Task 1: Python Testing & Troubleshooting

Repository: <https://github.com/RenzoPerriniDEV/WazuhProject>

**1A Identify and Document Issues**

**First issue:**

def extract\_errors(log\_data: str) -> list:

The function is receiving a string data as parameter, if you send an array or list it will trew an error. With this structure you can only check errors for one record.

**Second issue: Wrong way of using split**

lines = log\_data.split('\n')

The way implemented of split() is not the correct. It should handle each line iterating the list.

errors = [line.split('\n') for line in log\_data if 'ERROR' in line]

**Third issue: It is not handling multiples cases for ERROR word**

errors = [line for line in lines if 'ERROR' in line]

The current function might not handle all possible cases. For example, if the word 'ERROR' appears in upper or lower case, the current function would only capture those that match exactly. It might be useful to consider case insensitivity to make it more robust.

Another possible scenario that can arise is that the function is not removing whitespace around the word 'ERROR'.

**1B Develop a set of unit tests based on a real-world situation**

A screenshot of a computer

Description automatically generatedImplemented in this file:

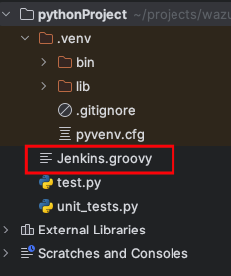
**1C Refactor the function to resolve the issues**

def extract\_errors(log\_data) -> list:  
 return [line.strip().split(‘\n’) for line in log\_data if ‘ERROR’ in line.upper()]

Task 2: Jenkins CI/CD Pipeline Creation and Management

**2A Draft a Jenkins Pipeline Script to achieve the requirements from the described scenario**

Implemented in this file:



**2B Document your choices and any prerequisites or configurations**

Script Choices:

* Declarative Pipeline: I chose to use a declarative pipeline instead of a script-style pipeline because of its structured and readable syntax.
* Three Main Stages:
  + Building Docker Image: The first stage builds the Docker image of the application. This stage is important to ensure the image is ready for deployment.
  + Pushing Docker Image: In the second stage, the Docker image is pushed to the specified Docker registry. This is crucial to make the image available for deployment in any environment.
  + Deploying to EC2: The third stage handles deploying the Docker image to an AWS EC2 instance. This involves stopping and removing any existing containers and then running a new container with the updated image.
* Handling Credentials: Jenkins-stored credentials are used to access AWS (access keys and region) and the Docker registry (registry credentials and SSH key to access the EC2 instance).

Prerequisites and Configurations:

* Jenkins Credentials: You need to configure the following credentials in Jenkins:
  + AWS access key with access ID and secret key.
  + Credentials for the Docker registry.
  + Private SSH key to access the EC2 instance.
* Environment Variable Configuration: Replace the values of environment variables like 'your-aws-region', 'your-docker-registry', 'your-docker-image', 'your-ec2-instance-id' with the corresponding values for your environment.
* EC2 Instance Configuration: Ensure you have an EC2 instance set up and ready to receive deployment. The instance should have Docker installed and configured to run Docker containers.
* Access Permissions: Ensure the EC2 instance has appropriate permissions to pull Docker images, stop/remove existing containers, and run new containers.
* Firewall and Ports: Ensure the EC2 instance's firewall allows traffic on port 80 (or the port you're using to expose your application).

These are the steps and configurations needed to use this Jenkins script to automate the process of building and deploying a Dockerized application to an AWS EC2 instance. If you need further assistance or have any questions, feel free to ask.

**2C Explain how the pipeline would handle errors or unexpected failures**

//This block defines actions to be performed after the pipeline stages have completed.  
post {  
 //If the pipeline succeeds, it shows the message "Deployment successful!".  
 success {  
 echo 'Deployment successful'  
 }  
 //If the pipeline fails, it shows the message "Deployment failed".  
 failure {  
 echo 'Deployment failed'  
 }  
 //If the last pipeline build and the current one failed, it shows the message "The current build failed!"  
 regression {  
 echo 'Tha current build failed'  
 }  
}

Task 3: Quick Feature Testing & Feedback

**3A Draft a Quick Test Plan**

**3B Execute a Few Test Cases and Document Results**