DESIGN AND DEVELOPMENT OF VOICE AUTOMATED INFRASTRUCTURE MANAGEMENT

By

Astitva Singh (1601410027)

Mohd. Bilal (1601410065)

Rishabh Umrao (1601410086)

Submitted to the Department of Computer Science & Engineering

in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

in

Computer Science & Engineering

Department of Computer Science and Engineering

Shri Ram Murti Smarak College of Engineering & Technology, Bareilly

Dr A.P.J Abdul Kalam Technical University, Lucknow

August, 2020

DESIGN AND DEVELOPMENT OF VOICE AUTOMATED INFRASTRUCTURE MANAGEMENT

By

Astitva Singh (1601410027)

Mohd. Bilal (1601410065)

Rishabh Umrao (1601410086)

Submitted to the Department of Computer Science & Engineering

in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

in

Computer Science & Engineering

Department of Computer Science and Engineering

Shri Ram Murti Smarak College of Engineering & Technology,

Bareilly

Dr A.P.J Abdul Kalam Technical University, Lucknow

August, 2020

Table of Contents

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature…………………………………	*Signature………………………………*
Name……………………………………..	*Name…………………………………..*
Roll No…………………………………..	*Roll No…………………………………*
Date……………………………………..	*Date…………………………… ……..*

Signature…………………………………

Name……………………………………..

Roll No…………………………………..

Date……………………………………..

CERTIFICATE

This is to certify that the Project Report entitled "Design and Development of Voice Automated Infrastructure Management" which is submitted by ***Astitva Singh (1601410027), Mohd. Bilal (1601410065) and Rishabh Umrao (1601410086)*** *is a record of the candidates own work carried out by them under my supervision. The matter embodied in this work is original and has not been submitted for the award of any other work or degree.*

Dr. L. S. Maurya	*Mr. Durgesh Tripathi*	*Mr. Shahjahan Ali*
HOD (CSE/IT)	***PROJECT INCHARGE***	***SUPERVISOR***

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Assistant Professor Mr. Shahjahan Ali, Computer Science and Engineering, S.R.M.S.C.E.T, Bareilly for his constant support and guidance throughout the course of our work. Her sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only her cognizant efforts that our endeavours have seen light of the day.

We also take the opportunity to acknowledge the contribution of Dr. L. S. Maurya, Head, Department of Computer Science & Engineering, S.R.M.S.C.E.T, Bareilly for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature…………………………………	*Signature………………………………*
Name……………………………………	*Name…………………………………*
Roll No…………………………………	*Roll No…………………………………*
Date……………………………………	*Date……………………………………….*

Signature………………………………

Name……………………………………

Roll No…………………………………

Date……………………………………

ABSTRACT

VoiceAIM (formerly known as VAIM) is an advance automated system that can be useful to employees and the managers in any functional organization. VoiceAIM gives the facility to automate the complete infrastructure using voice commands and some automation scripts.

This tools helps employees and managers to check the health, automate the infrastructure and much more using a single application and some voice commands.

This project aims at maximum automation of the infrastructure without a prior knowledge of underlying technologies and operating things with the help of pre built scripts. One needs nothing more than a mobile application to send some voice commands and everything will be automated and provisioned. This reduces the tasks of Operations Team in a company and gives a lot of control to managerial teams to check, view and query the current infrastrucure status and plans for upcoming designs.

CHAPTER 1- INTRODUCTION

VoiceAIM (formerly known as VAIM) is an advance automated system that can be useful to employees and the managers in any functional organization. VoiceAIM gives the facility to automate the complete infrastructure using voice commands and some automation scripts.

Many tools are there alread to automate the infrastructure but they all needs to be scripted and some prior knowledge about the tools is required. VoiceAIM aims to achieve a better user interface to manage and control the projects using the combination of such many free and opensource tools along with the capability to connect to any other third party tool to extend its features.

1.1 Purpose

It is the need of future to reduce the stress of managing the scripts and logging into systems to check the status or provision complete infrastructure. With this projet we acn easily manage the scripts with github or other SCM and can provision any of the projects with a bunch of voice command that can leverage the power of many underlying technologies.

This can help any one to check for the health and status of the infrastructure and maintain it easily with a mobile app interface.

1.2 Existing System

For any infrastructure to get up and running we need operations team to be active 24x7. This is important because of the remotely located teams and rapidly increasing demands of other development teams for new servers and changes in the configurations.

In this type of system, It becomes very difficult to manage and being available to make required changes and deploy the latest changes to the infrastructure on demand and make things ready for testing and production as soon as possible.

This leads to increase in the workload, stress and discomfort among the team members. And eventually a bad and unsatisfying project and team management.

1.3 Proposed System

Our new system is using many devops tools and agile principles to achieve the most automation and a fast and easy way of deployment (from testing to production). This approach creates, destroys, monitor, health checks and many more using a single mobile application that can accept many voice commands to perform operations for their existing and new projects. It will help you to take your projects to any public, private or hybrid cloud using some voice commands within minutes.

This uses many prebuilt scripts according to the need of the project, hence it is highly customisable, repeatable and portable.

Also the prebuilt scipts help us to maintain versions of our infrastructure and the configurations. So if our current changes are not satisfyable then we can easily roll back to previous infrastructure without making much efforts.

The most advantageous feature of this system is that anyone with the least knowledge of underlying technology can operate the complete infrastructure with some basic voice commands and a fully up and running project can be given to the company clients withing seconds or minutes (depending upon the infrastructure size and complexity).

CHAPTER 2 - LITERATURE SURVEY

VoiceAIM (formerly known as VAIM) is a solution to reduce the stress of having the knowledge about all the tools and technologies for the automation of infrastructure thus providing a simple tool that aims at providing a better user interface to manage and control the projects using the Voice Commands.

The VoiceAIM can dramatically increase the productivity and availability as a less knowledgeable person with the least knowledge of underlying technology can operate the complete infrastructure with some basic voice commands. Beside this having a mobile app to control the Infrastructure also provides us with the ease to manage cloud and local systems On The Fly from any remote location.

The System uses the prebuilt scripts that are customizable according to the need of the project. This will help in maintaining the version of our System and with each customization in script or in configuration a new version will be rolled out. Since every System once or ever deals with the faults and errors and if this happens with our System then this Version Setup will help us to roll us back to the previous version without having any much efforts and worries.

The project aims at reducing the task of the operations team in a company as one needs nothing more than a mobile application to send some voice commands to provision and automate the infrastructure. A major question that will arise in anyone’s mind does this have any impact on Operations Teams Job. The answer is simply No it won’t affect the jobs but aims mainly on reducing the workload of those guys. After all it will be those guys only who will provision and manage the Infrastructure and will configure the scripts, roll out the new versions of the System.

CHAPTER 3 - SOFTWARE AND HARDWARE REQUIREMENTS

2.1 Software Requirements

A set of programs associated with the operation of a computer is called software. Software is the part of the computer system which enables the user to interact with several physical hardware devices.

The minimum software requirement specifications for developing this project are as follows:

Designing frontend	:	Flutter Mobile Framework
Middleware	:	Python3.6 +
Backend	:	Hybrid cloud Platforms
Scripting	:	Ansible and Terraform
Web Server	:	Flask CGI
OS	:	RHEL7+ or CentOS7+ (tested)

2.2 Hardware Requirement Specification

The Collection of internal electronic circuits and external physical devices used in building a computer is called Hardware.

The minimum hardware requirement specification for developing this project is as follows:

Processor	:	Pentium IV
RAM	:	512MB RAM
Hard Disk	:	10 GB

CHAPTER 4 - SOFTWARE REQUIREMENTS ANALYSIS

4.1 Overview

Main focus of deployment phase of software delivery is on "What should be our production environment and how to arrange it". But most of our time goes on managing the keys and replicating the same environment and configuring it to make it work and test for performance and reliability. There should be some tool that can easy up the management of multiple resources, templates, scripts, etc and allows anyone to use those without having any much technical knowledge about the working of underlying technology.

4.2 Problem Description

One has to take great care to replicate the whole infrastructure and install all the required packages/software to make the systems ready for production environment. In existing system, most of these things are done manually by some technical experts. Also, in this era of rapidly increasing cloud migrations by several companies there is a huge gap in the technical requirement. Hence the current technical experts have a lot of burden to complete the tasks in as least time as possible. This can lead to overburden, stress, and poor quality of work. Along with that, this leads to a time consuming and error prone configurations sometimes.

4.3 Solution

VoiceAIM is designed to target most of the above-mentioned problems. This is a system that can leverage the power of multiple Infrastructure as Code tools (like Terraform) and Configuration management tools (like Ansible) to manage infrastructure and their configurations using some prebuilt templates.

This also keeps track of all the changes made so that they do not make the same changes again. And with the help of voice commands one can invoke these templates to get the basic structure ready for deployment.

This tool not only reduces the efforts by using the prebuild templates but also uses the power of voice commands to invoke these scripts. So, the user only needs to invoke a voice command and all the configuration and infrastructure will be ready in few minutes. The most advantage of this tool is that the user needs not to know or worry about the underlying commands and technologies used. This is a one-time development that requires some level of technical experts and the same templates can be used in future as many times as we want.

4.4 Modules

- Configuration Templates
- Infrastructure Templates
- Listener API
- Extra modules

4.5 Module Description

- **Configuration Templates** are the templates that are used by the extra modules to configure the target system. These uses ansible templates to configure any type of OS supported by ansible.
- **Infrastructure Templates** are the templates that are used by extra modules to create a new infrastructure on many different providers like AWS, GCP, Azure, Digital Ocean, Kubernetes, etc. These uses Terraform scripts to achieve the goal.
- **Listener API** is the API that listens to the voice commands. It extracts the meaningful information from the invoked voice command and then performs a desired task with desired parameters. Flask API is used for this.
- **Extra Modules** are the pluggable modules that are small and compact. Desired to achieve a specific task (hence modular). These are designed and a new rule is added to the Listener API to make system aware about the newly added modules.

CHAPTER 5 – SYSTEM ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system.

5.1 Basic Application Architecture

Software installed in one computer, for example – Installing s/w of a Calculator, Adobe Photoshop, MS Office, AutoCAD for faster in terms of management and access. Secured from Data hacking and virus. In the basic application architecture, there is only a single listener API and all the tools are installed in the same machine. This is good for development, testing and learning of the tool.

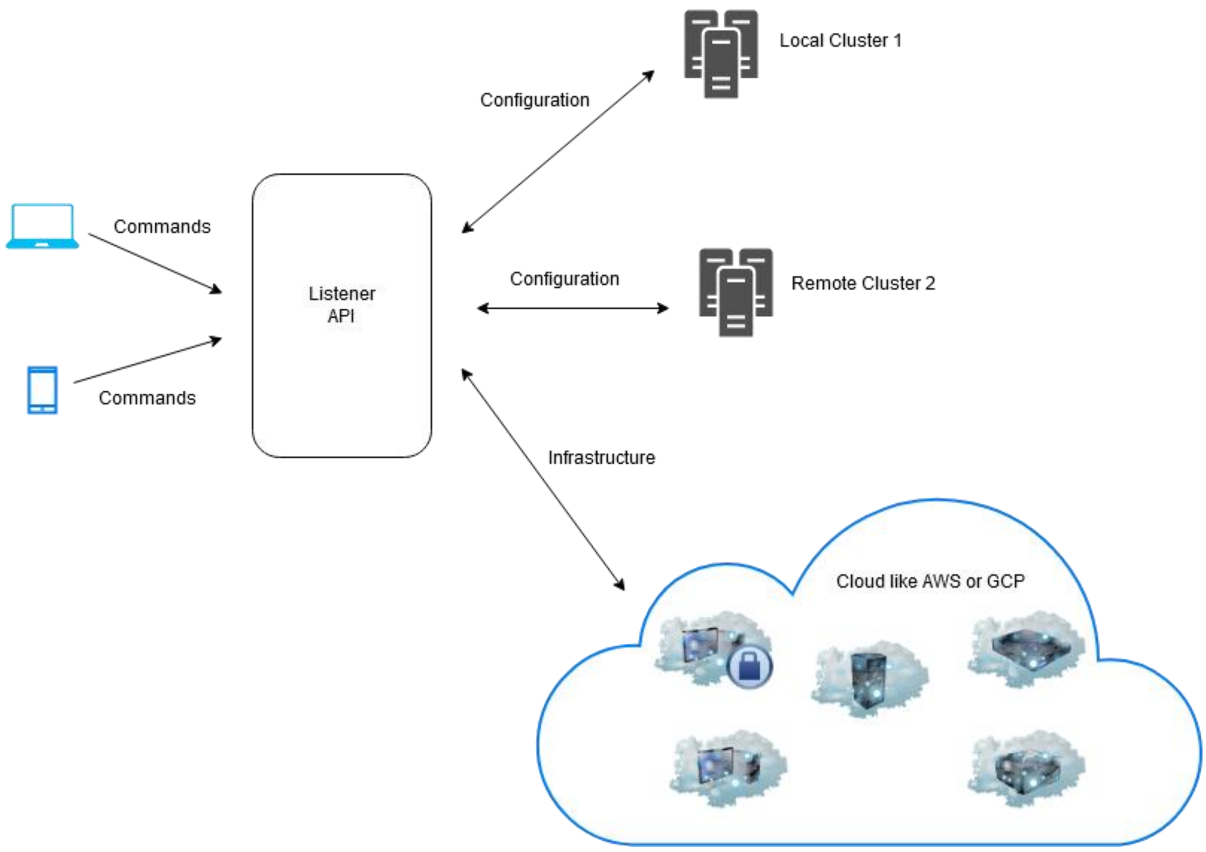


Fig 5.1.1

This is a simple to use, easy to install, and budget friendly architecture. This architecture is suitable for less critical systems that can accept some down time to recover or upgrade system. This is a 2-tier architecture with a client-side application and a server-side application.
Client-Side Application is an application that authenticates the user to the Listener API server. Another important task of the client-side application is to convert the voice command or message to text (Speech to Text) and pass it to the Listener API using a HTTP request.
We have used an android and IOS application for mobile platforms. These applications are made with flutter framework.

Server-side application comprises of many microservices like Listener API, Infrastructure Templates, Configuration Templates, Extra modules.

Listener API is the one responsible for listening the voice commands from client side and then extracting the knowledge out of it. Depending upon the extracted knowledge, the Listener API selects the appropriate module and passes desired and appropriate parameters to it.
Extra modules leverage the power of infrastructure templates and configuration templates to achieve the final desired results.

This system is helpful for small scale projects but if our requirement is to have a high availability structure then we can decouple the Listener API to a high availability cluster (either on docker swarm or Kubernetes or any other hypervisor)

CHAPTER 6 – CODING

6.1 General

A programming tool or software tool is a program or application that software developers use to create, debug, maintain, or otherwise support other programs and applications. The term usually refers to relatively simple programs that can be combined together to accomplish a task. The Chapter describes about the software tool that is used in our project.

6.2 Flask Server

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

"Micro" does not mean that your whole web application has to fit into a single Python file (although it certainly can), nor does it mean that Flask is lacking in functionality. The "micro" in microframework means Flask aims to keep the core simple but extensible. Flask won't make many decisions for you, such as what database to use. Those decisions that it does make, such as what templating engine to use, are easy to change. Everything else is up to you, so that Flask can be everything you need and nothing you don't.

By default, Flask does not include a database abstraction layer, form validation or anything else where different libraries already exist that can handle that. Instead, Flask supports extensions to add such functionality to your application as if it was implemented in Flask itself. Numerous extensions provide database integration, form validation, upload handling, various open authentication technologies, and more. Flask may be "micro", but it's ready for production use on a variety of needs.

Flask Advantages

- Development server and debugger
- Integrated support for unit testing
- RESTful request dispatching
- Uses Jinja templating
- Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based
- Extensive documentation
- Google App Engine compatibility
- Extensions available to enhance features desired

6.3 Python3

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

6.4 Ansible

Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows. It includes its own declarative language to describe system configuration. Ansible was written by Michael DeHaan and acquired by Red Hat in 2015. Ansible is agentless, temporarily connecting remotely via SSH or Windows Remote Management (allowing remote PowerShell execution) to do its tasks.

Control machines have to be a Linux/Unix host (for example SUSE Linux Enterprise, Red Hat Enterprise Linux, Debian, CentOS, macOS, BSD, Ubuntu), and Python 2.7 or 3.5 is required.

Managed nodes, if they are Unix-like, must have Python 2.4 or later. For managed nodes with Python 2.5 or earlier, the python-simple json package is also required. Since version 1.7, Ansible can also manage Windows nodes. In this case, native PowerShell remoting supported by the WS-Management protocol is used, instead of SSH.

The design goals of Ansible include:

- Minimal in nature. Management systems should not impose additional dependencies on the environment.
- Consistent. With Ansible one should be able to create consistent environments.
- Secure. Ansible does not deploy agents to nodes. Only OpenSSH and Python are required on the managed nodes.
- Highly reliable. When carefully written, an Ansible playbook can be idempotent, to prevent unexpected side-effects on the managed systems. It is entirely possible to have a poorly written playbook that is not idempotent.
- Minimal learning required. Playbooks use an easy and descriptive language based on YAML and Jinja templates

How Does Ansible Work ?

In Ansible, there are two categories of computers: the control node and managed nodes. The control node is a computer that runs Ansible. There must be at least one control node, although a backup control node may also exist. A managed node is any device being managed by the control node.

Ansible works by connecting to nodes (clients, servers, or whatever you're configuring) on a network, and then sending a small program called an Ansible module to that node. Ansible executes these modules over SSH and removes them when finished. The only requirement for this interaction is that your Ansible control node has login access to the managed nodes. SSH keys are the most common way to provide access, but other forms of authentication are also supported.

Ansible Architecture

Unlike most configuration-management software, Ansible does not require a single controlling machine where orchestration begins. Ansible works against multiple systems in your infrastructure by selecting portions of Ansible's inventory, stored as edit-able, version-able ASCII text files. Not only is this inventory configurable, but you can also use multiple inventory files at the same time and pull inventory from dynamic or cloud sources or different formats (YAML, INI, etc). Any machine with Ansible utilities installed can leverage a set of files/directories to orchestrate other nodes. The absence of a central-server requirement greatly simplifies disaster-recovery planning. Nodes are managed by this controlling machine - typically over SSH. The controlling machine describes the location of nodes through its inventory. Sensitive data can be stored in encrypted files using Ansible Vault since 2014. In contrast with other popular configuration-management software — such as Chef, Puppet, and CFEngine — Ansible uses an agentless architecture, with Ansible software not normally running or even installed on the controlled node. Instead, Ansible orchestrates a node by installing and running modules on the node temporarily via SSH. For the duration of an orchestration task, a process running the module communicates with the controlling machine with a JSON-based protocol via its standard input and output. When Ansible is not managing a node, it does not consume resources on the node because no daemons are executing of software installed.

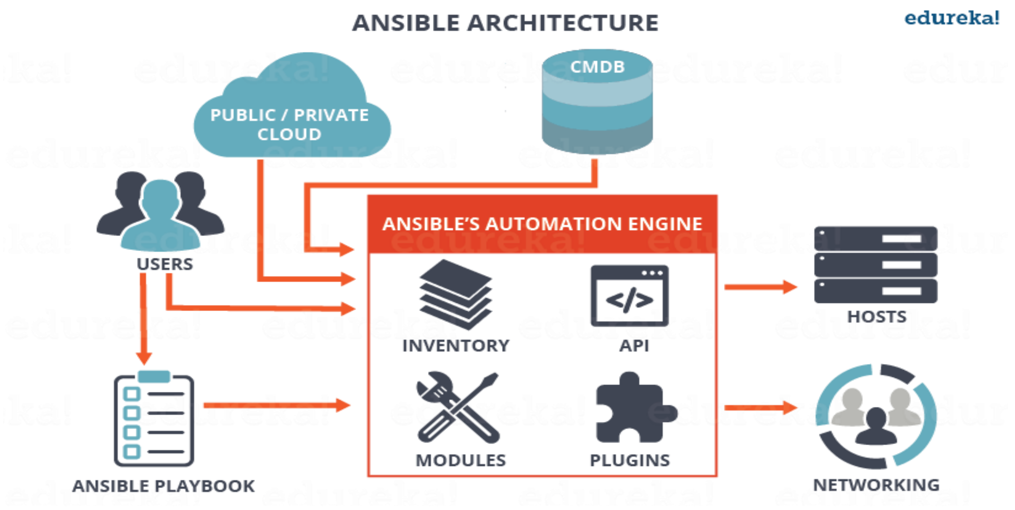


Fig 6.4.1

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).

Ansible in DevOps

The fit of Ansible in the DevOps lifecycle can also provide viable insight into the benefits of Ansible. Ansible provides support for integrating development and operations in contemporary test-driven application design. It provides a stable environment for the development and operations team, thereby leading to smooth orchestration. Ansible automation helps considerably with the representation of Infrastructure as Code (IAC).

IAC involves provisioning and management of computing infrastructure and related configuration through machine-processable definition files. Therefore, administrators could find opportunities to work in collaboration with developers that improves development speed. Also, the benefits of ansible help in focusing more on performance tuning and experimenting rather than fixing issues.

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:

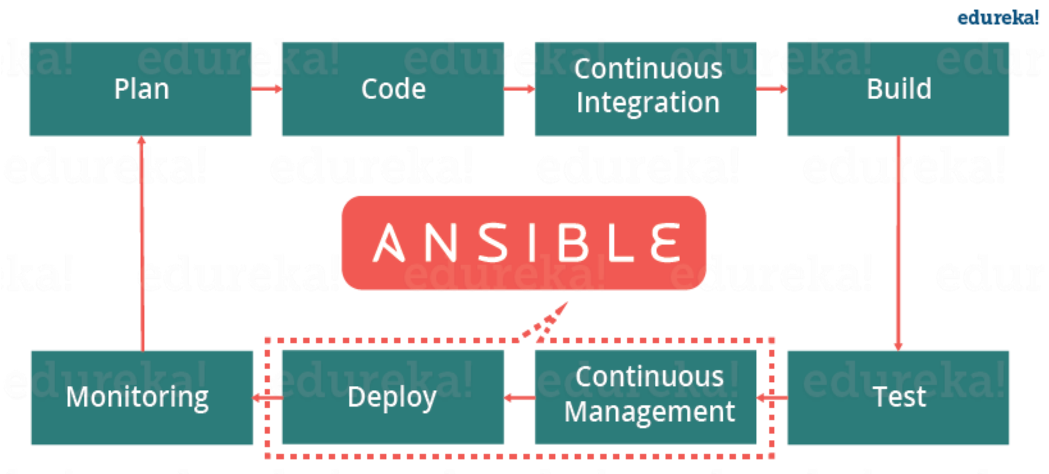


Fig 6.4.2

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.

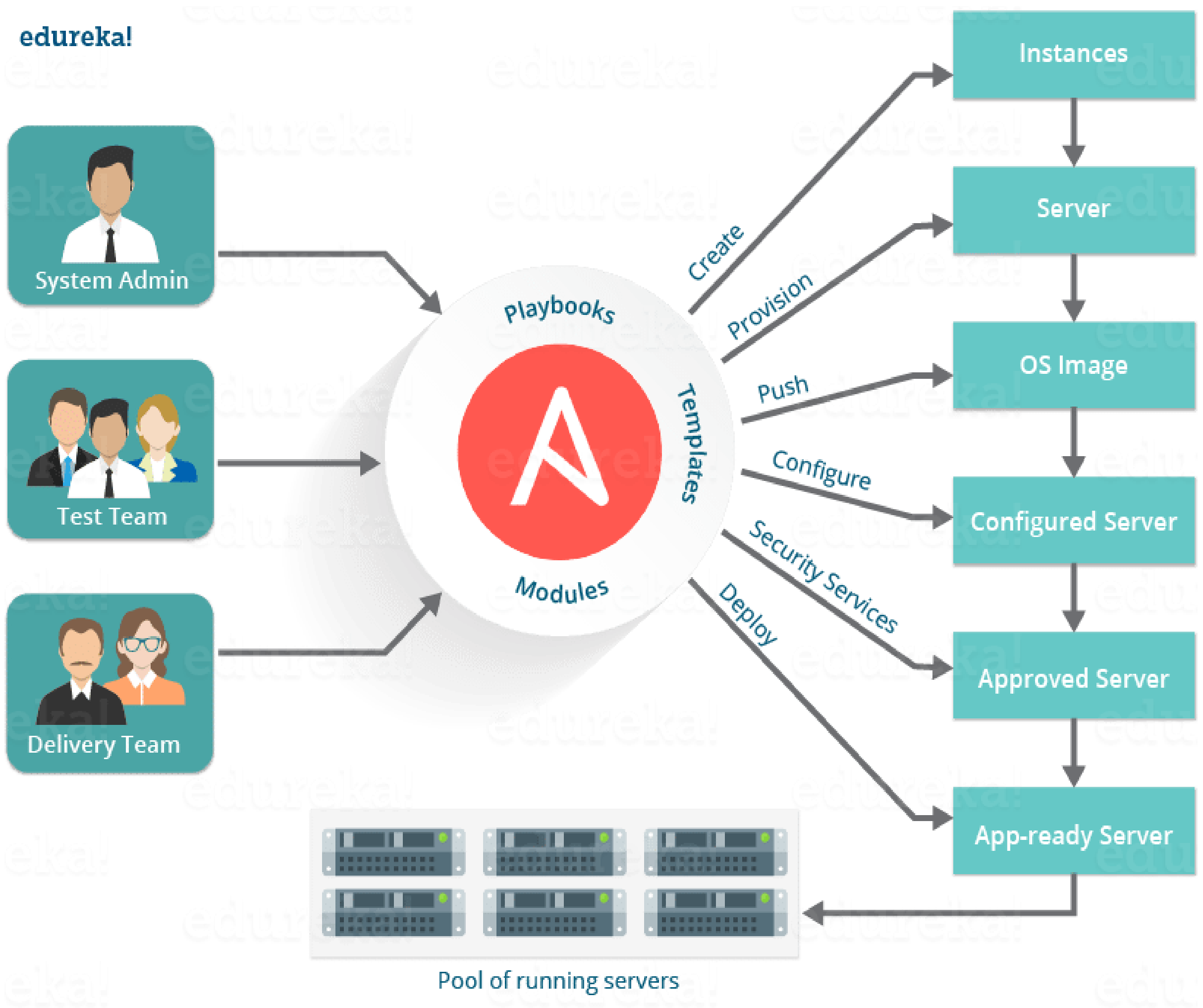


Fig 6.4.3

Advantages of ansible

With so many simplifications, Ansible can surely present many advantages. Let us have a closer look!

Simple to Learn

The foremost mention among advantages of Ansible refers to its simplicity. The simplicity is not only meant for professionals but also for beginners. It is easy to learn, and so, users could learn to use Ansible quickly along with better productivity. Ansible receives the support of comprehensive and easily interpretable documentation.

Therefore, you can learn the logic of Ansible operations and the workflow in a limited period. The lack of a dependency system could imply that Ansible tasks execute sequentially and stop when identifying an error. As a result, troubleshooting becomes a lot easier, even in the initial stages of learning about ansible.

Easily Understandable Python Language

One of the prominent advantages of Ansible also refers to the language in which it is written. Python is a human-readable language and serves as the basis for Ansible. It provides better facilities for getting up Ansible and running it due to the presence of Python libraries on the majority of Linux distributions by default.

Python is a highly ideal alternative for administration and scripting tasks implying higher popularity among engineers and system administrators. Another interesting aspect of Ansible is the facility of Ansible modules that can improve its functionality. The Ansible modules can be written in any language. However, the important concern, in this case, is that the module should return data in JSON format.

No Dependency on Agents

The next important addition among the benefits of Ansible refers to its agentless nature. Ansible manages all the master-agent communications through Standard SSH or Paramiko module. The Paramiko module is a Python implementation of SSH2 and is crucial for managing nodes. Therefore, Ansible does not require any form of agents installed on remote systems for ensuring management. As a result, maintenance overheads and performance degradations reduce considerably by huge margins with Ansible.

Playbooks are written in YAML

The use of Playbooks in Ansible is also another reason for the major advantages of Ansible. Playbooks are Ansible configuration files, and the language for writing them is YAML. The interesting factor, in this case, is that YAML is a better alternative for configuration management and automation.

The superiority of YAML over other formats like JSON makes Ansible better configuration management and automation tool. Ansible makes it easy to read and supports comments. Most important of all, it also includes the use of anchors to reference other items.

Ansible Galaxy

Another notable entry among advantages of Ansible refers to the Ansible Galaxy. Ansible Galaxy is a portal that acts as the central repository for locating, reusing, and sharing Ansible-related content. The best advantage of Ansible Galaxy is in the example of downloading reusable Roles for installing application or server configuration. The downloads are ideal for use in a particular user’s playbooks and can contribute substantially to an increase in deployment speed.

Disadvantages Of Using Ansible

Insufficient User Interface

The first entry in the disadvantages of Ansible is the crude user interface. Ansible was initially a command-line only tool. The first effort of Ansible at making a user interface was with AWX graphical user interface. The other component in the UI was the REST endpoint that is meant for easier infrastructure management.

Subsequently, the AWX turned into the Ansible Tower, which is a web management UI. Ansible Tower offers visual management features and a team-based workflow instrument. However, the Ansible Tower requires considerable improvements. For example, almost 85% of tasks that could be completed through the command line can be achieved through the UI.

You could also come across another mention of Ansible disadvantages arising from its UI. The failure of synchronization between the GUI and the command line can lead to conflicting query results. On a general basis, Ansible Tower is still in the development stages and could not do everything like a command-line interface.

Lack of any Notion of State

Another prominent mention among the disadvantages of Ansible is the lack of any notion of state. Ansible does not have any notion of state like other automation tools such as Puppet. Ansible does not track dependencies and simply executes sequential tasks and stops when tasks finish, fail, or any error comes.

These traits are not ideal for users who want the automation tool to maintain a detailed catalog for ordering. The catalog can help in reaching a specific state without any influence of changes in environmental conditions. However, Ansible lacks it and presents a formidable disadvantage.

Limited Windows Support

The next prominent mention among Ansible disadvantages is the half-built Windows support. Ansible version 1.7 supports Windows as well as Linux/Unix nodes. In the case of Windows, Ansible employs a native PowerShell remoting rather than SSH. As a result, a Linux control machine is mandatory for the management of Windows hosts. The limited support for Windows in Ansible presents one of the formidable setbacks with the configuration management and automation tool.

Ansible does not have Experience

The lack of enterprise support experience also draws down the appeal of Ansible. Ansible does not have a full-fledged working experience with large enterprises like its competitors, such as Puppet and Chef. Even though Ansible claims the facility of enterprise-grade extended support options, limited practical experience reduces the accountability of Ansible.

Ansible is New to the Market

Finally, you can note one of the most common entries in Ansible advantages and disadvantages as a prominent setback of Ansible. Ansible is new to the market, unlike its renowned competitors. As a result, it does not have a large developer or user community. Furthermore, the new presence of Ansible on the market implies the possibilities of undiscovered bugs, software issues, and edge scenarios.

Cloud integration

Ansible can deploy to bare metal hosts, virtualized systems and cloud environments, including Amazon Web Services, Atomic, CenturyLink, Cloudscale, CloudStack, DigitalOcean, Dimension Data, Docker, Google Cloud Platform, KVM, Linode, LXC, LXD, Microsoft

Azure, OpenStack, Oracle Cloud, OVH, oVirt, Packet, Profitbricks, PubNub, Rackspace, Scaleway, SmartOS, SoftLayer, Univention, VMware, Webfaction, and XenServer.

6.5 Terraform

Terraform is an open-source infrastructure as code software tool created by HashiCorp. It enables users to define and provision a datacenter infrastructure using a high-level configuration language known as Hashicorp Configuration Language (HCL), or optionally JSON. Terraform supports a number of cloud infrastructure providers such as Amazon Web Services, IBM Cloud (formerly Bluemix), Google Cloud Platform, DigitalOcean, Linode, Microsoft Azure, Oracle Cloud Infrastructure, OVH, Scaleway, VMware vSphere or Open Telekom Cloud as well as OpenNebula and OpenStack.

HashiCorp also supports a Terraform Module Registry launched in 2017 during HashiConf 2017 conferences. In 2019 Terraform introduced the paid version called Terraform Enterprise for larger organizations. Terraform has four major commands: Terraform init, Terraform Plan, Terraform Apply, Terraform Destroy.

Infrastructure as Code

If you are new to infrastructure as code as a concept, it is the process of managing infrastructure in a file or files rather than manually configuring resources in a user interface. A resource in this instance is any piece of infrastructure in a given environment, such as a virtual machine, security group, network interface, etc.

At a high level, Terraform allows operators to use HCL to author files containing definitions of their desired resources on almost any provider (AWS, GCP, GitHub, Docker, etc) and automates the creation of those resources at the time of apply.

Workflows

A simple workflow for deployment will follow closely to the steps below. We will go over each of these steps and concepts more in-depth throughout this track, so don't panic if you don't understand the concepts immediately.

- **Scope** - Confirm what resources need to be created for a given project.
- **Author** - Create the configuration file in HCL based on the scoped parameters
- **Initialize** - Run terraform init in the project directory with the configuration files. This will download the correct provider plug-ins for the project.
- **Plan & Apply** - Run terraform plan to verify creation process and then terraform apply to create real resources as well as state file that compares future changes in your configuration files to what actually exists in your deployment environment.

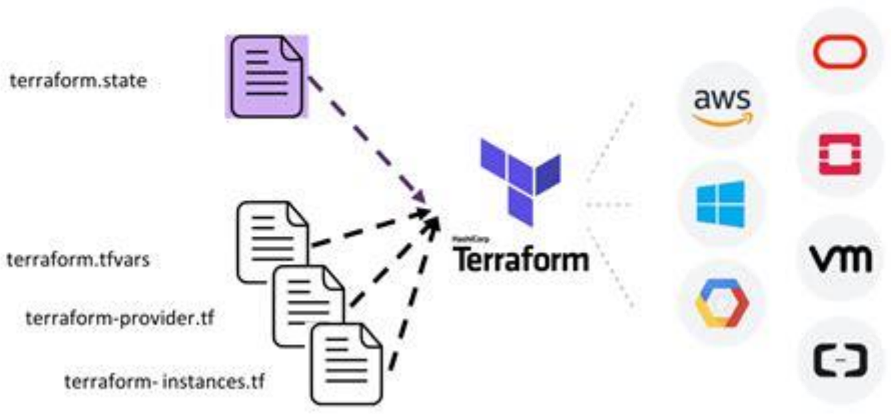


Fig 6.5.1

Advantages of Terraform

While many of the current offerings for infrastructure as code may work in your environment, Terraform aims to have a few advantages for operators and organizations of any size.

- Platform Agnostic
- State Management
- Operator Confidence

Platform Agnostic

In a modern datacenter, you may have several different clouds and platforms to support your various applications. With Terraform, you can manage a heterogenous environment with the same workflow by creating a configuration file to fit the needs of your project or organization.

State Management

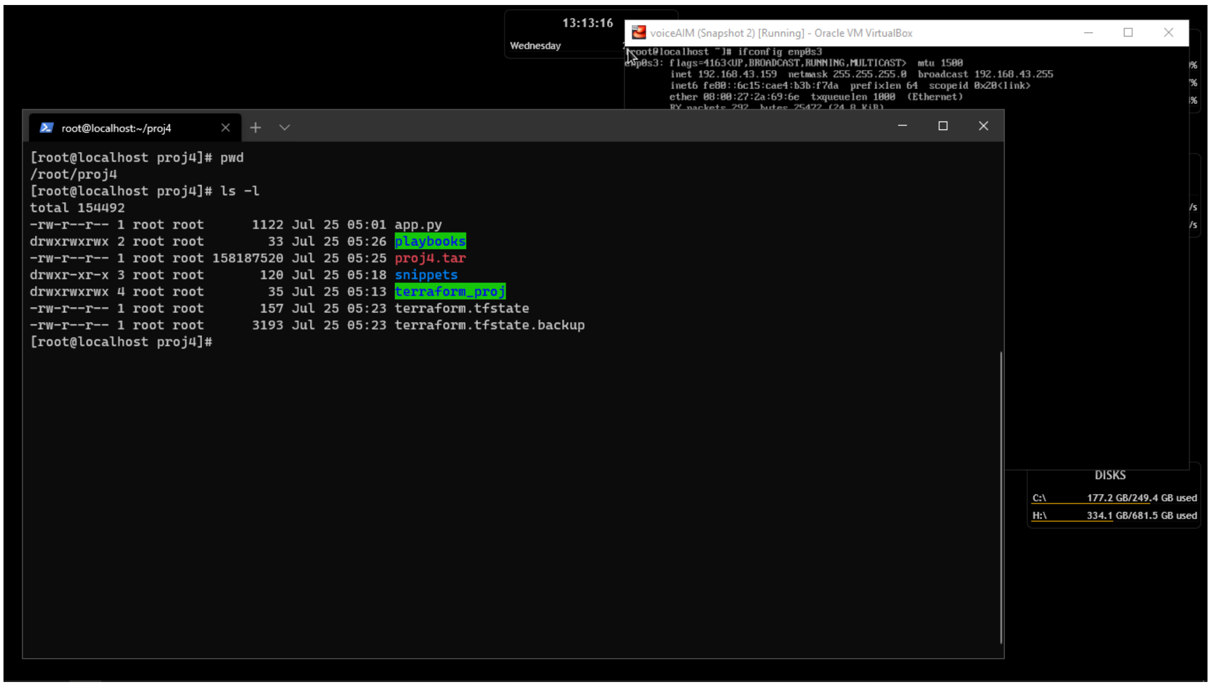
Terraform creates a state file when a project is first initialized. Terraform uses this local state to create plans and make changes to your infrastructure. Prior to any operation, Terraform does a refresh to update the state with the real infrastructure. This means that Terraform state is the source of truth by which configuration changes are measured. If a change is made or a resource is appended to a configuration, Terraform compares those changes with the state file to determine what changes result in a new resource or resource modifications.

Operator Confidence

The workflow built into Terraform aims to in still confidence in users by promoting easily repeatable operations and a planning phase to allow users to ensure the actions taken by Terraform will not cause disruption in their environment. Upon terraform apply, the user will be prompted to review the proposed changes and must affirm the changes or else Terraform will not apply the proposed plan.

CHAPTER 7 – GRAPHICAL USER INTERFACE

1. Directory Structure



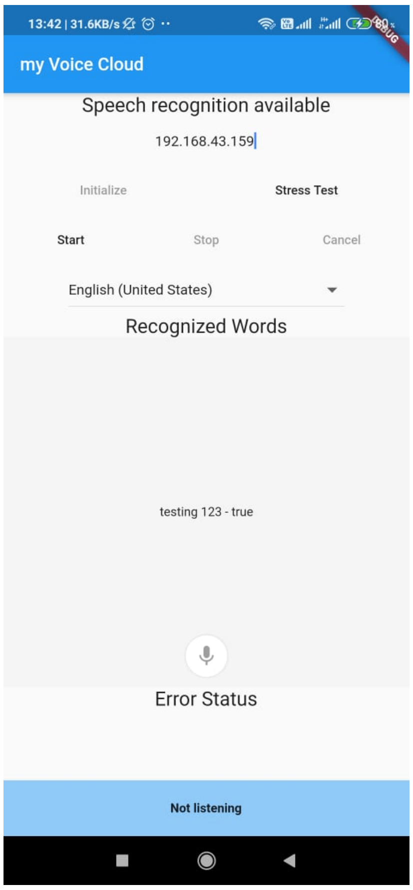
13:13:16
Wednesday
voiceAIM (Snapshot 2) [Running] - Oracle VM VirtualBox
root@localhost:~/proj4
DISKS
C:\ 177.2 GB/249.4 GB used
H:\ 334.1 GB/681.5 GB used

```
[root@localhost ~]# ifconfig enp0s3
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.159  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::6c15:cae4:b3b:f7da  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:2a:69:6e  txqueuelen 1000  (Ethernet)
```

```
[root@localhost proj4]# pwd
/root/proj4
[root@localhost proj4]# ls -l
total 154492
-rw-r--r-- 1 root root      1122 Jul 25 05:01 app.py
drwxrwxrwx 2 root root        33 Jul 25 05:26 playbooks
-rw-r--r-- 1 root root 158187520 Jul 25 05:25 proj4.tar
drwxr-xr-x 3 root root       120 Jul 25 05:18 snippets
drwxrwxrwx 4 root root        35 Jul 25 05:13 terraform_proj
-rw-r--r-- 1 root root       157 Jul 25 05:23 terraform.tfstate
-rw-r--r-- 1 root root      3193 Jul 25 05:23 terraform.tfstate.backup
[root@localhost proj4]#
```

Fig 7.1.1

2. Client side Application



Speech recognition available

192.168.43.159

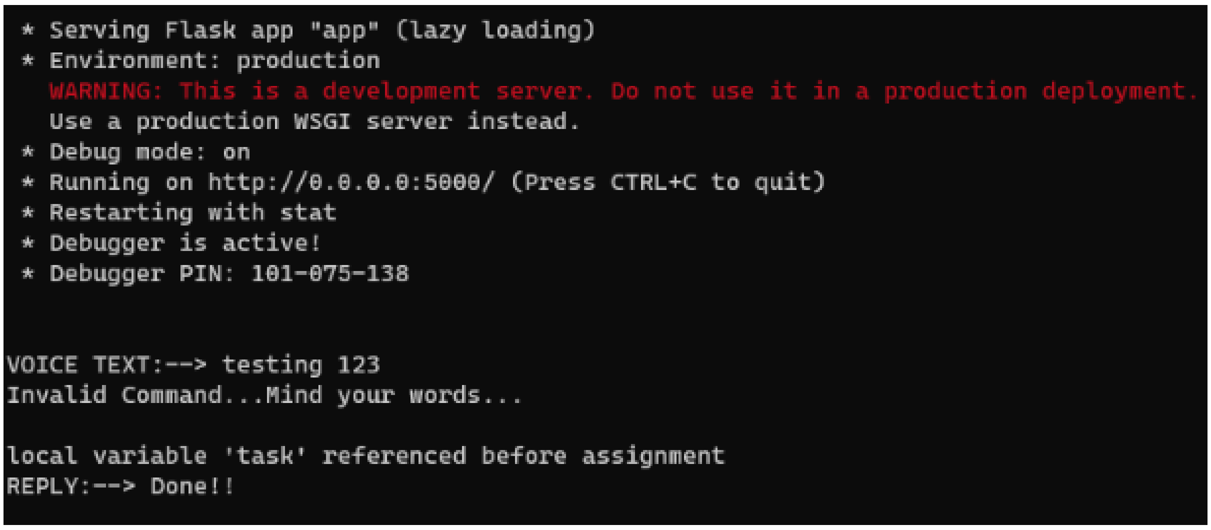
English (United States)

Recognized Words

testing 123 - true

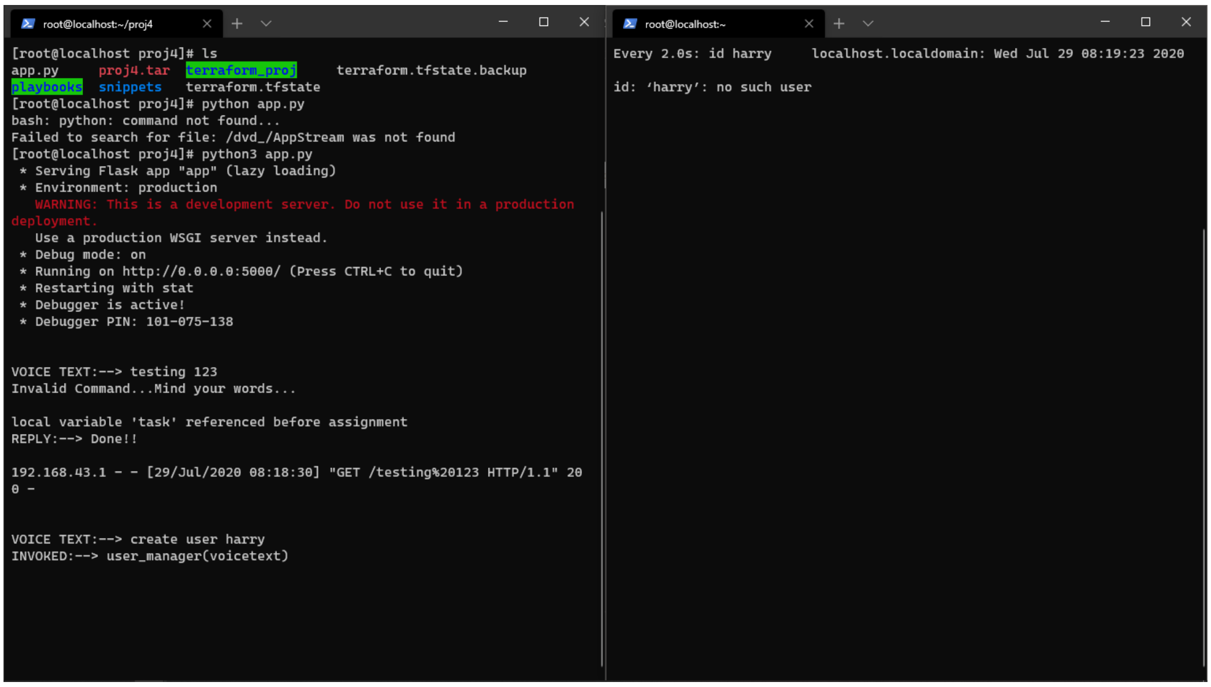
Error Status

Fig 7.2.1



```
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 101-075-138


VOICE TEXT:--> testing 123
Invalid Command...Mind your words...

local variable 'task' referenced before assignment
REPLY:--> Done!!
```

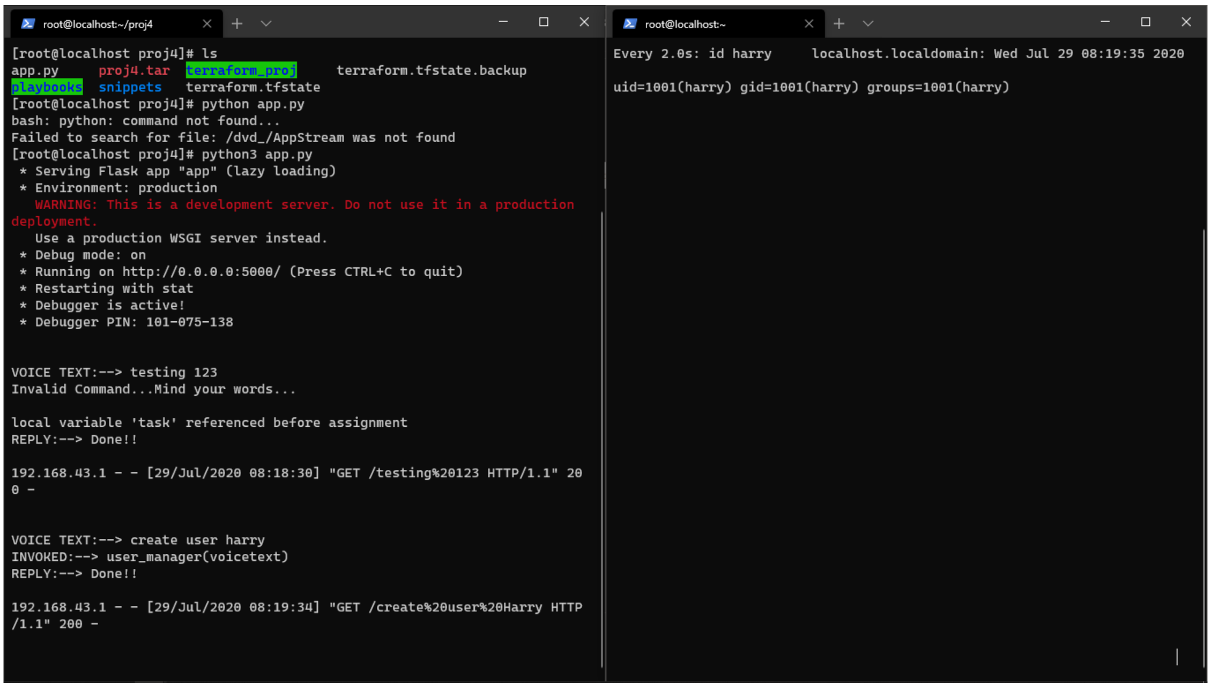
Fig 7.2.2



```
root@localhost:~/proj4

[root@localhost proj4]# ls
app.py     proj4.tar  terraform_proj     terraform.tfstate.backup
playbooks  snippets   terraform.tfstate
[root@localhost proj4]# python app.py
bash: python: command not found...
Failed to search for file: /dvd_/AppStream was not found
[root@localhost proj4]# python3 app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 101-075-138


VOICE TEXT:--> testing 123
Invalid Command...Mind your words...

local variable 'task' referenced before assignment
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:18:30] "GET /testing%20123 HTTP/1.1" 20
0 -


VOICE TEXT:--> create user harry
INVOKED:--> user_manager(voicetext)
```

```
root@localhost:~

Every 2.0s: id harry     localhost.localdomain: Wed Jul 29 08:19:23 2020

id: 'harry': no such user
```

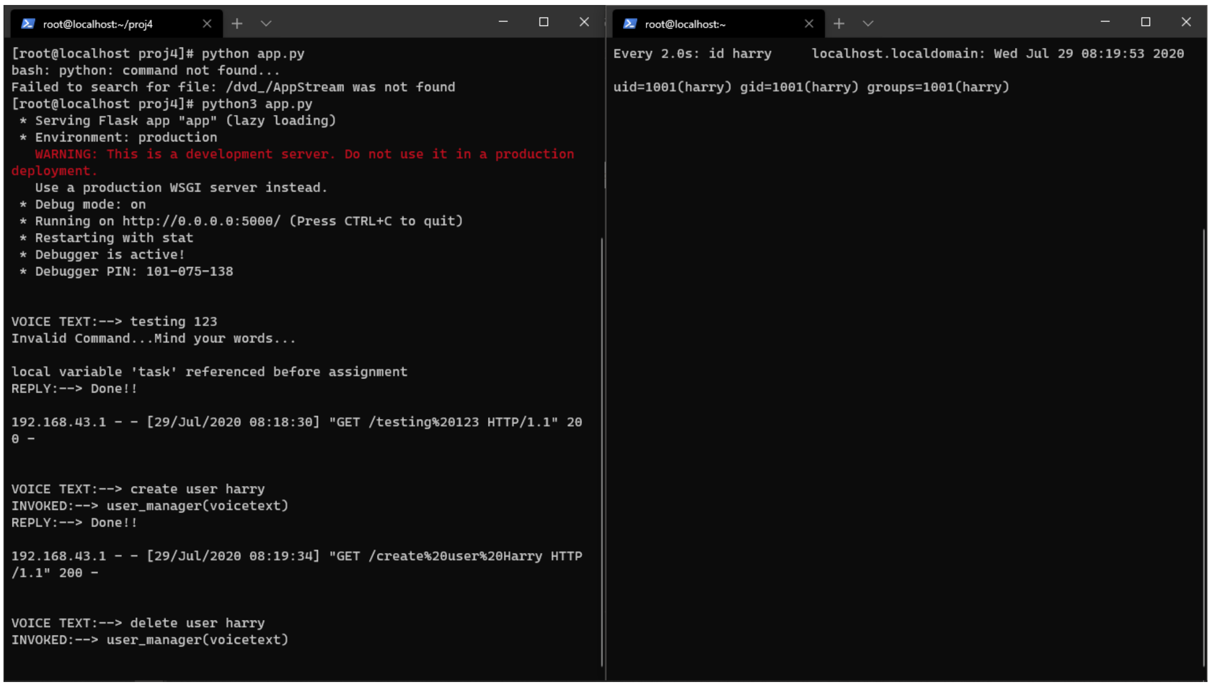
Fig 7.2.3



```
root@localhost:~/proj4

[root@localhost proj4]# ls
app.py     proj4.tar  terraform_proj     terraform.tfstate.backup
playbooks  snippets   terraform.tfstate
[root@localhost proj4]# python app.py
bash: python: command not found...
Failed to search for file: /dvd_/AppStream was not found
[root@localhost proj4]# python3 app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 101-075-138


VOICE TEXT:--> testing 123
Invalid Command...Mind your words...

local variable 'task' referenced before assignment
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:18:30] "GET /testing%20123 HTTP/1.1" 20
0 -


VOICE TEXT:--> create user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:19:34] "GET /create%20user%20Harry HTTP
/1.1" 200 -
```

```
root@localhost:~

Every 2.0s: id harry     localhost.localdomain: Wed Jul 29 08:19:35 2020

uid=1001(harry) gid=1001(harry) groups=1001(harry)
```

Fig 7.2.4



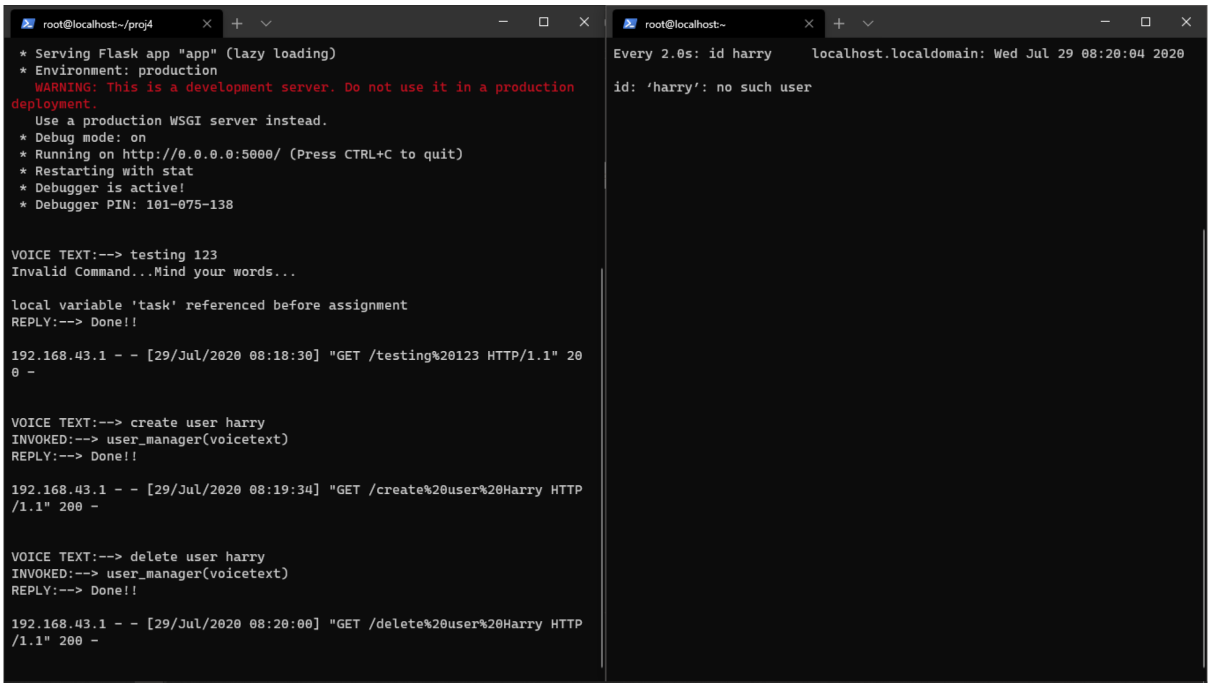
root@localhost:~/proj4

```
[root@localhost proj4]# python app.py
bash: python: command not found...
Failed to search for file: /dvd_/AppStream was not found
[root@localhost proj4]# python3 app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 101-075-138


VOICE TEXT:--> testing 123
Invalid Command...Mind your words...

local variable 'task' referenced before assignment
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:18:30] "GET /testing%20123 HTTP/1.1" 20
0 -


VOICE TEXT:--> create user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:19:34] "GET /create%20user%20Harry HTTP
/1.1" 200 -


VOICE TEXT:--> delete user harry
INVOKED:--> user_manager(voicetext)
```

root@localhost:~

```
Every 2.0s: id harry     localhost.localdomain: Wed Jul 29 08:19:53 2020

uid=1001(harry) gid=1001(harry) groups=1001(harry)
```

Fig 7.2.5



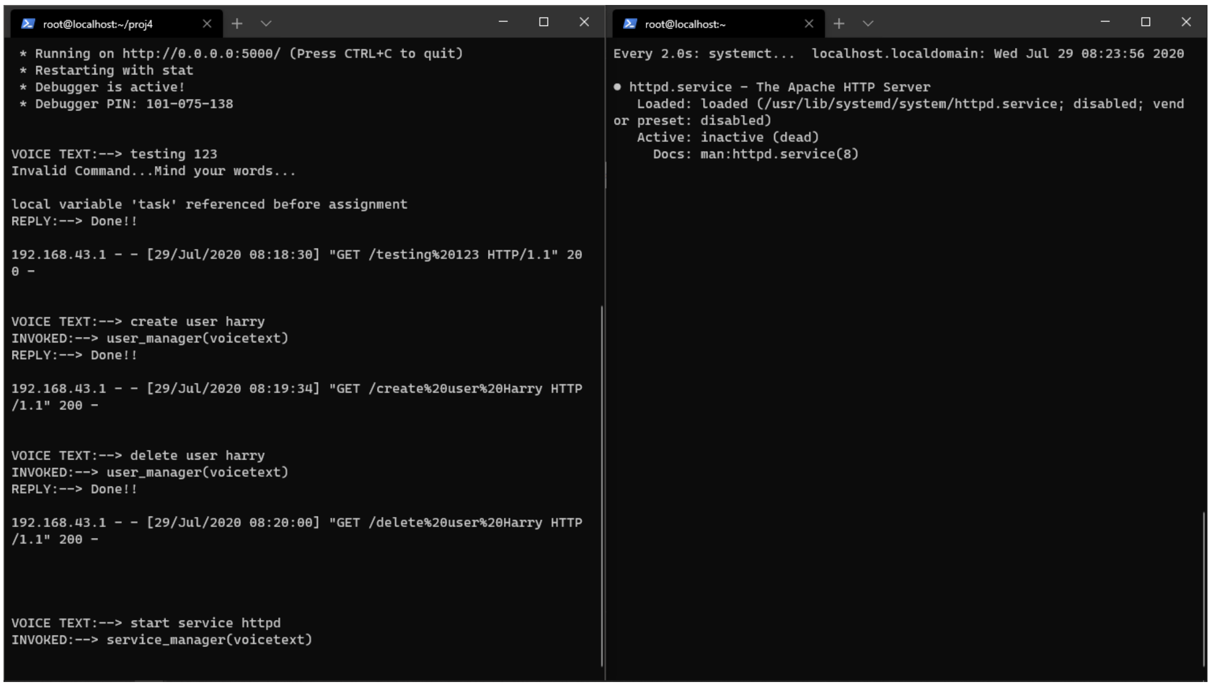
root@localhost:~/proj4

```
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 101-075-138


VOICE TEXT:--> testing 123
Invalid Command...Mind your words...

local variable 'task' referenced before assignment
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:18:30] "GET /testing%20123 HTTP/1.1" 20
0 -


VOICE TEXT:--> create user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:19:34] "GET /create%20user%20Harry HTTP
/1.1" 200 -


VOICE TEXT:--> delete user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:20:00] "GET /delete%20user%20Harry HTTP
/1.1" 200 -
```

root@localhost:~

```
Every 2.0s: id harry     localhost.localdomain: Wed Jul 29 08:20:04 2020

id: ‘harry’: no such user
```

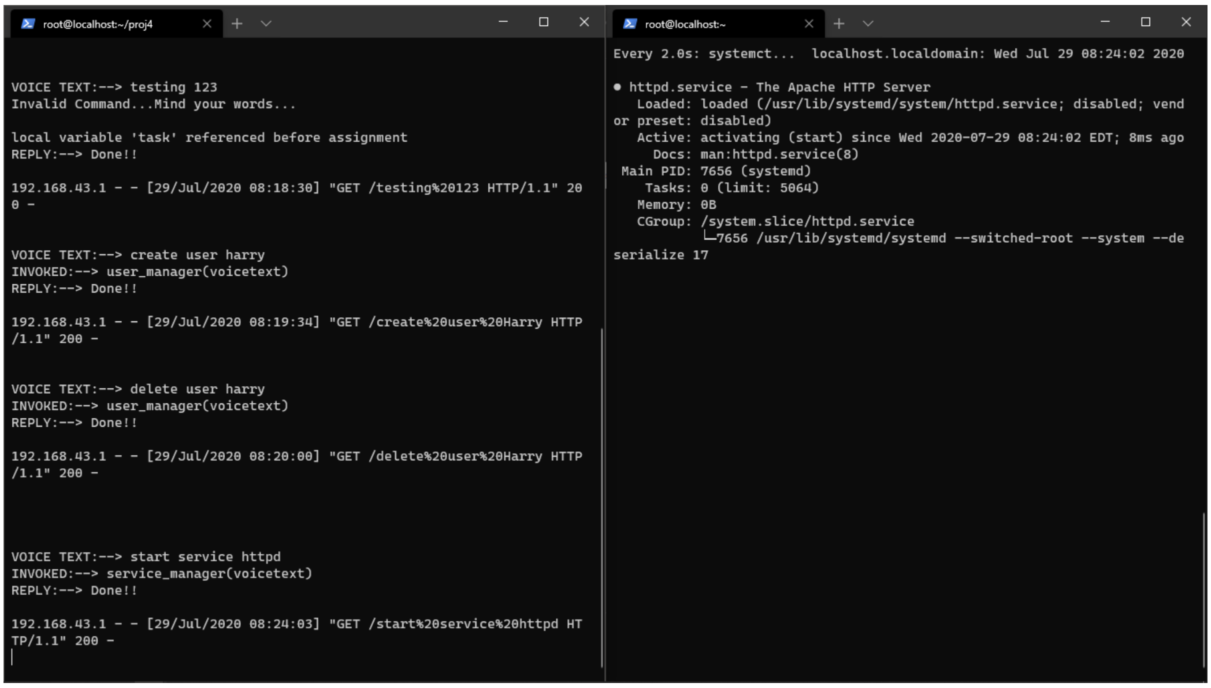
Fig 7.2.6



```
root@localhost:~/proj4

 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 101-075-138


VOICE TEXT:--> testing 123
Invalid Command...Mind your words...

local variable 'task' referenced before assignment
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:18:30] "GET /testing%20123 HTTP/1.1" 20
0 -


VOICE TEXT:--> create user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:19:34] "GET /create%20user%20Harry HTTP
/1.1" 200 -


VOICE TEXT:--> delete user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:20:00] "GET /delete%20user%20Harry HTTP
/1.1" 200 -




VOICE TEXT:--> start service httpd
INVOKED:--> service_manager(voicetext)
```

```
root@localhost:~

Every 2.0s: systemct...  localhost.localdomain: Wed Jul 29 08:23:56 2020

● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vend
or preset: disabled)
   Active: inactive (dead)
     Docs: man:httpd.service(8)
```

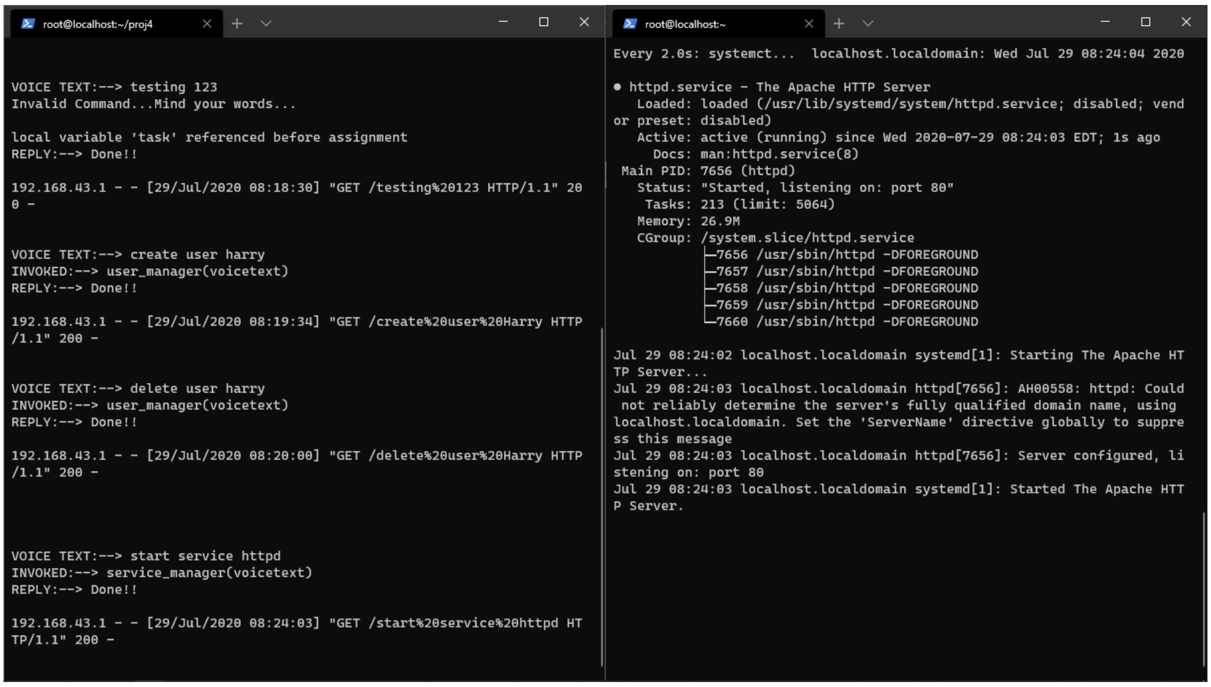
Fig 7.2.7



```
root@localhost:~/proj4


VOICE TEXT:--> testing 123
Invalid Command...Mind your words...

local variable 'task' referenced before assignment
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:18:30] "GET /testing%20123 HTTP/1.1" 20
0 -


VOICE TEXT:--> create user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:19:34] "GET /create%20user%20Harry HTTP
/1.1" 200 -


VOICE TEXT:--> delete user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:20:00] "GET /delete%20user%20Harry HTTP
/1.1" 200 -




VOICE TEXT:--> start service httpd
INVOKED:--> service_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:24:03] "GET /start%20service%20httpd HT
TP/1.1" 200 -
```

```
root@localhost:~

Every 2.0s: systemct...  localhost.localdomain: Wed Jul 29 08:24:02 2020

● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vend
or preset: disabled)
   Active: activating (start) since Wed 2020-07-29 08:24:02 EDT; 8ms ago
     Docs: man:httpd.service(8)
 Main PID: 7656 (systemd)
    Tasks: 0 (limit: 5064)
   Memory: 0B
   CGroup: /system.slice/httpd.service
           └─7656 /usr/lib/systemd/systemd --switched-root --system --de
serialize 17
```

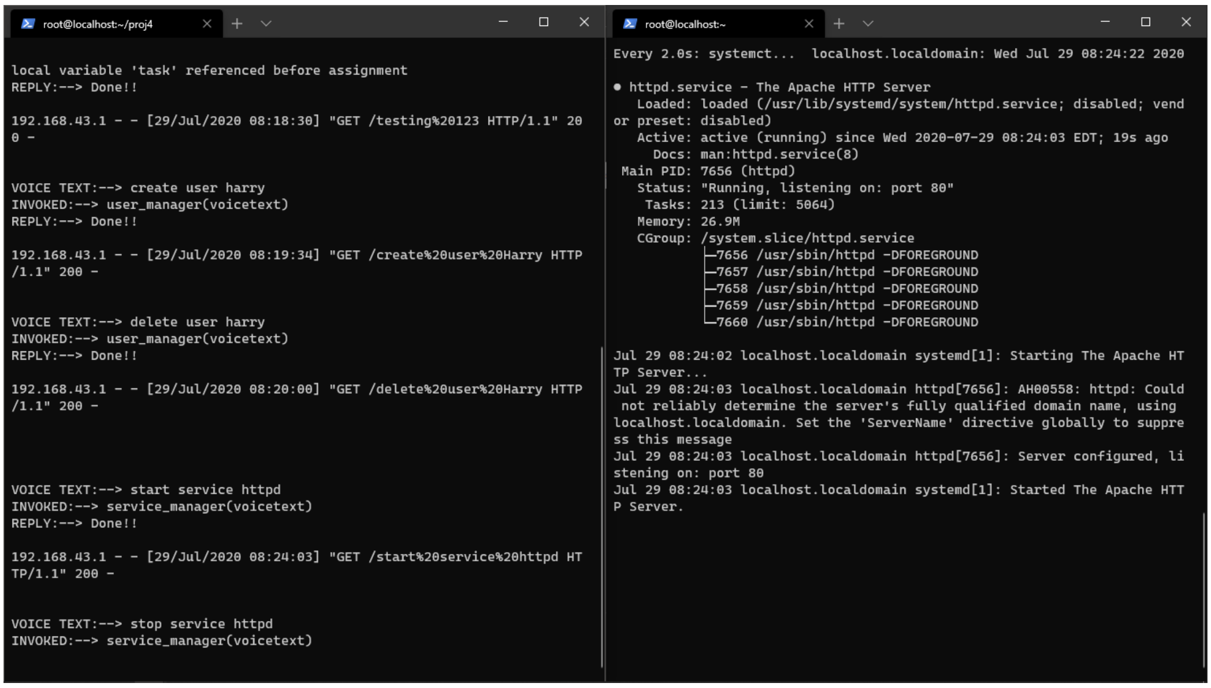
Fig 7.2.8



```
VOICE TEXT:--> testing 123
Invalid Command...Mind your words...

local variable 'task' referenced before assignment
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:18:30] "GET /testing%20123 HTTP/1.1" 20
0 -


VOICE TEXT:--> create user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:19:34] "GET /create%20user%20Harry HTTP
/1.1" 200 -


VOICE TEXT:--> delete user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:20:00] "GET /delete%20user%20Harry HTTP
/1.1" 200 -



VOICE TEXT:--> start service httpd
INVOKED:--> service_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:24:03] "GET /start%20service%20httpd HT
TP/1.1" 200 -
```

```
Every 2.0s: systemct...  localhost.localdomain: Wed Jul 29 08:24:04 2020

● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vend
or preset: disabled)
   Active: active (running) since Wed 2020-07-29 08:24:03 EDT; 1s ago
     Docs: man:httpd.service(8)
 Main PID: 7656 (httpd)
   Status: "Started, listening on: port 80"
    Tasks: 213 (limit: 5064)
   Memory: 26.9M
   CGroup: /system.slice/httpd.service
           ├─7656 /usr/sbin/httpd -DFOREGROUND
           ├─7657 /usr/sbin/httpd -DFOREGROUND
           ├─7658 /usr/sbin/httpd -DFOREGROUND
           ├─7659 /usr/sbin/httpd -DFOREGROUND
           └─7660 /usr/sbin/httpd -DFOREGROUND

Jul 29 08:24:02 localhost.localdomain systemd[1]: Starting The Apache HT
TP Server...
Jul 29 08:24:03 localhost.localdomain httpd[7656]: AH00558: httpd: Could
 not reliably determine the server's fully qualified domain name, using
localhost.localdomain. Set the 'ServerName' directive globally to suppre
ss this message
Jul 29 08:24:03 localhost.localdomain httpd[7656]: Server configured, li
stening on: port 80
Jul 29 08:24:03 localhost.localdomain systemd[1]: Started The Apache HTT
P Server.
```

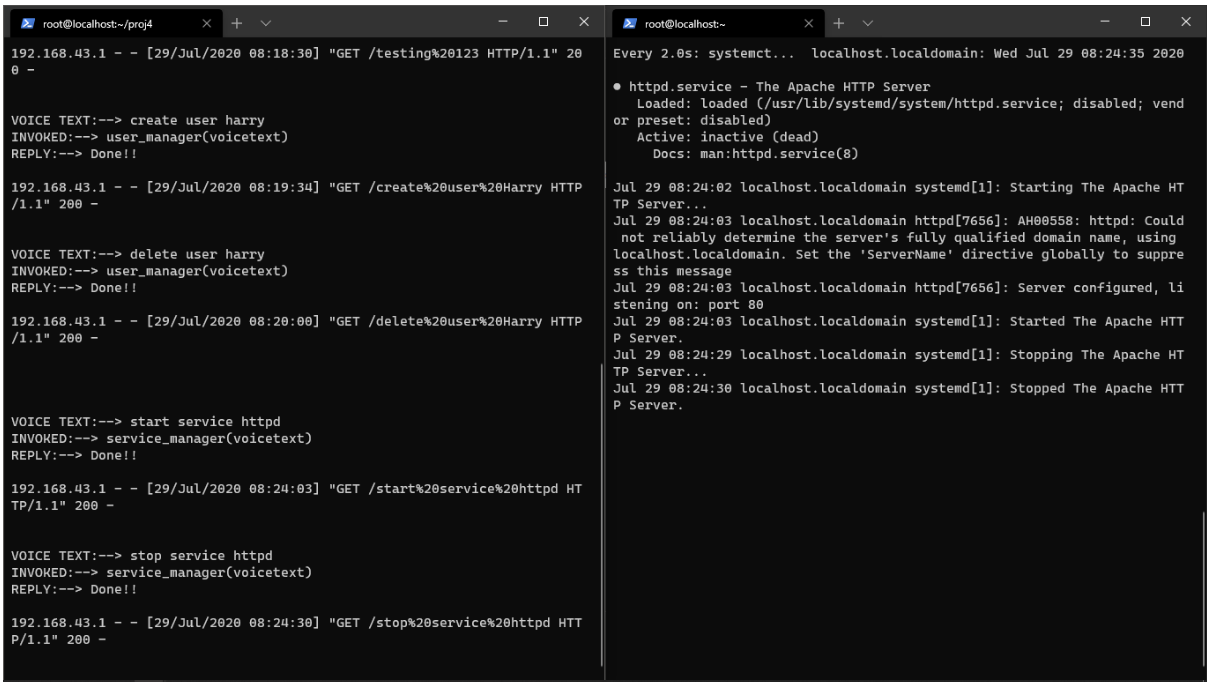
Fig 7.2.9



```
local variable 'task' referenced before assignment
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:18:30] "GET /testing%20123 HTTP/1.1" 20
0 -


VOICE TEXT:--> create user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:19:34] "GET /create%20user%20Harry HTTP
/1.1" 200 -


VOICE TEXT:--> delete user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:20:00] "GET /delete%20user%20Harry HTTP
/1.1" 200 -



VOICE TEXT:--> start service httpd
INVOKED:--> service_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:24:03] "GET /start%20service%20httpd HT
TP/1.1" 200 -


VOICE TEXT:--> stop service httpd
INVOKED:--> service_manager(voicetext)
```

```
Every 2.0s: systemct...  localhost.localdomain: Wed Jul 29 08:24:22 2020

● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vend
or preset: disabled)
   Active: active (running) since Wed 2020-07-29 08:24:03 EDT; 19s ago
     Docs: man:httpd.service(8)
 Main PID: 7656 (httpd)
   Status: "Running, listening on: port 80"
    Tasks: 213 (limit: 5064)
   Memory: 26.9M
   CGroup: /system.slice/httpd.service
           ├─7656 /usr/sbin/httpd -DFOREGROUND
           ├─7657 /usr/sbin/httpd -DFOREGROUND
           ├─7658 /usr/sbin/httpd -DFOREGROUND
           ├─7659 /usr/sbin/httpd -DFOREGROUND
           └─7660 /usr/sbin/httpd -DFOREGROUND

Jul 29 08:24:02 localhost.localdomain systemd[1]: Starting The Apache HT
TP Server...
Jul 29 08:24:03 localhost.localdomain httpd[7656]: AH00558: httpd: Could
 not reliably determine the server's fully qualified domain name, using
localhost.localdomain. Set the 'ServerName' directive globally to suppre
ss this message
Jul 29 08:24:03 localhost.localdomain httpd[7656]: Server configured, li
stening on: port 80
Jul 29 08:24:03 localhost.localdomain systemd[1]: Started The Apache HTT
P Server.
```

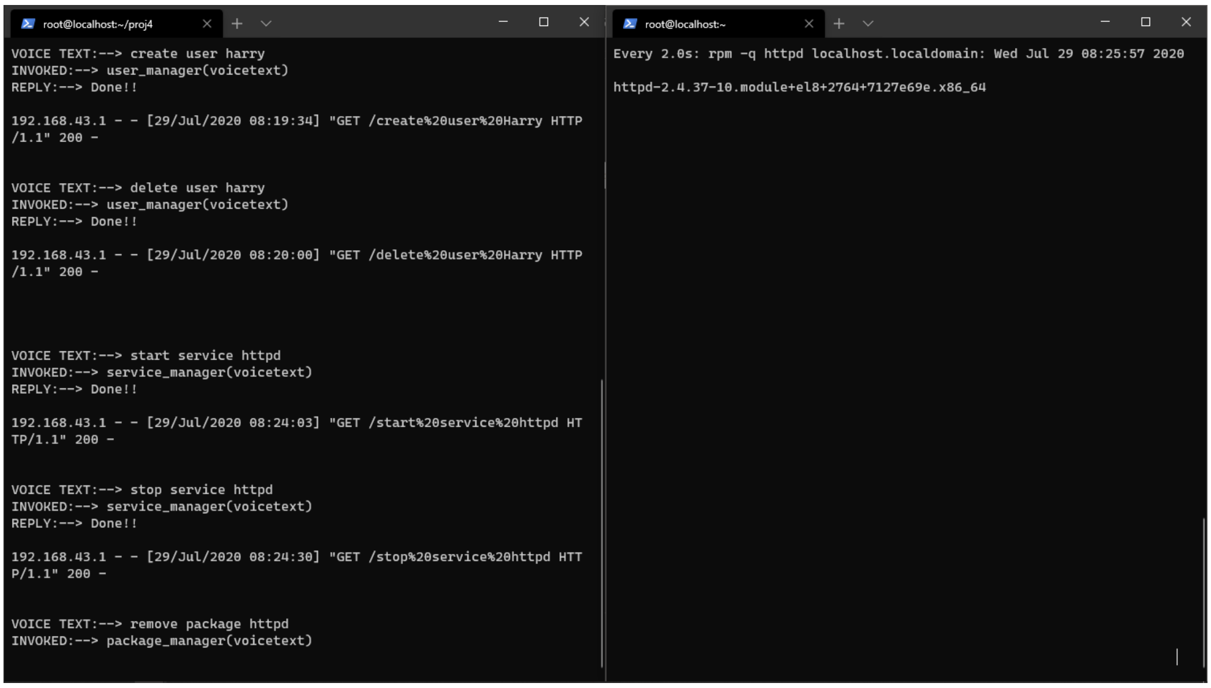
Fig 7.2.10



```
192.168.43.1 - - [29/Jul/2020 08:18:30] "GET /testing%20123 HTTP/1.1" 20
0 -


VOICE TEXT:--> create user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:19:34] "GET /create%20user%20Harry HTTP
/1.1" 200 -


VOICE TEXT:--> delete user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:20:00] "GET /delete%20user%20Harry HTTP
/1.1" 200 -




VOICE TEXT:--> start service httpd
INVOKED:--> service_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:24:03] "GET /start%20service%20httpd HT
TP/1.1" 200 -


VOICE TEXT:--> stop service httpd
INVOKED:--> service_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:24:30] "GET /stop%20service%20httpd HTT
P/1.1" 200 -
```

```
Every 2.0s: systemct...  localhost.localdomain: Wed Jul 29 08:24:35 2020

● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vend
or preset: disabled)
   Active: inactive (dead)
     Docs: man:httpd.service(8)

Jul 29 08:24:02 localhost.localdomain systemd[1]: Starting The Apache HT
TP Server...
Jul 29 08:24:03 localhost.localdomain httpd[7656]: AH00558: httpd: Could
 not reliably determine the server's fully qualified domain name, using
localhost.localdomain. Set the 'ServerName' directive globally to suppre
ss this message
Jul 29 08:24:03 localhost.localdomain httpd[7656]: Server configured, li
stening on: port 80
Jul 29 08:24:03 localhost.localdomain systemd[1]: Started The Apache HTT
P Server.
Jul 29 08:24:29 localhost.localdomain systemd[1]: Stopping The Apache HT
TP Server...
Jul 29 08:24:30 localhost.localdomain systemd[1]: Stopped The Apache HTT
P Server.
```

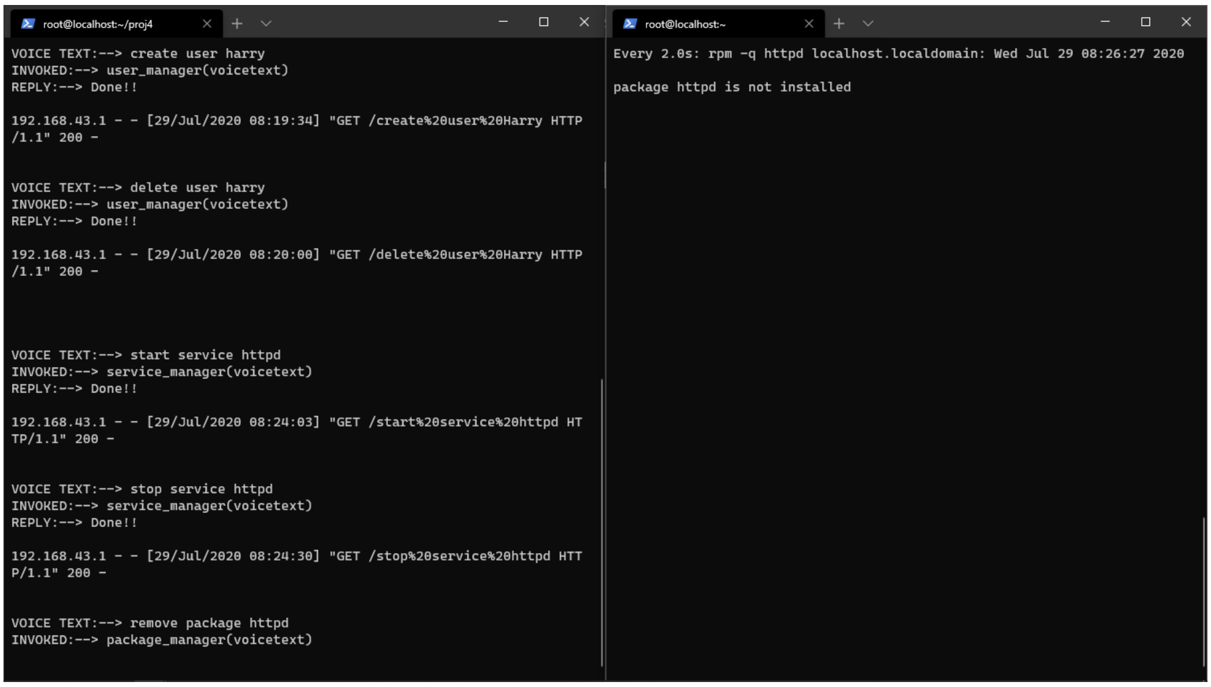
Fig 7.2.10



```
VOICE TEXT:--> create user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:19:34] "GET /create%20user%20Harry HTTP
/1.1" 200 -


VOICE TEXT:--> delete user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:20:00] "GET /delete%20user%20Harry HTTP
/1.1" 200 -




VOICE TEXT:--> start service httpd
INVOKED:--> service_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:24:03] "GET /start%20service%20httpd HT
TP/1.1" 200 -


VOICE TEXT:--> stop service httpd
INVOKED:--> service_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:24:30] "GET /stop%20service%20httpd HTT
P/1.1" 200 -


VOICE TEXT:--> remove package httpd
INVOKED:--> package_manager(voicetext)
```

```
Every 2.0s: rpm -q httpd localhost.localdomain: Wed Jul 29 08:25:57 2020

httpd-2.4.37-10.module+el8+2764+7127e69e.x86_64
```

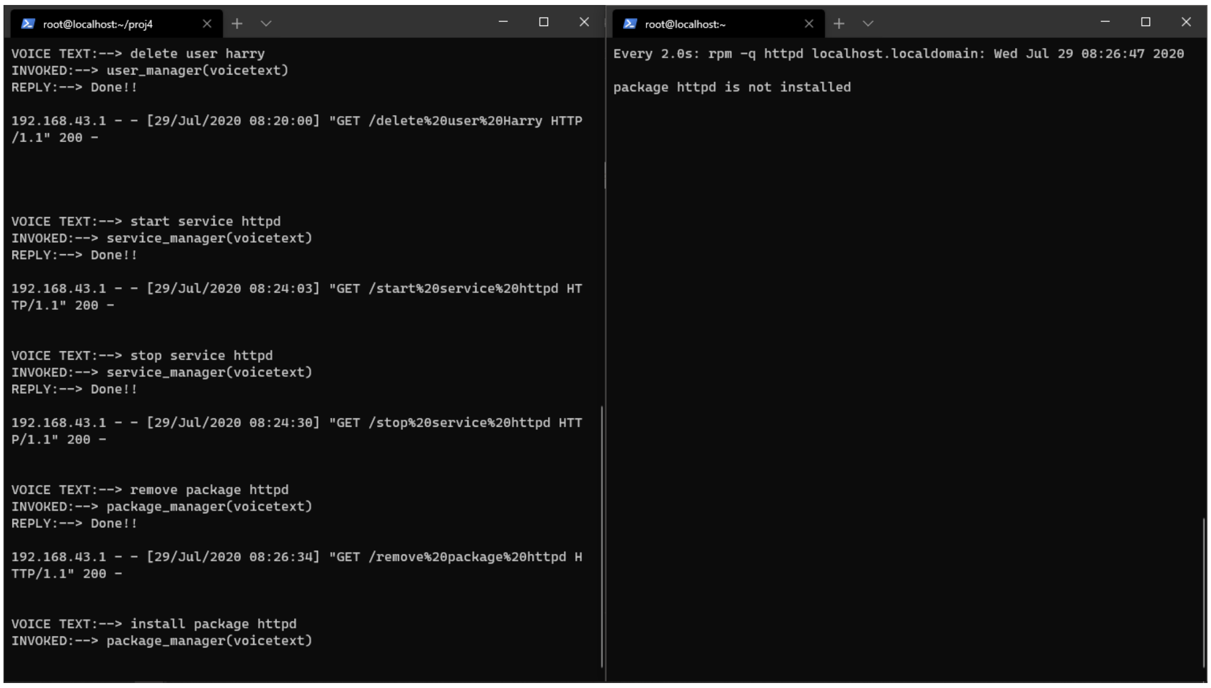
Fig 7.2.11



```
VOICE TEXT:--> create user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:19:34] "GET /create%20user%20Harry HTTP
/1.1" 200 -


VOICE TEXT:--> delete user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:20:00] "GET /delete%20user%20Harry HTTP
/1.1" 200 -



VOICE TEXT:--> start service httpd
INVOKED:--> service_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:24:03] "GET /start%20service%20httpd HT
TP/1.1" 200 -


VOICE TEXT:--> stop service httpd
INVOKED:--> service_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:24:30] "GET /stop%20service%20httpd HTT
P/1.1" 200 -


VOICE TEXT:--> remove package httpd
INVOKED:--> package_manager(voicetext)
```

```
Every 2.0s: rpm -q httpd localhost.localdomain: Wed Jul 29 08:26:27 2020

package httpd is not installed
```

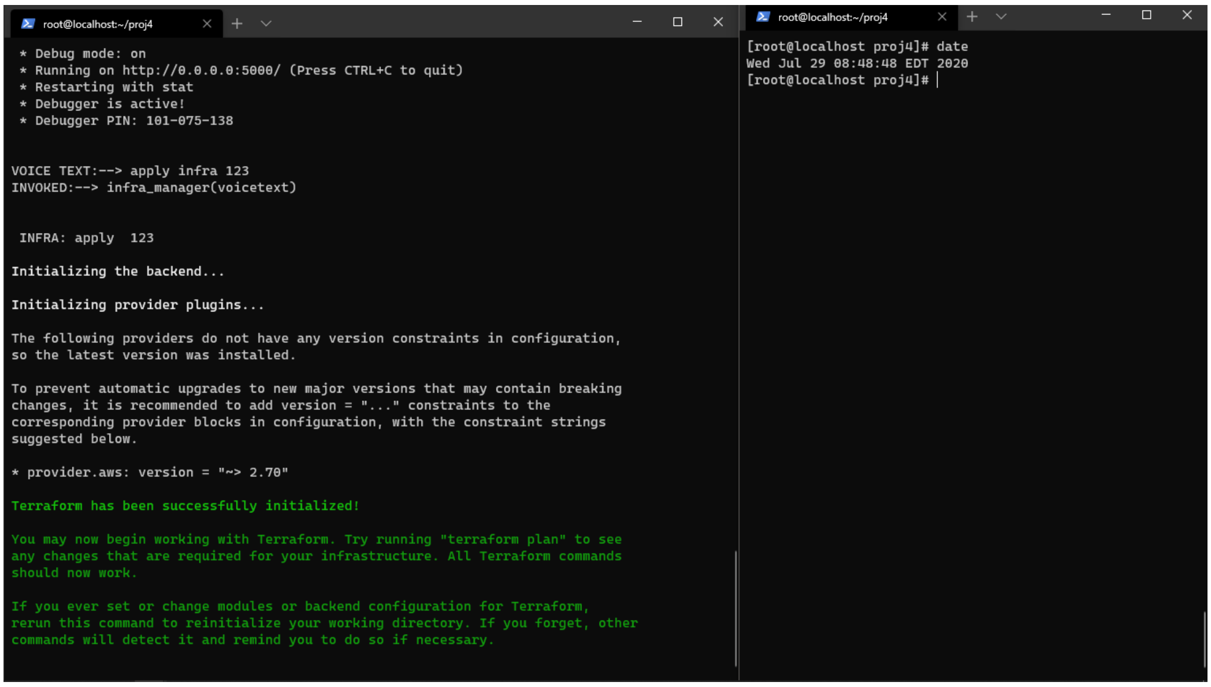
Fig 7.2.12



```
VOICE TEXT:--> delete user harry
INVOKED:--> user_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:20:00] "GET /delete%20user%20Harry HTTP
/1.1" 200 -



VOICE TEXT:--> start service httpd
INVOKED:--> service_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:24:03] "GET /start%20service%20httpd HT
TP/1.1" 200 -


VOICE TEXT:--> stop service httpd
INVOKED:--> service_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:24:30] "GET /stop%20service%20httpd HTT
P/1.1" 200 -


VOICE TEXT:--> remove package httpd
INVOKED:--> package_manager(voicetext)
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:26:34] "GET /remove%20package%20httpd H
TTP/1.1" 200 -


VOICE TEXT:--> install package httpd
INVOKED:--> package_manager(voicetext)
```

```
Every 2.0s: rpm -q httpd localhost.localdomain: Wed Jul 29 08:26:47 2020

package httpd is not installed
```

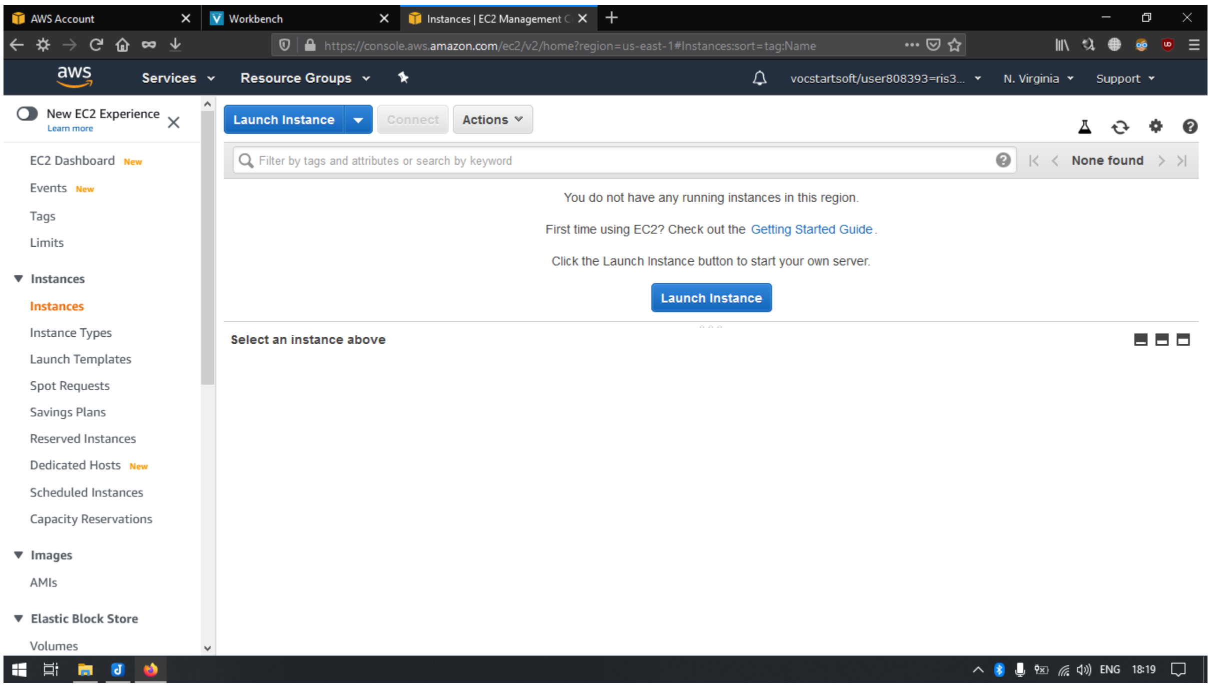
Fig 7.2.13



```
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 101-075-138


VOICE TEXT:--> apply infra 123
INVOKED:--> infra_manager(voicetext)


 INFRA: apply  123

Initializing the backend...

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 2.70"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

```
[root@localhost proj4]# date
Wed Jul 29 08:48:48 EDT 2020
[root@localhost proj4]#
```

Fig 7.2.14



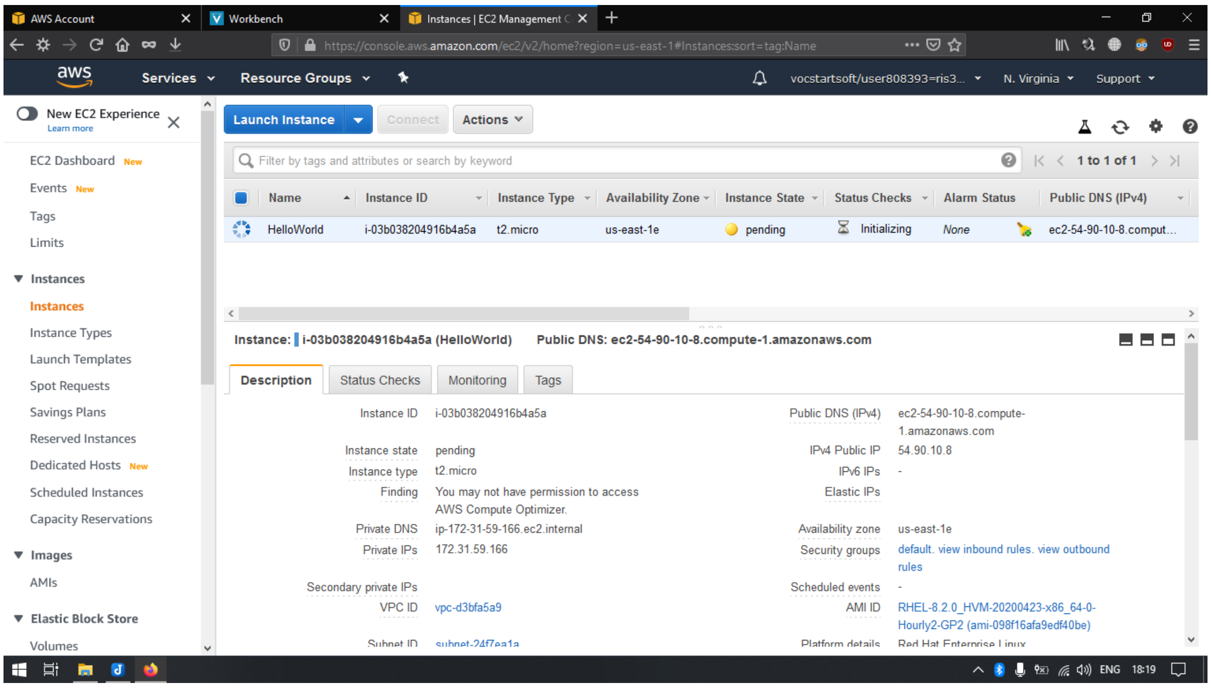
You do not have any running instances in this region.

First time using EC2? Check out the Getting Started Guide.

Click the Launch Instance button to start your own server.

Fig 7.2.15

F

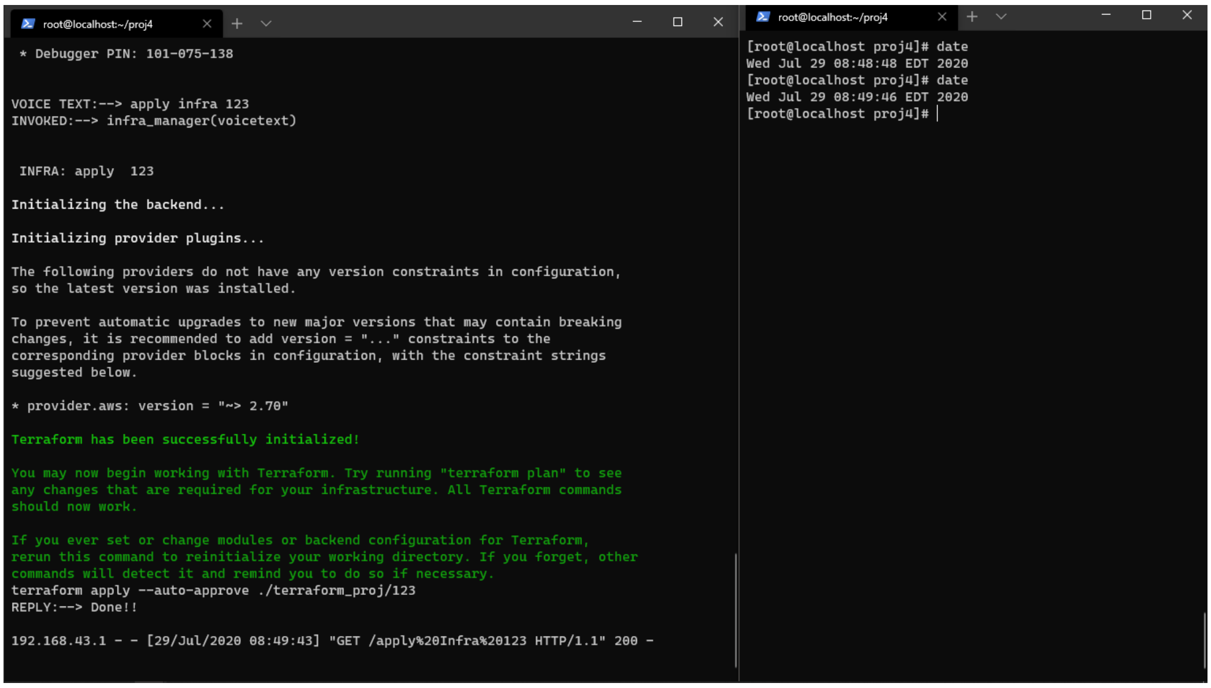


Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
HelloWorld	i-03b038204916b4a5a	t2.micro	us-east-1e	pending	Initializing	None	ec2-54-90-10-8.comput...

Instance: i-03b038204916b4a5a (HelloWorld) Public DNS: ec2-54-90-10-8.compute-1.amazonaws.com

Instance ID	i-03b038204916b4a5a	Public DNS (IPv4)	ec2-54-90-10-8.compute-1.amazonaws.com
Instance state	pending	IPv4 Public IP	54.90.10.8
Instance type	t2.micro	IPv6 IPs	-
Finding	You may not have permission to access AWS Compute Optimizer.	Elastic IPs	
Private DNS	ip-172-31-59-166.ec2.internal	Availability zone	us-east-1e
Private IPs	172.31.59.166	Security groups	default. view inbound rules. view outbound rules
Secondary private IPs		Scheduled events	-
VPC ID	vpc-d3bfa5a9	AMI ID	RHEL-8.2.0_HVM-20200423-x86_64-0-Hourly2-GP2 (ami-098f16afa9edf40be)
Subnet ID	subnet-24f7ea1a	Platform details	Red Hat Enterprise Linux

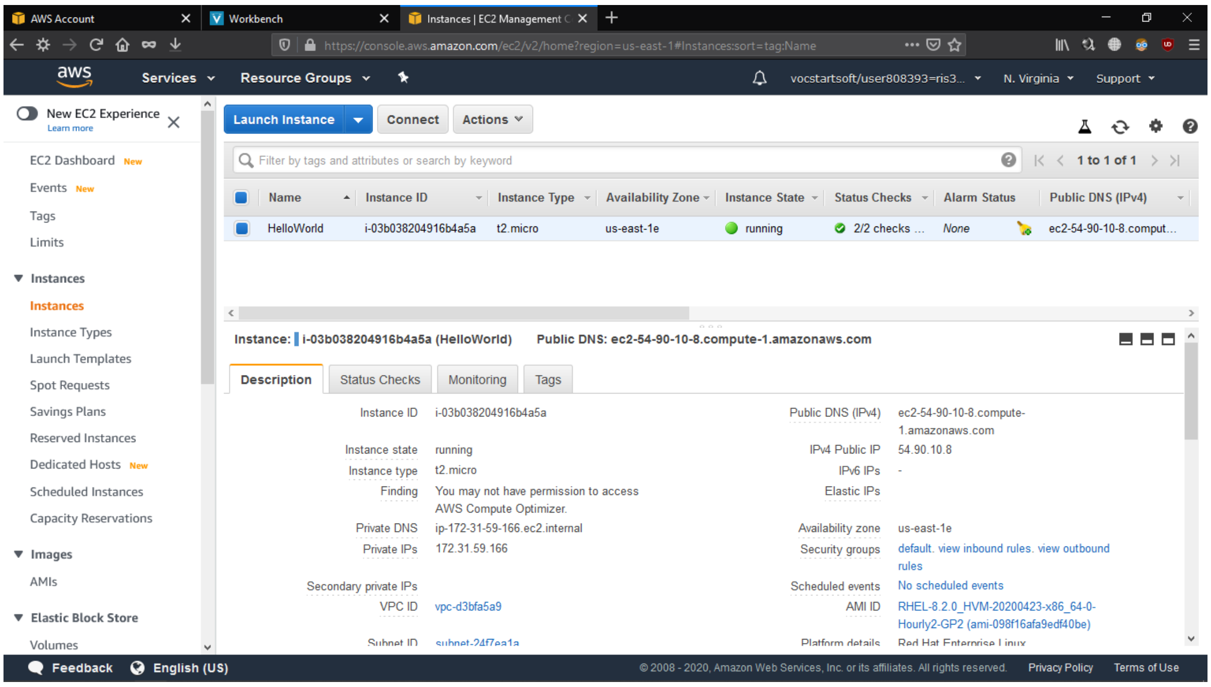
Fig 7.2.16



```
 * Debugger PIN: 101-075-138


VOICE TEXT:--> apply infra 123
INVOKED:--> infra_manager(voicetext)


 INFRA: apply  123

Initializing the backend...

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 2.70"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
terraform apply --auto-approve ./terraform_proj/123
REPLY:--> Done!!

192.168.43.1 - - [29/Jul/2020 08:49:43] "GET /apply%20Infra%20123 HTTP/1.1" 200 -
```

```
[root@localhost proj4]# date
Wed Jul 29 08:48:48 EDT 2020
[root@localhost proj4]# date
Wed Jul 29 08:49:46 EDT 2020
[root@localhost proj4]# |
```

Fig 7.2.17



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
HelloWorld	i-03b038204916b4a5a	t2.micro	us-east-1e	running	2/2 checks ...	None	ec2-54-90-10-8.comput...

Instance: i-03b038204916b4a5a (HelloWorld)   Public DNS: ec2-54-90-10-8.compute-1.amazonaws.com

Description | Status Checks | Monitoring | Tags

Instance ID	i-03b038204916b4a5a	Public DNS (IPv4)	ec2-54-90-10-8.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	54.90.10.8
Instance type	t2.micro	IPv6 IPs	-
Finding	You may not have permission to access AWS Compute Optimizer.	Elastic IPs	
Private DNS	ip-172-31-59-166.ec2.internal	Availability zone	us-east-1e
Private IPs	172.31.59.166	Security groups	default. view inbound rules. view outbound rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-d3bfa5a9	AMI ID	RHEL-8.2.0_HVM-20200423-x86_64-0-Hourly2-GP2 (ami-098f16afa9edf40be)
Subnet ID	subnet-24f7ea1a	Platform details	Red Hat Enterprise Linux

Fig 7.2.18

CHAPTER 8 - CONCLUSION

This tool helps to configure and manage the infrastructure in hybrid effectively with basic voice commands from remote client. This project helps in reducing the stress for the operations team allowing them to configure and deploy the infrastructure in a production ready aspect. The VoiceAIM can dramatically increase the productivity and availability as a less knowledgeable person with the least knowledge of underlying technology can operate the complete infrastructure with some basic voice commands. Pre-scripted templates of ansible and terraform are used to save time and deploy highly configurable and reproducible infrastructure and configurations.

LIMITATIONS

- History of changes is present in the local system.
- Limited number of voice commands can be used.
- Needs a network Connectivity between client app, Listener API, and the cloud or cluster to be configured.

FUTURE ENHANCEMENTS

- Advanced NLP for better knowledge extraction and vivid voice commands.
- Installer Scripts to install the whole setup in minutes.
- Addon for configuring templates on voice commands.

REFERENCES

- "Infrastructure as code: Deploying Terraform with OVH". Infrastructure as code: Deploying Terraform with OVH. 2019-10-09. Archived from the original on 2019-10-10. Retrieved 2019-10-09.
- https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software
- https://en.wikipedia.org/wiki/Ansible
- Simmons, Dan (2003). Ilium (hbk. ed.). New York: Eos/HarperCollins. p. 98. ISBN 0-380-97893-8. I can see Nightenhelser madly taking notes on his recorder ansible.
- Ronacher, Armin. "Opening the Flask" (PDF). Archived from the original (PDF) on 2016-12-17. Retrieved 2011-09-30
- https://pypi.org/project/Flask

APPENDIX / CODE

1. Listener_API.py

```
from flask import Flask
from snippets.packagemanager import package_manager
from snippets.servicemanager import service_manager
from snippets.usermanager import user_manager
from snippets.inframanager import infra_manager
import subprocess
app = Flask(__name__)
def listener(voicetext):
    reply = 'Done!!'
    try:
        if "package" in voicetext:
            task = "package"
        elif "service" in voicetext:
            task = "service"
        elif "user" in voicetext:
            task = "user"
        elif "infra" in voicetext:
            task = "infra"
        print("INVOKED:--> "+ task +"_manager(voicetext)")
        exec(task+"_manager(voicetext)")
    except Exception as e:
        print(f"Invalid Command...Mind your words...\n\n{e}")
    return reply

@app.route("/<command>")
def index(command):
    voicetext = command.lower()
    print("\n\nVOICE TEXT:--> "+ voicetext)
    reply =  listener(voicetext)
    print("REPLY:--> "+ reply, end="\n\n")
    return reply

if __name__ == "__main__":
    host = "0.0.0.0"
    port = 5000
    debug = True
    app.run(host, port, debug)
```

2. Snippets/inframanager.py

```
import subprocess


def infra_manager(voicetext):
    text = voicetext.strip().lower().split()
    todo = text[text.index("infra")-1]
    infra_code = text[text.index("infra")+1]

    if todo in ("create", "apply"):
        todo = "apply"
    elif todo in ("remove", "destroy", "delete"):
        todo = "destroy"

    print("\n\n INFRA: "+ todo +"  "+ infra_code)
    init = subprocess.getoutput(f"terraform init ./terraform_proj/{infra_code}")
    print(init)
    print(f"terraform {todo} --auto-approve ./terraform_proj/{infra_code}")

    output = subprocess.getoutput(f"terraform {todo} --auto-approve ./terraform_proj/{infra_code}")
    return(output)
```

3. Snippets/packagemanager.py

```
import subprocess

def package_manager(voicetext):
    text = voicetext.strip().lower().split()
    name = text[text.index("package")+1]
    state = text[text.index("package")-1]
    if state in ("install", "present", "download"):
        state = "present"
    else:
        state = "absent"

    ansible_data = '''
    - hosts: localhost
      tasks:
        - name: "Handling package... "
          package:
            name: "{0}"
            state: "{1}"
    '''.format(name, state)

    with open("./playbooks/package_manager.yml","w") as f:
        f.write(ansible_data)

    output = subprocess.getoutput("ansible-playbook ./playbooks/package_manager.yml")
    return(output)
```

4. Snippets/Servicemanager.py

```
import subprocess


def service_manager(voicetext):
    text = voicetext.strip().lower().split()
    name = text[text.index("service")+1]
    state = text[text.index("service")-1]

    if state in ("start", "started"):
        state = "started"
    elif state in ("stopped","stop"):
        state = "stopped"
    elif state in ("restart", "restarted"):
        state = "restarted"
    ansible_data = '''
    - hosts: localhost
      tasks:
        - name: "Handling Service... "
          service:
            name: "{0}"
            state: "{1}"
    '''.format(name, state)

    with open("./playbooks/service_manager.yml","w") as f:
        f.write(ansible_data)
    output = subprocess.getoutput("ansible-playbook ./playbooks/service_manager.yml")
    return(output)
```

5. Snippets/Usermanager.py

```
import subprocess

def user_manager(voicetext): #create user harry
    text = voicetext.strip().lower().split()
    name = text[text.index("user")+1]
    action = text[text.index("user")-1]

    if action in ("create", "add"):
        ansible_data = '''
- hosts: localhost
  tasks:
    - name: "Adding User..."
      user:
        name: "{}"
        generate_ssh_key: yes
        ssh_key_bits: 2048
        ssh_key_file: .ssh/id_rsa
        '''.format(name)

    elif action in ("remove", "delete"):
        ansible_data = '''
- hosts: localhost
  tasks:
    - name: "removing User..."
      user:
        name: "{}"
        state: absent
        remove: yes
        '''.format(name)

    with open("./playbooks/user_manager.yml","w") as f:
        f.write(ansible_data)

    output = subprocess.getoutput("ansible-playbook ./playbooks/user_manager.yml")
    return(output)
```

6. Terraform_proj/123/123.tf

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "web" {
  ami           = "ami-098f16afa9edf40be"
  instance_type = "t2.micro"

  tags = {
    Name = "HelloWorld"
  }
}
```

7. Terraform_proj/web/web.tf

```
# Defining the Provider of the Terraform


provider "aws" {
        access_key = "${var.access_key}"
        secret_key = "${var.secret_key}"


        region = "${var.region}"
}


# Defining the VPC


resource "aws_vpc" "terraform_vpc" {
  cidr_block = "10.0.0.0/26"
  instance_tenancy = "default"
  enable_dns_hostnames = true


  tags = {
    Name = "terraform_vpc"
  }
}


# Defining the VPC Subnets


resource "aws_subnet" "pub-sub1" {
  cidr_block = "10.0.0.0/28"
  vpc_id = "${aws_vpc.terraform_vpc.id}"
  availability_zone = "${var.pub_sub_az}"
  map_public_ip_on_launch = true
  tags = {
    Name = "pub-sub-A"
  }
}

/*
resource "aws_subnet" "pub-sub2" {
  cidr_block = "10.0.0.16/28"
  vpc_id = "${aws_vpc.terraform_vpc.id}"
  availability_zone = "us-east-1b"
  map_public_ip_on_launch = true

  tags = {
    Name = "pub-sub-B"
  }
}

resource "aws_subnet" "pri-sub1" {
  cidr_block = "10.0.0.32/28"
  vpc_id = "${aws_vpc.terraform_vpc.id}"
  availability_zone = "us-east-1c"

  tags = {
    Name = "pri-sub-A"
  }
}

*/
```

```
resource "aws_subnet" "pri-sub2" {
  cidr_block = "10.0.0.48/28"
  vpc_id = "${aws_vpc.terraform_vpc.id}"
  availability_zone = "${var.pri_sub_az}"

  tags = {
    Name = "pri-sub-B"
  }
}

# Defining the VPC Internet Gateway

resource "aws_internet_gateway" "igw" {
  vpc_id = "${aws_vpc.terraform_vpc.id}"

  tags = {
    Name = "terraform_vpc_igw"
  }
}

# Defining the Elastic IP Address for NAT

resource "aws_eip" "nat" {
vpc      = true
}

# Defining the VPC NAT Gateway

resource "aws_nat_gateway" "terraform_vpc_nat" {
  allocation_id = "${aws_eip.nat.id}"
  subnet_id     = "${aws_subnet.pub-sub1.id}"
  depends_on = ["aws_internet_gateway.igw"]
  tags = {
    Name = "terraform_vpc_nat"
  }
}

# Defining the route table for public subnet

resource "aws_route_table" "pub-route" {
  vpc_id = "${aws_vpc.terraform_vpc.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_internet_gateway.igw.id}"
  }

  tags = {
    Name = "Public_Route"
  }
}

# Associating the Public subnet to the Internet exposed route table

resource "aws_route_table_association" "aws_rt_association" {
  route_table_id = "${aws_route_table.pub-route.id}"
  subnet_id = "${aws_subnet.pub-sub1.id}"
}

# Defining the route table for private subnet
```

```
resource "aws_route_table" "pri-route" {
  vpc_id = "${aws_vpc.terraform_vpc.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_nat_gateway.terraform_vpc_nat.id}"
  }

  tags = {
    Name = "Private_Route"
  }
}

# Associating the Private subnet to the NAT exposed route table

resource "aws_route_table_association" "aws_rt_association2" {
  route_table_id = "${aws_route_table.pri-route.id}"
  subnet_id = "${aws_subnet.pri-sub2.id}"
}



# Defining the Public Subnet Security Group

resource "aws_security_group" "sg_public" {
  name = "sg_public"
  description = "Allowing Internet Access"
  vpc_id = "${aws_vpc.terraform_vpc.id}"
  tags = {
    Name = "sg_public_subnet"
  }

  ingress {
    from_port = 80
    to_port = 80
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port = 22
    to_port = 22
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

    egress {
    from_port       = 0
    to_port         = 0
    protocol        = "-1"
    cidr_blocks     = ["0.0.0.0/0"]
  }

}

# Defining the Private Subnet Security Group

resource "aws_security_group" "sg_private" {
  name = "sg_private"
  description = "Restricted Access"
  vpc_id = "${aws_vpc.terraform_vpc.id}"

  tags = {
    Name = "sg_private_subnet"
  }

  ingress {
    from_port = 22
    to_port = 22
    protocol = "tcp"
    cidr_blocks = [
      aws_subnet.pub-sub1.cidr_block]
  }

  egress {
    from_port = 0
    to_port = 0
    protocol = "-1"
    cidr_blocks = [
      "0.0.0.0/0"]
  }
}

# Creating the aws ec2 instance in the Public Subnet



resource "aws_instance" "web-instance" {
        ami = "${var.aws_ami}"
```

```
                    subnet_id = "${aws_subnet.pub-sub1.id}"
        instance_type = "t2.micro"
        key_name = "${var.key_name}"
        user_data = "${file("httpd.sh")}"
        vpc_security_group_ids = ["${aws_security_group.sg_public.id}"]
  tags = {
    Name = "web-instance"
  }
}
```

# Creating the aws ec2 instance in the Private Subnet with SSH accessible from Public Subnet CIDR

```
resource "aws_instance" "private-instance" {
        ami = "${var.aws_ami}"
                    subnet_id = "${aws_subnet.pri-sub2.id}"
        instance_type = "t2.micro"
        key_name = "${var.key_name}"
        vpc_security_group_ids = ["${aws_security_group.sg_private.id}"]
  tags = {
    Name = "private-instance"
  }
}
```

8. Terraform_proj/web/var.tf

```
variable "access_key" {
description = "AWS Access key"
default = "......................."
}

variable "secret_key" {
description = "AWS Secret Key"
default = "........................"
}



variable "region" {
description = "AWS region for hosting our your network"
default = "ap-south-1"
}



variable "aws_ami" {
description = "AWS region for hosting our your network"
default = "ami-02d55cb47e83a99a0"
}



variable "key_name" {
description = "Key name for SSH into EC2"
default = "railway"
}
```

```
variable "pub_sub_az" {

description = "Key name for SSH into EC2"

default = "ap-south-1a"

}


variable "pri_sub_az" {

description = "Key name for SSH into EC2"

default = "ap-south-1b"

}
```

Client Android App code

1. **Main.dart**

```
import 'dart:async';

import 'package:flutter/material.dart';
import 'package:speech_to_text/speech_recognition_error.dart';
import 'package:speech_to_text/speech_recognition_result.dart';
import 'package:speech_to_text/speech_to_text.dart';
import 'package:http/http.dart' as http;

void main() => runApp(MyApp());

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  bool _hasSpeech = false;
  bool _stressTest = false;
  double level = 0.0;
  int _stressLoops = 0;
  String lastWords = "";
  String lastError = "";
  String lastStatus = "";
  String _currentLocaleId = "";
  List<LocaleName> _localeNames = [];
  final SpeechToText speech = SpeechToText();
  final name_controller = TextEditingController();
  String url = '';

  @override
  void initState() {
    super.initState();
  }

  Future<void> initSpeechState() async {
    bool hasSpeech = await speech.initialize(
        onError: errorListener, onStatus: statusListener);
    if (hasSpeech) {
      _localeNames = await speech.locales();

      var systemLocale = await speech.systemLocale();
      _currentLocaleId = systemLocale.localeId;
    }
  if (!mounted) return;

  setState(() {
    _hasSpeech = hasSpeech;
  });
}

@override
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: const Text('my Voice Cloud'),
      ),
      body: Column(children: [
        Center(
          child: Text(
            'Speech recognition available',
            style: TextStyle(fontSize: 22.0),
          ),
        ),
        Center(
          child: TextField(
            textAlign: TextAlign.center,
            controller: name_controller,
            decoration: InputDecoration(
              border: InputBorder.none,
              hintText: 'Server Name/IP'
            ),
          ),
        ),
        Container(
          child: Column(
            children: <Widget>[
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceAround,
                children: <Widget>[
                  FlatButton(
                    child: Text('Initialize'),
                    onPressed: _hasSpeech ? null : initSpeechState,
                  ),
                  FlatButton(
                    child: Text('Stress Test'),
                    onPressed: stressTest,
                  ),
        ],
      ),
      Row(
        mainAxisAlignment: MainAxisAlignment.spaceAround,
        children: <Widget>[
          FlatButton(
            child: Text('Start'),
            onPressed: !_hasSpeech || speech.isListening
                ? null
                : startListening,
          ),
          FlatButton(
            child: Text('Stop'),
            onPressed: speech.isListening ? stopListening : null,
          ),
          FlatButton(
            child: Text('Cancel'),
            onPressed: speech.isListening ? cancelListening : null,
          ),
        ],
      ),
      Row(
        mainAxisAlignment: MainAxisAlignment.spaceAround,
        children: <Widget>[
          DropdownButton(
            onChanged: (selectedVal) => _switchLang(selectedVal),
            value: _currentLocaleId,
            items: _localeNames
                .map(
                  (localeName) => DropdownMenuItem(
                    value: localeName.localeId,
                    child: Text(localeName.name),
                  ),
                )
                .toList(),
          ),
        ],
      )
    ],
  ),
),
Expanded(
  flex: 4,
  child: Column(
    children: <Widget>[
      Center(
        child: Text(
          'Recognized Words',
          style: TextStyle(fontSize: 22.0),
        ),
      ),
      Expanded(
        child: Stack(
          children: <Widget>[
            Container(
              color: Theme.of(context).selectedRowColor,
              child: Center(
                child: Text(
                  lastWords,      /* useless variable for label speech text*/
                  textAlign: TextAlign.center,
                ),
              ),
            ),
            Positioned.fill(
              bottom: 10,
              child: Align(
                alignment: Alignment.bottomCenter,
                child: Container(
                  width: 40,
                  height: 40,
                  alignment: Alignment.center,
                  decoration: BoxDecoration(
                    boxShadow: [
                      BoxShadow(
                          blurRadius: .26,
                          spreadRadius: level * 1.5,
                          color: Colors.black.withOpacity(.05))
                    ],
                    color: Colors.white,
                    borderRadius:
                        BorderRadius.all(Radius.circular(50)),
                  ),
                  child: IconButton(icon: Icon(Icons.mic)),
                ),
              ),
            ),
          ],
        ),
      ),
    ],
  ),
),
      Expanded(
        flex: 1,
        child: Column(
          children: <Widget>[
            Center(
              child: Text(
                'Error Status',
                style: TextStyle(fontSize: 22.0),
              ),
            ),
            Center(
              child: Text(lastError),
            ),
          ],
        ),
      ),
      Container(
        padding: EdgeInsets.symmetric(vertical: 20),
        color: Theme.of(context).backgroundColor,
        child: Center(
          child: speech.isListening
              ? Text(
                  "I'm listening...",
                  style: TextStyle(fontWeight: FontWeight.bold),
                )
              : Text(
                  'Not listening',
                  style: TextStyle(fontWeight: FontWeight.bold),
                ),
        ),
      ),
    ]),
  ),
);
}

void stressTest() {
if (_stressTest) {
  return;
}
_stressLoops = 0;
_stressTest = true;
print("Starting stress test...");
startListening();
}
```

```
void changeStatusForStress(String status) {
  if (!_stressTest) {
    return;
  }
  if (speech.isListening) {
    stopListening();
  } else {
    if (_stressLoops >= 100) {
      _stressTest = false;
      print("Stress test complete.");
      return;
    }
    print("Stress loop: $_stressLoops");
    ++_stressLoops;
    startListening();
  }
}

void startListening() {
  lastWords = "";
  lastError = "";
  speech.listen(
      onResult: resultListener,
      listenFor: Duration(seconds: 10),
      localeId: _currentLocaleId,
      onSoundLevelChange: soundLevelListener,
      cancelOnError: true,
      partialResults: true);
  setState(() {});
}

void stopListening() {
  speech.stop();
  setState(() {
    level = 0.0;
  });
}

void cancelListening() {
  speech.cancel();
  setState(() {
    level = 0.0;
  });
}

void resultListener(SpeechRecognitionResult result) {
    setState(() {
      lastWords = "${result.recognizedWords} - ${result.finalResult}";
      if(result.finalResult == true){   //* print the true values only *//
        // print(lastWords);
        // print(name_controller.text);
        url = "http://"+name_controller.text+":5000/"+result.recognizedWords;
        print(url);
        var res = http.get(Uri.encodeFull(url));
        // print(res);
      }
    });
  }

  void soundLevelListener(double level) {
    setState(() {
      this.level = level;
    });
  }

  void errorListener(SpeechRecognitionError error) {
    setState(() {
      lastError = "${error.errorMsg} - ${error.permanent}";
    });
  }

  void statusListener(String status) {
    changeStatusForStress(status);
    setState(() {
      lastStatus = "$status";
    });
  }

  _switchLang(selectedVal) {
    setState(() {
      _currentLocaleId = selectedVal;
    });
    print(selectedVal);
  }
}
```

2. Pubsec.yml

```
name: voice_cloud

description: A new Flutter project.


# The following defines the version and build number for your application.

# A version number is three numbers separated by dots, like 1.2.43

# followed by an optional build number separated by a +.

# Both the version and the builder number may be overridden in flutter

# build by specifying --build-name and --build-number, respectively.

# In Android, build-name is used as versionName while build-number used as versionCode.

# Read more about Android versioning at https://developer.android.com/studio/publish/versioning

# In iOS, build-name is used as CFBundleShortVersionString while build-number used as CFBundleVersion.

# Read more about iOS versioning at

# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html

version: 1.0.0+1


environment:

  sdk: ">=2.1.0 <3.0.0"


dependencies:

  http: ^0.12.0+4

  speech_to_text: ^2.1.0

  flutter:

    sdk: flutter


  # The following adds the Cupertino Icons font to your application.

  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^0.1.2

dev_dependencies:
  flutter_test:
    sdk: flutter


# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  # assets:
  #  - images/a_dot_burr.jpeg
  #  - images/a_dot_ham.jpeg

  # An image asset can refer to one or more resolution-specific "variants", see
  # https://flutter.dev/assets-and-images/#resolution-aware.

  # For details regarding adding assets from package dependencies, see
  # https://flutter.dev/assets-and-images/#from-packages

  # To add custom fonts to your application, add a fonts section here,
  # in this "flutter" section. Each entry in this list should have a
  # "family" key with the font family name, and a "fonts" key with a
  # list giving the asset and other descriptors for the font. For
  # example:
  # fonts:
  #   - family: Schyler
  #     fonts:
  #       - asset: fonts/Schyler-Regular.ttf
  #       - asset: fonts/Schyler-Italic.ttf
  #         style: italic
  #   - family: Trajan Pro
  #     fonts:
  #       - asset: fonts/TrajanPro.ttf
  #       - asset: fonts/TrajanPro_Bold.ttf
  #         weight: 700
  #
  # For details regarding fonts from package dependencies,
  # see https://flutter.dev/custom-fonts/#from-packages
```

3. Build.gradle – android

```
def localProperties = new Properties()
def localPropertiesFile = rootProject.file('local.properties')
if (localPropertiesFile.exists()) {
    localPropertiesFile.withReader('UTF-8') { reader ->
        localProperties.load(reader)
    }
}

def flutterRoot = localProperties.getProperty('flutter.sdk')
if (flutterRoot == null) {
    throw new GradleException("Flutter SDK not found. Define location with flutter.sdk in the local.properties file.")
}

def flutterVersionCode = localProperties.getProperty('flutter.versionCode')
if (flutterVersionCode == null) {
    flutterVersionCode = '1'
}

def flutterVersionName = localProperties.getProperty('flutter.versionName')
if (flutterVersionName == null) {
    flutterVersionName = '1.0'
}

apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply from: "$flutterRoot/packages/flutter_tools/gradle/flutter.gradle"

android {
    compileSdkVersion 28

    sourceSets {
        main.java.srcDirs += 'src/main/kotlin'
    }

    lintOptions {
        disable 'InvalidPackage'
    }

    defaultConfig {
        // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).
        applicationId "com.example.voice_cloud"
        minSdkVersion 21
        targetSdkVersion 28
        versionCode flutterVersionCode.toInteger()
        versionName flutterVersionName
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            // TODO: Add your own signing config for the release build.
            // Signing with the debug keys for now, so `flutter run --release` works.
            signingConfig signingConfigs.debug
        }
    }
}

flutter {
    source '../..'
}

dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test:runner:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
}
```