Q.1. Note on

**a. What is Terraform** ?

Terraform is an open-source infrastructure as code (IaC) software tool created by HashiCorp. It allows users to define and provision a data center infrastructure using a high-level configuration language called HashiCorp Configuration Language (HCL) or JSON. Terraform enables developers and operators to automate the deployment and management of infrastructure resources such as virtual machines, storage accounts, network configurations, and more, across various cloud providers (like AWS, Azure, Google Cloud) and on-premises environments. It supports a declarative syntax, where you specify the desired state of your infrastructure, and Terraform figures out how to achieve that state. This approach makes infrastructure provisioning more predictable, repeatable, and scalable compared to traditional manual processes.

**b.why Terraform ?**

Terraform is widely used for several key reasons:

1. Infrastructure as Code (IaC): Terraform allows infrastructure to be managed as code. This means you can define your infrastructure setup in a declarative configuration file (using HCL or JSON), which can be version-controlled, reviewed, and reused.

2. Multi-Cloud and Hybrid Cloud Support: Terraform supports multiple cloud providers (AWS, Azure, Google Cloud, etc.) and even on-premises environments. This allows organizations to manage infrastructure consistently across different platforms and avoid vendor lock-in.

3. Automation and Consistency: With Terraform, infrastructure provisioning and management tasks are automated. This reduces human error, ensures consistency in deployments, and speeds up the process of spinning up new environments or updating existing ones.

4. Immutable Infrastructure: Terraform encourages the practice of immutable infrastructure, where infrastructure components are never modified in place but are instead replaced with updated versions. This approach promotes stability and reliability.

5. State Management: Terraform keeps track of the state of your infrastructure. This state file allows Terraform to know what resources were created, modified, or deleted, and helps in making changes incrementally without affecting other parts of the infrastructure.

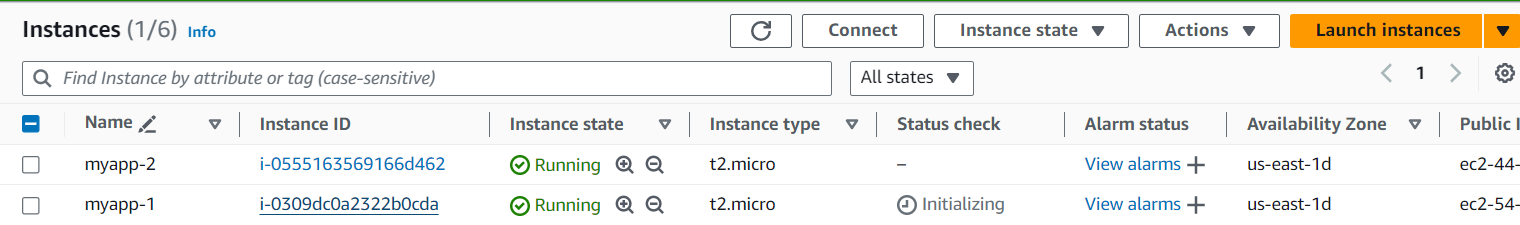
6. Community and Ecosystem: Terraform has a large and active community that contributes modules, plugins, and best practices. This ecosystem makes it easier to adopt and extend Terraform to fit specific needs.

7. Scalability: Terraform can manage infrastructure deployments of any scale, from small environments to large enterprise deployments. It supports modularization, allowing complex infrastructures to be broken down into manageable parts.

1. **Benefits of Terraform**

Terraform offers a range of benefits that streamline and enhance infrastructure management in modern IT environments. By enabling Infrastructure as Code (IaC), Terraform allows teams to define and manage infrastructure configurations using declarative code, such as HashiCorp Configuration Language (HCL) or JSON. This approach brings automation, repeatability, and version control to infrastructure provisioning, reducing manual errors and ensuring consistency across deployments. Additionally, Terraform supports multi-cloud and hybrid cloud deployments, providing flexibility to deploy and manage resources seamlessly across different cloud providers or on-premises environments. Its modular architecture encourages reusability and scalability, allowing organizations to build and share standardized infrastructure components. Terraform's state management capabilities further enhance predictability and auditability, enabling teams to track changes to infrastructure over time and ensuring compliance with regulatory requirements. Moreover, Terraform integrates smoothly with CI/CD pipelines, facilitating automated testing and deployment of infrastructure changes alongside application updates. Overall, these features make Terraform a powerful tool for improving efficiency, agility, and reliability in managing infrastructure at scale, aligning closely with the needs of modern DevOps practices.

1. Q.2 Launch two EC2 instances with names as “myapp-1” and “myapp-2” using Amazon-Linux OS in ‘ap-south-1’ region.



1. Q.3 Install Terraform on local machine (Laptop), integrate aws and terraform with VS code. Using VS code launch an EC2 instances with name ‘myserver’ using Windows OS in ‘ap-south-1’ region.

