CREATING A BASIC CI/CD PIPELINE IN GITLAB: STEP-BY-STEP GUIDE FOR BUILD AND TEST JOBS

Step 1: Create a New Repository

1. Log in to your GitLab account.
2. Click on the **“+”** icon in the top-right corner of the screen and select **“New project/repository”**.
3. Choose **“Create blank project”**.
4. Enter the project name, e.g., simple-ci-cd-pipeline.
5. *(Optional)* Add a description for the project.
6. Make the project **public** or **private** based on your preference.
7. Click **Create project**.

Step 2: Open the Web IDE

1. In your newly created project, click on **"Web IDE"** from the repository's main page.

Step 3: Create the .gitlab-ci.yml File

1. In the Web IDE, create a new file by clicking the **“New File”** button.
2. Name the file **.gitlab-ci.yml** (this is the configuration file for the GitLab CI/CD pipeline).
3. Add the following code for the **build job**:

```
create_file:
  image: alpine
  script:
    - echo "Building ..."
    - mkdir build
    - touch build/somefile.txt
```

1. create_file:

This is the **job name**. In GitLab CI/CD, jobs define the individual tasks that the pipeline will execute. Here, the job is named create_file. You can name it anything descriptive based on the purpose of the job.

---

2. image: alpine

- This specifies the **Docker image** that will be used to run the job.
- alpine is a lightweight Linux distribution that is widely used in CI/CD pipelines for its simplicity and small size.
- The job runs inside a container based on this image.

---

3. script:

- The script section contains the commands that will execute as part of the job.
- These commands run sequentially in the Alpine container.

Commands in the script section:

1. **echo "Building ..."**
    - Prints the message "Building ..." to the console.

    - This is useful for logging purposes so you can track the pipeline's progress.

2. **mkdir build**
    - Creates a new directory named build in the working directory.
    - This is commonly used in build pipelines to organize files or outputs.
3. **touch build/somefile.txt**
    - Creates an empty file named somefile.txt inside the build directory.
    - The touch command is often used to create placeholder files or verify directory structures in pipelines.
4. In the Web IDE, go to the **source control** section, type **"Add build job"** as the commit message, and commit the changes to the **main branch** in your repository.
5. Go back to your GitLab project dashboard, refresh the page, and verify that the file is added to the repository.
6. Navigate to the **CI/CD > Pipelines** section and check if the pipeline ran successfully to create the build/somefile.txt file.

Step 4: Add the Test Job

1. Open the .gitlab-ci.yml file in the Web IDE.
2. Replace the existing content with the following code to include both **build** and **test** jobs:

```
stages:
  - build
  - test


create_file:
  image: alpine
  stage: build
  script:
    - echo "Building ..."
    - mkdir build
    - touch build/somefile.txt
  artifacts:
    paths:
      - build/

test_file:
  image: alpine
  stage: test
  script:
    - test -f build/somefile.txt
```

1. stages:

- This defines the pipeline **stages** that the jobs belong to.
- Here, there are two stages:
  - **build**: Represents the stage where files or outputs are created.
  - **test**: Represents the stage where the created files or outputs are tested/verified.
- Stages are executed in order (build first, then test).

---

2. create_file:

This is the **first job** in the pipeline, part of the build stage.

Key Elements:

- **image: alpine**
    - Specifies the **Docker image** for this job.
    - alpine is a lightweight Linux-based image, ideal for simple tasks.
- **stage: build**
    - Assigns this job to the build stage, as defined in the stages section.
- **script:**
    - Contains the commands to be executed for this job.
    - Commands:
        1. **echo "Building ..."**
            - Outputs "Building ..." to the pipeline logs for tracking.
        2. **mkdir build**
            - Creates a directory named build to represent a "build output folder."
        3. **touch build/somefile.txt**
            - Creates an empty file named somefile.txt inside the build directory.
- **artifacts:**
    - Specifies files or directories that should be saved and passed to subsequent stages.
    - **paths:**
        - Lists the paths to be stored as artifacts.
        - In this case, the build/ directory is saved so that it can be used by jobs in later stages (e.g., test_file).

3. test_file:

This is the **second job**, part of the test stage.

Key Elements:

- **image: alpine**
  - Uses the same lightweight Alpine image to run this job.
- **stage: test**
  - Assigns this job to the test stage, as defined in the stages section.
- **script:**
  - Contains the commands to test the output from the create_file job.
  - Commands:
    1. **test -f build/somefile.txt**
       - Checks if the file somefile.txt exists in the build/ directory.
       - If the file exists, the job passes. If it doesn’t, the job fails, and the pipeline stops.

Workflow of the Pipeline

1. **Stage 1: Build**
   - The create_file job runs.
   - It creates a build/ directory and an empty file somefile.txt.
   - The build/ directory is saved as an artifact for the next stage.
2. **Stage 2: Test**
   - The test_file job runs.
   - It verifies that the file somefile.txt exists in the build/ directory created in the previous stage.

3. In the Web IDE, type **"Add test job"** as the commit message, and commit the changes to the **main branch**.

Step 5: Trigger the Pipeline

1. Once the .gitlab-ci.yml file is updated and committed, GitLab automatically triggers the pipeline.
2. Go to the **CI/CD > Pipelines** section in the left-hand menu to view the pipeline's status.

Step 6: Verify the Pipeline Jobs

1. In the **Pipelines** section, you'll see the pipeline with two stages: **Build** and **Test**.
2. Click on the pipeline to view the jobs:
    - **Build job**: Creates the directory and file (build/somefile.txt).
    - **Test job**: Checks if the file exists.
3. Each job runs sequentially based on the defined stages.

Step 7: Check the Logs

1. Click on each job to view the logs and confirm the actions:
    - **Build job**: The log should show the creation of the build/ folder and the somefile.txt file.
    - **Test job**: The log should confirm the existence of the somefile.txt.

Step 8: (Optional) Make Modifications

1. If you want to add more jobs or stages, return to the Web IDE.

2. Edit the .gitlab-ci.yml file as needed.
3. Commit the changes to trigger the updated pipeline.