Terraform is an open-source infrastructure-as-code (IaC) tool that allows you to define and manage your infrastructure in a declarative way. It provides a way to create, modify, and destroy infrastructure resources such as virtual machines, networks, storage, and other cloud resources.

In DevOps, Terraform can be used to automate the provisioning and management of infrastructure resources for application deployment and testing. With Terraform, you can define your infrastructure as code, version control it, and apply changes to it as needed. This helps to reduce errors and ensure consistency in your infrastructure deployments.

Terraform integrates with various cloud providers, including Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), and others. This makes it a powerful tool for managing cloud infrastructure across multiple platforms.

Overall, Terraform can help streamline the deployment and management of infrastructure resources in a DevOps environment, making it a popular choice for many organizations.

Facts:

1. IAAC | Automate Infrastructure
2. Defines Infrastructure State
3. Ansible, puppet or chef automates mostly OS related tasks

- Defines Machines state

1. Terraform automates infra itself:

- like AWS, GCP, AZURE, DIGITAL OCEAN ETC

1. Terraform works with automation softwares like ansible to give a powerful automation, after infra is setup and ready
2. No Programming knowledge is needed, its own syntax is similar to JSON

Installation guides:

1. open <https://www.terraform.io> click on downloads
2. scroll down and select your operating system (Windows 64bits)

* windows = extract the downloaded file and save in your C:/ drive
* Linux = /usr/local/bin

1. Windows is the easiest way, open your powershell as administrator

choco install terraform (These would install it complete) copy the save path

* Click on start and search for Environment variable
* click on Environment variable under startup and recovery
* click on path and click on edit
* click on new, type: C:\ProgramData\chocolatey\lib\terraform\tools (Check the powershell for the path)
* click on ok

Or:

Download the file, unzip it to your C;/ drive

* Click on start and search for Environment variable
* click on Environment variable under startup and recovery
* click on path and click on edit
* click on new, type: c:\terraform
* click on ok

1. **Authentication:**
2. In your powershell: choco install awscli -y
3. Once the installation is complete, Go to your AWS console and search for IAM User
4. click on Add user:

User name = terradmin

check on programmatic access and click on Next

Click on Attach existing policies and check Administrator Access

click on Next, give a Tag & Value if necessary

click on next and the click on create

download and store the credentials safely. You will need it for step 9 below

1. Go to GitBash
2. aws configure

AWS access key ID = paste from your IAM User .csv file saved

AWS Secret Access key = paste from your IAM User .csv file saved

Default region name = insert the current region

Default output format = json

1. terraform - -help
2. **USING TERRAFORM TO CREATE AN EC2 INSTANCE**
3. cd c:/
4. mkdir terraform-scripts
5. cd terraform-scripts/
6. mkdir exercise1
7. cd exercise1/
8. open Vs Code and select the folder path you created above:

* Study the file: first\_instance.tf (C:\gitrepos\cloudcodesandsecurity\DevOps\Terraform\terraform-exercises\exercise1)
* create: the keypair and Security groups manually, Automatic creation will be taught below
* custom TCP / 22 / My IP
* security group ID is what we used in the terraform file
* save it

1. cd exercise1/ (make sure you are in the exercise1 dir)
2. terraform init (this is to run the code. It can be in powershell or GitBash)
3. ls
4. ls .terraform/plugins/
5. terraform validate (to check for errors in the code)
6. terraform fmt (helps to format the code, rearranging it neatly)
7. terraform plan (show the execution steps to be carried out)
8. terraform apply (If you are OK with the settings)
9. Go to your AWS account and check the EC2 instance if created
10. **To clean up**
11. terraform destroy
12. **WORKING WITH VARIABLES**

Variables are used:

* To Move secrets to another file
* Values that change: (AMI, tags, keypair etc)
* Reuse your code

1. open Vs Code and select the folder path

* Study the file: var.tf, instance.tf and providers.tf (C:\gitrepos\cloudcodesandsecurity\DevOps\Terraform\terraform-exercises\exercise2)

1. Use GitBash
2. cd /c/terraform-scripts/exercise2/
3. ls
4. terraform init (this is to run the code. It can be in powershell or GitBash)
5. ls
6. ls .terraform/plugins/
7. terraform validate (to check for errors in the code)
8. terraform fmt (helps to format the code, rearranging it neatly)
9. terraform plan (show the execution steps to be carried out)
10. terraform apply (If you are OK with the settings)
11. Go to your AWS account and check the EC2 instance if created
12. **WORKING WITH PROVISIONING:**
13. k