Technical Design Document for Azure Pipeline Template Library

Purpose

This document serves as a comprehensive guide for developers to understand and effectively use the Azure Pipeline Template Library. It ensures standardization and best practices while abstracting the complexities of CI/CD processes.

---

1. Architectural Overview

The pipeline template library simplifies the creation and management of CI/CD pipelines for microservices. It provides modular, reusable, and parameterized templates designed to enforce consistency and scalability across projects.

Design Principles:

- **Encapsulation:** Developers use pre-defined `jobs` for tasks instead of interacting directly with `steps`.
- **Reusability:** Templates are parameterized to fit various use cases.
- **Modularity:** The library separates responsibilities into specialized job files.
- **Scalability:** Easily expandable for new services, stages, or environments.

Template Structure:

The repository az-pipelines-golang contains pipeline templates specifically designed for Go projects. The directory structure is as follows:

```
├── go
│   ├── jobs
│   │   ├── build.yml   # Template for Docker image builds, scan and push
│   │   ├── tests.yml   # Template for code analysis and testing
│   │   ├── deploy.yml  # Template for deployments
│   ├── steps
│   │   ├── build
│   │   │   ├── docker-build.yml
│   │   │   ├── docker-push.yml
│   │   │   ├── snyk-container-scan.yml
│   │   │   ├── clean.yml
│   │   ├── deploy
│   │   │   └── deploy.yml
│   │   ├── tests
│   │       ├── sonar-code-analysis.yml
│   │       ├── snyk-app-security-analysis.yml
│   │       ├── integrations.yml
├── README.md           # Developer documentation
```

---

2. Job Templates for Developers

Developers interact only with the `jobs` templates, which encapsulate all logic. Here's an overview of available job templates:

2.1 Build Job (jobs/build.yml)

Handles building Docker images, scanning for vulnerabilities, and pushing images to a container registry.

Parameters:

- `repository` *(string, required)*: Docker repository name.
- `tag` *(string, optional)*: Docker image tag (default: `$(Build.BuildId)`).
- `containerRegistry` *(string, optional)*: Container registry name (default: `uservice`).

2.2 Test Job (jobs/tests.yml)

Performs code analysis, security scanning, and integration tests.

Parameters:

- `GoVersion` *(string, optional)*: Go version (default: `1.23.4`).

2.3 Deployment Job (jobs/deploy.yml)

Manages deployments to staging and production environments.

Parameters:

- `environment` *(string, required)*: Target environment (e.g., `staging`, `prod`).

---

3. Main Pipeline File

Developers integrate the job templates into their Azure Pipelines for GoLang Repo as follows:

azure-pipelines.yml:

```yaml
trigger:
 - main
 - develop

resources:
  repositories:
  - repository: templates
    name: 'peng/az-pipelines-go'
    type: 'git'
    ref: 'refs/heads/main'

variables:
  repository: 'test-service'
  tag: '$(Build.BuildId)'
```

```yaml
  golang_version: '1.23.4'

stages:
- stage: Build
  jobs:
    - template: golang/jobs/build.yml@templates
      parameters:
        repository: $(repository)
        tag: $(tag)

- stage: Tests
  jobs:
    - template: golang/jobs/tests.yml@templates
      parameters:
        GoVersion: $(golang_version)

- stage: Deploy
  jobs:
    - template: golang/jobs/deploy.yml@templates
      parameters:
        environment: 'staging'
    - template: golang/jobs/deploy.yml@templates
      parameters:
        environment: 'prod'
```