Organização do Desenvolvimento de Software

Mestrado em Engenharia Informática

PL 5

Project - Part 1

Hugo Silva
Nuno Bettencourt
Nuno Ferreira
Nuno Melo

Dep. de Engenharia Informática – ISEP

2022/2023

Project - Part 1: Content

Content

- Sequential Scripted Pipeline;
- Parallel Scripted Pipeline;

Project - Part 1: Rules

- This Project is divided in two tasks that are comprised of smaller Goals;
- Each Task only differs in the way the Pipeline is designed and implemented;
- You must create at least one issue in your Bitbucket repository for each Task and Goal; Task status must be automatically transitioned by the commit messages;
- You should update the Readme file with your analysis of the solution;
- A pipeline design must be created and committed to the repository before the implementation begins;
- A performance evaluation report comparing both tasks' execution must be written and added to your repository in the Readme file;
- A self-assessment form must be filled by the team and added to the repository;
- All Jenkins jobs should be added to the Git repository;
- All Tasks should be developed independently of the Team size;
- Bonus points are given to teams that present and discuss their finished work to the class before the deadline;
- This assignment must be completed and submitted on Moodle by November, 13th.

Remember: Solutions may differ in several criteria, such as performance or fault tolerance or extensibility.

Project - Part 1: Setup

For this assignment, **you can use any project that fits the Project Requirements**. For this, on your first lab class you should fill the Project Requirements form (available on Moodle), show it to your lab teacher (to check if it suits the requirements) and submit it on Moodle by the end of the first week.

If you do not wish to use your own project, **you can use the code from the provided project**. The link is available on Moodle.
You should:

- Download the repository into your system (**Note: Do NOT clone it! If cloning it, delete the ".git" hidden folder before proceeding to the next step**).
- Copy **all** the files inside the folder to your git repository **Do not forget the hidden file ".gitignore"!**.

The provided CMS Project is based on Vaadin and Spring Boot. To better understand the project structure, please follow the tutorial:
`https://vaadin.com/docs/latest/tutorial/overview`.

Project - Part 1: Tasks

Task 1 Configure a Jenkins Pipeline using a Scripted Pipeline[1], to perform a sequential build;

Task 2 Configure a Jenkins Pipeline using a Scripted Pipeline, to perform a parallel build.

Stages in a sequential build are sequentially executed while stages in a parallel build should be executed in parallel when possible.

Think about dependencies when optimizing the parallel build and justify your options in the analysis.

In both situations (sequential or parallel build), the overall "build" should fail if any of the stages fail.

[1] for more information about Scripted vs. Declarative pipelines check `https://www.jenkins.io/doc/book/pipeline/`

Project - Part 1: Goal 1: Pipeline Design (Team Goals)

- For each Task, a sketch/design for each Pipeline, along with a description must be provided:
- You should reason about how you are going to orchestrate the Pipeline taking into consideration factors such as stage dependency, fail fast, etc. This reasoning must be explicitly stated in the Readme.

Project - Part 1: Goal 2: Pipeline Stages (1/5) (Team Goals)

These are some of the goals for each Task. Please note that the presentation order does not force the execution order. **Tasks should be implemented on Gradle as much as possible.**

The following goals should be performed by all team members:

- **Repository Checkout** Checkout the GIT repository;
- **Deployment file** Build and Publish the deployment file on Jenkins (*e.g.* for CMS: *.jar*);
- **Javadoc** Generate and Publish the Javadoc on Jenkins;

Evaluation is individual, so it is expected to see commits for all team members for the work each has carried out.

Even though all students must be able to perform all the following goals, each group of goals should be implemented by one student on the repository. Assessment question can be related to your goals group or any other.

Project - Part 1: Goal 2: Pipeline Stages (2/5) (Individual Goals)

Goals 2.a):

- **Integration Tests Execution** Execute the Integration Tests;
- **Integration Tests HTML Report Generation** Generate the Integration Tests HTML Report;
- **Integration Tests HTML Report Publishing** Publish the Integration Tests HTML Report on Jenkins;
- **Integration Tests HTML Coverage Report Generation** Generate the Integration Tests HTML Coverage Report;
- **Integration Tests HTML Coverage Report Publishing** Publish the Integration Tests HTML Coverage Report on Jenkins;

Project - Part 1: Goal 2: Pipeline Stages (3/5) (Individual Goals)

Goals 2.b):

- **Unit Tests Execution** Execute the Unit Tests;
- **Unit Tests HTML Report Generation** Generate the Unit Tests HTML Report;
- **Unit Tests HTML Report Publishing** Publish the Unit Tests HTML Report on Jenkins;
- **Unit Tests HTML Coverage Report Generation** Generate the Unit Tests HTML Coverage Report;
- **Unit Tests HTML Coverage Report Publishing** Publish the Unit Tests HTML Coverage Report on Jenkins;

Project - Part 1: Goal 2: Pipeline Stages (4/5) (Individual Goals)

Goals 2.c):

- **Mutation Tests Execution** Execute the Mutation Tests;
- **Mutation Tests HTML Coverage Report Generation** Generate the Mutation Tests HTML Coverage Report;
- **Mutation Tests HTML Coverage Report Publishing** Publish the Mutation Tests HTML Coverage Report on Jenkins;

Project - Part 1: Goal 2: Pipeline Stages (5/5) (Individual Goals)

Goals 2.d):

- **Staging Deployment** Deploy the application (*e.g.* *.jar* file) to a pre-configured staging server (*e.g.* Tomcat Server instance). To simulate a staging environment you can use:
  - a remote machine (*e.g.* DEI's virtual servers infrastructure, Heroku,...)
  - a local Docker machine
- **System Test** Perform an automatic **smoke test**. This smoke test can be as simple as using **curl** to check if the base url of the application is responsive after staging deployment (*e.g.* on the Tomcat Server), ensuring that the application is properly deployed to the Staging Environment;
- **UI Acceptance Manual Tests** A user should be notified by email of the successful execution of all the previous tests and be asked to perform a manual test. In order to cancel the progression or proceed, a UI Acceptance Manual Test must take place. The pipeline should wait for a user manual confirmation on Jenkins; The email to send must include, among others: 1) the link to the running application that is going to be manually tested, and 2) the link to the pipeline where confirmation/cancellation action is to be performed;
- **Continuous Integration Feedback** Push a tag to the repository with the Jenkins build number and status (*e.g.* Build#XX-Passed or Build#XX-Failed). The XX should be replaced by the build number.

Project - Part 1: Goal 3: Pipeline Implementation (Team/Individual Goals)

- Each team should create only two jobs on Jenkins, one for each Task implementation;
- The Pipeline must work either on Linux and Windows systems;
- Each student implements one of the group goals;
- Apart from using a central Jenkins Server instance for your group's Pipeline, each student should be able to demonstrate the Pipeline execution on their own PC, using the same Jenkins Pipeline Configuration as all other students. **Remember that the Jenkins job configuration must be in your repository and treated as any other source code artefact.**

Project - Part 1: Goal 4: Pipeline Evaluation Report (Team Goals)

- A performance evaluation report for each Task Pipeline execution should be written;
- You should provide the mean result for at least three executions of each Pipeline;
- This report should compare the mean time taken to execute a successful build for each Task;
- The report should be written in the Readme file.