INFRASTRUCTURE AS CODE

Using Terraform and Ansible

Cloud Computing Project

By (Group 9):

Faiz Usmani	1323197
Parag Tambalkar	1322596
Pranay Raman	1321759
Shubham Girdhar	1323003

Supervised by: Prof. Dr. Christian Baun

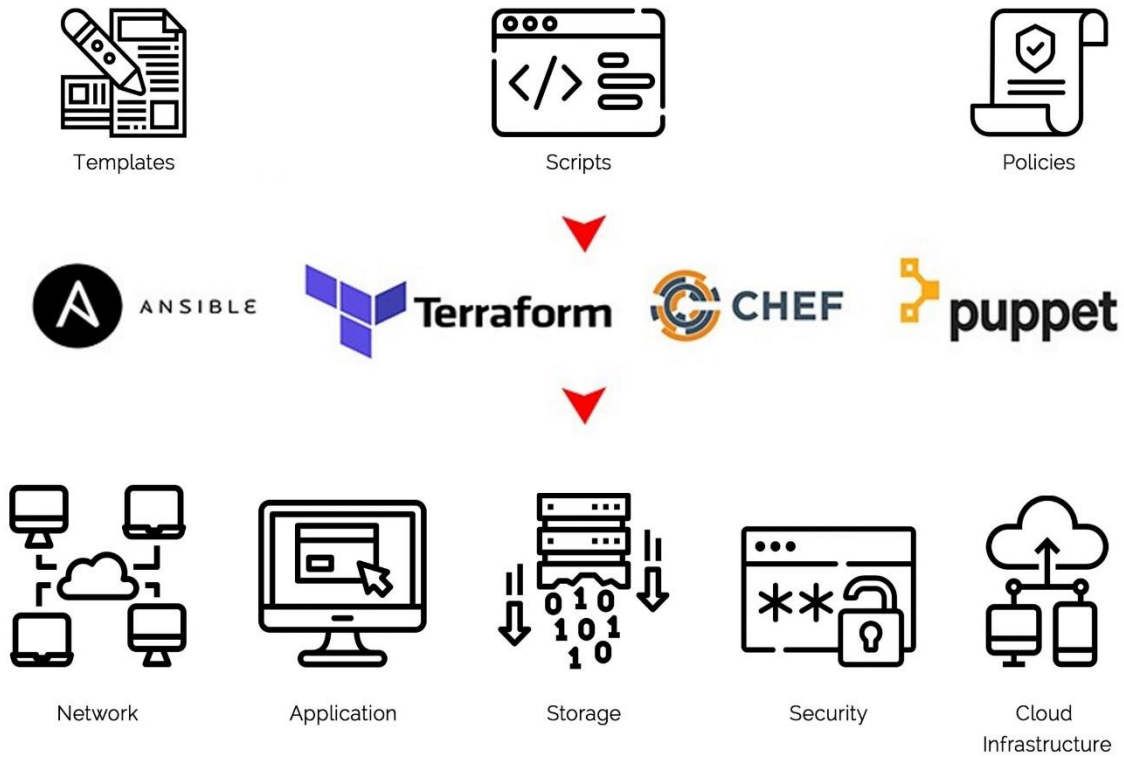


FIGURE 1

What is IaC ?

- IaC enables us to provision and configure the virtual servers on cloud and onsite physical servers easily in minutes just by executing a script written in a user-friendly language.
- Allows us to deal with servers, networks, security groups, databases, etc. as if they are a part of the software.
- The leading solutions in the market nowadays for IaC are provided by Terraform, Ansible, Chef, Puppet.
- Some Advantages:
  - Code Once, Iterate Multiple times
  - Immune to Human error
  - Scaling is as easy as it gets

Understanding the tools (1)

TERRAFORM

- A tool for building, changing, and versioning infrastructures safely and efficiently.
- A starting project mainly has three kind of files – main.tf, variables.tf, terraform.tfvars.
- The configuration files are written in HashiCorp Configuration Language (HCL). Terraform then goes ahead and produce an execution plan describing the steps to reach the desired state, and then executes it.
- Also, the main difference between Terraform and the other IaC tools is that it does not re-provision resources that are successfully provisioned.

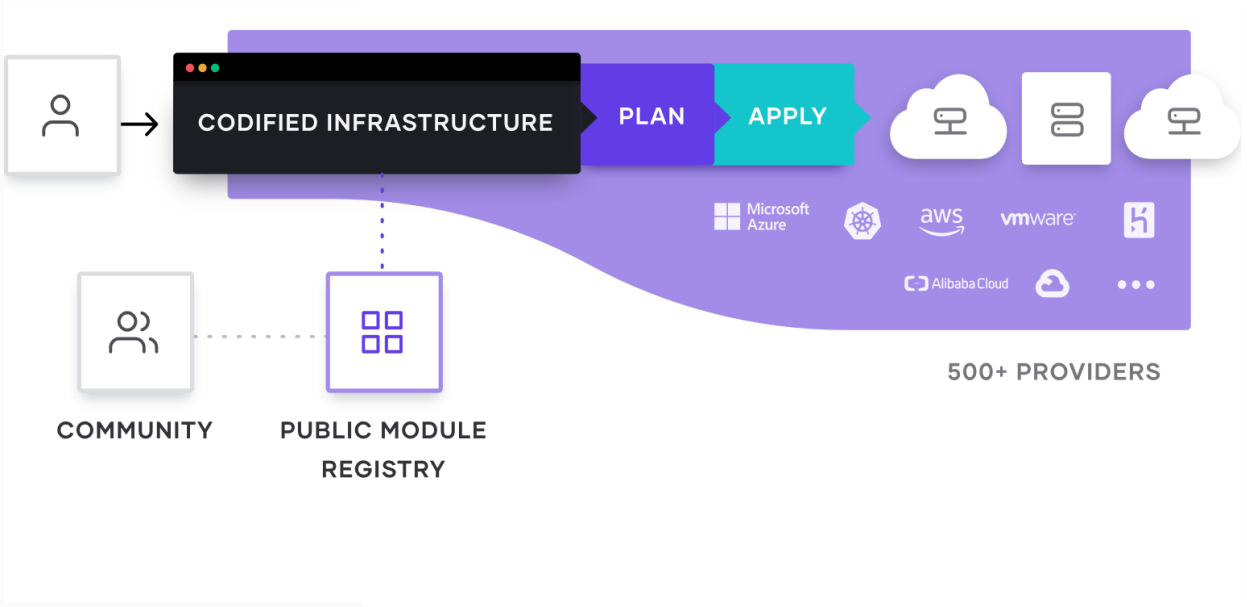


FIGURE 2

Understanding the tools (2)

ANSIBLE

- The Configuration expert although Ansible is a tool that can do a lot more than configuring existing infrastructures. It is primarily known for the configuration tasks as it is very easy to do it using Ansible.
- The primary architecture of Ansible is shown in Figure 3 which consists of two kinds of nodes – Control and Managed nodes.
- The Control node has the list of IP addresses of the managed nodes and sends out Ansible modules to them to configure the managed nodes to reach a desired state.

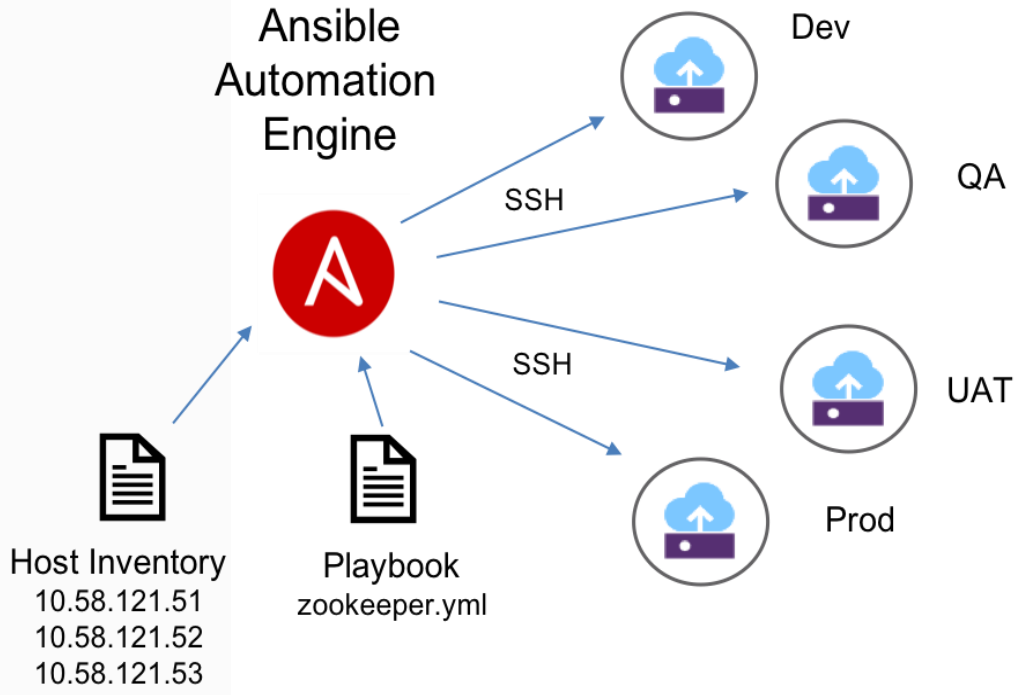


FIGURE 3

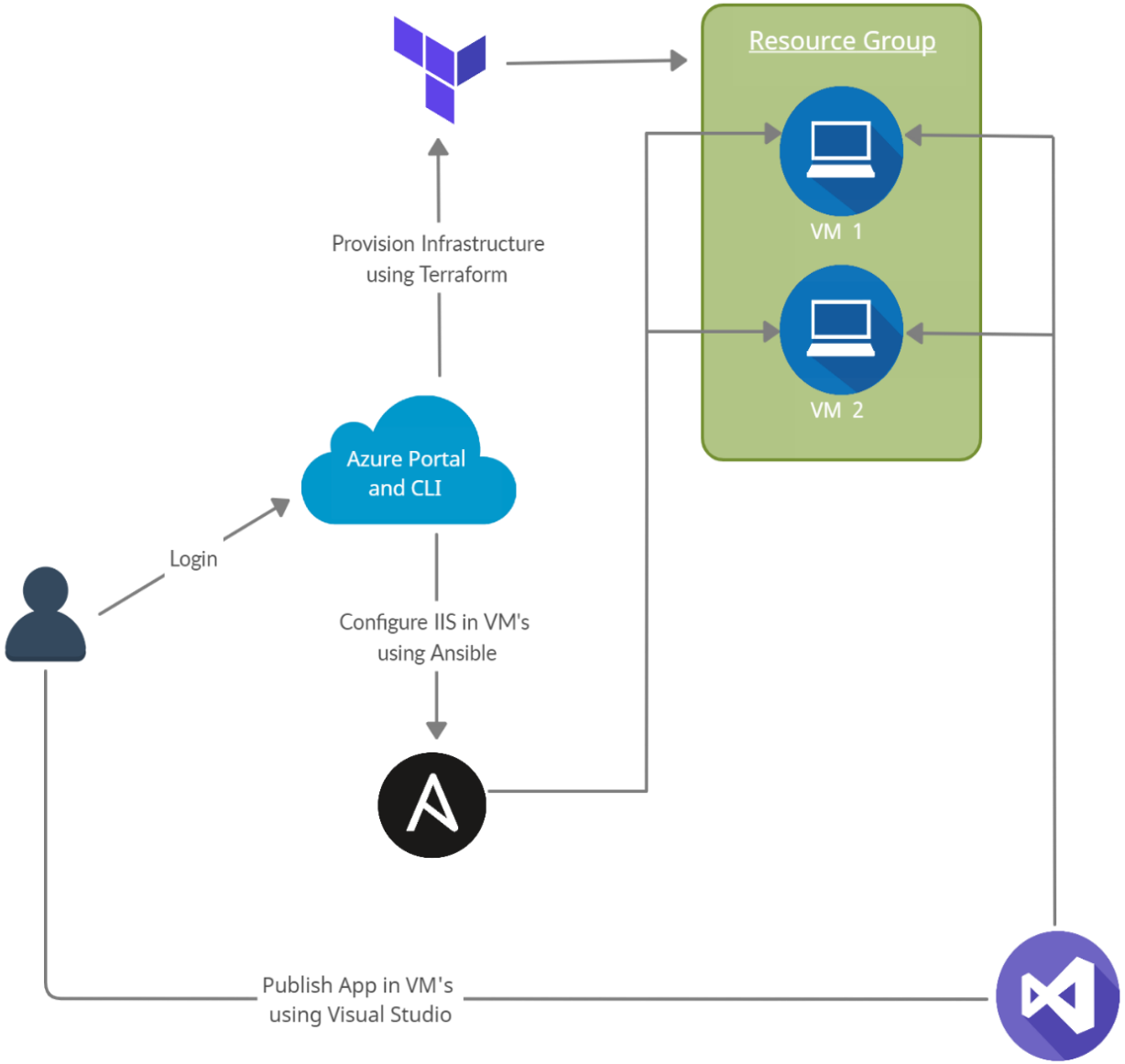
Understanding the tools (3)

AZURE AND CLOUD SHELL

- The cloud computing service provider by Microsoft supporting multitudes of services.
- Cloud shell is a browser integrated shell that is used to manage resources on Azure. As Azure has built in support for Terraform and Ansible, we leveraged this to run terraform and Ansible scripts.

VISUAL STUDIO

- Being a Microsoft product, used this powerful IDE because of the ease it offers to directly deploy apps to the VM's hosted on Azure.



Deployment Plan

- The user will login into the Azure portal and use Azure Cloud Shell to run Terraform and Ansible scripts.
- Terraform scripts are run and a resource group with two Windows VMs having proper network configurations are created.
- After the successful creation of VMs, IIS (Internet Information Services – a web server software package specifically designed for windows) and some other modules are configured on both VMs using their IP's. They are required to run an ASP.NET web app on a Windows server.
- A self-created web app or app cloned from the provided GitHub link is published on both VMs using Visual Studio 2019 and accessed through the public IP of the load balancer.

Provisioning Infrastructure with Terraform

Major Components of Terraform are:

- Terraform Infrastructure Configuration coded in below files,

  - Main.tf
  - terraform.tfvars
  - variables.tf

- Process of Building Cloud Infrastructure using Terraform configuration code for which we need below three important commands,

  - terraform init
  - terraform plan
  - terraform apply

Running IIS using Ansible

Main Components of Ansible:

- Ansible playbook: Main script with the set of instructions that need to be implemented on multiple hosts
- Inventory: Part that maintains the structure of the network environment.

Invoking Ansible Playbook:

- Making a new directory and get the ansible playbook and inventory file from GitHub.
- Update the inventory with IP addresses of the VMs provisioned by Terraform.
- Execute the ansible-playbook command to do the configuration.

Deploying the Web Application

- By this point, we have the Infrastructure built using Terraform and configured with Ansible. The next step is to deploy the application.

Prerequisites:

- Visual Studio installed with the workloads- ASP.NET Core cross-platform development, ASP.NET web development, Azure development.

Wrap Up

Concluding Remarks:

- We were able to provision infrastructure using Terraform, configure the infrastructure using Ansible and finally deploy a .net core web application on our infrastructure.

Next Steps…

- Further automation into a CI/CD (Continuous Integration/Continuous Deployment) pipeline.
- Replace manual execution of Terraform and Ansible scripts with triggering build and release pipelines.
- While the process of configuring these steps into pipelines could be a little tricky, once achieved, this could make the life of the DevOps or cloud engineer much easier.

References :

- Figure 1: https://www.suntechnologies.com/wp-content/uploads/2020/04/image-1-2-scaled.jpg
- Figure 2: https://www.terraform.io/_next/static/images/how-terraform-works-33c33a9c82bf5aef0bf9f75eb5f5f9b2.svg
- Figure 3: : Ansible_ov.png (904×602) (s81c.com)

Thank You