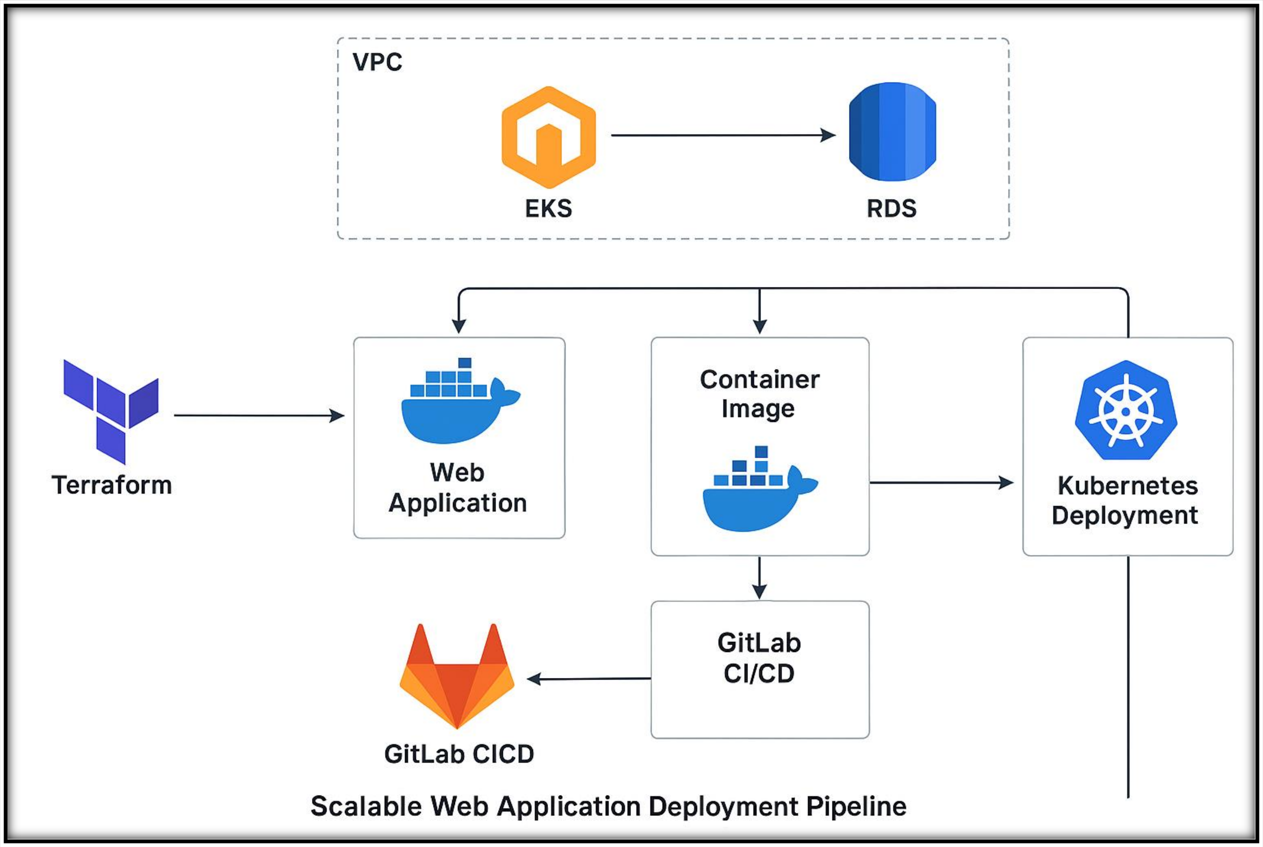
Project Title: Scalable Web Application Deployment Pipeline

Objective:

Design and deploy a scalable web application using infrastructure-as-code, containerization, orchestration, and CI/CD automation.



Scalable Web Application Deployment Pipeline

Tech Stack Overview:

- **AWS**: Cloud infrastructure (EC2, VPC, RDS, S3, IAM, EKS)
- **Terraform**: Infrastructure as Code (IaC) to provision AWS resources
- **Docker**: Containerize the application
- **Kubernetes (EKS)**: Orchestrate containers
- **GitLab CI/CD**: Automate build, test, and deployment pipelines

Project Components:

1. Infrastructure Provisioning (Terraform + AWS)

- Use **Terraform** to create:
  - **VPC** with subnets and security groups
  - **EKS cluster** (AWS-managed Kubernetes)
  - **RDS database** for persistent storage
  - Optional: S3 bucket for logs or static assets
- This ensures that everything is automated and reproducible.

2. Application Containerization (Docker)

- Build a **Docker image** for your web application.
- Optimize using multi-stage builds.
- Push the image to a container registry (GitLab Registry or AWS ECR).

3. Kubernetes Deployment (EKS)

- Deploy the containerized app to **EKS** using:
  - **Deployment** and **Service** manifests
  - **Ingress** for external access (via ALB or NGINX)
  - **ConfigMaps** and **Secrets** for environment variables
- Add **Horizontal Pod Autoscaler (HPA)** for scalability.

4. CI/CD Automation (GitLab)

- Create a **GitLab pipeline** (.gitlab-ci.yml) with stages:
  - **Build**: Create Docker image
  - **Test**: Run unit/integration tests
  - **Push**: Upload image to registry
  - **Deploy**: Apply Kubernetes manifests to EKS using kubectl

5. Monitoring & Logging (Optional but Impressive)

- Integrate Prometheus & Grafana for metrics
- Use Fluentd or CloudWatch for logs

Deliverables:

- Terraform codebase
- Dockerfile & Kubernetes manifests
- GitLab CI/CD pipeline config (.gitlab-ci.yml)
- Documentation (README with architecture diagram, setup steps, and usage)