# Multi-Cloud Auto Deployment Report (AWS + GCP)

## 1. Project Overview

**Objective:** Deploy web servers on both AWS and GCP using Terraform, and validate availability with a health check script. Traffic is routed to the healthy server using local DNS (DNSMasq or hosts file).

**Tools:**

- Terraform
- AWS Free Tier (EC2)
- GCP Free Tier (Compute Engine)
- NGINX / Apache
- DNSMasq (local machine)
- Bash script (health check)

## 2. Infrastructure Deployed

| Cloud | Resource | Details |
|---|---|---|
| AWS | EC2 Instance | Ubuntu, t2.micro, HTTP 80, Public IP |
| GCP | Compute Engine | Ubuntu , HTTP 80, Public IP |
| Local | DNSMasq / Hosts | multi-cloud.local resolves to healthy server |
| Local | Health Script | Monitors AWS & GCP server every 5 seconds |

## 3. Deployment Steps

1. **Configure Terraform Providers**
2. AWS provider with region `us-east-1`.
3. GCP provider with project ID, region, and zone.
4. **Deploy Resources**

```
terraform init
terraform apply
```

- Terraform creates EC2 and Compute Engine instances.
- Instances have Apache/Nginx serving “Hello from AWS/GCP”.

- **Retrieve Public IPs**

```
terraform output
```

- Note the AWS and GCP public IPs.

- **Configure Local Hostnames**

- Edit `/etc/hosts` (WSL) or DNSMasq:

```
<AWS_PUBLIC_IP> aws.local
<GCP_PUBLIC_IP> gcp.local
multi-cloud.local -> points to healthy server
```

- **Create Health Check Script**

- Bash script `healthcheck.sh` monitors HTTP response every 5 seconds.

- Updates local DNS for failover if one server goes down.

- **Run Health Check**

```
./healthcheck.sh
```

# 4. Validation Steps

## Step 1: Verify Instances

- **AWS EC2:** Ensure instance is running in AWS Console.
- **GCP Compute Engine:** Ensure instance is running in GCP Console.

## Step 2: Verify Web Servers

- Access via public IPs:
- AWS: `http://<AWS_PUBLIC_IP>` → should return “Hello from AWS”.
- GCP: `http://<GCP_PUBLIC_IP>` → should return “Hello from GCP”.

## Step 3: Test Health Check

- Run `healthcheck.sh`.
- Confirm script prints:

```
AWS server healthy
GCP server healthy
```

- Stop one server (simulate failure) and confirm script prints:

```
AWS server DOWN
GCP server healthy
```

- Confirm `multi-cloud.local` resolves to the healthy server.