**Devops**

* **Ansible-** Configuration Management
* **Docker-** Virtualization
* **Kubernetes:** Managing Virtualization
* **GitHub:** Code Management
* **Terraform:** Infrastructure as code (IAC)
* **Shell Scripting**

**Main Topics**

* Ansible
* Docker
* Kubernetes
* terraform
* Shell Scripting

**Good to have.**

* Jenkins
* Chef
* Puppet
* Ant
* Maven
* Cradle

My Infrastructure was completely automated.no Human intervention**---- Terraform or IAC**

**TERRAFORM OR IAC:**

* Open-Source Tool
* Terraform commands: **init, plan, apply.**
* Terraform is a tool which will automatically create AWS/Azure/GCP infrastructure
* Terraform Script or Terraform File
* Terraform v/s AWS CLI
* Version 1.4.6

**Example**

1. **One Network**
2. **Four subnets-** Two Public Two private
3. **Two Public-** should connect to both Internet gateways and egress gateways.
4. **Private Subnets –** Should connect to NAT gateways. **(NAT Gateway Require Elastic IP’s)**
5. **Public one -**enable to everyone**.**
6. **Private one -**enable to one outbound IP.
7. Create two instances in each subnet (both Private and Public) and having tag name as I1, I2, I3, I4 and **create security group to allow Port Number**
8. **Private Subnet**: Create instance with AMI have No SQL which will allow Port number 3306 and IP 10.10.10.32
9. **All this infrastructure** creates in Mumbai, Singapore, North Virginia and Frankfurt

* **Egress Gateway**
  + **assign\_generated\_ipv6\_cidr\_block = true s**hould be mentioned in while creating VPC in case pf Egress Gateway creation
* **NAT gateways** are at Subnet level, so only one private subnet can be attached or at max only one subnet, and **Internet Gateways** are at VPC level, so multiple subnets can be attached.
* **Elastic IP** does not depend on VPC, EIP is public IP whereas **Private IP** depends on **VPC.**
* **Route Tables**

**Concepts of Terraform**

1. AWS Console
2. AWS CLI (Command line Interface)
3. Using Lambda function
4. Cloud formation Templates
5. Provisioning and managing infrastructure IAAC (Infrastructure As a Code)- Terraform -AWS, GCP, Azure

**Commands**

**Main commands:**

* **init Prepare your working directory for other commands**
* **validate Check whether the configuration is valid**
* **plan Show changes required by the current configuration**
* **apply Create or update infrastructure.**
* **destroy Destroy previously created infrastructure**

**Other Commands:**

* console Try Terraform expressions at an interactive command prompt
* fmt Reformat your configuration in the standard style
* force-unlock Release a stuck lock on the current workspace
* get Install or upgrade remote Terraform modules
* graph Generate a Graphviz graph of the steps in an operation
* import Associate existing infrastructure with a Terraform resource
* login Obtain and save credentials for a remote host
* logout Remove locally-stored cr edentials for a remote host
* metadata Metadata related commands
* output Show output values from your root module
* providers Show the providers required for this configuration
* refresh Update the state to match remote systems
* show Show the current state or a saved plan
* state Advanced state management
* taint Mark a resource instance as not fully functional
* test Experimental support for module integration testing
* untaint Remove the 'tainted' state from a resource instance
* version Show the current Terraform version.
* workspace Workspace management

**Life Cycle**

1. Alpha
2. Beta
3. Released candidate (RC)

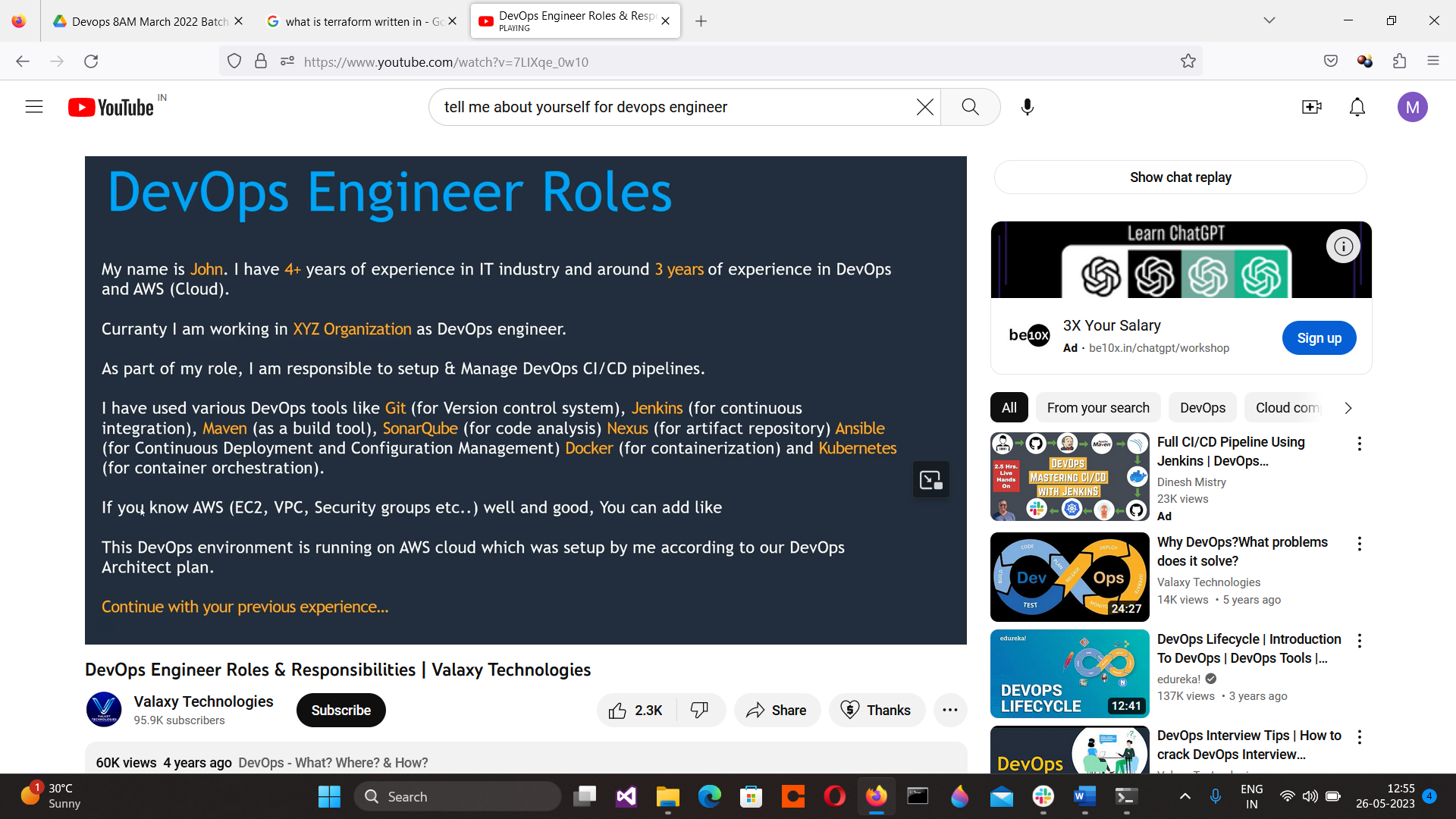
**Terraform Version 0.12.11 to 0.13.0-Beta1 is big change.**

* [terraform\_1.5.0-beta2](https://releases.hashicorp.com/terraform/1.5.0-beta2/)
* [terraform\_1.5.0-beta1](https://releases.hashicorp.com/terraform/1.5.0-beta1/)
* [terraform\_1.5.0-alpha20230504](https://releases.hashicorp.com/terraform/1.5.0-alpha20230504/)
* [terraform\_1.5.0-alpha20230405](https://releases.hashicorp.com/terraform/1.5.0-alpha20230405/)
* [**terraform\_1.4.6**](https://releases.hashicorp.com/terraform/1.4.6/)
* [terraform\_1.4.5](https://releases.hashicorp.com/terraform/1.4.5/)
* [terraform\_1.4.4](https://releases.hashicorp.com/terraform/1.4.4/)
* [terraform\_1.4.3](https://releases.hashicorp.com/terraform/1.4.3/)
* [terraform\_1.4.2](https://releases.hashicorp.com/terraform/1.4.2/)
* [terraform\_1.4.1](https://releases.hashicorp.com/terraform/1.4.1/)
* [terraform\_1.4.0](https://releases.hashicorp.com/terraform/1.4.0/)
* [terraform\_1.4.0-rc1](https://releases.hashicorp.com/terraform/1.4.0-rc1/)
* **How Many issues in each Release-** Go to AWS Terraform GitHub and you can see issues.
* Terraform is coded in language called **GoLang**

**How Terraform Works**

**Terraform providers-** basically version or library or cloud that terraform is supporting.

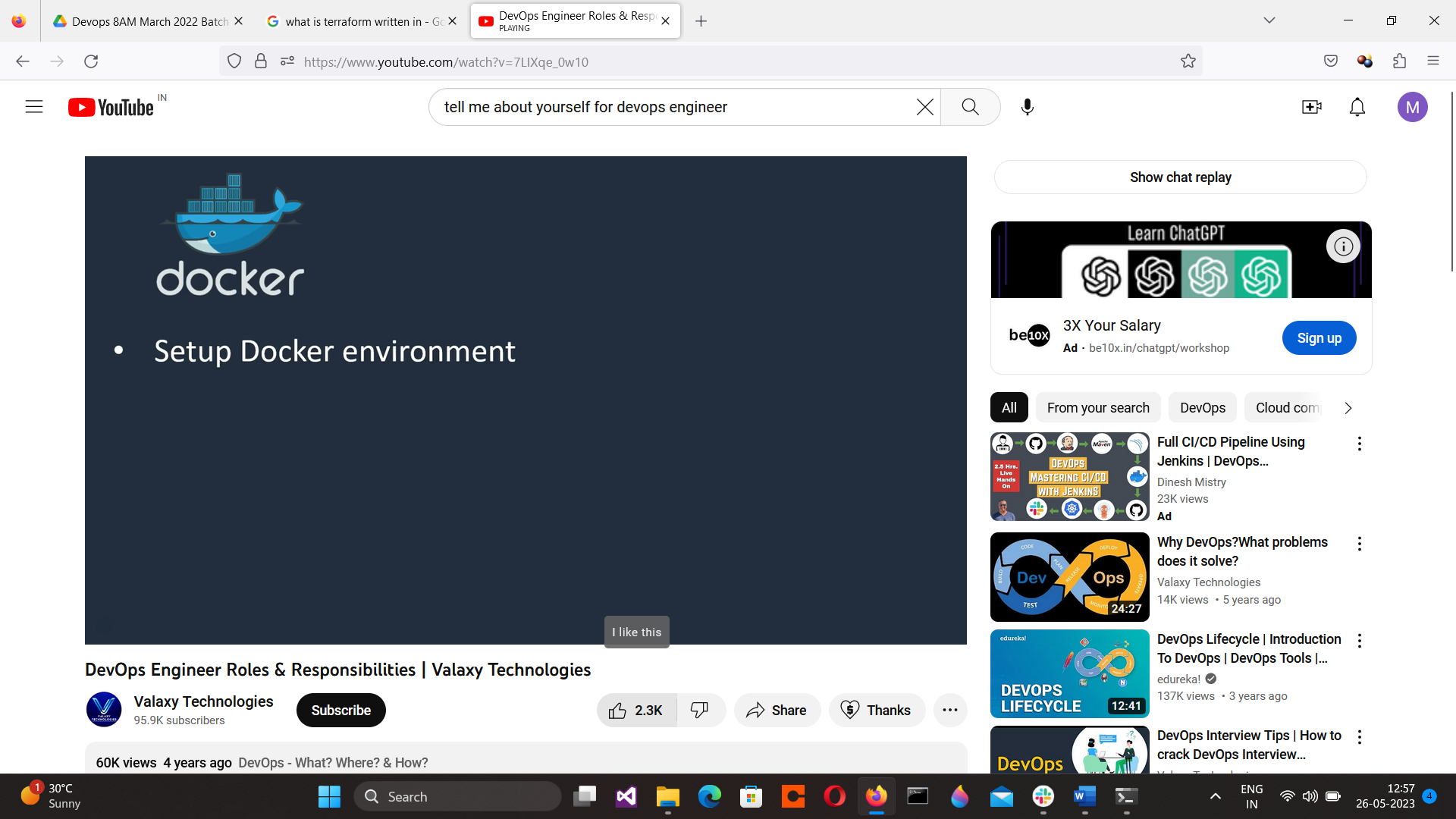
* **Init**: prepare your working directory for other commands.
* Inside tf file, block of provider will be there, terraform will look for the providers and its dependencies.



A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

.