [1] | AWS, "What is cloud computing?," [Online]. Available: https://aws.amazon.com/what-is-cloud-computing/. [Accessed 2 12 2022]. |
| [2] | H. Sharma, "AWS Cheat Sheet," [Online]. Available: https://intellipaat.com/blog/tutorial/amazon-web-services-aws-tutorial/aws-cheat-sheet/. [Accessed 2 12 2022]. |
| [3] | AWS, "AWS Auto Scaling," [Online]. Available: https://aws.amazon.com/autoscaling/. [Accessed 2 12 2022]. |
| [4] | AWS, "Elastic Load Balancing," [Online]. Available: https://aws.amazon.com/elasticloadbalancing/. [Accessed 2 12 2022]. |
| [5] | AWS, "Amazon EC2," [Online]. Available: https://aws.amazon.com/ec2/instance-types/. [Accessed 2 12 2022]. |
| [6] | openSource, "What is Ansible?," [Online]. Available: https://opensource.com/resources/what-ansible. [Accessed 2 12 2022]. |
| [7] | C. BasuMallick, "What Is Ansible? Uses, Working, Architecture, Features," [Online]. Available: https://www.spiceworks.com/tech/devops/articles/what-is-ansible/. [Accessed 2 12 2022]. |
| [8] | A. Doc, "Using Ansible modules and plugins," [Online]. Available: https://docs.ansible.com/ansible/latest/module\_plugin\_guide/index.html. [Accessed 2 12 2022]. |
| [9] | T. Point, "Ansible-Playbooks," [Online]. Available: https://www.tutorialspoint.com/ansible/ansible\_playbooks.htm. [Accessed 2 12 2022]. |
| [10] | "Ansible Plugins," [Online]. Available: https://www.educba.com/ansible-plugins/. |
| [11] | Intellipaat, "Ansible Installation & Configuration on AWS | Install & Configure Ansible on EC2 | Intellipaat," [Online]. Available: https://www.youtube.com/watch?v=Km3BCQnV6sw. [Accessed 2 12 2022]. |
| [12] | Simplilearn, "What is an Ansible Playbook and How to Write One on Your Own," [Online]. Available: https://www.simplilearn.com/what-is-ansible-playbook-article. [Accessed 2 12 2022]. |