Assignment 1: Implementing Your First CI/CD Pipeline

Overview

In this assignment, you will implement your first CI/CD pipeline using GitLab. The pipeline will automatically prepare, test, build, and deploy a web application. This will help you gain hands-on experience with Continuous Integration and Continuous Deployment (CI/CD) workflows.

In addition to the CI/CD pipeline, you will also implement a simple monitoring system for your web application using a cron job. The monitoring system will help you track the status of the web server over time.

Requirements

1. Node.js (v20.13.1)
2. GitLab account
3. Web application source code located within this assignment. Find the source code in the `react-tictactoe` directory.

Part 1: Implementing CI/CD Pipeline

In this part, you will implement a CI/CD pipeline using GitLab. The pipeline will automatically **prepare**, **test**, **build**, and **deploy** a web application.

Step 1: Check the Web Application

1. Navigate to the `react-tictactoe` directory.
2. Install the dependencies by running the following command:

   ```
   npm install
   ```

3. Start the application by running the following command:

   ```
   npm run start
   ```

4. Open your browser and navigate to the provided URL to view the web application.
5. Stop the application by pressing `Ctrl + C` in the terminal.

Step 2: Prepare the GitLab Repository

1. Create a new GitLab account if you don't have one.
2. Create a new project in GitLab.
3. Push the web application source code to the GitLab repository.

Step 3: Setup Gitlab Runner

1. Install GitLab Runner on your local machine or a server.
2. Register the GitLab Runner with your GitLab project.

Step 4: Implement the CI/CD Pipeline

1. Create a `.gitlab-ci.yml` file in the root of your project.
2. Define the stages for the pipeline: `prepare`, `build`, `test`, and `deploy`.
3. Define the jobs for each stage: `prepare`, `build`, `test`, and `deploy`.
4. Configure the pipeline to run the following commands:
   - **Prepare**: Install the dependencies by running `npm install`.
   - **Test**: Run the tests by executing `npm run test`.
   - **Build**: Build the application by executing `npm run build`.
   - **Deploy**: Deploy the application using a Linux service. You should put static files (generated by `npm run build`) in a specific directory and serve them using `serve -s /path/to/build` (you can install `serve` via `npm install serve`) in the background within a Linux service. You should place the service configuration file in the root of your project.
5. Commit and push the `.gitlab-ci.yml` file to your GitLab repository.
6. Check the pipeline status in GitLab.

Step 5: Verify the CI/CD Pipeline

1. Make a change to the web application source code.
2. Commit and push the changes to your GitLab repository.
3. Check the pipeline status in GitLab.

Step 6: Verify the Deployment

1. Open your browser and navigate to http://localhost:3000 to view the deployed web application.
2. Verify that the web application is working as expected.
3. Restart the server and check if the web application is still accessible. It should automatically start when the server restarts.

Part 2: Implementing Monitoring

In this part, you will implement a simple monitoring of your web application using a cron job. The monitoring will help you track the status of the web server over time.

Step 1: Implement the Monitoring Script

1. Create a monitoring script that checks the status of the web server.

    - The script should make an HTTP request to the web server and log the response status code. You can use `curl` to make the HTTP request.
    - The script should log the status code to a file named `monitor.log`.
2. Save the script as `monitor.sh` in the root of your project.
    - The script should be executable.
3. Test the monitoring script by running it manually.
4. Check the log file generated by the monitoring script.

Step 2: Setup the Cron Job

1. Create a cron job that runs the monitoring script every 1 minute.
2. Save the cron job configuration in a file named `cronjob`.
3. Add the cron job configuration to the crontab.

Step 3: Verify the Monitoring

1. Start the cron job.
2. Monitor the status of the web server over time.
3. Check the log file generated by the monitoring script.
4. Stop your web server via `sudo systemctl stop <your-service-name>` and check if the monitoring script detects the failure.

Deliverables

1. A link to your GitLab repository containing the web application source code, `.gitlab-ci.yml`, `monitor.sh`, and a Linux service configuration file.
2. Proof of the CI/CD pipeline running successfully in GitLab. You should provide screenshots.
    - Screenshots of the pipeline status and job logs in GitLab.
    - Screenshots of the service running on the server. You can use `systemctl status <your-service-name>` to check the status.
    - Screenshots of the web application running in the browser.
3. Proof of the monitoring script running successfully. You should provide screenshots.
    - Screenshots of the contents of the `monitor.log` file for successful requests.
    - Screenshots of the contents of the `monitor.log` file for failed requests.
    - Screenshots of the cron job configuration. Use `crontab -l` to list the cron jobs.

Grading Criteria

1. **Part 1: Implementing CI/CD Pipeline (70%)**
    - The CI/CD pipeline is implemented correctly.
    - The pipeline runs successfully on GitLab.

  - The web application is deployed and accessible.
  - The pipeline is triggered automatically on code changes.

2. **Part 2: Implementing Monitoring (30%)**
   - The monitoring script is implemented correctly.
   - The cron job is set up correctly.
   - The monitoring script logs the status of the web server.
   - The monitoring script detects server failures.