WIS DevOps Practical Test Pipeline

Pipeline Overview

This is a comprehensive CI/CD workflow implemented using GitHub Actions. It encompasses the entire software development lifecycle from code commit to deployment, integrating build, quality assurance, security scanning, and deployment steps. The pipeline is designed to ensure code quality, security, and smooth deployment while utilizing best practices in DevOps.

Pipeline Structure

The pipeline is structured into four distinct jobs: `build`, `quality_gate`, `security_scan`, and `deploy`. Each job is configured to run on `ubuntu-latest` and is dependent on the successful completion of preceding jobs to ensure a robust and reliable pipeline.

1. Build Job

Purpose: The `build` job compiles the code, packages the build artifacts, and prepares them for further processing.

Steps:

- **Checkout Code**

```yaml
- name: Checkout code
  uses: actions/checkout@v3
```

Description: This step clones the repository to the runner environment, ensuring that the latest codebase is available for subsequent steps.

- **Make Script Executable**

```
- name: Make scripts executable
  run: chmod +x ./.github/workflows/scripts/build.sh
```

Description: Ensures that the build script has execute permissions, which is necessary for running shell scripts on Unix-based systems.

- **Set Up Node.js**

```
- name: Set up Node.js
  uses: actions/setup-node@v3
  with:
    node-version: '21'
```

Description: Configures Node.js version 21 on the runner. This setup is critical for projects that require specific Node.js versions for dependency management and script execution.

- **Install dependencies**

```
- name: Install Dependencies
  run: npm install
  working-directory: ./wp-content/themes/twentytwentyfour
```

Description: Installs the npm dependencies required for the WordPress theme. The `working-directory` parameter ensures that the command is run in the correct directory.

- **Run Build Script**

```
- name: Run build script
  run: ./.github/workflows/scripts/build.sh
```

Description: Executes the build script, which is responsible for compiling and packaging the project.

- **Create Artifacts**

```
- name: Create Artifacts
  run: tar -czf /tmp/artifacts.tar.gz ./ > /dev/null
```

Description: Archives the project files into a compressed tarball. This step prepares the build output for artifact storage and transfer.

- **Upload Compiled Assets**

```
- name: Upload Compiled Assets
  uses: actions/upload-artifact@v3
  with:
    name: artifacts
    path: /tmp/artifacts.tar.gz
```

Description: Uploads the tarball as an artifact, making it available for download in subsequent jobs. This facilitates the transfer of build outputs between jobs.

Choice Explanation:

- **Node.js Version 21:** Selected to leverage the latest features and optimizations provided by Node.js, ensuring compatibility with modern dependencies.
- **Artifact Management:** Using tarballs and GitHub Actions artifact features simplifies artifact transfer and versioning.

2. Quality Gate Job

Purpose: The `quality_gate` job performs static code analysis and integrates with SonarCloud to ensure code quality and maintainability.

Steps:

- **Checkout Code**

```
- name: Checkout code
  uses: actions/checkout@v3
```

Description: Retrieves the codebase for analysis, ensuring that the latest changes are evaluated.

- **Set Up PHP**

```
- name: Set up PHP
  uses: shivammathur/setup-php@v2
  with:
    php-version: '8.1'
```

Description: Configures PHP 8.1, which is essential for running PHPStan and ensuring compatibility with the project's PHP code.

- **Make Scripts Executable**

```
- name: Make scripts executable
  run: chmod +x ./.github/workflows/scripts/phpstan_analysis.sh ./.github/work
```

Description: Ensures that the static analysis scripts are executable, necessary for running them in the CI environment.

- **Run PHPStan Analysis**

```
- name: Run PHPStan analysis
  run: ./.github/workflows/scripts/phpstan_analysis.sh
```

Description: Executes PHPStan to perform static code analysis, helping to identify potential issues in the codebase.

- **Run Sonar Quality Gate**

```
- name: Run Sonar Quality Gate
  uses: SonarSource/sonarcloud-github-action@master
  continue-on-error: true
  env:
    GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
    SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
```

Description: Integrates with SonarCloud to perform comprehensive code quality and security checks. The `continue-on-error` option ensures that the pipeline does not fail immediately on errors but still reports them.

Choice Explanation:

- **PHP 8.1:** Provides a stable environment compatible with the codebase and analysis tools.
- **SonarCloud Integration:** Offers deep insights into code quality and security while allowing the pipeline to proceed even if quality issues are detected.

3. Security Scan Job

Purpose: The `security_scan` job performs security scans to identify vulnerabilities in the codebase.

Steps:

- **Checkout Code**

```
- name: Checkout code
  uses: actions/checkout@v3
```

Description: Ensures that the codebase is available for security scanning.

- **Make Scripts Executable**

```
- name: Make scripts executable
  run: chmod +x ./.github/workflows/scripts/security-scan.sh
```

Description: Grants execution permissions to the security scan script.

- **Run Security Scan**

```
- name: Run security scan
  run: ./.github/workflows/scripts/security-scan.sh
```

1. **Description:** Executes the security scan to identify vulnerabilities and potential security issues in the codebase.

Choice Explanation:

- Security Scan Integration: Ensures that vulnerabilities are detected early, contributing to a more secure codebase and reducing the risk of deploying insecure code.

4. Deploy Job

Purpose: The `deploy` job handles the deployment of the application to the target environment.

Steps:

- **Download Source Code**

```yaml
- name: Download source code
  uses: actions/download-artifact@v3
  with:
    name: artifacts
```

Description: Retrieves the build artifacts from the artifact storage for deployment.

- **Extract Artifacts**

```yaml
- name: Extract artifacts
  run: mkdir artifacts && tar -xzf artifacts.tar.gz -C artifacts > /dev/null 2>
```

Description: Unpacks the tarball into a directory, preparing the files for deployment.

- **Make Scripts Executable**

```yaml
- name: Make scripts executable
  run: chmod +x ./artifacts/.github/workflows/scripts/deploy.sh
```

Description: Ensures the deployment script has execution permissions.

- **Run Deploy Script**

```
- name: Run deploy script
  run: ./artifacts/.github/workflows/scripts/deploy.sh
  env:
   DB_NAME: ${{ secrets.DB_NAME }}
   DB_USER: ${{ secrets.DB_USER }}
   DB_PASSWORD: ${{ secrets.DB_PASSWORD }}
   SSH_HOST: ${{ secrets.SSH_HOST }}
   SSH_USER: ${{ secrets.SSH_USER }}
   SSH_KEY: ${{ secrets.SSH_KEY }}
   SSH_PORT: ${{ secrets.SSH_PORT }}
   DEPLOY_PATH: ${{ secrets.DEPLOY_PATH }}
```

Description: Executes the deployment script, deploying the project to the target environment. The environment variables securely manage sensitive information required for deployment.

Choice Explanation:

- **Artifact Deployment:** Ensures that only verified and tested code is deployed, reducing the risk of introducing bugs.
- **Environment Variables:** Utilizes GitHub Secrets to manage sensitive information securely, enhancing the security of the deployment process.

Best Practices and Considerations

- **Dependency Management:** By installing dependencies in a specific directory and using modern Node.js and PHP versions, the pipeline ensures compatibility and reduces issues during the build and analysis phases.
- **Error Handling:** The `continue-on-error` option in the quality gate job prevents the pipeline from failing prematurely, providing flexibility in managing quality and security issues.
- **Secure Credential Management:** GitHub Secrets are used to manage sensitive information, ensuring that credentials and deployment paths are handled securely.