Phase 1

DevOps Goals

Niyata IT

Phase 1

Niyata DevOps

01 SDLC Process Setup with Gitlab

02 Continuous Integration Model

03 Continuous Deployment Model

04 Continuous Monitoring Setup-Iteration 01

Phase 1

MILESTONE

01

SDLC Process Setup with Gitlab

- Gathering Requirements
- Define Milestones
- Design and Document
- Tag Commits and MR to requirements
- Estimate and Track time spent
- Process Visualization
- Release Management

Phase 1

Target

A Session on Gitlab SDLC

01

- ✓ Developers and Leads will collaborate and analyse requirements.
- ✓ That requirements will be transformed into User stories, Which will be created as Issues in Gitlab. Appropriate labels will be assigned.
- ✓ Project Milestones will be defined and Issues will be mapped to it. Time will be estimated for each Issues.
- ✓ Relevant Designs and Documentation for each milestone will be managed in Wiki.
- ✓ Developers will tag the relevant issue on each commit and MR and track the time spent on each Issue.
- ✓ Feature branches will be created for each module, and integrated to target branch after code review
- ✓ Separate Board For each activity, Dev, QA, Discussion, Deployment, Defects

Phase 1

MILESTONE

02

Continuous Integration Model

- Runner/Agent Setup
- Standard Check (PEP8,Build)
- Code Quality
- Unit Test
- Security Analysis
- Coverage Report
- Build Image for containers

Target

CI/CD Session with GitLab

02

- ✓ Install and Configure Runners for handling single or concurrent Jobs.
- ✓ CodeStyle and Standard Check for Both python(PEP8) and Javascript (TSLint, ESLint, Build job). Pipeline will exit, if this job fails.
- ✓ Setup a Code Analysis stage for Pylint and Rate code for Python. Pipeline continues with a warning if this job fails.
- ✓ Unit Test for Both FrontEnd and BackEnd. Pipeline exits if this job fails. Minimal Code Coverage for earlier Adoption.
- ✓ Security Analysis For Python. Pipeline exits if this Job fails.
- ✓ Unit Test Report will be visible in Gitlab Pipeline. MR only be allowed if required Jobs passes. MR will show coverage status. Pipeline and quality status will be shown as badges in repository.

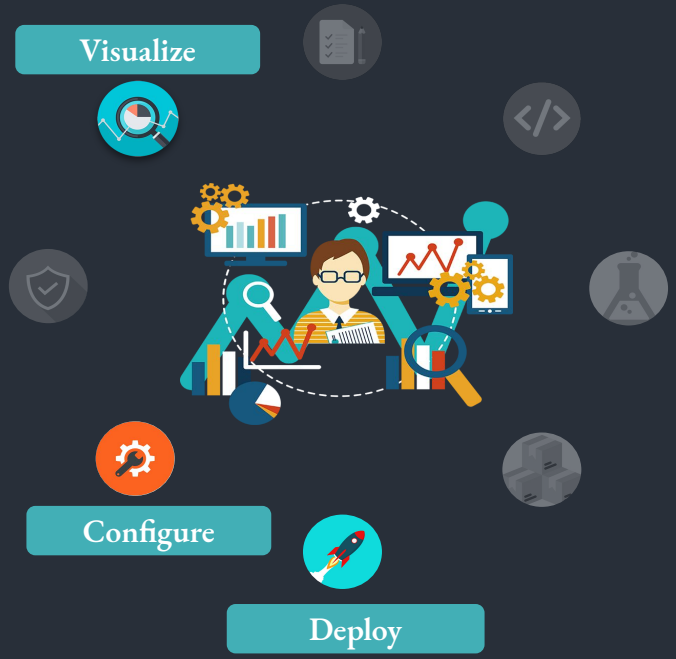
Phase 1

MILESTONE

03

Continuous Deployment Model

- Environment Definition in Gitlab
- Deploy to environment ~ Backend
- Deploy to environment ~ Frontend
- Visualize Deployment Status
- Integrated with GitLab Environment



Phase 1

Target

CI/CD Session with GitLab

03

- ✓ Install and Configure Runners for handling single or concurrent Jobs.
- ✓ Environments will be defined/preconfigured(Dev, Staging, production) before automation.
- ✓ After Successful CI Stages, The code will be automatically deployed into Dev environment First(Development Branch is required).
- ✓ On Successful CI Stages, The code will be automatically deployed to Next environment (Staging/Master Branch is required).
- ✓ Each Deployment Job will be visualized and integrated to Gitlab and related issues. This will provide end to end visibility.

Phase 1

MILESTONE

04

Monitoring Base Setup

- Configure & Secure Prometheus Server
- Node Exporter service
- Infra Dashboard For Grafana
- Loki 2.0
- App and Nginx Logs
- Sentry
- Prometheus-Gitlab

Target

Prometheus-Loki-Grafana Session

04

- ✓ Upgrade and Secure the existing Prometheus Server. Restricted IP Access.
- ✓ Configure Node Exporter in each server. Pull metrics and collect in Centralized Prometheus Server.
- ✓ Create an Infra Dashboard For Niyata Servers(CPU, Memory, Disk, N/W stats).
- ✓ Upgrade Loki to new version. Secure.
- ✓ Collect Backend Service Logs using Promtail. Visualize Logs with Grafana Dashboard.
- ✓ Sentry Integration to required projects. Integrate with Gitlab.
- ✓ Prometheus Dashboard with base metrics in Gitlab(CPU, Memory).