High-Level Design Document: Azure DevOps, Terraform, and Pipelines

1. Purpose

* Define the overall goal of implementing Azure DevOps, Terraform, and CI/CD Pipelines.
* Outline how these tools will streamline infrastructure provisioning, deployment, and management.

1. Audience

* DevOps Engineers
* Cloud Architects
* Developers
* IT Operations and Support Teams
* Project Stakeholders

1. Introduction

* Overview of Azure DevOps as a platform for source code management, build, release pipelines, and collaboration.
* Terraform as Infrastructure as Code (IaC) tool for provisioning AWS/Azure resources.
* Pipelines for automating build, test, and deployment processes.
* Benefits: consistency, automation, reduced errors, faster delivery.

1. Architecture and Design

* High-Level Architecture Diagram (mock/realistic representation)
  + Azure DevOps Repository → Terraform Modules → CI/CD Pipeline → Target Cloud Environment (AWS/Azure)
* Design Description
  + Version control in Azure Repos
  + Automated Infrastructure provisioning with Terraform
  + CI/CD Pipeline stages: Build → Test → Deploy → Monitor
  + Integration with AWS services (S3, EC2, RDS, Lambda, etc.) or Azure resources
  + Rollback and recovery strategy

1. List of Components and Services | Component | Service | Purpose | |———–|——–|———| | Source Control | Azure Repos | Version control for code and Terraform scripts | | CI/CD | Azure Pipelines | Build, Test, and Deployment automation | | IaC | Terraform | Infrastructure provisioning and management | | Artifact Repository | Azure Artifacts | Store and version reusable components | | Monitoring | Azure Monitor / CloudWatch | Observability and metrics | | Secrets Management | Azure Key Vault / AWS Secrets Manager | Secure storage of credentials and sensitive information |
2. Screenshots

* Azure DevOps project dashboard (mock)
* Sample Terraform module structure (mock)
* Example Pipeline stages (Build/Test/Deploy) (mock)
* Deployment logs and execution outputs (mock)

1. Observability (Backup and Restore)

* Backup strategies for repositories and pipelines
* Terraform state backup (remote backend in S3/Azure Storage)
* Pipeline execution logs retention
* Rollback strategies for failed deployments
* Disaster recovery steps for Azure DevOps and deployed infrastructure

1. Common Services

* Azure DevOps Extensions
* Terraform Providers
* Monitoring & Alerting tools (Azure Monitor, CloudWatch)
* Logging frameworks
* Security compliance tools (Azure Security Center, AWS Config)