DevOps: Assignment 3

Mini-Project

Jonathan D'penha

PRN: 23030142005

Lab 01 (10 Marks):

Option 1: Consume any publically available API and display result.

• Consume any publically available API (e.g. https://github.com/public-apis/public-apis)

• Display Result (local machine demo allowed)

Explanation:

The provided Python script appears to be a simple program for fetching and displaying the definition of a word using an online API. Here's a breakdown of how it works:

1. The **fetch_dictionary_entry** function takes an API URL as input, sends a GET request to that URL using the **requests** library, and returns the JSON response if the status code is 200 (indicating success). Otherwise, it prints an error message and returns **None**.
2. The **display_definition** function takes the fetched entry data (if available) and prints out the word, part of speech, and definition(s) for that word. It iterates over each meaning and its corresponding definitions.
3. In the main block, it prompts the user to input a word. It constructs the API URL using the input word and then calls the **fetch_dictionary_entry** function to get the entry data. Finally, it calls the **display_definition** function to display the definition(s) for the word.

DevOps: Assignment 3

Jonathan D’penha

PRN: 23030142005

Lab 02 (20 Marks):

• Create a GitHub Repo.

• Create pipeline in Jenkins to build code automatically when code is updated in Github Repo (latest code should be downloaded on Jenkins server and build it locally).

• Execute any Two Options from below once build is done (local demo allowed)

- o **Option 1:** Create pipeline and send an email with status of build based on code change
- o **Option 2:** Create pipeline to store record status of job in database
- o **Option 3:** Send Slack notification on build status

Explanation:

The Jenkins job is configured to connect to the GitHub repository and monitor the main branch for any changes in the code.

Build Steps:

1. The Jenkins job performs build steps such as compiling and running the Java code and pulling updates from the GitHub repository.
2. The build steps include compiling a Java file (‘Helloworld.java’), running the compiled program, and pulling updates from the GitHub repository.

Post-build, the project sends notifications:

1. **Option 1 (Email):** After the build, an email is sent to the specified recipients with the build status. This is done through the Extended Email Publisher plugin.
2. **Option 3 (Slack):** Additionally, a Slack notification is sent to the specified Slack channel based on the build status. This is done through the Slack Notifier plugin.
3. This Jenkins job, when triggered by changes to the GitHub repository, will automatically build the latest code and notify stakeholders via email and notify stakeholders via email and Slack.

This Jenkins configuration represents a freestyle project designed to automate the build process for a GitHub repository. Here's a step-by-step explanation of how it works in the context of the provided requirements:

DevOps: Assignment 3

Jonathan D’penha

PRN: 23030142005

Step-by-Step Explanation:

Source Control Management (SCM) Configuration:

- The project is configured to use Git SCM (**hudson.plugins.git.GitSCM**) to fetch the source code from the specified GitHub repository (https://github.com/jonathandpenha1/helloworld.git).
- It's set to monitor changes on the main branch.

Build Trigger Configuration:

- The project is configured to be triggered by GitHub push events (**com.cloudbees.jenkins.GitHubPushTrigger**). Whenever there's a push event in the specified GitHub repository, Jenkins will automatically trigger a build.

Builders Configuration:

The project contains a shell script task (hudson.tasks.Shell) responsible for executing the following commands:

- **javac Helloworld.java:** Compiles the Java source code file Helloworld.java.
- **java Helloworld:** Executes the compiled Java program.
- **git pull:** Pulls the latest changes from the GitHub repository to ensure that Jenkins has the most up-to-date code before each build.

Publishers Configuration:

Extended Email Publisher:

- Configured to send email notifications (**hudson.plugins.emailext.ExtendedEmailPublisher**) based on different build outcomes (always, on failure, on success).
- Email recipients are specified in the <**recipientList**> tag.

Slack Notifier:

- Configured to send Slack notifications (**jenkins.plugins.slack.SlackNotifier**) based on build status.
- Includes settings for the Slack team domain, authentication token, and notification preferences for different build outcomes.