Project Title: Multi-Cloud Auto Deployment using Terraform (AWS + GCP Free Tier)

---

1. Objective

The goal of this project is to design and implement an infrastructure-as-code (IaC) solution to automatically deploy resources across multiple cloud providers—**Amazon Web Services (AWS)** and **Google Cloud Platform (GCP)**—using **Terraform**, while leveraging only their **Free Tier** resources to ensure cost efficiency.

---

2. Introduction

Cloud computing services offer flexibility, scalability, and cost-effectiveness. However, managing infrastructure across different cloud platforms can become complex. **Terraform**, an open-source IaC tool by HashiCorp, simplifies provisioning and management of cloud infrastructure across multiple providers with a single configuration language.

This project demonstrates how to:

- Provision AWS and GCP resources simultaneously.
- Manage infrastructure using Terraform.
- Operate within the constraints of Free Tier services.

---

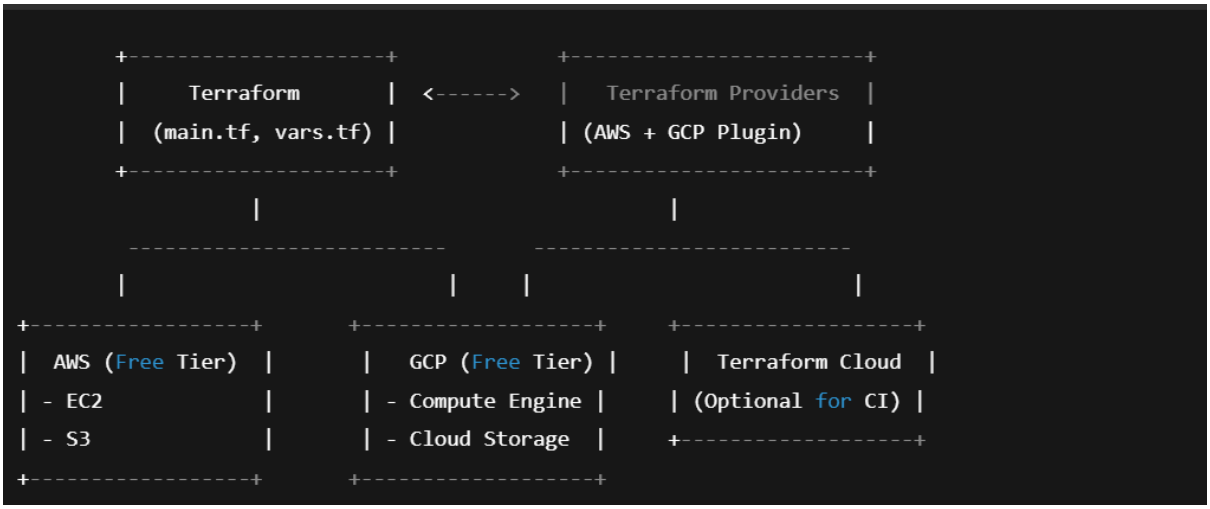
3. Tools and Technologies Used

Tool/Technology	Purpose
Terraform	Infrastructure as Code tool for provisioning
AWS Free Tier	Cloud provider – used for EC2, S3, IAM, etc.
GCP Free Tier	Cloud provider – used for Compute Engine, Cloud Storage
Git & GitHub	Version control and collaboration
VS Code	Code editor
Terraform Cloud (Optional)	Remote state management and collaboration
Google Cloud SDK	Command-line access to GCP
AWS CLI	Command-line access to AWS

---

4. Architecture Diagram

SQL



---

5. Terraform Configuration Overview

- main.tf: Defines resources for both AWS and GCP.
- provider.tf: Configures providers (AWS & GCP) with authentication.
- variables.tf: Manages input variables for dynamic provisioning.
- outputs.tf: Displays resource info like public IPs and URLs.
- terraform.tfvars: Stores variable values (credentials, instance types).

---

6. Resources Deployed

AWS

- EC2 Instance (t2.micro - Free Tier)
- S3 Bucket
- IAM User/Role (optional)

GCP

- Compute Engine VM (e2-micro - Free Tier)
- Cloud Storage Bucket

---

7. Steps to Execute

- Terraform, AWS CLI, GCP SDK.
- **Configure Credentials**:
    - AWS: aws configure

  - GCP: gcloud auth application-default login

- **Initialize Terraform**:

```bash
terraform init
```

- **Plan Deployment**:

```bash
terraform plan
```

- **Apply Deployment**:

```bash
terraform apply
```

- **Verify Resources**: Check AWS & GCP consoles for provisioned services.

---

8. Challenges Faced

- Authentication configuration for multiple cloud providers.
- Managing limits of Free Tier services.
- Handling provider-specific differences in resource definitions.

---

9. Future Improvements

- Integrate with **CI/CD tools** like GitHub Actions.
- Add monitoring and logging via **CloudWatch** and **Stackdriver**.
- Expand to **Azure** for full multi-cloud support.

---

10. Conclusion

This project demonstrates the power of **Terraform** to unify infrastructure management across multiple clouds. It enables consistent, repeatable deployments while utilizing the Free Tier offerings, ideal for prototyping and educational purposes.