Continuous Deployment

EECS 2311 - Software Development Project

Click to edit Master text styles
Second level
Third level
Fifth level

Wednesday, February 26, 2020

YORK UNIVERSITÉ UNIVERSITY

But first, as promised…

- Once a system is released, users will find bugs or will want more features
- You may also have to react to the requirements of important stakeholders
- In our case, a stakeholder will visit our lab on Monday. You will have a chance to ask him about his requirements
- There are also **two** additional features for the final release...

New requirements

1. Your system must allow the users to select multiple objects at once in order to customize them
   - E.g. select all objects in the intersection, and increase their font size
2. Your system must also implement an Undo / Redo mechanism
   - This is an important feature that will require some research on how to implement it
   - Do not wait until the last week to start on it

Software deployment

- For your midterm submission, you created a release of your software manually
- In practice, the steps that create a new release that gets deployed once new code is pushed to the master branch can be automated
  - This is **Continuous Deployment**

Gradle

- Gradle is a modern build automation tool that a software project can use to automate tasks related to deployment
- Gradle comes installed with the latest version of Eclipse
  - If you don’t have it, you can install it through the Eclipse Marketplace (search for Buildship)

Gradle in Eclipse

- In the Eclipse Package Explorer, click on the little downward triangle, select Filters, and uncheck the Gradle build folder, and Gradle sub projects so that they are visible
- If you cannot see the Gradle Tasks window: Window → Show View → Other → Gradle → Gradle Tasks

Adding Gradle to your project

- You can add Gradle to any existing Eclipse project
  - Right-click on the project name → Configure → Add Gradle Nature
  - Then open the Gradle Tasks window
    - Window → Show View → Other → Gradle → Gradle Tasks
  - Under build setup, right-click on init → Run Gradle Tasks
  - If you now right-click on the project name → Gradle → Refresh Gradle Project, you should be able to see the **`build.gradle`** file in Package Explorer
  - The **`build.gradle`** file describes all the tasks necessary to build and deploy your system

Gradle code

- Gradle uses a language called Groovy to express the necessary tasks
- We provide a sample **`build.gradle`** file that you can use as a starting point for your project
  - See link in course website
- For most builds for this course, this is all you will need
  - Copy it and replace the main class for the jar file

Building with Gradle in Eclipse

- In the Gradle Tasks window, expand your project, expand the **`build`** task group, and double-click on **`build`**
- This runs several tasks, such as
  - Resolving dependencies
  - Compiling all code
  - Running your tests
  - Creating a runnable jar

Building with Gradle in Eclipse

- Results are shown in the Gradle Executions window
- If something goes wrong, detailed information can be found in the Console window

Continuous Integration/Deployment

- Building with Gradle provides many benefits, such as resolving dependencies for all teammates in the same way
- It also allows for continuous integration/deployment
- As soon as changes are pushed to the master branch on github, the Gradle build executes automatically to:

1. Ensure tests are passing (continuous integration)
2. Upload the latest version as a release (continuous deployment)

- We will use Circle CI for this purpose

Circle CI

- Go to circleci.com, click on Log In, click on Log In with Github, and authenticate with your Github credentials
- This way Circle CI has access to your Github repositories
- Find your EECS 2311 project and click on Set Up Project
- Circle CI should be able to detect that you have a Gradle project
  - If not, select Gradle from the drop down menu

Circle CI

- If the top level of your github repository contains the **`build.gradle`** file, then you can click on Start Building
- If everything is inside a directory (typical for Eclipse projects), edit the provided **`config.yml`** as in the posted example on the course website
- In either case, you will need to add the last part of the provided **`config.yml`** in order to automatically release
- For more information on Circle CI, see https://circleci.com/docs/2.0/

Circle CI

- In order to automatically release, you must setup three environment variables in CircleCI
- Click on Project Settings, then Environment Variables. Add values for GITHUB_TOKEN, PROJECT_USERNAME, and PROJECT_REPONAME
- For the GITHUB_TOKEN value: Go to Settings under your profile, click on Developer Settings, Personal Access Tokens, Generate New Token
  - Add permissions for **`repo`**, and **`admin:repo_hook`**

Full pipeline

- Once everything is set up correctly, you should be able to:

1. Push a new version to the master branch of your project
2. Relax while the new version is tested automatically and a new release is created in your github page for your users

   If something goes wrong (such as tests that failed), you will be notified by email

Lab Task

- Set up the Gradle – Circle CI pipeline for your project (as a team)
- Demonstrate to the TA that pushing a new version to the master branch creates a new release on your github page