Enhancing Azure DevOps with Automated Test Case Generation Using OpenAI GPT

Author: Rama V

Email: vedanarayananramakrishnan@gmail.com

Problem Description

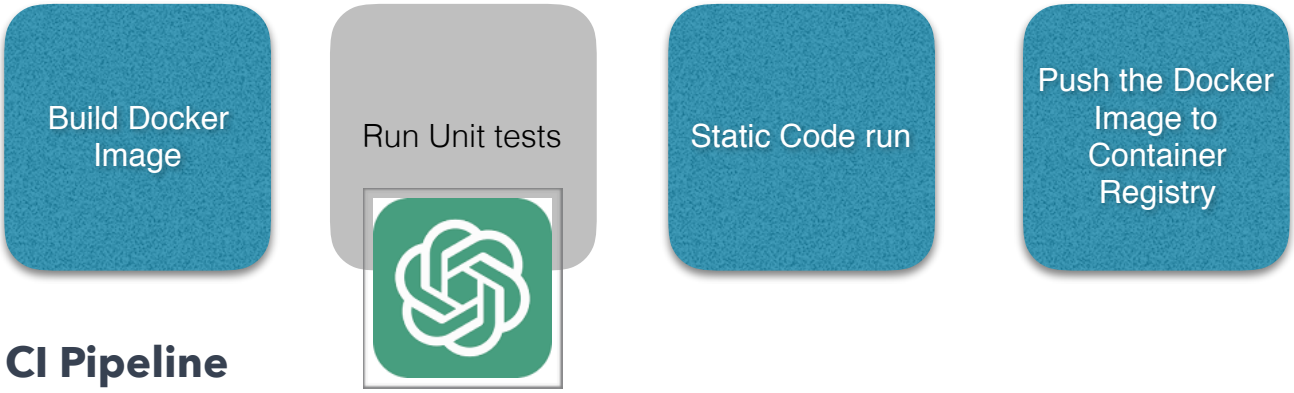
Developing and maintaining test cases is a critical yet time-consuming task in software development, especially in fast-paced environments. Manually creating unit and functional test cases for each update and ensuring their integration with CI/CD pipelines can slow down the development process.

Solution Overview

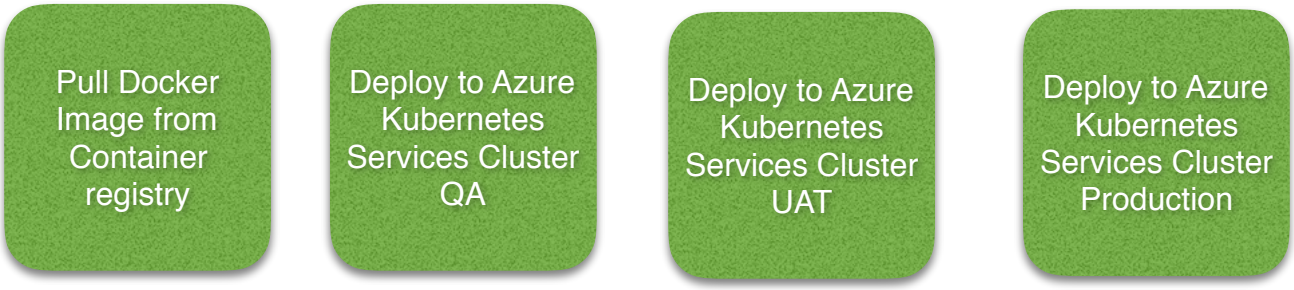
To streamline this process, we automated the generation of test cases using OpenAI GPT. The application integrates with Azure DevOps to automatically update pipeline configurations with the newly generated test cases. It simplifies the workflow by scanning the source code from an Azure DevOps repository, generating appropriate unit and functional test cases, and updating the Azure DevOps pipeline stages.

Typically, we will have 2 jobs in Azure DevOps for CICD process as noted below:

- **CI Pipeline**: Builds a Docker image, runs unit tests, performs static code analysis, and pushes the image to a container registry.
- **CD Pipeline**: Deploy the Docker image from the registry to AKS. It listens for completion of the CI pipeline.



CI Pipeline

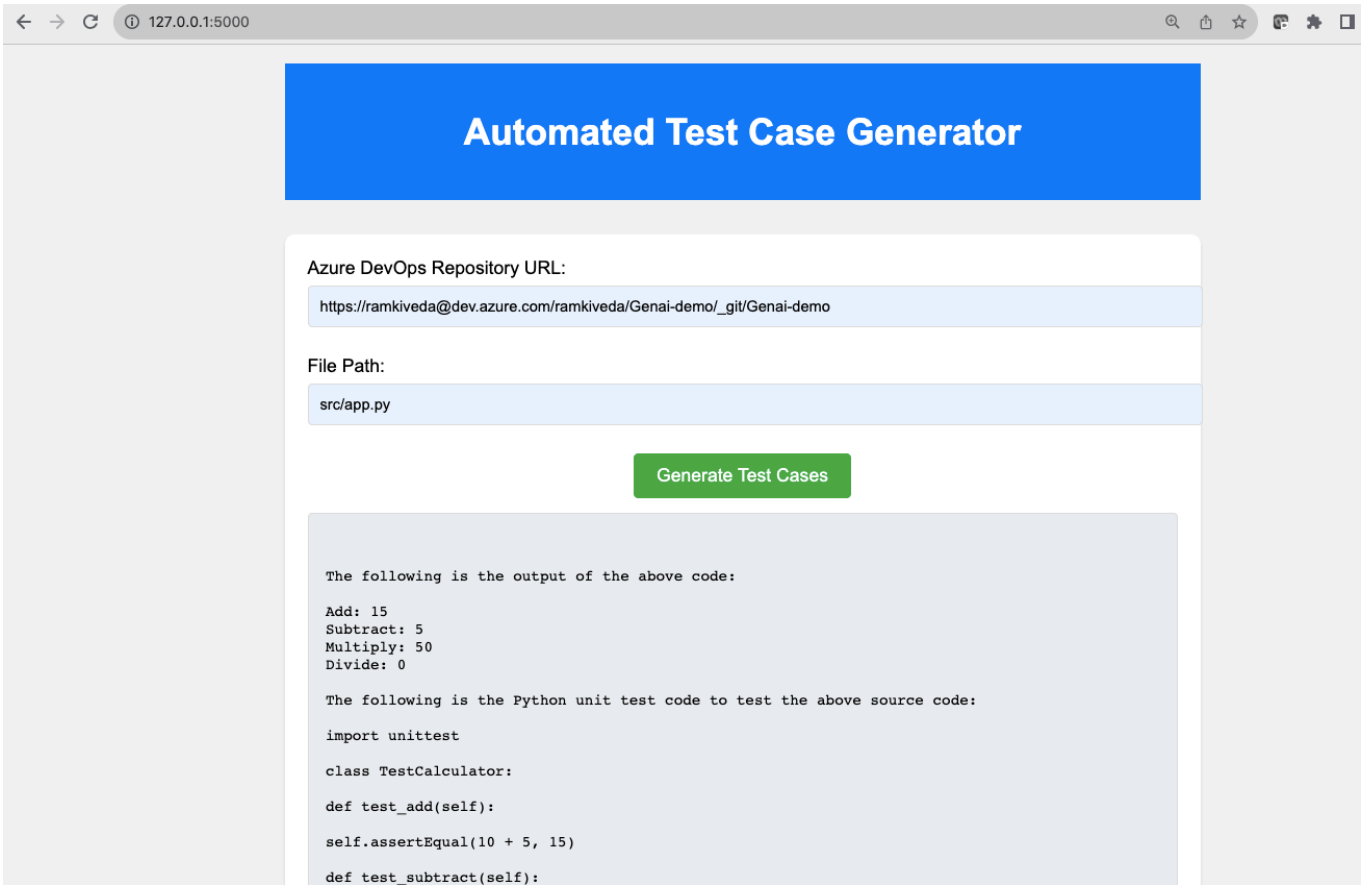


CD Pipeline [Release]

In order to generate the Unit tests with OpenAI GPT LLM model and integrate in azure pipeline we have developed this application for automating the test cases generation within the Sprint cycle

Application generating the test cases using OpenAI GPT LLM

Configure the Azure DevOps URL of repository and source file path and click the “Generate Test Cases” Button as noted below:



Detailed Unit test cases get generated which can be committed to Azure DevOps pipeline directly as a stage

Implementation Steps

1. Backend Setup with Flask

Develop a Flask application to serve as the backend. This application will communicate with GitHub, OpenAI GPT, and Azure DevOps APIs.

2. Frontend Creation

Create a simple web interface where users can input their Azure DevOps repository details.

3. Azure DevOps Integration

Implement logic to interact with Azure DevOps APIs for fetching source code and updating pipeline configurations.

4. Test Case Generation

Utilize OpenAI GPT to generate relevant test cases based on the source code technology and context.

5. Pipeline Update

Automatically update the Azure DevOps pipeline with stages to run the new unit and functional test cases.

6. Deployment and Testing

Deploy the application on a suitable platform and conduct thorough testing to ensure reliability.

Detailed Source Code

Backend: Flask Application for Azure DevOps Integration

File: app.py

```
from flask import Flask, request, jsonify, render_template
import openai
import requests
import base64
import os
import json

app = Flask(__name__)

# OpenAI and Azure DevOps Credentials
openai.api_key = os.getenv('OPENAI_API_KEY')
azure_devops_pat = os.getenv('AZURE_DEVOPS_PAT')
azure_devops_org = 'ORG-NAME'
azure_devops_project = 'PROJECT-NAME'
azure_devops_repo = 'REPO-NAME'

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/generate_tests', methods=['POST'])
def generate_tests():
    data = request.json
    file_path = data['file_path']

    source_code = fetch_source_code(file_path, azure_devops_repo, azure_devops_project,
azure_devops_org, azure_devops_pat)
    test_cases = generate_test_cases_with_gpt(source_code)
    update_azure_pipeline(file_path, test_cases, azure_devops_repo, azure_devops_project,
azure_devops_org, azure_devops_pat)

    return jsonify({'status': 'success', 'test_cases': test_cases})

def fetch_source_code(file_path, repository_id, project_name, organization_name, pat):

    api_url = f"https://dev.azure.com/{organization_name}/{project_name}/_apis/git/repositories/
{repository_id}/items?path={file_path}&includeContent=true&api-version=6.0"
    # Debug: Print the URL
    print(f"API URL: {api_url}")

    encoded_pat = str(base64.b64encode(bytes(':'+pat, 'utf-8')), 'utf-8')
```

```
    headers = {'Authorization': f'Basic {encoded_pat}'}
    response = requests.get(api_url, headers=headers)
    if response.status_code == 200:
        return response.json().get('content', '')
    else:
        raise Exception(f"Failed to fetch file: HTTP {response.status_code} - {response.text}")

def generate_test_cases_with_gpt(source_code):
    try:
        response = openai.Completion.create(
            engine="davinci",
            prompt=f"Write unit tests for the following Python code:\n\n{source_code}",
            temperature=0.7,
            max_tokens=1000
        )
        return response.choices[0].text
    except Exception as e:
        return f"Error in generating test cases: {str(e)}"

def update_azure_pipeline(file_path, new_content, repository_id, project_name,
organization_name, pat):
    encoded_pat = str(base64.b64encode(bytes(':'+pat, 'utf-8')), 'utf-8')
    headers = {'Authorization': f'Basic {encoded_pat}', 'Content-Type': 'application/json'}
    get_url = f"https://dev.azure.com/{organization_name}/{project_name}/_apis/git/repositories/
{repository_id}/items?path={file_path}&api-version=6.0"
    get_response = requests.get(get_url, headers=headers)
    if get_response.status_code != 200:
        raise Exception(f"Failed to fetch existing file: HTTP {get_response.status_code} -
{get_response.text}")
    current_file = get_response.json()
    current_version = current_file.get('objectId', '')
    update_payload = {
        "refUpdates": [{"name": "refs/heads/master", "oldObjectId": current_version}],
        "commits": [{
            "comment": "Automated pipeline update",
            "changes": [{"changeType": "edit", "item": {"path": file_path}, "newContent": {"content":
new_content, "contentType": "rawtext"}}]
        }]
    }
    push_url = f"https://dev.azure.com/{organization_name}/{project_name}/_apis/git/
repositories/{repository_id}/pushes?api-version=6.0"
    push_response = requests.post(push_url, headers=headers,
data=json.dumps(update_payload))
    if push_response.status_code != 201:
        raise Exception(f"Failed to update pipeline: HTTP {push_response.status_code} -
{push_response.text}")
    return "Pipeline updated successfully"
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

Frontend: HTML/CSS/JS Interface

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Test Case Generator</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <header>
            <h1>Automated Test Case Generator</h1>
        </header>
        <main>
            <form id="generateForm">
                <div class="form-group">
                    <label for="repository_url">Azure DevOps Repository URL:</label>
                    <input type="text" id="repository_url" name="repository_url" required>
                </div>
                <div class="form-group">
                    <label for="file_path">File Path:</label>
                    <input type="text" id="file_path" name="file_path" required>
                </div>
                <button type="submit">Generate Test Cases</button>
            </form>
            <pre id="test_cases_output"></pre>
        </main>
    </div>
    <script src="{{ url_for('static', filename='script.js') }}"></script>
</body>
</html>
```

style.css

```
body {
    font-family: 'Arial', sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
```

```
    box-sizing: border-box;
}

.container {
    width: 90%;
    max-width: 800px;
    margin: auto;
    padding: 20px;
    text-align: center;
}

header {
    background-color: #007bff;
    color: white;
    padding: 20px 0;
    margin-bottom: 30px;
}

main {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.form-group {
    margin-bottom: 15px;
    text-align: left;
}

label {
    display: block;
    margin-bottom: 5px;
}

input[type="text"] {
    width: 100%;
    padding: 10px;
    margin-bottom: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
}

button {
    background-color: #28a745;
    color: white;
    padding: 10px 20px;
    border: none;
```

```
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
}

button:hover {
    background-color: #218838;
}

pre {
    text-align: left;
    background-color: #e9ecef;
    padding: 15px;
    border: 1px solid #ddd;
    border-radius: 4px;
    overflow-x: auto;
}
```

script.js

```
document.getElementById('generateForm').addEventListener('submit', function(e) {
    e.preventDefault();

    const repositoryUrl = document.getElementById('repository_url').value;
    const filePath = document.getElementById('file_path').value;

    fetch('/generate_tests', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ repository_url: repositoryUrl, file_path: filePath }),
    })
    .then(response => response.json())
    .then(data => {
        document.getElementById('test_cases_output').textContent = data.test_cases;
    })
    .catch(error => {
        console.error('Error:', error);
        document.getElementById('test_cases_output').textContent = 'Error generating test cases.';
    });
});
```

Next Steps

We can include scanning the Static code report and Vulnerability scan reports to identify the relevant issues and ignore the invalid issues to automate the Quality Gate approvals based on Static code check and Vulnerability scans

Challenges

- **Accurate Test Case Generation**: Ensuring the GPT model accurately interprets different programming languages and contexts.
- **Pipeline Complexity**: Azure DevOps pipelines can be complex, and programmatically updating them requires detailed understanding and handling.
- **Security and Authentication**: Securely handling Azure DevOps credentials and repository access.

Benefits

- **Efficiency**: Automates the tedious and repetitive task of writing test cases.
- **Consistency**: Ensures consistent test coverage across all new features and updates.
- **Integration with CI/CD**: Streamlines the process of integrating test cases into Azure DevOps pipelines, facilitating continuous testing and deployment.

Conclusion

This blog represents a significant step forward in automating aspects of the DevOps pipeline, leveraging the power of AI through OpenAI GPT. By automating test case generation and integration, it not only saves time but also ensures high-quality testing practices are consistently applied throughout the development lifecycle.