**Phase 3: Development – Cloud-Based Disaster Recovery**

**1. Objectives**

* Implement a disaster recovery (DR) environment that can be activated automatically in the event of a failure.
* Use Infrastructure as Code (IaC) to provision environments consistently.
* Enable monitoring, backup, and automated failover using AWS services.

**2. Project Repository**

**GitHub Repo**: <https://github.com/atulkamble/cloud-dr-solution-enterprises>

**Key Directories and Files**:

* cloud\_dr\_resources.yaml – CloudFormation template to deploy DR infrastructure
* lambda\_start\_ec2.py – AWS Lambda function script for automating EC2 instance startup
* backup\_to\_s3.sh – Shell script to perform regular backups to Amazon S3
* README.md – Provides setup instructions and architecture explanation

**3. Development Components**

**a. Infrastructure as Code (IaC)**

Using the cloud\_dr\_resources.yaml template, the following resources are provisioned:

* VPC with Subnets
* EC2 Instances (web/app tier)
* Amazon RDS (with Multi-AZ option)
* S3 buckets for backup
* Lambda function roles and permissions

**b. Automation Scripts**

**Backup Script (backup\_to\_s3.sh)**

* Scheduled using cron to back up data from instances to S3
* Ensures all critical data is periodically stored in the cloud

**Lambda Function (lambda\_start\_ec2.py)**

* Triggered during a DR event
* Starts EC2 instances and updates Route 53 DNS settings

**c. Monitoring and Failover**

* **Route 53 Health Checks** monitor endpoints in the primary region
* **CloudWatch Alarms** trigger Lambda functions based on monitored events
* **Route 53 DNS Failover** redirects traffic to the DR region upon failure

**4. Deployment Steps**

1. **Clone the Repository**
2. git clone https://github.com/atulkamble/cloud-dr-solution-enterprises.git
3. cd cloud-dr-solution-enterprises
4. **Deploy Infrastructure**
   * Go to the AWS CloudFormation Console
   * Upload and deploy the cloud\_dr\_resources.yaml template
5. **Configure IAM Roles**
   * Ensure Lambda and EC2 instances have proper IAM roles
   * Grant necessary S3, EC2, RDS, and Route 53 permissions
6. **Set Up Backup Jobs**
   * Upload backup\_to\_s3.sh to each EC2 instance
   * Schedule it with cron:
   * crontab -e
   * 0 \*/6 \* \* \* /path/to/backup\_to\_s3.sh
7. **Deploy and Test Lambda**
   * Deploy lambda\_start\_ec2.py via AWS Lambda Console
   * Configure necessary environment variables and permissions
   * Simulate a primary region failure and observe DNS failover

**5. Version Control and Branching**

* Git is used for version control
* main branch contains stable, production-ready code
* dev branch is used for feature testing and improvements

**6. Best Practices Followed**

* CloudFormation templates are parameterized for flexibility
* Role-based access (IAM) ensures secure automation
* Logging and monitoring integrated via CloudWatch
* Cost-optimized design using a "cold standby" DR approach