Assignment: Deploy OLake on Minikube with Terraform

Objective

Provision cloud infrastructure using Terraform and deploy the OLake application on Minikube. This assignment evaluates your Infrastructure as Code (IaC) skills, Kubernetes knowledge, and DevOps best practices.

Estimated Time: 6-8 hours
Deadline: 5 days from assignment date

Requirements Overview

You will:

1. Write Terraform code to provision a cloud VM
2. Use Terraform to install Minikube and dependencies
3. Deploy OLake using Helm via Terraform
4. Verify the deployment by logging into the OLake UI
5. Document everything for reproducibility

Part 1: Terraform Infrastructure Setup (30 points)

Requirements

1. **Cloud Provider**: You are free to choose any cloud provider (AWS, GCP, Azure etc.)
2. **VM Specifications**:
    - 4 vCPU minimum
    - 8GB RAM minimum
    - 50GB storage
    - Ubuntu 22.04 LTS or later
3. **Terraform Code Must Include**:
    - Provider configuration
    - VM/compute instance resource
    - Firewall/Security group rules:
        - **Port 22**: SSH access for management
        - **Port 8000**: For accessing OLake UI
    - Output variables (VM IP, instance ID, etc.)
    - Remote state configuration (S3, GCS, or local)
4. **Infrastructure as Code Best Practices**:
    - Proper resource naming conventions
    - Tags/labels for resource organization

Part 2: Automated Minikube Installation (25 points)

Requirements

Use Terraform to:

1. **Install Dependencies**:
   - Docker or containerd
   - kubectl
   - Minikube
   - Helm
2. **Start Minikube** with appropriate configuration, for example:
   `minikube start --cpus=3 --memory=6144 --driver=docker`

3. **Enable Required Addons**:
   - Ingress controller: `minikube addons enable ingress`
   - Storage class: `minikube addons enable storage-provisioner` (for NFS persistent volumes)

Warning: Ensure a default storage class is configured in your Minikube cluster before deploying OLake. This is required for persistent volume claims to work properly.

Implementation Options

Choose one approach:

- **Option A**: Terraform remote-exec provisioner with bash script
- **Option B**: Cloud-init user data script

Part 3: OLake Helm Deployment via Terraform (25 points)

Requirements

1. **Use Terraform Helm Provider** to deploy OLake:
```
provider "helm" {
  kubernetes {
    config_path = "~/.kube/config"
  }
}
```
2. Add OLake Helm Repository:
```
helm repo add olake https://datazip-inc.github.io/olake-helm
helm repo update
```

Part 4: Ingress Configuration & UI Access (20 points)

Requirements

Configure Ingress in your custom values.yaml file to make the OLake UI accessible from outside the cluster.

1. **Configure Ingress in values.yaml**:
   - Enable ingress in the Helm chart configuration
   - Expose OLake UI service on port 8000
   - Configure appropriate ingress settings (host, paths, etc.)
   - Ensure the service is accessible from outside the cluster via the VM's public IP
2. **Example values.yaml structure**:

   ```
   ingress:
     enabled: true
     # ... other ingress settings
   ```

   Documentation for setting up Ingress:
   https://olake.io/docs/install/kubernetes/#ingress-configuration
   Refer to chart's values.yaml for exact structure

3. **Access & Verify OLake UI**:
   - Access OLake UI via the VM's public IP on port 8000 (e.g., http://<VM_IP>:8000)
   - Login using the default credentials:
     - **Username**: admin
     - **Password**: password
   - Take screenshots of the login page and authenticated dashboard

Part 5: Documentation (10 points)

Required Documentation

Create a comprehensive README.md in your repository root:

1. Prerequisites
   List all tools and accounts needed:
   - Cloud provider
   - Terraform version
2. Setup Instructions
   A step-by-step guide on how to set up the infrastructure from your assignment, including how to access the UI.
3. Cleanup Instructions

Deliverables Checklist

Submit a single ZIP file containing the following:

1. Project Files:

- [ ] Complete Terraform code (all .tf files)
- [ ] Custom values.yaml for Helm
- [ ] terraform.tfvars.example (without secrets)
- [ ] Comprehensive README.md

2. Screenshots (in a screenshots/ folder):

- [ ] 01-terraform-apply.png - Successful terraform apply output
- [ ] 02-olake-login.png - OLake UI login page (accessed via VM IP)
- [ ] 03-olake-dashboard.png - OLake UI after successful authentication
- [ ] 04-kubectl-pods.png - All pods running (kubectl get pods -A)
- [ ] 05-terraform-destroy.png - Successful terraform destroy output

What We're Looking For:

- ✅ **Clean, readable Terraform code**
- ✅ **Proper use of variables and outputs**
- ✅ **Automated end-to-end deployment**
- ✅ **Working OLake deployment accessible via browser**
- ✅ **No secrets in submission files**
- ✅ **Excellent documentation**

Submission Instructions

Step 1: Prepare Your Submission

1. Ensure no secrets are in any files
2. Test your terraform destroy works completely
3. Verify all documentation is complete and accurate
4. Organize all files according to the deliverables checklist

Step 2: Create ZIP Archive

Create a ZIP file named: OLake_Assignment_[YourName].zip

Structure:

```
OLake_Assignment_YourName/
├── terraform/
│   ├── main.tf
│   ├── variables.tf
│   ├── outputs.tf
│   └── ...
├── values.yaml
├── screenshots/
│   ├── 01-terraform-apply.png
│   ├── 02-olake-login.png
│   ├── 03-olake-dashboard.png
│   ├── 04-kubectl-pods.png
│   └── 05-terraform-destroy.png
└── README.md
```

Step 3: Submit via Email

Send to: **[schitiz@dataip.io]**

Email Subject: DevOps Intern Assignment - [Your Name]

Email Body:

Name: [Your Full Name]
Email: [Your Email]
Contact Number: [Your Phone Number]

Cloud Provider Used: [Your chosen provider]

Attached:
- OLake_Assignment_[YourName].zip (complete submission)
- Resume

Important Notes

⚠️ **Security Warnings**:

- DO NOT include cloud provider credentials in ZIP file
- DO NOT include Terraform state files with sensitive data

⚠️ **Cost Management - IMPORTANT**:

- **You do NOT need to keep infrastructure running after testing**
- Test your deployment, take screenshots, then destroy immediately
- We will reproduce your work on our infrastructure for evaluation
- Use the smallest VM size that meets requirements
- Consider using spot/preemptible instances if your cloud provider supports them (AWS Spot, GCP Preemptible, Azure Spot)
- Run terraform destroy as soon as you finish testing and taking screenshots

⚠️ **Testing Requirements**:

- Verify terraform destroy cleans up all resources
- Test access from a different browser/incognito mode
- Ensure documentation is accurate

Good luck! We're excited to see your work. 🚀