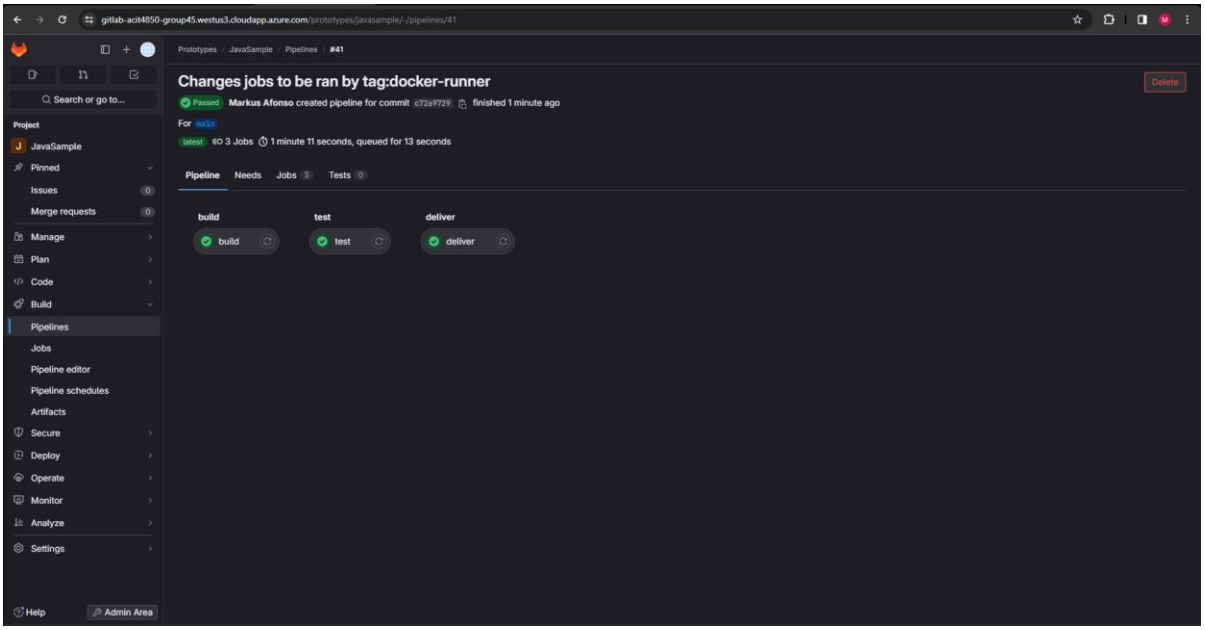
Part A:

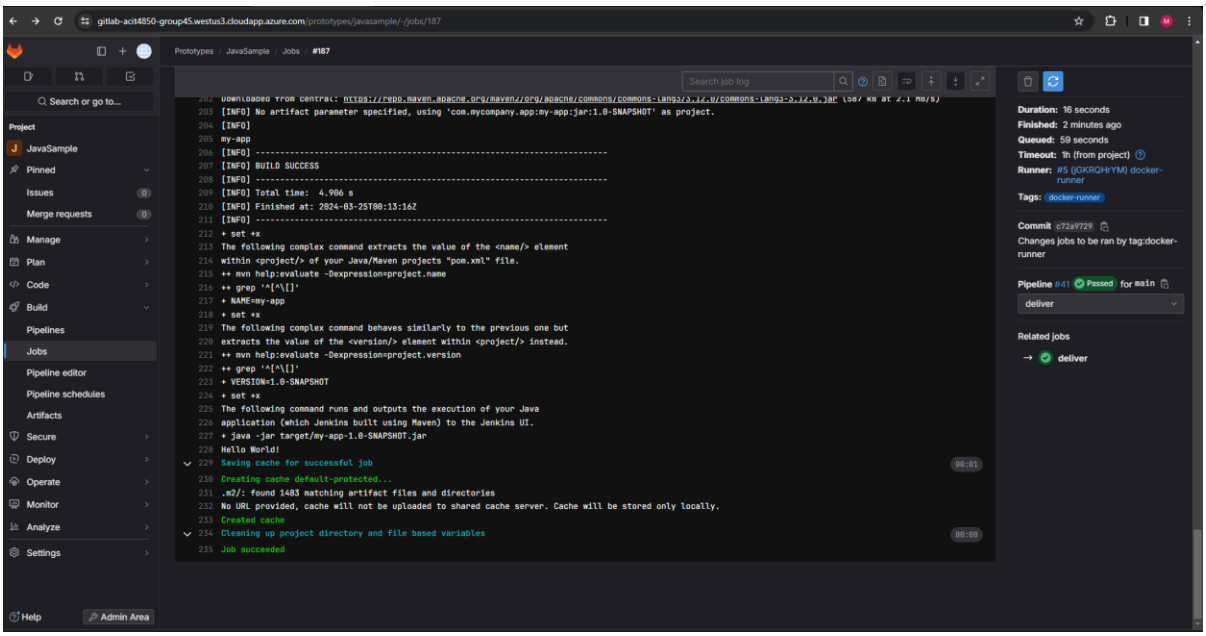
Screenshot of your Docker runner, proving that it is a Docker running on your GitLab VM.

```
azureuser@GitLabCE:~$ sudo docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS     NAMES
7238194c77af   gitlab/gitlab-runner:latest    "/usr/bin/dumb-init …"   34 minutes ago   Up 34 minutes             gitlab-runner
```

Screenshot of your pipeline running successfully with the above stages. It must show the URL of your GitLab installation (with your Group #).



Screenshot of the output of your Deliver stage showing the output of the built Java program. It must show the URL of your GitLab installation (with your Group #).



Part B:

1. Identify three similarities between Jenkins and GitLab CI
   Both Jenkins and GitLab CI are continuous integration and continuous deployment (CI/CD) tools designed to automate the software development lifecycle.

   They both support pipeline-based workflows, allowing developers to define their build, test, and deployment processes as code.

   Both platforms offer extensive plugin ecosystems, enabling users to extend their functionality and integrate with various tools and services.

2. Identify three differences between Jenkins and GitLab CI
   Jenkins is primarily focused on CI/CD, while GitLab CI is part of a larger integrated DevOps platform offered by GitLab, which includes source code management, issue tracking, and more.

   GitLab CI is tightly integrated with GitLab's version control system, providing a single interface for managing code repositories and CI/CD pipelines, whereas Jenkins requires additional setup and configuration to integrate with version control systems like Git.

   GitLab CI emphasizes simplicity and convention over configuration, providing predefined CI/CD templates and workflows out of the box, while Jenkins offers more flexibility but requires more manual configuration.

3. Define two scenarios – i.e., companies with software teams – that would use a CI/CD tool. For one scenario, recommend and justify Jenkins as the preferred choice. For the other scenario, recommend and justify GitLab CI as the preferred choice.

   Scenario 1: Let’s say a small startup with limited resources requires a CI/CD tool that is easy to set up, maintain, and integrates seamlessly with their version control system. Jenkins might be the preferred choice here due to its flexibility and extensive plugin ecosystem, allowing the team to customize their CI/CD pipelines according to their specific needs without the overhead of a comprehensive DevOps platform.

   Scenario 2: Let’s say a large enterprise with multiple software teams and a fully operating DevOps culture may prefer GitLab CI as part of the GitLab DevOps platform. GitLab CI offers a comprehensive solution that includes not only CI/CD but also version control, issue tracking, code review, and more, all integrated into a single platform. This integration simplifies the workflow, enhances collaboration, and provides better visibility across the software development lifecycle.