INSTITUTE FOR ADVANCED COMPUTING
AND
SOFTWARE DEVELOPMENT
AKURDI, PUNE

DOCUMENTATION ON

"Seamless Deployments:Orchestrating AWS DevOps CI/CD with GitHub, Jenkins, Ansible, and Docker"

PG-DITISS March-2023

SUBMITTED BY:

GROUP NO: 4

CHINMAY GHATE (233411)

MR. KARTIK AWARI
PROJECT GUIDE

MR.ROHIT PURANIK
CENTRE COORDINATOR

INDEX

1. INTRODUCTION

In a business world where time is money and every dollar counts, efficiency and productivity are critical success factors. Continuous Integration (CI)/Continuous Deployment (CD) is a process that helps organizations rapidly release software with confidence The process relies heavily on automation and modern cloud-based services to provide tools for build management, monitoring, testing, and deployment automation. In this Project, we will explore how CI/CD works and how you can implement the process in your organization for rapid, reliable, and automated software delivery.

- Life Before CI/CD

Before CI/CD teams faced long, slow-release cycles that were full of uncertainties, manual deployments were the norm, and any change required extensive testing to ensure quality control. Teams would spend days or even weeks in a “staging” environment, where they tested and validated the release. If something went wrong, there was often no way to track what happened during the release, making troubleshooting near impossible.

These long release cycles had another drawback. Since everything was manual, it was difficult to scale up the process to meet increased demand.With the advent of CI/CD, all that has changed. Deployments are now automated, predictable, frequent, and quick. The release process follows a well-defined and structured series of steps. No manual intervention is required; CI/CD is a proven methodology that can help any organization bring about significant transformation in their business. Let’s explore further.

- The Three Phases of CI/CD

Successful CI/CD relies on structure and automation. The process consists of three distinct phases; continuous integration, continuous delivery, and continuous deployment. Each stage handles specific tasks in the build and release cycle.

- Continuous Integration

Continuous integration is a software development practice where team members integrate their work frequently.

- Continuous Delivery

Continuous delivery is an extension of continuous integration in which software is built, tested, and released more frequently. Continuous delivery automates software release processes such that changes can be released automatically anytime new code gets pushedto the repository. The aim is to eliminate the need for human intervention in the software deployment process.

- Continuous Deployment

  Continuous deployment is an extension of continuous delivery where all production code changes are automatically deployed into production after passing automated tests. This practice reduces the need for extensive manual testing before deployment and allows teams to react quickly when problems arise.

  Continuous deployment is especially ideal for large-scale projects where changes are deployed often, automated testing is essential, and the business impact of a single change is significant. Continuous deployment is often associated with agile development methods, but it can be beneficial for any team that depends on the frequent delivery of new code.

CI/CD pipelines are composed of a series of steps that are executed automatically when code changes are pushed to the code repository. These steps typically include:

- Build: The code is compiled and packaged into an executable form.
- Test: Automated tests are run to verify the functionality of the code changes.
- Deploy: The code changes are deployed to a staging or production environment.

There are a number of different CI/CD tools and services available, so it is important to choose the right one for your organization. Some of the most common CI/CD tools and servicesinclude:

- Build Automation: Jenkins

Jenkins is a popular open-source build automation tool. It can be used to automate the build process, including the compilation of code, testing of code, and deployment of code.

It can also be used to monitor applications, track code changes, and report metrics. Since Jenkins is open .source, it’s free to use, and it’s widely used. It’s a great choice for almost any team. It’s easy to get started with Jenkins, and there are lots of resources to help you along the way.

- Deployment Automation: Puppet, Ansible, Chef etc.

Puppet is a popular tool used to automate the deployment process, including the systemconfiguration and software installations and managing updates. Puppet has a built-in package manager that makes installing new modules easy. This is different from many other tools that require downloading and compiling code.

- Docker

Docker is a platform that enables developers to automate the deployment, scaling, and management of applications using containerization technology. Containers are lightweight, portable, and self-sufficient units that encapsulate an application and all its dependencies, including libraries, runtime, and settings, ensuring consistent behavior across different environment.

Here are some key concepts related to Docker:

Containers: Containers are isolated environments that package an application and its dependencies together. They provide a consistent runtime environment, making it easier to deploy applications across various systems without worrying about differences in underlying infrastructure.

Images: An image is a lightweight, stand-alone, and executable software package that includes everything needed to run an application, including code, runtime, libraries, and system tools. Images are used as the basis for creating containers.

Dockerfile: A Dockerfile is a text file containing instructions for building a Docker image. It specifies the base image, application code, dependencies, and other configuration settings necessary to create a functional image.

Container Registry: A container registry is a repository for storing and distributing Docker images. Popular container registries include Docker Hub, which is a public registry, and private registries that organizations can set up for their internal use.

Docker Compose: Docker Compose is a tool for defining and running multi-container applications. It uses a YAML file to define the services, networks, and volumes required for an application's components to work together.

Docker Swarm: Docker Swarm is Docker's native clustering and orchestration solution. It allows you to create and manage a cluster of Docker nodes, making it easier to deploy and scale applications across multiple machines.

2. EXISTING SYSTEM : (WITHOUT CI/CD)

Before CI/CD teams faced long, slow-release cycles that were full of uncertainties, manual deployments were the norm, and any change required extensive testing to ensure quality control. Teams would spend days or even weeks in a “staging” environment, where theytested and validated the release. If something went wrong, there was often no way to track what happened during the release, making troubleshooting near impossible.

These long release cycles had another drawback. Since everything was manual, it was difficult to scale up the process to meet increased demand.

3. PROPOSED SYSTEM :

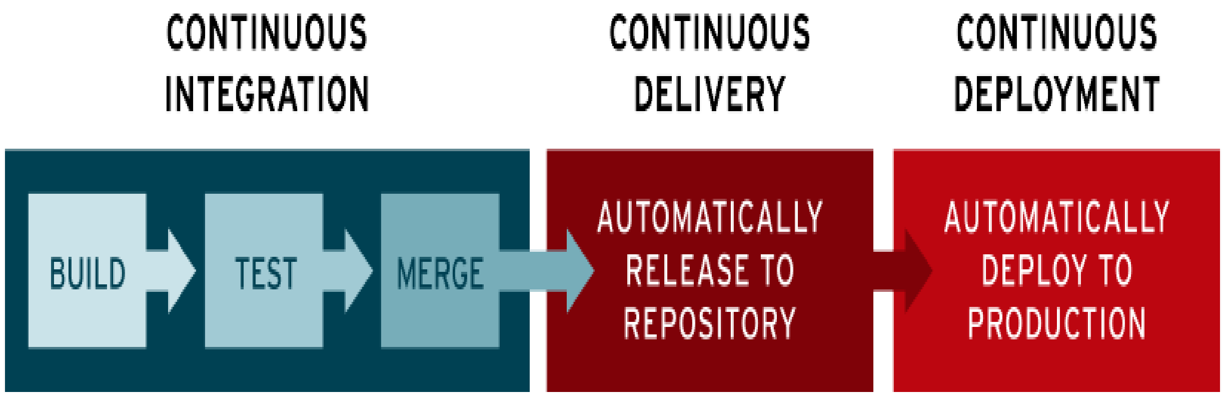
Here's a tool chain for CI/CD system:

- Version Control: Git and Git-Hub
- Continuous Integration: Jenkins
- Configuration Management: Ansible
- Deployment tool :- Docker(web server)

Importance of CI/CD-

Each tool performs a specific function and can be integrated with other tools to create a seamless CI/CD process.

A pipeline is a process that drives software development through a path of building, testing, and deploying code, also known as CI/CD. By automating the process, the objective is to minimize human error and maintain a consistent process for how software is released. Tools that are included in the pipeline could include compiling code, unit tests, code analysis, security, and binaries creation. For containerized environments, this pipeline would also include packaging the code into a container image to be deployed across a hybrid cloud.CI/CD is the backbone of a DevOps methodology, bringing developers and IT operations teams together to deploy software. As custom applications become key to how companies differentiate, the rate at which code can be released has become a competitive differentiator.



Before Devops, the model used for software development was the “Waterfall” Model. This model is best suited when all the requirements are present beforehand. Here beforehand signifiesthe planning stage. All the requirements should be finalized at that time only. No later changes can be entertained. No work is in parallel for our software development in this model. Previously, After the product planning phase, we move to the product development phase and then the testing phase and then the product building phase. No effort is done in parallel.

So, the software release lifecycle is also very long. No new feature can be proposed in the product development cycle. There is no flexibility to change the requirements after the planning phase isover. Also, previously as there were no integrated build mechanism, development team may use an environment while operations team might try to maintain it in a different environment. So, at the time or integration, if the operations team get any bug, development team may respond that the feature is working fine in their dev setup and they may ask to sync the same setof setup. Now, different dev teams might be using different dev setups. It will be very difficult for the integration team to sync to every dev team setup and a lot of time gets wasted unnecessarily.

4. TECHNOLOGY USED:

4.1 Hardware Requirement :

- RAM:2/4GB(minimum requirements)
- HDD: 20 GB(minimum requirements)

4.2 Software Requirement :

- Operating System: Linux(ubuntu)

  Tools: Git, Git-hub, Jenkins, ansible, Apache.

4.3:Block Diagram:-

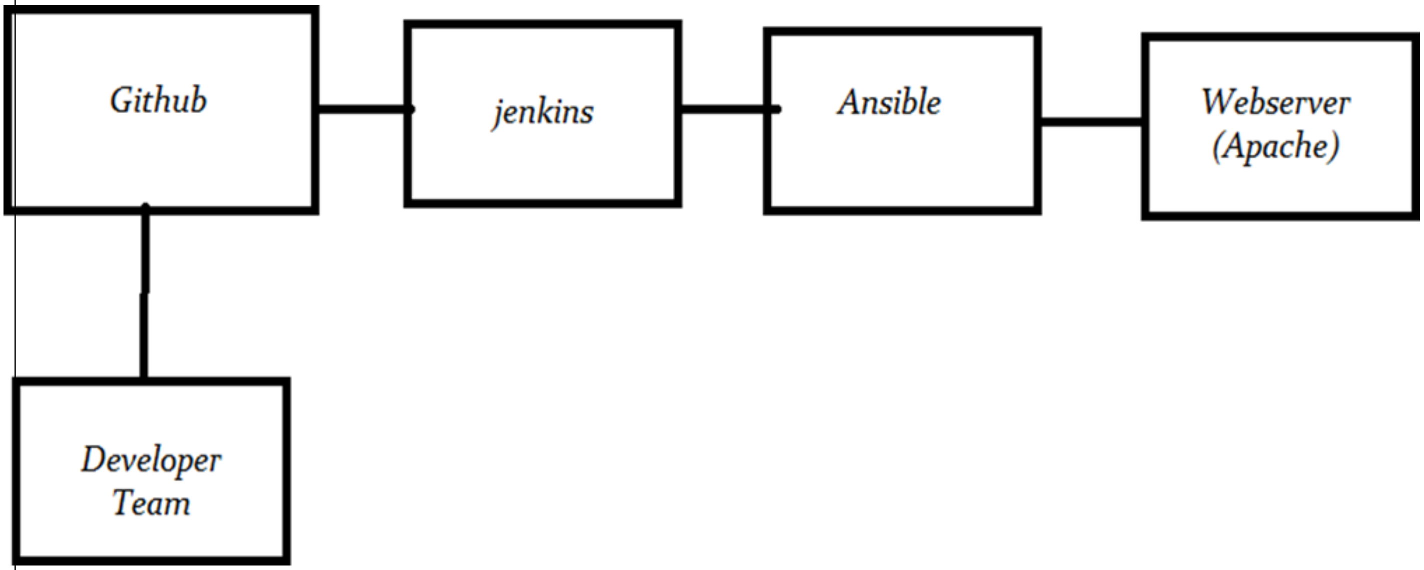


Fig: Flow of CI/CD Pipeline

5. GIT

Git is an open-source distributed version control system. It is designed to handle minor to majorprojects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

Git is foundation of many services like GitHub and GitLab, but we can use Git without usingany other Git services. Git can be used privately and publicly.

Git was created by Linus Torvalds in 2005 to develop Linux Kernel. It is also used as animportant distributed version-control tool for the DevOps.

Git is easy to learn, and has fast performance. It is superior to other SCM tools like Subversion,CVS, Perforce, and ClearCase.

- **Features of Git**

Some remarkable features of Git are as follows:

Scalability
Security
Open Source
Features of Git
Speed

- Open Source
  Git is an open-source tool. It is released under the GPL (General Public License) license.
- Scalable
  Git is scalable, which means when the number of users increases, the Git can easily handle such situations.
- Distributed
  One of Git's great features is that it is distributed. Distributed means that instead of switching the project to another machine, we can create a "clone" of the entire repository. Also, instead of just having one central repository that you send changes to, every user has their own repository that contains the entire commit history of the project. We do not need to connect to the remote repository; the change is just stored on our local repository. If necessary, we can push these changes to a remote repository.

- Security

  Git is secure. It uses the SHA1 (Secure Hash Function) to name and identify objects within its repository. Files and commits are checked and retrieved by its checksum at the time of checkout. It stores its history in such a way that the ID of particular commits depends upon the complete development history leading up to that commit. Once it is published, one cannot make changes to its old version.
- Speed

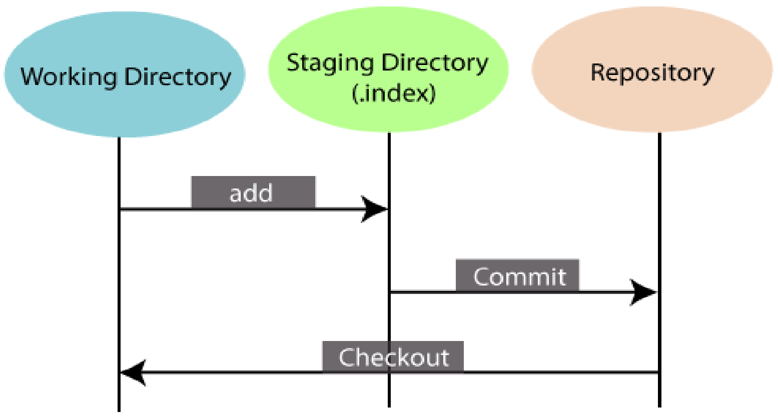
  Git is very fast, so it can complete all the tasks in a while. Most of the git operations are done on the local repository, so it provides a huge speed. Also, a centralized version control system continually communicates with a server somewhere. Performance tests conducted by Mozilla showed that it was extremely fast compared to other VCSs. Fetching version history from a locally stored repository is much faster than fetching it from the remote server. The core part of Git is written in C, which ignores runtime overheads associated with other high-level languages. Git was developed to work on the Linux kernel; therefore, it is capable enough to handle large repositories effectively. From the beginning, speed and performance have been Git's primary goals.
- Supports

  non-linear developmentGit supports seamless branching and merging, which helps in visualizing and navigatinga non-linear development. A branch in Git represents a single commit. We can construct the full branch structure with the help of its parental commit.
- Branching and Merging

  Branching and merging are the great features of Git, which makes it different from the other SCM tools. Git allows the creation of multiple branches without affecting each other. We can perform tasks like creation, deletion, and merging on branches, and these tasks take a few seconds only. Below are some features that can be achieved by branching:

  - We can create a separate branch for a new module of the project, commit and delete it whenever we want.
  - We can have a production branch, which always has what goes into production and can be merged for testing in the test branch.
  - We can create a demo branch for the experiment and check if it is working. We can also remove it if needed.
  - The core benefit of branching is if we want to push something to a remote repository, we do not have to push all of our branches. We can select a few of our branches, or all of them together.

- Data Assurance
  The Git data model ensures the cryptographic integrity of every unit of our project. It provides a unique commit ID to every commit through a SHA algorithm. We can retrieve and update the commit by commit ID. Most of the centralized version control systems do not provide such integrity by default.
- Staging Area
  The Staging area is also a unique functionality of Git. It can be considered as a preview of our next commit, moreover, an intermediate area where commits can be formatted and reviewed before completion. When you make a commit, Git takes changes that are in the staging area and make them as a new commit. We are allowed to add and remove changes from the staging area. The staging area can be considered as a place where Git stores the changes. Although, Git doesn't have a dedicated staging directory where it can store some objects representing file changes (blobs). Instead of this, it uses a file called index.

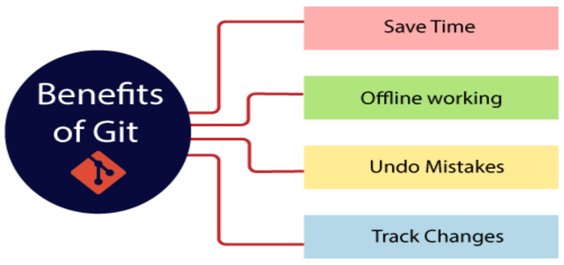


  Another feature of Git that makes it apart from other SCM tools is that it is possible to quickly stage some of our files and commit them without committing other modified files in our working directory.
- Maintain the clean historyGit facilitates with Git Rebase; It is one of the most helpful features of Git. It fetches the latest commits from the master branch and puts our code on top of that. Thus, it maintains a clean history of the project.

5.1 :Benefits of Git

A version control application allows us to keep track of all the changes that we make in the files of our project. Every time we make changes in files of an existing project, we can push those changes to a repository. Other developers are allowed to pull your changes from the repository and continue to work with the updates that you added to the project files.

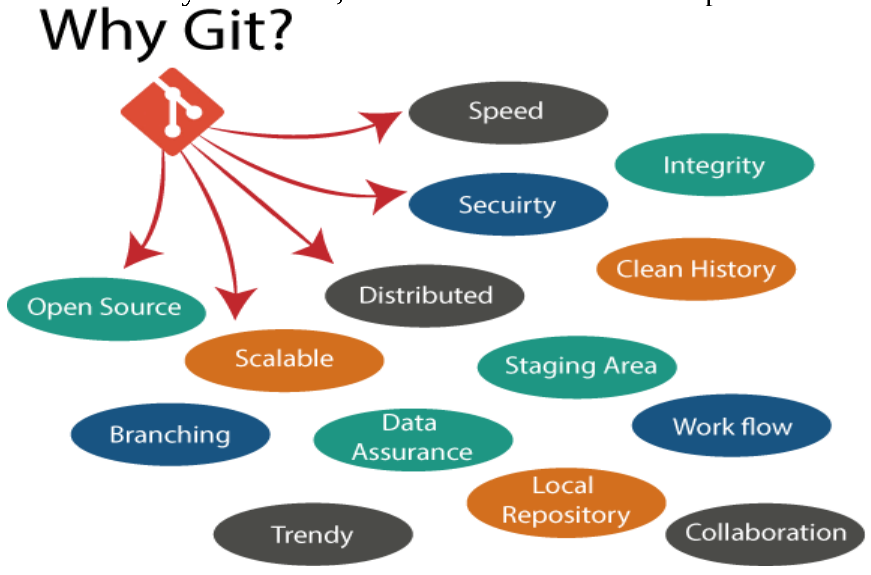
Some significant benefits of using Git are as follows:



- Saves Time
  Git is lightning fast technology. Each command takes only a few seconds to execute so we can save a lot of time as compared to login to a GitHub account and find out its features.
- Offline Working
  One of the most important benefits of Git is that it supports offline working. If we are facing internet connectivity issues, it will not affect our work. In Git, we can do almost everything locally. Comparatively, other CVS like SVN is limited and prefer the connection with the central repository.
- Undo Mistakes
  One additional benefit of Git is we can Undo mistakes. Sometimes the undo can be asavior option for us. Git provides the undo option for almost everything.
- Track the Changes
  Git facilitates with some exciting features such as Diff, Log, and Status, which allows usto track changes so we can check the status, compare our files or branches.

5.2 :Why Git?

We have discussed many features and benefits of Git that demonstrate the undoubtedly Git as the leading version control system. Now, we will discuss some other points about why should we choose Git.



- Git Integrity
  Git is developed to ensure the security and integrity of content being version controlled. It uses checksum during transit or tampering with the file system to confirm that information is not lost. Internally it creates a checksum value from the contents of the file and then verifies it when transmitting or storing data.
- Trendy Version Control
  System Git is the most widely used version control system. It has maximum projects among all the version control systems. Due to its amazing workflow and features, it is a preferred choice of developers.
- Everything isLocalAlmost All operations of Git can be performed locally; this is a significant reason for the use of Git. We will not have to ensure internet connectivity.
- Collaborate to Public
  ProjectsThere are many public projects available on the GitHub. We can collaborate on those projects and show our creativity to the world. Many developers are collaborating on public projects. The collaboration allows us to stand with experienced developers and learn a lot from them; thus, it takes our programming skills to the next level.
- Impress Recruiters
  We can impress recruiters by mentioning the Git and GitHub on our resume. Send your GitHub profile link to the HR of the organization you want to join. Show your skills and influence them through your work. It increases the chances of getting hired.

6. ANSIBLE

Ansible is an agentless automation tool that you install on a single host (referred to as the control node). From the control node, Ansible can manage an entire fleet of machines and other devices (referred to as managed nodes) remotely with SSH, Power shell remoting, and numerous other transports, all from a simple command-line interface with no databases or
daemons required.

Control node requirements

For your control node (the machine that runs Ansible), you can use nearly any UNIX-like machine with Python 3.9 or newer installed. This includes Red Hat, Debian, Ubuntu, macOS, BSDs, and Windows under a Windows Subsystem for Linux (WSL) distribution. Windows without WSL is not natively supported as a control node; see Matt Davis blog post for more information.

Managed node requirements

The managed node (the machine that Ansible is managing) does not require Ansible to be installed, but requires Python 2.7, or Python 3.5 - 3.11 to run Ansible library code. The managed node also needs a user account that can SSH to the node with an interactive POSIX shell.

6.1 :Ansible concepts

These concepts are common to all uses of Ansible. We should understand them before using Ansible or reading the documentation.

- Control node
- Managed nodes
- Inventory
- Playbooks
    - Plays
        - Roles
        - Tasks
        - Handlers
- Modules
- Plugins
- Collections
- AAP

Control node

The machine from which you run the Ansible CLI tools ansible-playbook, ansible ansible-vault and others). You can use any computer that meets the software requirements as a control node -laptops, shared desktops, and servers can all run Ansible. Multiple control nodes are possible, but Ansible itself does not coordinate across them, see AAP for such features.

Managed nodes

Also referred to as ‘hosts’, these are the target devices (servers, network appliances or any computer) you aim to manage with Ansible. Ansible is not normally installed on managed nodes, unless you are using ansible-pull but this is rare and not the recommended setup.

Inventory

A list of managed nodes provided by one or more ‘inventory sources’. Your inventory can specify information specific to each node, like IP address. It is also used for assigning groups, that both allow for node selection in the Play and bulk variable assignment. To learn more about inventory, see the Working with Inventory section. Sometimes an inventory source file is also referred to as a ‘hostfile’.

Playbooks

They contain Plays (which are the basic unit of Ansible execution). This is both an ‘execution concept’ and how we describe the files on which `ansible-playbook` operates. Playbooks are written in YAML and are easy to read, write, share and understand. To learn more about playbooks, see Ansible playbooks.

Plays

The main context for Ansible execution, this playbook object maps managed nodes (hosts) to tasks. The Play contains variables, roles and an ordered lists of tasks and can be run repeatedly. It basically consists of an implicit loop over the mapped hosts and tasks and defines how to iterate over them.

Roles

A limited distribution of reusable Ansible content (tasks, handlers, variables, plugins, templates and files) for use inside of a Play. To use any Role resource, the Role itself must be imported into the Play.

Tasks

The definition of an ‘action’ to be applied to the managed host. Tasks must always be contained in a Play, directly or indirectly (Role, or imported/included task list file). You can execute a single task once with an ad hoc command using `ansible` or `ansible-console` (both create a virtual Play).

Handlers

A special form of a Task, that only executes when notified by a previous task which resulted ina ‘changed’ status.

Modules

The code or binaries that Ansible copies to and executes on each managed node (when needed) to accomplish the action defined in each Task. Each module has a particular use, from administering users on a specific type of database to managing VLAN interfaces on a specific type of network device. You can invoke a single module with a task, or invoke several different modules in a playbook. Ansible modules are grouped in collections. For an idea of how many collections Ansible includes, see the Collection Index.

Plugins

Pieces of code that expand Ansible’s core capabilities, they can control how you connect to a managed node (connection plugins), manipulate data (filter plugins) and even control what is displayed in the console (callback plugins). See Working with plugins for details.

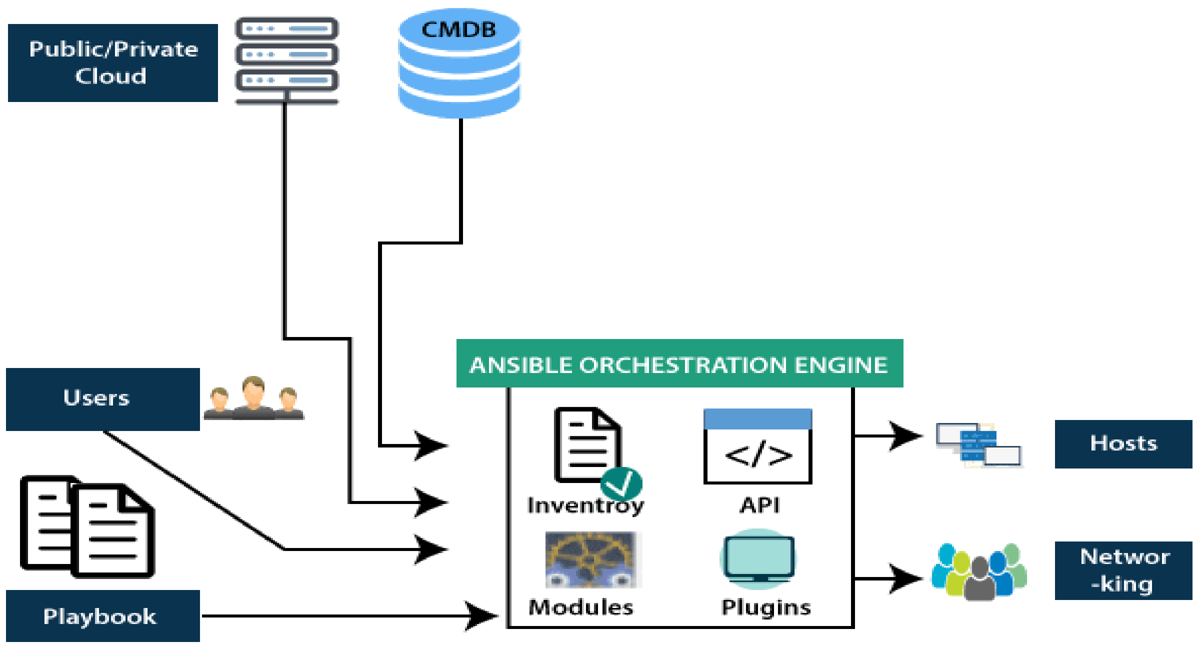
Collections

A format in which Ansible content is distributed that can contain playbooks, roles, modules,and plugins. You can install and use collections through Ansible Galaxy. To learn more about collections, see Using Ansible collections. Collection resources can be used independently and discretely from each other.

AAP

Short for ‘Ansible Automation Platform’. This is a product that includes enterprise levelfeatures and integrates many tools of the Ansible ecosystem: ansible-core, awx, galaxyNG, and so on.

6.2 :Ansible architecture:



Here’s what a plain text inventory file looks like:

[webservers]

Node1 Node2

[db-servers]

Server1

Server2

Once inventory hosts are listed, variables can be assigned to them in simple text files (in asubdirectory called ‘group_vars/’ or ‘host_vars/’) or directly in the inventory file.

Or, as already mentioned, use a dynamic inventory to pull your inventory from data sourceslike EC2, Rackspace, or OpenStack.

6.3:Playbooks

Playbooks can finely orchestrate multiple slices of your infrastructure topology, with very detailed control over how many machines to tackle at a time. This is where Ansible starts to getmost interesting.

Ansible's approach to orchestration is one of finely-tuned simplicity, as we believe your automation code should make perfect sense to you years down the road and there should be very little to remember about special syntax or features.

Simple playbook for installing apache2 service on Debian-

```
---
- name: Apache server installed

hosts: all

Become:yes
tasks:
    - name: Installing apache2 serviceapt:
-name:apache2
state:latest
```

Modules, module utilities, plugins, playbooks, and roles can live in multiple locations. If you write your own code to extend Ansible's core features, you may have multiple files with similaror the same names in different locations on your Ansible control node. The search path determines which of these files Ansible will discover and use on any given playbook run.

Ansible's search path grows incrementally over a run. As Ansible finds each playbook and role included in a given run, it appends any directories related to that playbook or role to the search path. Those directories remain in scope for the duration of the run, even after the playbook or role has finished executing. Ansible loads modules, module utilities, and plugins in this order:

Directories adjacent to a playbook specified on the command line. If you run Ansible withansible-playbook path to .yaml file .Ansible appends these directories if they exist:

- /path/to/modules
- /path/to/module_utils/path/to/plugins

- Selecting an Ansible package and version to install

Ansible’s community packages are distributed in two ways: a minimalist language and runtime package called ansible-core and a much larger “batteries included” package called ansible which adds a community-curated selection of Ansible-collections for automating a wide variety of devices. Choose the package that fits your needs; The following instructions use ansible, but you can substitute ansible-core if you prefer to start with a more minimal package and separately install only the Ansible Collections you require. The ansible or ansible-core packages may be available in your operating systems package manager, and you are free to install these packages with your preferred method. These installation instructions only coverthe officially supported means of installing the python package with pip.

6.3 :Installing and upgrading Ansible

Locating python

Locate and remember the path to the Python interpreter you wish to use to run Ansible. The following instructions refer to this Python as python3, For example, if you’ve determined that you want the Python at /usr/bin/python3 to be the one that you’ll install Ansible under, specify that instead of python3.

Ensuring pip is available

To verify whether pip is already installed for your preferred Python:

$ python3 -m pip -V

If all is well, you should see something like the following.

$ python3 -m pip -V

pip 21.0.1 from /usr/lib/python3.9/site-packages/pip (python 3.9)If so, pip is

available, and you can move on to the next step.

If you see an error like no module pip , you’ll need to install pip under your chosen Python interpreter before proceeding. This may mean installing an additional OS package (for example,python3-pip), or installing the latest pip directly from the Python Packaging Authority by running the following:

```
$ curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

```
$ python3 get-pip.py --user
```

You may need to perform some additional configuration before you are able to run Ansible.See the Python documentation on installing to the user site for more information.

Installing Ansible

Use pip in your selected Python environment to install the Ansible package of your choice forthe current user:

```
$python3 -m pip install --user ansible
```

Alternately, you can install a specific version of `ansible-core` in this Python environment:

```
$ python3 -m pip install --user ansible-core==2.12.3
```

To upgrade an existing Ansible installation in this Python environment to the latest releasedversion, simply add --upgrade to the command above:

```
$ python3 -m pip install --upgrade --user ansible
```

You can test that Ansible is installed correctly by checking the version:

```
$ ansible –version-
```

The version displayed by this command is for the associated ansible version package that hasbeen installed.

To check the version of the ansible package that has been installed:

```
$ Python3 -m pip show ansible
```

Ansible automates tasks on managed nodes or “hosts” in your infrastructure, using a list or group of lists known as inventory. You can pass host names at the command line, but most Ansible users create inventory files. Your inventory defines the managed nodes you automate, with groups so you can run automation tasks on multiple hosts at the same time. Once your inventory is defined, you use patterns to select the hosts or groups you want Ansible to run against.

Ansible Inventory plugins support a range of formats and sources to make your inventory flexible and customizable. As your inventory expands, you may need more than a single file to organize your hosts and groups. Here are three options beyond the `/etc/ansible/hosts` file: - You can create a directory with multiple inventory files. See Organizing inventory in a directory. These can use different formats (YAML, ini, and so on). - You can pull inventory dynamically. For example, you can use a dynamic inventory plugin to list resources in one or more cloud providers. See Working with dynamic inventory. - You can use multiple sources forinventory, including both dynamic inventory and static files. See Passing multiple inventory sources.

Ansible playbooks

Ansible Playbooks offer a repeatable, re-usable, simple configuration management and multi- machine deployment system, one that is well suited to deploying complex applications. If you need to execute a task with Ansible more than once, write a playbook and put it under source control. Then you can use the playbook to push out new configuration or confirm the configuration of remote systems. The playbooks in the ansible-examples repository illustrate many useful techniques. You may want to look at these in another tab as you read the documentation.

Playbooks can:

- declare configurations
- orchestrate steps of any manual ordered process, on multiple sets of machines, in a defined order
- launch tasks synchronously or asynchronously

Playbook execution

A playbook runs in order from top to bottom. Within each play, tasks also run in order from top to bottom. Playbooks with multiple 'plays' can orchestrate multi-machine deployments, running one play on your webservers, then another play on your database servers, then a third play on your network infrastructure, and so on. At a minimum, each play defines two things:

- the managed nodes to target, using a pattern
- at least one task to execute

6.5:Introduction to ad hoc commands

An Ansible ad hoc command uses the /usr/bin/ansible command-line tool to automate a single task on one or more managed nodes. ad hoc commands are quick and easy, but they are not reusable. So why learn about ad hoc commands? ad hoc commands demonstrate the simplicity
and power of Ansible.

```
ansible all -m command/shell -a “uptime”
```

Above command will show uptime of all the node/client mentioned in hosts file(Inventory file).

7. JENKINS(AUTOMATION SERVER)

Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software. Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment technologies.

Jenkins can be installed through native system packages, Docker, or even run standalone byany machine with a Java Runtime Environment (JRE) installed.

Jenkins Installation On Debian-apt-

get update

apt-get upgrade

java -version

INSTALL JAVA if not present. .!apt-

get install default-jre

ADD THE KEY

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -REPO

sudo sh –c ' echo deb http://pkg.jenkins.io/debian-stable binary/
>/etc/apt/sources.list.d/jenkins.list' apt update
apt-get install jenkins

systemctl status jenkins

sudo usermod -a -G root jenkins

7.1:Publish over ssh (plugin used)-

This plugin includes a builder which enables the use of the publisher during the build process. This means that you can send newly created packages to a remote server and install them, start and stop services that the build may depend on and many other use cases.

8. DOCKER(WEB SERVER)

Apache HTTP Server is a free and open-source web server that delivers web content through the internet. It is commonly referred to as Apache and after development, it quickly became the most popular HTTP client on the web. It's widely thought that Apache gets its name from its development history and process of improvement through applied patches and modules but thatwas corrected back in 2000. It was revealed that the name originated from the respect of the Native American tribe for its resiliency and durability.

Why Apache Web Servers?

Apache is considered open source software, which means the original source code is freely available for viewing and collaboration. Being open source has made Apache very popular withdevelopers who have built and configured their own modules to apply specific functionality and improve on its core features. Apache has been around since 1995 and is responsible as a core technology that helped spur the initial growth of the internet in its infancy.

One of the pros of Apache is its ability to handle large amounts of traffic with minimal configuration. It scales with ease and with its modular functionality at its core, you can configure Apache to do what you want, how you want it. You can also remove unwanted modules to make Apache more lightweight and efficient.

Some of the most popular modules that can be added are SSL, Server Side Programming Support (PHP), and Load Balancing configs to handle large amounts of traffic. Apache can alsobe deployed on Linux, MacOS, and Windows. If you learn how to configure Apache on Linux, you can administer Apache on Windows and Mac. The only difference would be directory paths and installation processes.

Features of Apache Web Server

- Handling of static files
- Loadable dynamic modules
- Auto-indexing
- .htaccess
- Compatible with IPv6
- Supports HTTP/2
- FTP connections
- Gzip compression and decompression
- Bandwidth throttling
- Perl, PHP, Lua scripts
- Load balancing
- Session tracking
- URL rewriting
- Geolocation based on IP address

How does Apache Web Server Work?

Apache functions as a way to communicate over networks from client to server using the TCP/IP protocol. Apache can be used for a wide variety of protocols, but the most common is HTTP/S. HTTP/S or Hyper Text Transfer Protocol (S stands for Secure) is one of the main protocols on the web, and the one protocol Apache is most known for.

HTTP/S is used to define how messages are formatted and transmitted across the web, with instructions for browsers and servers on how to respond to various requests and commands. Hypertext Transfer Protocol Secure is usually through port 443 with the unsecured protocol being through port 80.

The Apache server is configured via config files in which modules are used to control its behavior. By default, Apache listens to the IP addresses configured in its config files that are being requested. This is where one of Apaches many strengths come into play.

With the Listen directive, Apache can accept and route specific traffic to certain ports and domains based on specific address-port combination requests. By default, Listen runs on port 80 but Apache can be bound to different ports for different domains, allowing for many different websites and domains to be hosted and a single server. You can have domain1.com listening on port 80, domain2.com on port 8080 and domain3.com on port 443 using HTTPS allon Apache.

Once a message reaches its destination or recipient, it sends a notice, or ACK message, basically giving acknowledgment to the original sender that their data has successfully arrived. If there’s an error in receiving data, or some packets were lost in transit, the destination host or client sends a Not Acknowledged, or NAK message, to inform the sender that the data needs to be retransmitted.

Who Uses Apache Web Server?

Apache HTTP web servers are used by over 67% of all web servers in the world. Apache web servers are easy to customize environments, they’re fast, reliable, and highly secure. This makes Apache web servers a common choice by best-in-class companies.

Alternatives for Apache HTTP Server

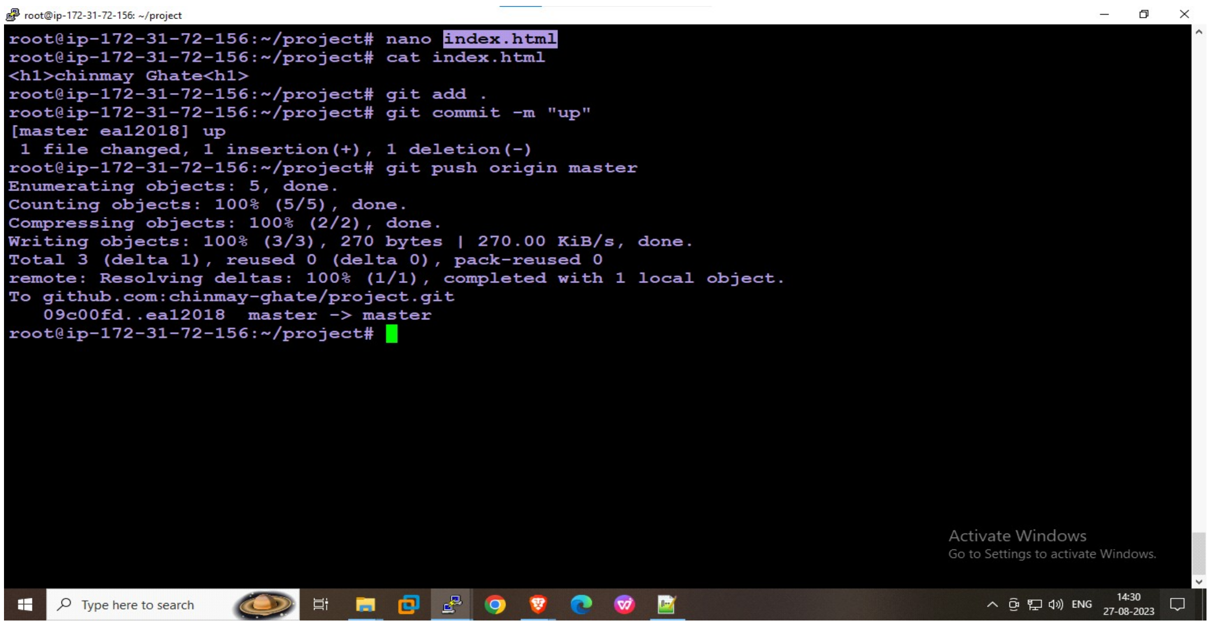
While Apache web servers are very popular, they’re not the only web servers on the market. Below are a number of alternatives for Apache HTTP servers.

- Nginx
- Apache Tomcat
- Node.js
- Lighttpd

9. SCREENSHOTS

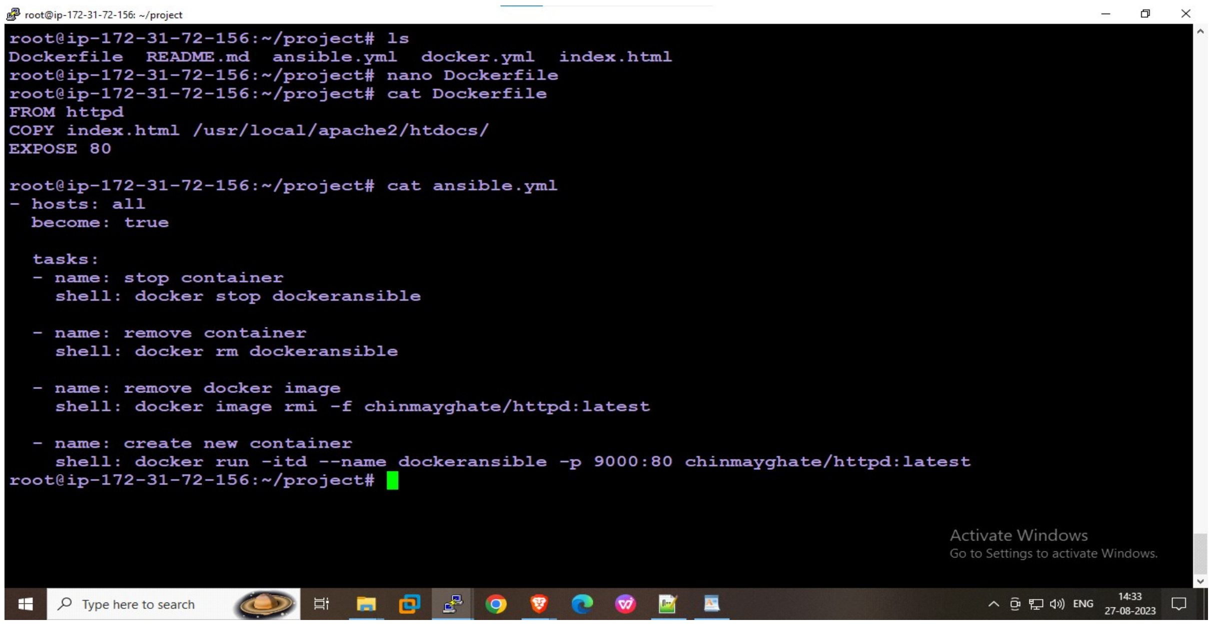
Git Machine (pvt ip =172.31.72.156)

index.html,add,commit,push



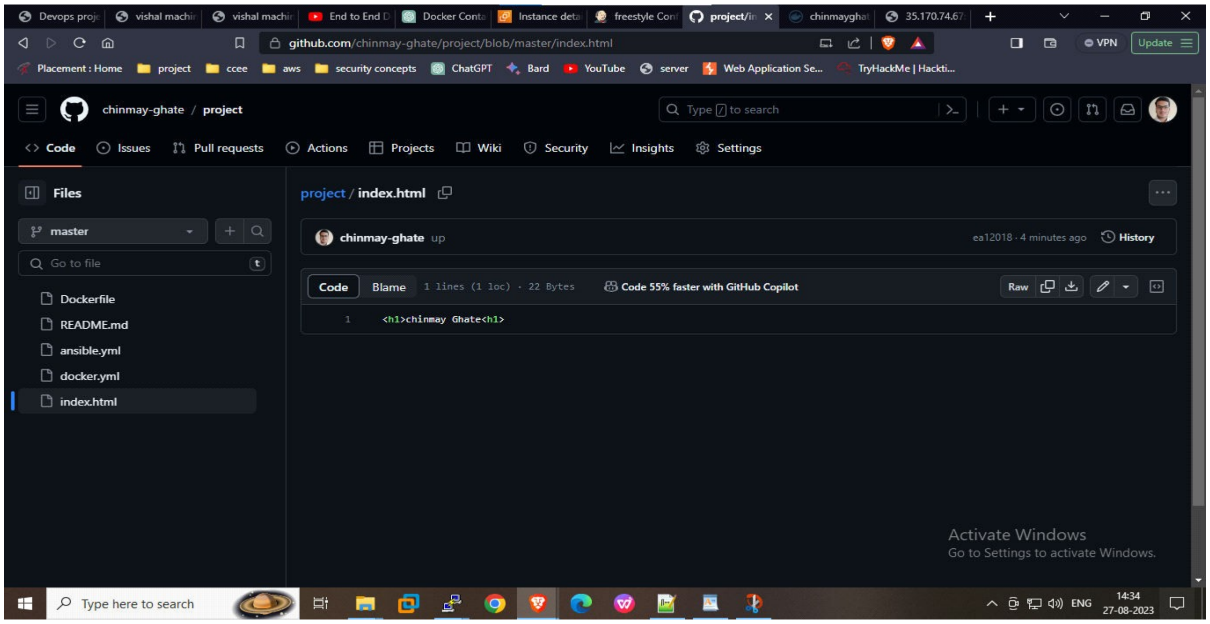
```
root@ip-172-31-72-156:~/project# nano index.html
root@ip-172-31-72-156:~/project# cat index.html
<h1>chinmay Ghate<h1>
root@ip-172-31-72-156:~/project# git add .
root@ip-172-31-72-156:~/project# git commit -m "up"
[master ea12018] up
 1 file changed, 1 insertion(+), 1 deletion(-)
root@ip-172-31-72-156:~/project# git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 270 bytes | 270.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:chinmay-ghate/project.git
   09c00fd..ea12018  master -> master
root@ip-172-31-72-156:~/project#
```

Dockerfile,YAML file

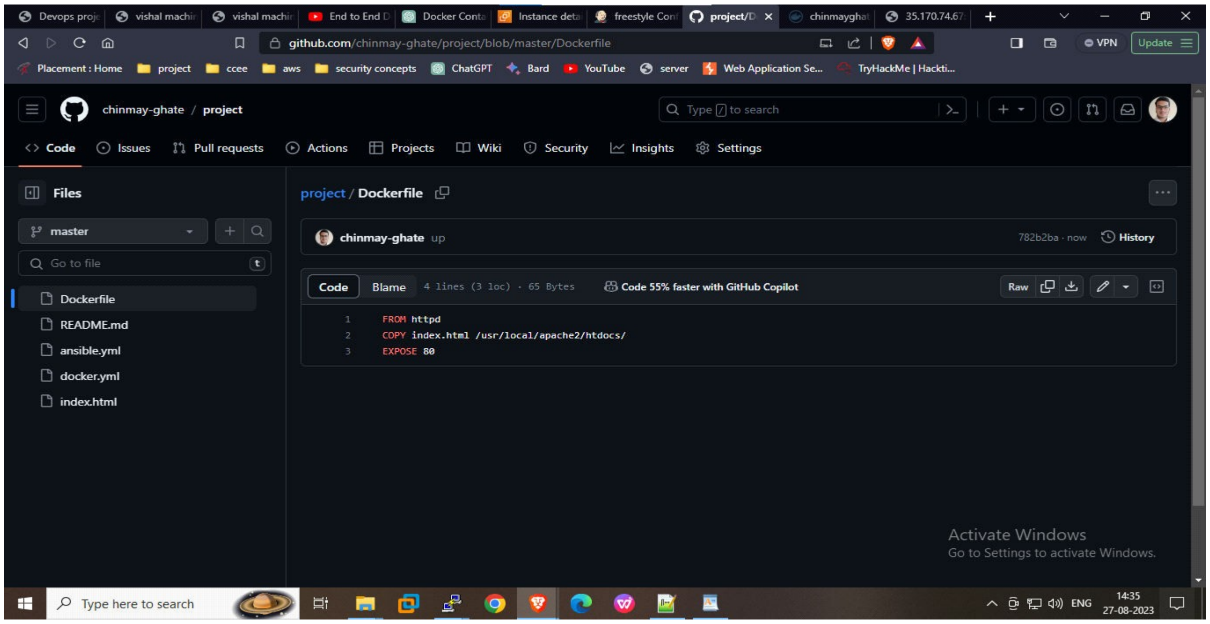


```
root@ip-172-31-72-156:~/project# ls
Dockerfile  README.md  ansible.yml  docker.yml  index.html
root@ip-172-31-72-156:~/project# nano Dockerfile
root@ip-172-31-72-156:~/project# cat Dockerfile
FROM httpd
COPY index.html /usr/local/apache2/htdocs/
EXPOSE 80

root@ip-172-31-72-156:~/project# cat ansible.yml
- hosts: all
  become: true

  tasks:
  - name: stop container
    shell: docker stop dockeransible

  - name: remove container
    shell: docker rm dockeransible

  - name: remove docker image
    shell: docker image rmi -f chinmayghate/httpd:latest

  - name: create new container
    shell: docker run -itd --name dockeransible -p 9000:80 chinmayghate/httpd:latest
root@ip-172-31-72-156:~/project#
```

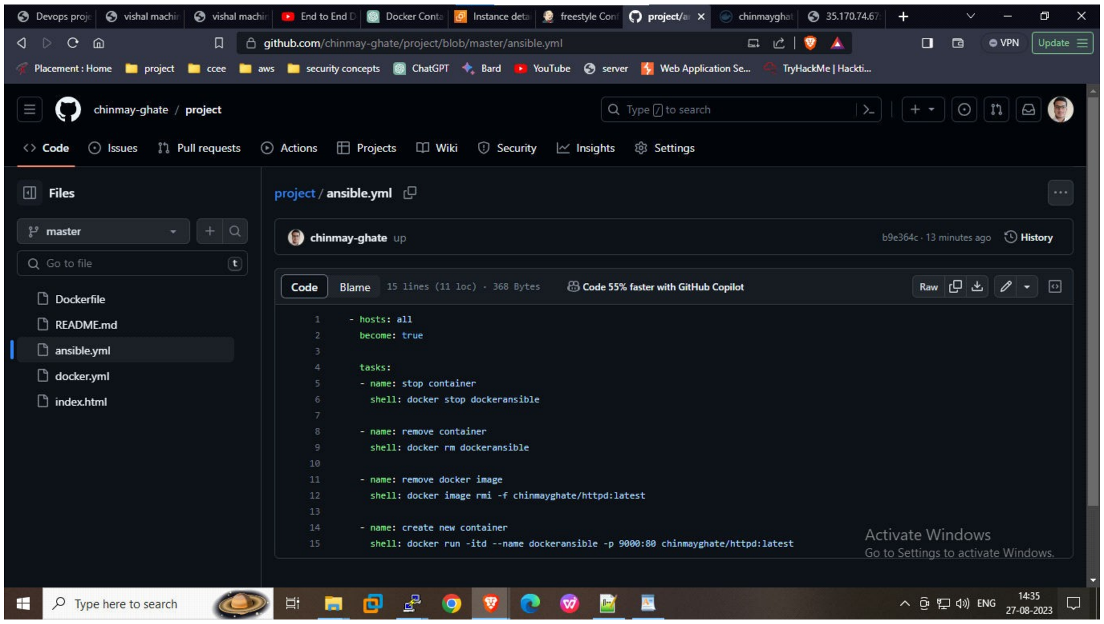
Github Repo


github.com/chinmay-ghate/project/blob/master/index.html
chinmay-ghate / project
Code Issues Pull requests Actions Projects Wiki Security Insights Settings
Files
master
Dockerfile
README.md
ansible.yml
docker.yml
index.html
project / index.html
chinmay-ghate up
ea12018 · 4 minutes ago
History
1 lines (1 loc) · 22 Bytes

```
<h1>chinmay Ghate<h1>
```



```
FROM httpd
COPY index.html /usr/local/apache2/htdocs/
EXPOSE 80
```



github.com/chinmay-ghate/project/blob/master/ansible.yml

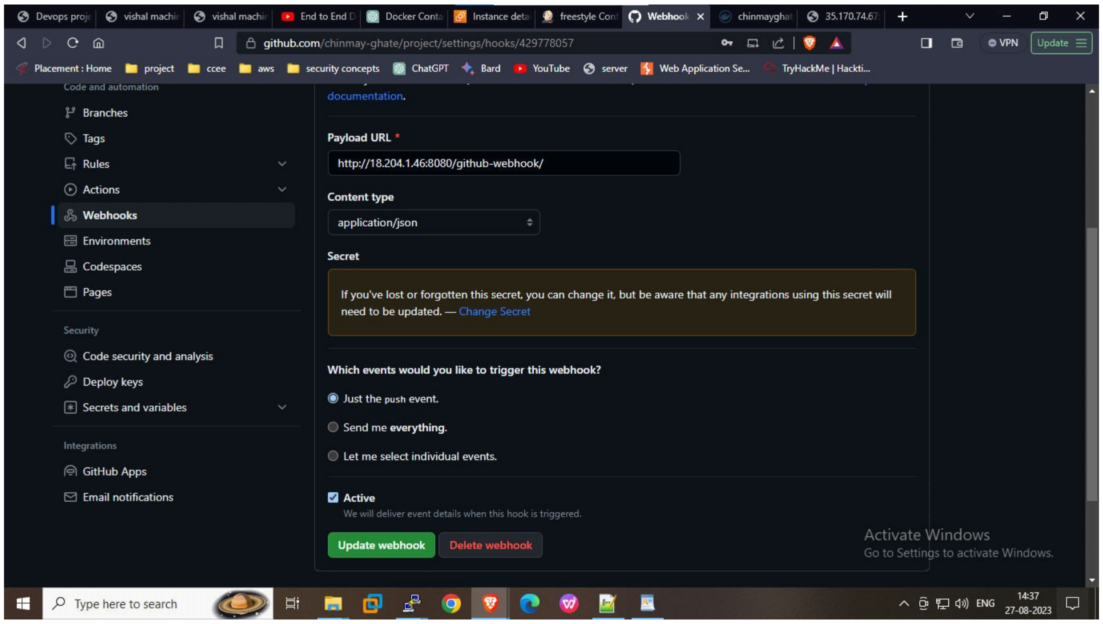
project / ansible.yml

chinmay-ghate up — b9e364c · 13 minutes ago

15 lines (11 loc) · 368 Bytes

```
- hosts: all
  become: true

  tasks:
  - name: stop container
    shell: docker stop dockeransible

  - name: remove container
    shell: docker rm dockeransible

  - name: remove docker image
    shell: docker image rmi -f chinmayghate/httpd:latest

  - name: create new container
    shell: docker run -itd --name dockeransible -p 9000:80 chinmayghate/httpd:latest
```

Webhook



github.com/chinmay-ghate/project/settings/hooks/429778057

Payload URL *

http://18.204.1.46:8080/github-webhook/

Content type

application/json

Secret

If you've lost or forgotten this secret, you can change it, but be aware that any integrations using this secret will need to be updated. — Change Secret

Which events would you like to trigger this webhook?

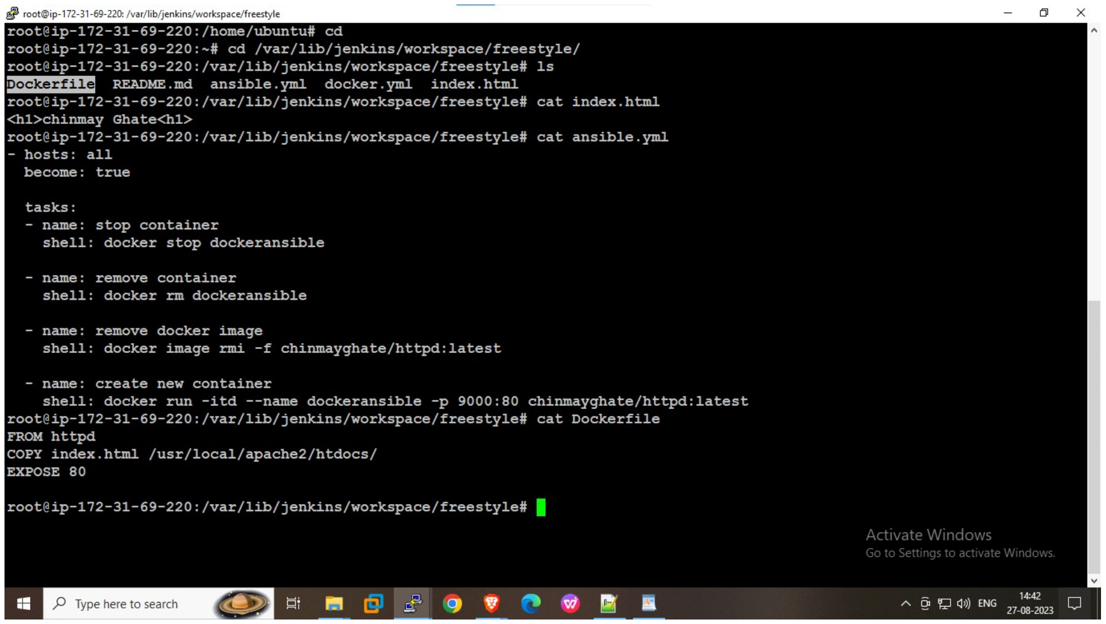
- [✓ Just the push event.]
- Send me everything.
- Let me select individual events.

[✓] Active

We will deliver event details when this hook is triggered.

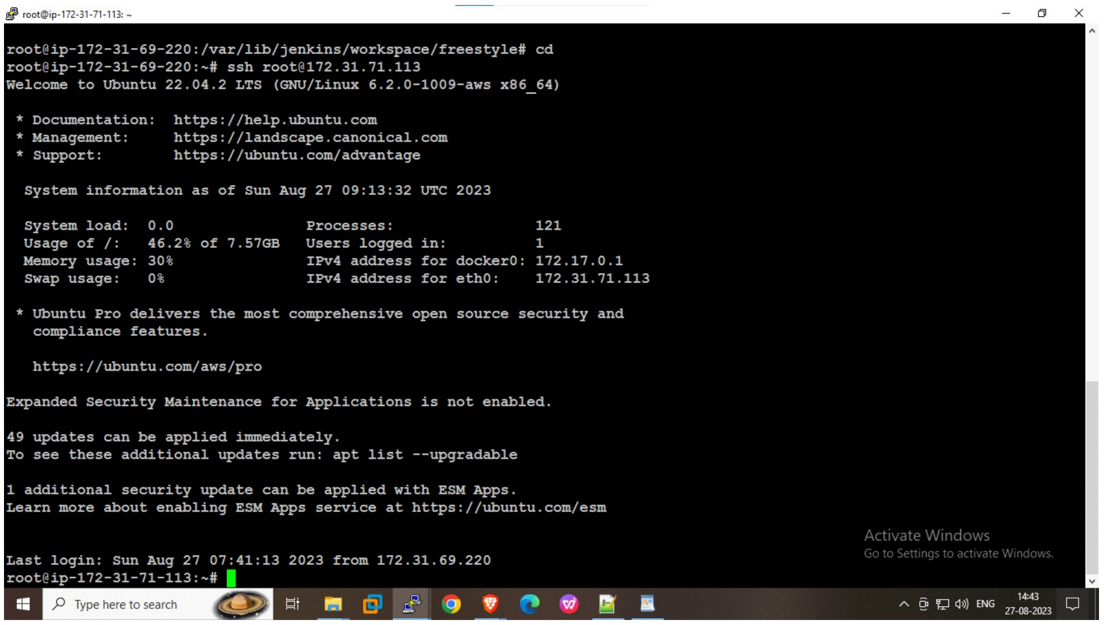
Update webhook   Delete webhook

Jenkins m/c (pvt ip=172.31.69.220)

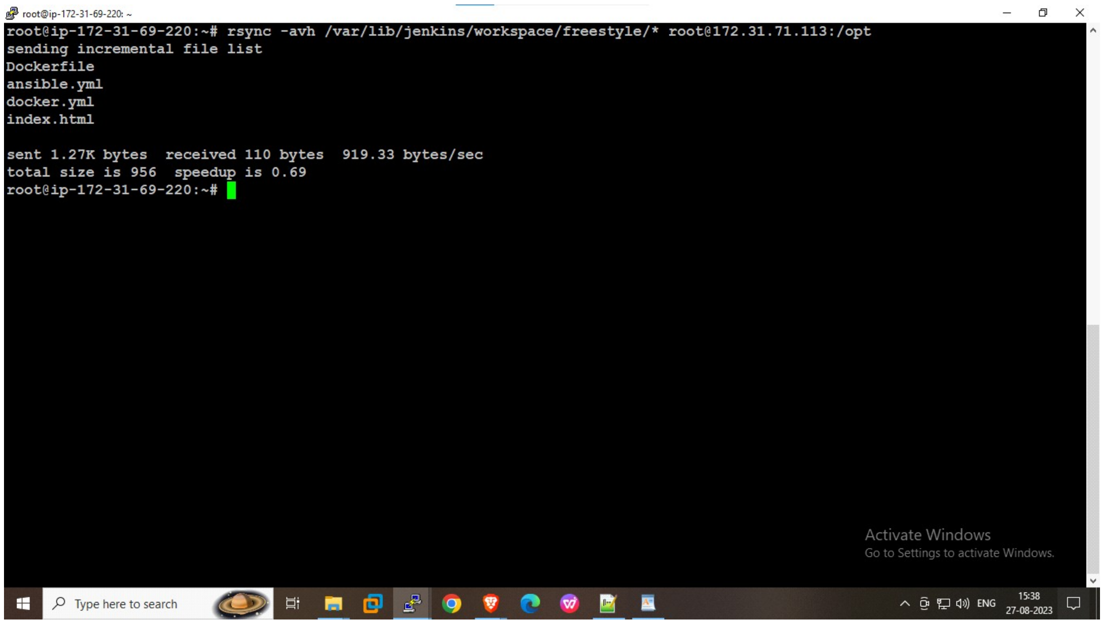

root@ip-172-31-69-220: /var/lib/jenkins/workspace/freestyle
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
14:42
27-08-2023

```
root@ip-172-31-69-220:/home/ubuntu# cd
root@ip-172-31-69-220:~# cd /var/lib/jenkins/workspace/freestyle/
root@ip-172-31-69-220:/var/lib/jenkins/workspace/freestyle# ls
Dockerfile  README.md  ansible.yml  docker.yml  index.html
root@ip-172-31-69-220:/var/lib/jenkins/workspace/freestyle# cat index.html
<h1>chinmay Ghate<h1>
root@ip-172-31-69-220:/var/lib/jenkins/workspace/freestyle# cat ansible.yml
- hosts: all
  become: true

  tasks:
  - name: stop container
    shell: docker stop dockeransible

  - name: remove container
    shell: docker rm dockeransible

  - name: remove docker image
    shell: docker image rmi -f chinmayghate/httpd:latest

  - name: create new container
    shell: docker run -itd --name dockeransible -p 9000:80 chinmayghate/httpd:latest
root@ip-172-31-69-220:/var/lib/jenkins/workspace/freestyle# cat Dockerfile
FROM httpd
COPY index.html /usr/local/apache2/htdocs/
EXPOSE 80

root@ip-172-31-69-220:/var/lib/jenkins/workspace/freestyle#
```

SSH to ansible from jenkins



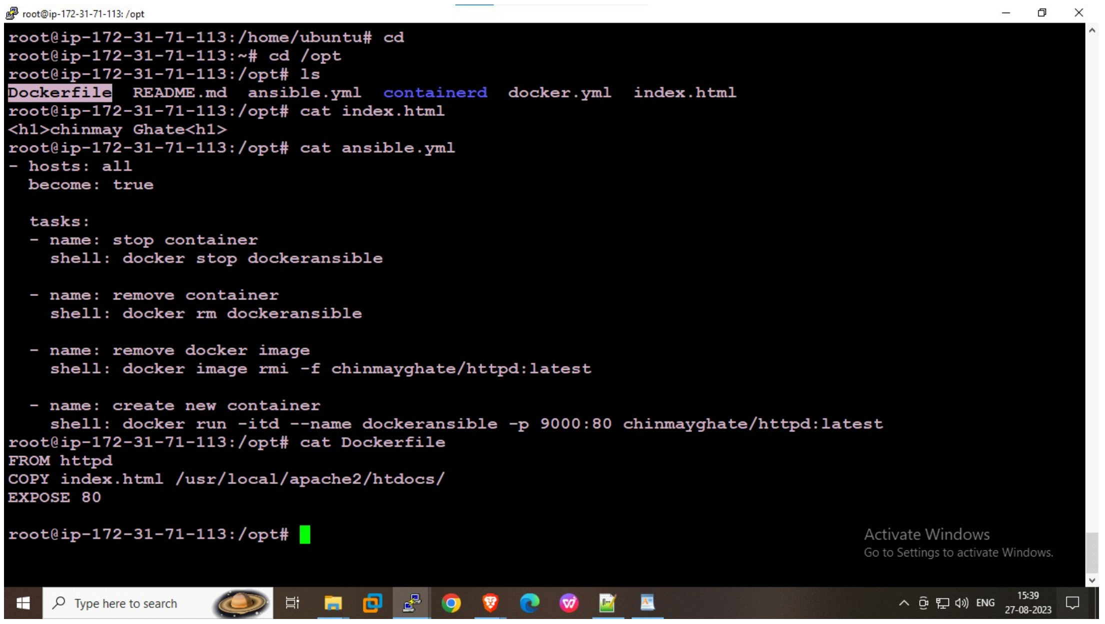
```
root@ip-172-31-69-220:/var/lib/jenkins/workspace/freestyle# cd
root@ip-172-31-69-220:~# ssh root@172.31.71.113
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 6.2.0-1009-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Aug 27 09:13:32 UTC 2023

  System load:  0.0               Processes:                121
  Usage of /:   46.2% of 7.57GB   Users logged in:          1
  Memory usage: 30%               IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                IPv4 address for eth0:    172.31.71.113

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

49 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


Last login: Sun Aug 27 07:41:13 2023 from 172.31.69.220
root@ip-172-31-71-113:~#
```



```
root@ip-172-31-69-220:~# rsync -avh /var/lib/jenkins/workspace/freestyle/* root@172.31.71.113:/opt
sending incremental file list
Dockerfile
ansible.yml
docker.yml
index.html

sent 1.27K bytes  received 110 bytes  919.33 bytes/sec
total size is 956  speedup is 0.69
root@ip-172-31-69-220:~#
```

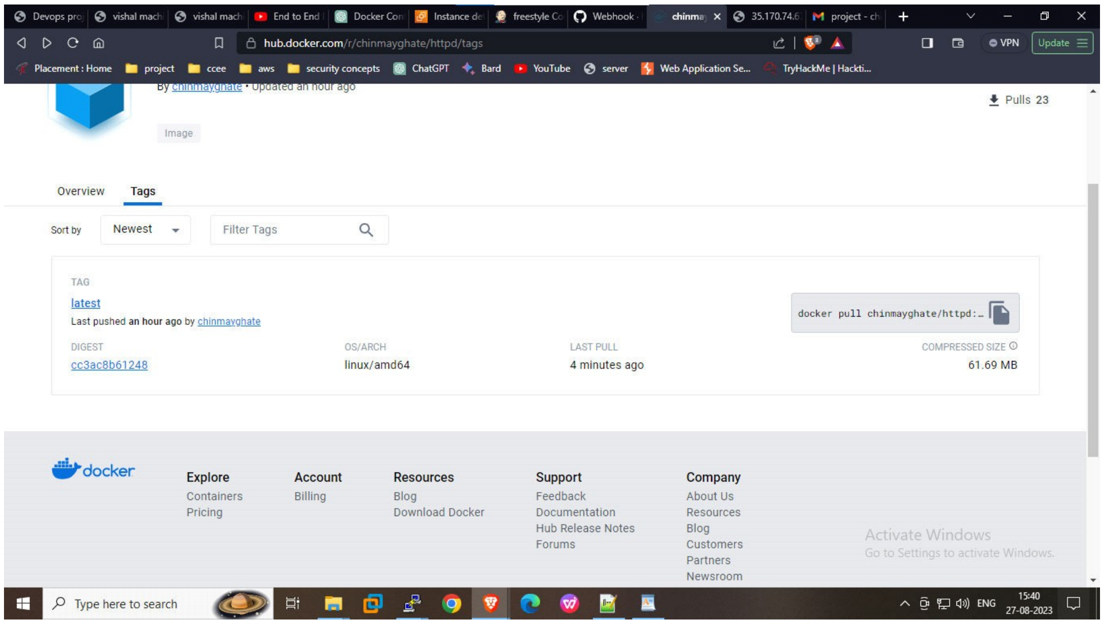
Ansible m/c (pvt ip=172.31.71.113)

Dockerfile,YAML file

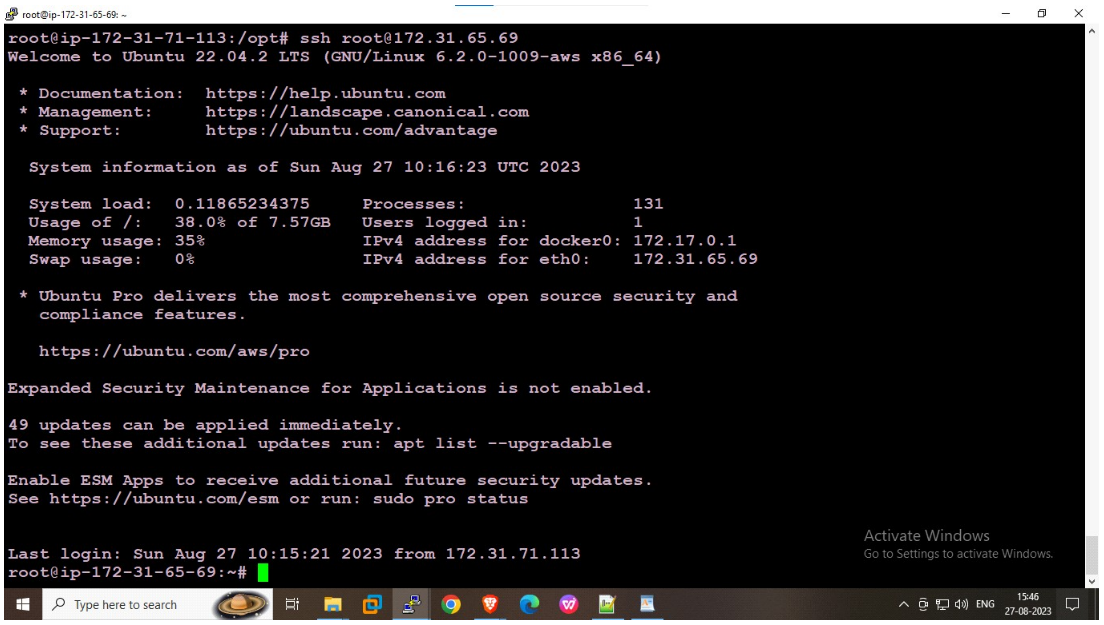


```
root@ip-172-31-71-113:/home/ubuntu# cd
root@ip-172-31-71-113:~# cd /opt
root@ip-172-31-71-113:/opt# ls
Dockerfile  README.md  ansible.yml  containerd  docker.yml  index.html
root@ip-172-31-71-113:/opt# cat index.html
<h1>chinmay Ghate<h1>
root@ip-172-31-71-113:/opt# cat ansible.yml
- hosts: all
  become: true

  tasks:
  - name: stop container
    shell: docker stop dockeransible

  - name: remove container
    shell: docker rm dockeransible

  - name: remove docker image
    shell: docker image rmi -f chinmayghate/httpd:latest

  - name: create new container
    shell: docker run -itd --name dockeransible -p 9000:80 chinmayghate/httpd:latest
root@ip-172-31-71-113:/opt# cat Dockerfile
FROM httpd
COPY index.html /usr/local/apache2/htdocs/
EXPOSE 80

root@ip-172-31-71-113:/opt#
```

DOCKERHUB (push and push image)


hub.docker.com/r/chinmayghate/httpd/tags
By chinmayghate • Updated an hour ago
Pulls 23
Overview
Tags
Sort by Newest
Filter Tags
TAG
latest
Last pushed an hour ago by chinmayghate
docker pull chinmayghate/httpd:...
DIGEST cc3ac8b61248
OS/ARCH linux/amd64
LAST PULL 4 minutes ago
COMPRESSED SIZE 61.69 MB

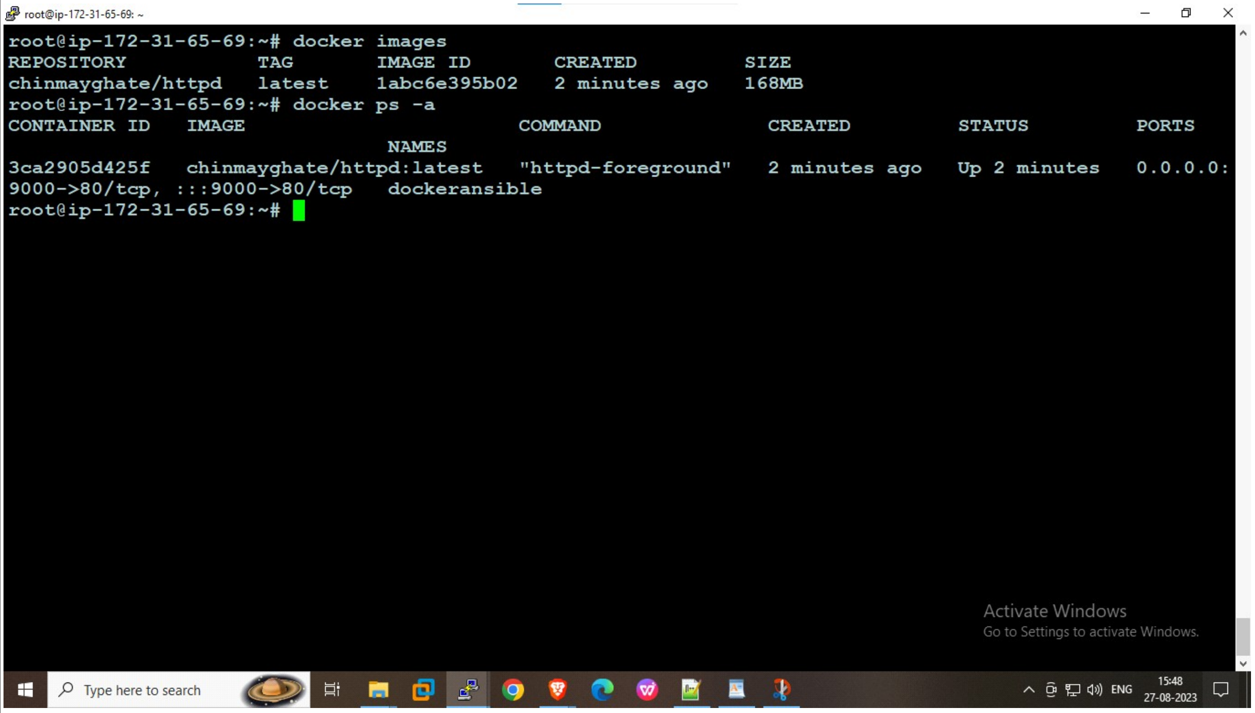
Ansible to docker ssh



```
root@ip-172-31-71-113:/opt# ssh root@172.31.65.69
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 6.2.0-1009-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Aug 27 10:16:23 UTC 2023

  System load:  0.11865234375     Processes:                131
  Usage of /:   38.0% of 7.57GB   Users logged in:          1
  Memory usage: 35%               IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                IPv4 address for eth0:    172.31.65.69

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

49 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Sun Aug 27 10:15:21 2023 from 172.31.71.113
root@ip-172-31-65-69:~#
```

Docker (Web server) m/c (pvt ip =172.31.65.69)

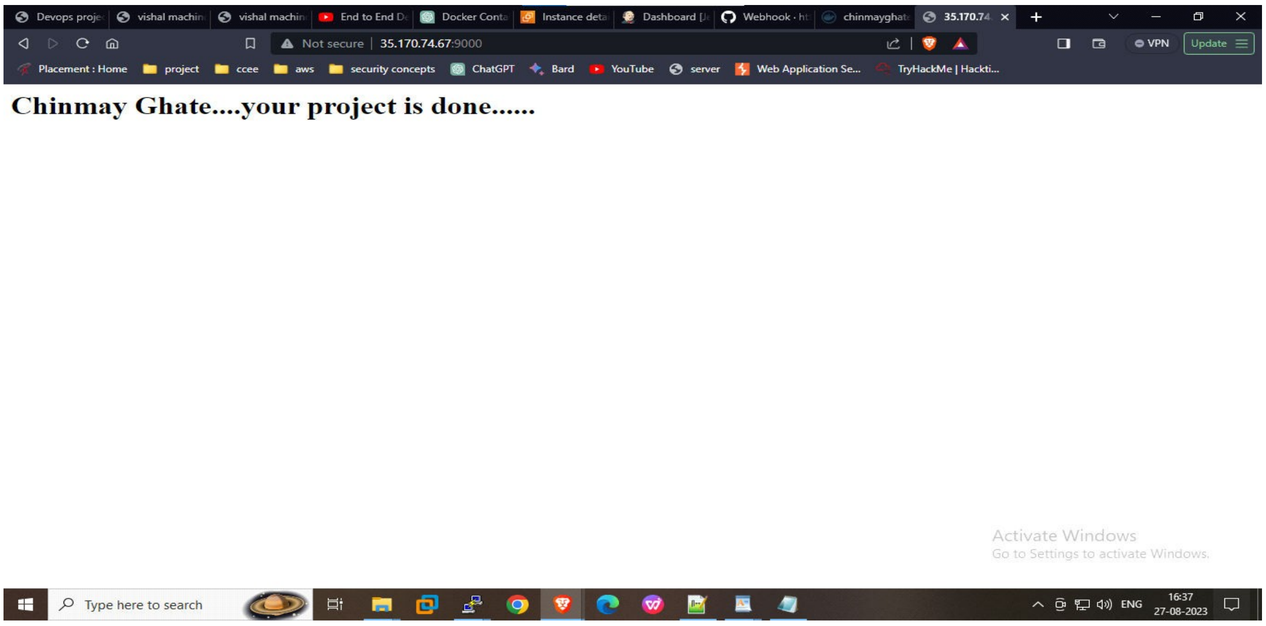
new image and container



root@ip-172-31-65-69: ~

```
root@ip-172-31-65-69:~# docker images
REPOSITORY           TAG       IMAGE ID       CREATED         SIZE
chinmayghate/httpd   latest    1abc6e395b02   2 minutes ago   168MB
root@ip-172-31-65-69:~# docker ps -a
CONTAINER ID   IMAGE                         COMMAND              CREATED         STATUS         PORTS
                                NAMES
3ca2905d425f   chinmayghate/httpd:latest   "httpd-foreground"   2 minutes ago   Up 2 minutes   0.0.0.0:
9000->80/tcp, :::9000->80/tcp   dockeransible
root@ip-172-31-65-69:~#
```

Final Output [(pub ip of dock):9000]



Not secure | 35.170.74.67:9000

Chinmay Ghate....your project is done......

10.CONCLUSION

The goal of this project is to gain the knowledge and skills needed for the working in devops field and CI/CD allows organizations to ship software quickly and efficiently. CI/CD facilitates an effective process for getting products to market faster than ever before, continuously delivering code into production, and ensuring an ongoing flow of new features and bug fixes via the most efficient delivery method.Continuous integration and continuous delivery provide an ideal scenario for any organization's application teams. Your developers simply push code to a repository. This code will be integrated, tested, deployed, tested again, merged with infrastructure, go through security and quality reviews, and be ready to deploy with extremely high confidence.When CI/CD is used, code quality is improved and software updates are delivered quickly and with high confidence that there will be no breaking changes. The impact of any release can be correlated with data from production and operations. It can be used for planning the next cycle, too—a vital DevOps practice in your organization's cloud transformation.

11.REFERENCE

- *https://www.javatpoint.com/git*
- *https://www.youtube.com/watch?v=Ri-URt8gPCk*
- *https://www.jenkins.io/doc/*
- *https://www.ansible.com/*
- *https://plugins.jenkins.io/publish-over-ssh/*
- *https://www.youtube.com/watch?v=nplH3BzKHPk*