#### 1. ****Introduction****

Bitbucket Pipelines and GitHub Actions are CI/CD automation tools that allow you to build, test, and deploy your code. While they serve the same purpose, their configuration structures and YAML syntaxes are different. This guide details how to convert a Bitbucket pipeline (bitbucket-pipelines.yml) to a GitHub Actions workflow (main.yml).

#### 2. ****Key Differences Between Bitbucket Pipelines and GitHub Actions****

| **Feature** | **Bitbucket Pipelines** | **GitHub Actions** |  |
| --- | --- | --- | --- |
| **Triggers** | Defined under pipelines: | Defined under on: (e.g., push, pull\_request, schedule) |  |
| **Image** | Global image: key | Use runs-on: for environments or container: for Docker |  |
| **Steps** | Defined under steps: in stages | Defined under steps: inside jobs: |  |
| **Caches** | caches: section for specific tools (e.g., node) | Use actions/cache@v3 action |  |
| **Artifacts** | artifacts: keyword | actions/upload-artifact@v3 for saving |  |
| **Services** | services: for databases and external dependencies | services: keyword in jobs |  |
| **Environment Variables** | variables: or inline in steps | Use env: at job or step level |  |

#### 3. ****Converting Each Section****

##### 3.1 ****Triggers****

In Bitbucket, pipeline triggers are defined under the pipelines: section, usually with options like default, branches, or pull-requests. In GitHub Actions, you define triggers under the on: section, such as for push, pull\_request, or schedule.

**Bitbucket Pipeline (Trigger Example)**:

yaml

Copy code

pipelines:

default:

- step:

script:

- echo "This runs on every push"

**GitHub Actions Workflow (Trigger Example)**:

yaml

Copy code

name: CI Workflowon:

push:

branches:

- main

##### 3.2 ****Docker Image****

In Bitbucket, you specify the Docker image globally using image:. In GitHub Actions, jobs run on virtual environments using runs-on:, or you can specify a Docker image with container:.

**Bitbucket Pipeline (Docker Image Example)**:

yaml

Copy code

image: node:14

**GitHub Actions Workflow (Docker Image Example)**:

yaml

Copy code

jobs:

build:

runs-on: ubuntu-latest

container:

image: node:14

##### 3.3 ****Steps****

Steps define the individual commands to run during the pipeline. Bitbucket steps are defined under steps: with script:, while in GitHub Actions, steps are inside jobs: and use the run: key for commands.

**Bitbucket Pipeline (Steps Example)**:

yaml

Copy code

pipelines:

default:

- step:

name: Build and Test

script:

- npm install

- npm run build

- npm test

**GitHub Actions Workflow (Steps Example)**:

yaml

Copy code

jobs:

build-and-test:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3 # Check out the code

- name: Install Dependencies

run: npm install

- name: Build the Project

run: npm run build

- name: Run Tests

run: npm test

##### 3.4 ****Caching****

Bitbucket allows you to cache dependencies between pipeline runs using caches:, for example, to speed up Node.js builds. In GitHub Actions, you use the actions/cache@v3 action to cache dependencies.

**Bitbucket Pipeline (Cache Example)**:

yaml

Copy code

pipelines:

default:

- step:

caches:

- node

script:

- npm install

**GitHub Actions Workflow (Cache Example)**:

yaml

Copy code

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- name: Cache Node modules

uses: actions/cache@v3

with:

path: ~/.npm

key: ${{ runner.os }}-node-${{ hashFiles('package-lock.json') }}

- run: npm install

##### 3.5 ****Artifacts****

Bitbucket uses artifacts: to store files generated during the pipeline. GitHub Actions uses actions/upload-artifact@v3 to upload and save artifacts from a job.

**Bitbucket Pipeline (Artifact Example)**:

yaml

Copy code

pipelines:

default:

- step:

script:

- npm run build

artifacts:

- dist/\*\*

**GitHub Actions Workflow (Artifact Example)**:

yaml

Copy code

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Build Project

run: npm run build

- name: Upload Artifact

uses: actions/upload-artifact@v3

with:

name: build

path: dist/\*\*

##### 3.6 ****Services****

Both Bitbucket and GitHub Actions allow the use of service containers like databases or external dependencies (e.g., MySQL). In GitHub Actions, you use services: inside a job to define the service.

**Bitbucket Pipeline (Service Example)**:

yaml

Copy code

pipelines:

default:

- step:

services:

- mysql

script:

- npm test

**GitHub Actions Workflow (Service Example)**:

yaml

Copy code

jobs:

test:

runs-on: ubuntu-latest

services:

mysql:

image: mysql:5.7

env:

MYSQL\_ROOT\_PASSWORD: password

ports:

- 3306:3306

steps:

- name: Run Tests

run: npm test

##### 3.7 ****Environment Variables****

In Bitbucket, you can define environment variables under variables: or inline within a step. In GitHub Actions, you define environment variables using env: at the job or step level.

**Bitbucket Pipeline (Environment Variables Example)**:

yaml

Copy code

pipelines:

default:

- step:

script:

- echo $MY\_VAR

variables:

MY\_VAR: "hello world"

**GitHub Actions Workflow (Environment Variables Example)**:

yaml

Copy code

jobs:

build:

runs-on: ubuntu-latest

env:

MY\_VAR: "hello world"

steps:

- name: Echo Environment Variable

run: echo $MY\_VAR

#### 4. ****Parallelism****

In Bitbucket, parallel steps are defined under parallel:. GitHub Actions supports parallel jobs using the strategy.matrix: feature, which allows jobs to run in parallel based on different configurations.

**Bitbucket Pipeline (Parallel Steps Example)**:

yaml

Copy code

pipelines:

default:

- parallel:

- step:

name: Unit Tests

script:

- npm run test:unit

- step:

name: Integration Tests

script:

- npm run test:integration

**GitHub Actions Workflow (Parallel Jobs Example)**:

yaml

Copy code

jobs:

test:

runs-on: ubuntu-latest

strategy:

matrix:

test-type: [unit, integration]

steps:

- uses: actions/checkout@v3

- name: Run ${{ matrix.test-type }} tests

run: npm run test:${{ matrix.test-type }}