**Name:** **Sakshi Mohan Kamble**

**PRN :2020BTEIT00205**

**Subject: Agile Lab**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Assignment no:12**

**Title:**

**Use Tool Gitlab for Agile practice OR check and demonstrate the feature of Gitlab for CI/CD on a sample codes**

GitLab is an all-in-one DevOps platform that includes features for source code management, continuous integration/continuous deployment (CI/CD), and project management. GitLab's integrated platform supports agile development practices, including Scrum and Kanban, by allowing teams to collaborate, manage projects, and automate the software delivery pipeline.

Agile practices involve breaking down work into small, iterative chunks that can be completed quickly and tested regularly. GitLab's project management features, such as issue tracking, boards, and milestones, allow teams to plan and prioritize work in an agile way. Teams can also use GitLab's built-in CI/CD pipelines to automate testing, building, and deploying software changes.

To demonstrate GitLab's CI/CD features, let's use a simple Python application that prints "Hello, World!" when run. We'll set up a GitLab repository, create a CI/CD pipeline, and configure it to automatically test and deploy the code.

1. Create a GitLab repository

First, we need to create a new GitLab repository for our project. We can do this by logging into GitLab and clicking on the "New project" button. We'll give the project a name, select the appropriate visibility settings, and create the repository.

1. Add code to the repository

Next, we'll add our Python application code to the repository. We can do this by cloning the repository locally, creating a new file called "hello\_world.py", and adding the following code:

bash

Copy code

print("Hello, World!")

We'll then commit and push the code to the GitLab repository.

1. Set up the CI/CD pipeline

Now that we have code in our repository, we can set up a CI/CD pipeline to automatically test and deploy the code. GitLab's CI/CD pipelines are defined using a YAML file called ".gitlab-ci.yml". We'll create this file in the root of our repository and add the following configuration:

yaml

Copy code

stages: - test - deploy test: stage: test image: python:3.9 script: - python -m unittest discover -v deploy: stage: deploy image: python:3.9 script: - echo "Deploying..."

This configuration sets up two stages: "test" and "deploy". In the "test" stage, we use a Python 3.9 Docker image to run unit tests using the unittest module. In the "deploy" stage, we simply echo a message to indicate that the code is being deployed.

1. Run the pipeline

With our CI/CD pipeline configured, we can now run it by committing and pushing changes to our GitLab repository. GitLab will automatically detect the changes and start running the pipeline.

The pipeline will run the "test" stage first, which will execute the unit tests in our Python code. If the tests pass, the pipeline will move on to the "deploy" stage and echo the "Deploying..." message.

If we make changes to the code and push them to the repository, GitLab will automatically detect the changes and run the pipeline again. This ensures that our code is always tested and deployed whenever changes are made.

**Conclusion:**

GitLab's integrated platform supports agile development practices by allowing teams to collaborate, manage projects, and automate the software delivery pipeline. Its CI/CD pipelines make it easy to test and deploy code changes quickly and reliably.