Multi-Cloud Auto Deployment Using Terraform (AWS + GCP Free Tier)

Introduction

Cloud computing allows organizations to leverage multiple cloud providers to increase redundancy, scalability, and availability. This project demonstrates **multi-cloud deployment** by provisioning resources simultaneously on **AWS Free Tier** and **GCP Free Tier** using **Terraform**, an infrastructure-as-code tool. NGINX web servers are deployed, and health checks simulate routing based on availability.

Abstract

The goal of this project is to automate deployment of web servers on multiple cloud platforms with a single command. By leveraging Terraform's multi-provider capability, the project provisions infrastructure in both AWS and GCP, ensuring that applications can run in a resilient, highly available environment. Local DNSMasq is used to simulate routing based on server health.

Tools Used

- **Terraform**: Infrastructure as Code (IaC) for provisioning cloud resources
- **AWS Free Tier**: EC2, Security Groups, Elastic IP
- **GCP Free Tier**: Compute Engine VM, VPC, Static IP
- **NGINX**: Web server deployment
- **DNSMasq**: Local routing and health checks
- **Terminal / CLI Tools**: Git, gcloud, AWS CLI

Steps Involved in Building the Project

1. **Setup Terraform Providers**
    - Configured AWS and GCP providers using `provider` blocks and variables for region, project, and zone.
2. **Define Infrastructure**
    - AWS: EC2 instance with Security Group, Elastic IP, and NGINX installation
    - GCP: VM instance with VPC, static IP, and NGINX installation (GCP requires billing account to enable Compute Engine API)

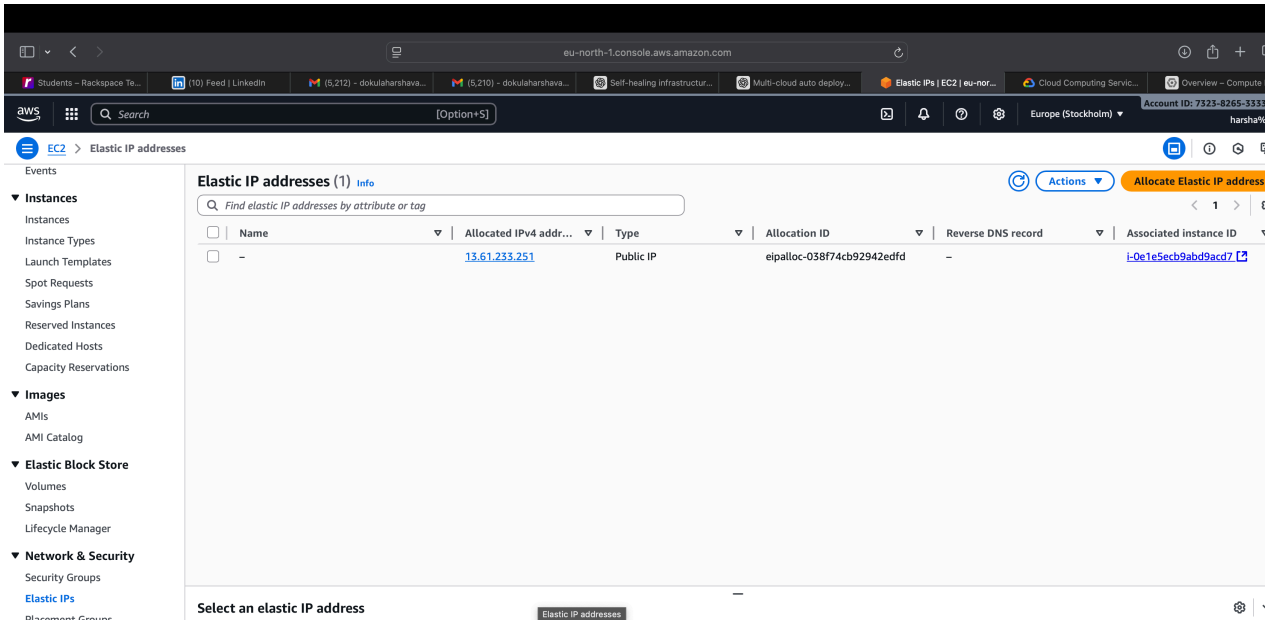
3. **Variables and Outputs**
    - Defined all configuration variables (`aws_region`, `gcp_project`, `ssh_key_name`) in `variables.tf`
    - Configured output files to display public IPs and instance details
4. **Deployment**
    - Initialized Terraform with `terraform init`
    - Deployed resources using `terraform apply`
    - Validated web server availability by accessing public IPs
5. **Health Checks and Routing**
    - Configured DNSMasq locally to simulate failover and routing based on server health

Conclusion

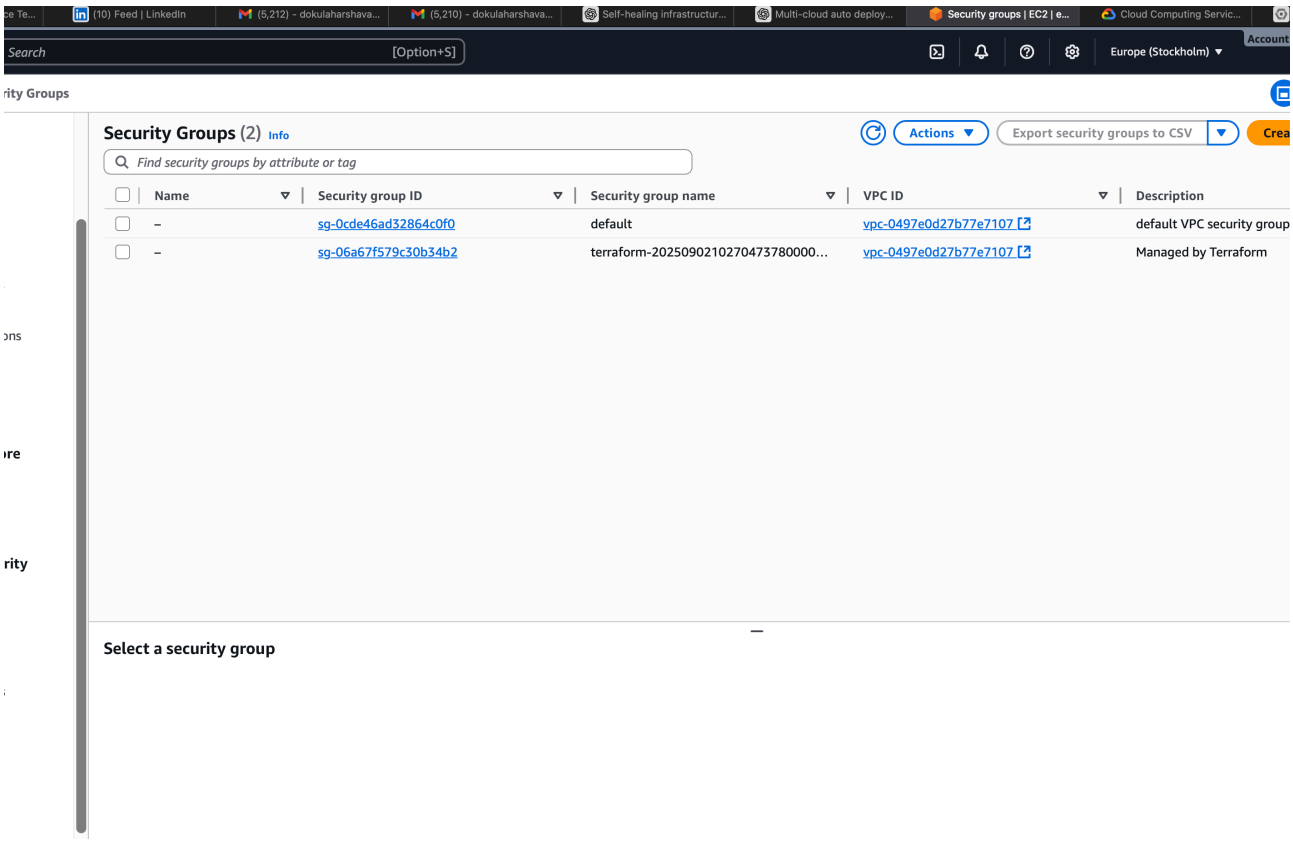
This project successfully demonstrates **multi-cloud auto deployment** using Terraform. AWS deployment was completed successfully with Free Tier resources. GCP deployment requires a billing account to activate Compute Engine API. The project highlights:

- Multi-cloud provisioning
- Infrastructure-as-Code automation
- Local testing of high availability and routing
- Cost-efficient use of Free Tier resources

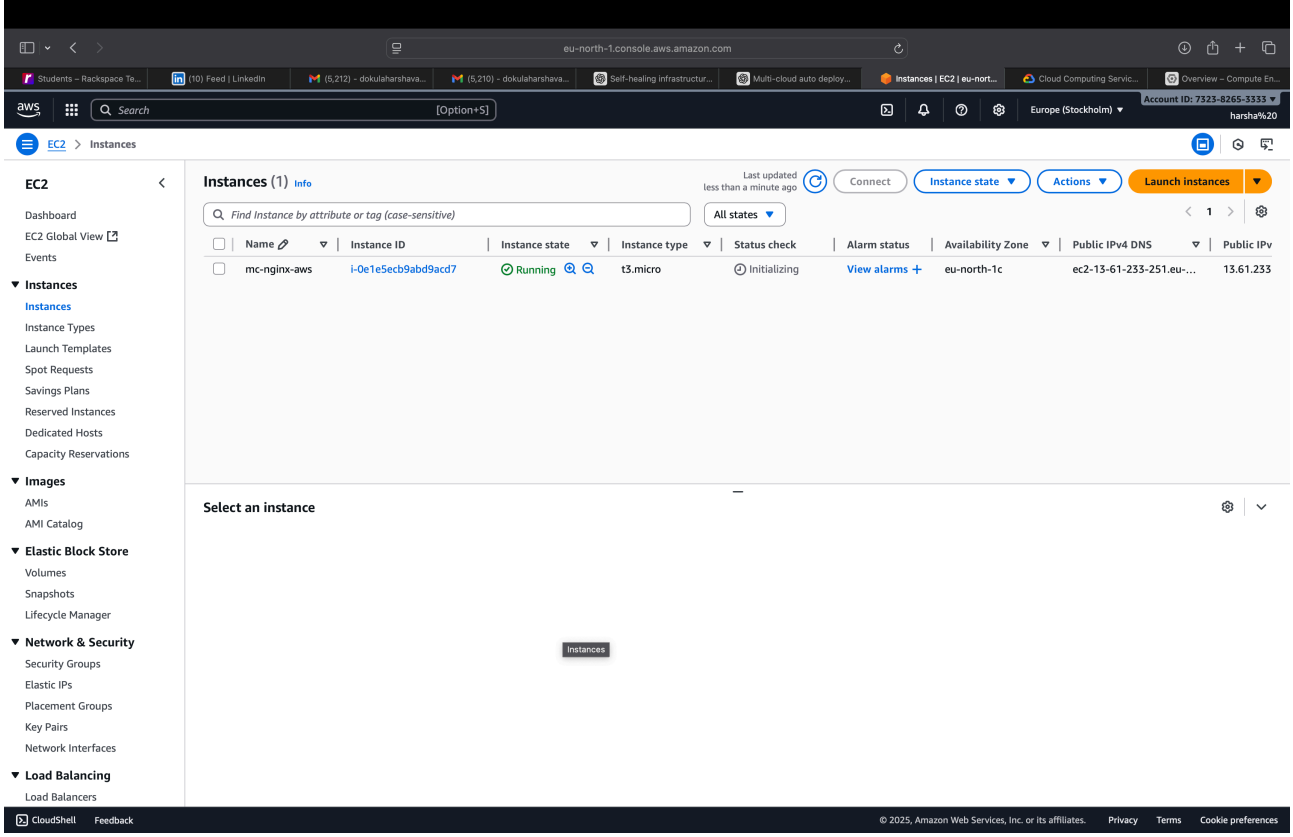
This setup can be extended for production environments with real traffic routing, automated failover, and monitoring tools.



Elastic ip



Security Group


eu-north-1.console.aws.amazon.com
Search
[Option+S]
Europe (Stockholm)
Account ID: 7323-8265-3333
harsha%20
EC2 > Instances
EC2
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Instances (1) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Public IPv
mc-nginx-aws
i-0e1e5ecb9abd9acd7
Running
t3.micro
Initializing
View alarms
eu-north-1c
ec2-13-61-233-251.eu-...
13.61.233
Select an instance
Instances
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Instance