Terraform GCP Infrastructure Deployment

This Terraform project deploys a complete data processing infrastructure on Google Cloud Platform (GCP), including a custom VPC network, a Dataproc cluster for big data processing, and a Cloud SQL MySQL instance for relational data storage.

Problem Statement

Manual infrastructure provisioning on GCP for data processing workloads presents several challenges:

- **Inconsistency**: Manual deployments lead to configuration drift across environments
- **Time-Consuming**: Setting up VPC networking, Dataproc clusters, and private Cloud SQL instances can take hours
- **Error-Prone**: Complex networking configurations (VPC peering, private service connections) are easy to misconfigure
- **Security Gaps**: Public IPs and improper firewall rules expose resources to security risks
- **Lack of Repeatability**: Difficult to replicate infrastructure across dev/staging/prod environments
- **No Version Control**: Infrastructure changes aren't tracked or reviewable

Solution

This Infrastructure-as-Code (IaC) solution addresses these challenges by:

✅ **Automated Provisioning**: Deploy entire infrastructure stack in 10-15 minutes with a single command
✅ **Consistent & Repeatable**: Same configuration across all environments, easily customizable via variables
✅ **Security-First Design**: All resources use private IPs only, with proper VPC peering and firewall rules
✅ **Version Controlled**: Track infrastructure changes in Git, enable code reviews and rollbacks
✅ **Modular Architecture**: Reusable modules for VPC, Dataproc, and Cloud SQL components
✅ **Production-Ready**: Includes service accounts, IAM roles, staging buckets, and proper dependencies

Architecture Overview

The infrastructure consists of three main modules:

- **VPC Module**: Creates a custom network with private Google access
- **Dataproc Cluster Module**: Deploys a managed Spark/Hadoop cluster
- **Cloud SQL Module**: Provisions a private MySQL 8.0 database instance

All components are deployed within a private network configuration for enhanced security.

```
terraform-deploy/
├── main.tf              # Root module - orchestrates all infrastructure
├── provider.tf          # Provider configuration
├── variables.tf         # Root-level variable definitions
├── terraform.tfvars     # Variable values (customize for your project)
└── modules/
    ├── vpc/
    │   ├── main.tf
    │   ├── variables.tf
    │   └── output.tf
    ├── dataproc-cluster/
    │   ├── main.tf
    │   └── variables.tf
    └── cloudsql/
        ├── main.tf
        └── variables.tf
```

Technical Challenges & Solutions

Challenge 1: Private Cloud SQL Connectivity

Problem: Cloud SQL instances with private IPs require VPC peering through Service Networking API, which involves complex networking setup.

Solution: Implemented automated VPC peering using:

- `google_compute_global_address` to reserve an IP range for Private Service Access
- `google_service_networking_connection` to establish the peering connection
- Proper dependency chain to ensure VPC is ready before SQL instance creation

Challenge 2: Dataproc Cluster IAM Configuration

Problem: Dataproc clusters need specific service accounts and IAM permissions to access GCS staging buckets and other GCP services.

Solution:

- Created dedicated service account with `dataproc.worker` role
- Granted `storage.admin` access to staging bucket
- Configured proper `service_account_scopes` for cluster nodes

Challenge 3: Internal-Only Networking

Problem: Resources needed to communicate privately without public IP exposure, requiring proper firewall rules and subnet configuration.

Solution:

- Enabled `private_ip_google_access` on subnet for Google API access
- Set `internal_ip_only = true` on Dataproc cluster
- Disabled IPv4 (`ipv4_enabled = false`) on Cloud SQL
- Created firewall rules allowing internal traffic on all protocols

Challenge 4: Resource Dependencies

Problem: Resources must be created in specific order (VPC → Subnet → Peering → SQL), but Terraform's parallel execution can cause failures.

Solution: Used explicit `depends_on` declarations:

- Cloud SQL depends on Service Networking connection
- Service Networking connection depends on VPC existence
- Firewall rules depend on VPC network creation

Challenge 5: Module Reusability

Problem: Need to deploy similar infrastructure across multiple environments without code duplication.

Solution:

- Separated infrastructure into reusable modules (VPC, Dataproc, Cloud SQL)
- Used variable-driven configuration with `terraform.tfvars`
- Implemented naming prefix system for multi-environment support