

### C1/CD using Github Actions

Let all the actions do the job for us



#### Who am i?

Muhammad Fakhri Putra Supriyadi [FAI]

More about me: f4r4w4y

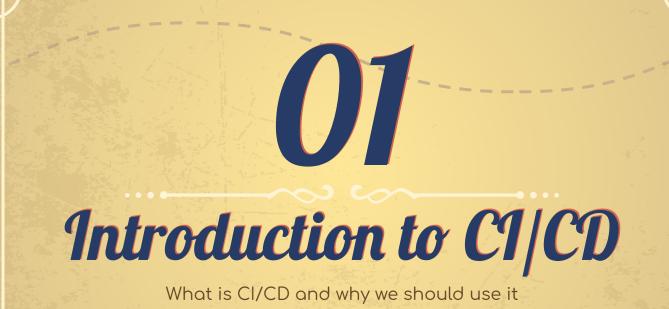
#### Table of Content (ToC)

- Ox1 Introduction to CI/CD
  - What is CI/CD and why we should use it
- Ox2 Introduction to Github Actions
  What is Github Actions and how to use it
- Ox3 Using Github Actions

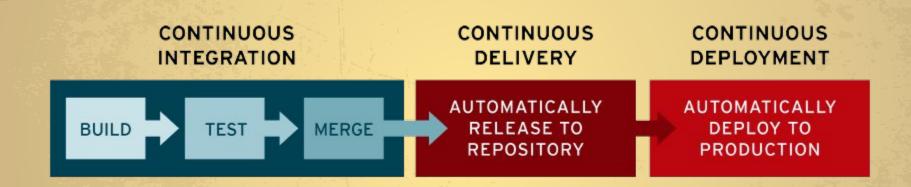
How to use actions built by others and how to create our own actions

0x4 QnA

Ask me anything and i will answer if i can



#### What is CI/CD



Source: What is CI/CD?

#### Why use CI/CD

#### **Integration Hell**

It is really hard to integrate each individual work into one big system

#### Project Management

It is also hard to maintain a big project manually all the time

#### Human Error

It is so much easier to make mistakes during integration and deployment

## 100,000

And more code repositories on GitHub contain secret access keys, source: Thousands of API and cryptographic keys leaking on GitHub every day

# 02

## Introduction to Github Actions

What is Github Actions and how to use it

#### What is Github Actions

"Automate, customize, and execute your software development workflows right in your repository with GitHub Actions. You can discover, create, and share actions to perform any job you'd like, including CI/CD, and combine actions in a completely customized workflow." - official documentation.

- CI/CD right inside your repository
- Set all CI/CD configuration in the same directory as with the project
- CI/CD as a package, (use something built by others, or build something yourself and let other people use it)

#### How to use Github Actions

#### Just do it in three simple steps:

- 1. Create directory structure like this \_github/workflows/ inside your repository
- 2. Create a workflow with .yml extension, ex: code-test.yml .
- 3. Commit all changes and push the repository to github

```
name: learn-github-actions
on: [push]
jobs:
   check-bats-version:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - uses: actions/setup-node@v2
    - run: npm install -g bats
    - run: bats -v
```

Description: Name of the action (optional)

```
name: learn-github-actions
on: [push]
jobs:
   check-bats-version:
    runs-on: ubuntu-latest
   steps:
    - uses: actions/checkout@v2
    - uses: actions/setup-node@v2
    - run: npm install -g bats
    - run: bats -v
```

Description: Event that will automatically trigger the workflow, for more detail check at Workflow syntax for GitHub Actions

```
name: learn-github-actions
on: [push]
jobs:
    check-bats-version:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - uses: actions/setup-node@v2
    - run: npm install -g bats
    - run: bats -v
```

Description: All jobs exist in this workflow and will run one-by-one

```
name: learn-github-actions
on: [push]
jobs:
   check-bats-version:
    runs-on: ubuntu-latest
   steps:
    - uses: actions/checkout@v2
    - uses: actions/setup-node@v2
    - run: npm install -g bats
    - run: bats -v
```

Description: Runner in which this workflow will be running

```
name: learn-github-actions
on: [push]
jobs:
   check-bats-version:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - uses: actions/setup-node@v2
    - run: npm install -g bats
    - run: bats -v
```

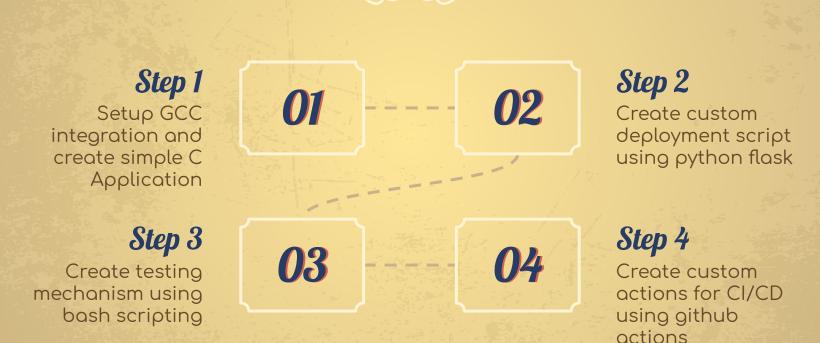
Description: All steps exist in check-bats-version job that will also run one-by-one

# 03

## **Using Github Actions**

How to use actions built by others and how to create our own actions

### Lets create C testing and deployment



### Step 1.A [GCC Integration]

We will be using egor-tensin/setup-gcc: GitHub action to set up GCC

```
steps:
    - uses: actions/checkout@v2

- name: Setup GCC Integration
    uses: actions/setup-node@v2
    with:
        version: latest
        platform: x64
```

#### Step 1.B [Simple C Application]

We will create a simple application that calculate give student score and output the quality of the score

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char **argv) {
     if (argc == 2) {
          int num = (int) strtoull(argv[1], &argv[1], 10);
     } else {
          printf("\n\
                  Example usage: ./calculate 5 \n\
          \r
                        output: Lumayan\n\
          \r");
```

#### Step 1.C [Simple C Application]

We will create a simple application that calculate give student score and output the quality of the score

```
int num = (int) strtoull(argv[1], &argv[1], 10);
if (num == 100) {
    printf("Sangat Baik");
} else if (num >= 80) {
    printf("Baik");
} else if (num >= 50) {
    printf("Lumayan");
} else if (num >= 25) {
    printf("Buruk");
} else if (num >= 0) {
    printf("Sangat Buruk");
}
```

#### Step 2 [Deployment Script Python]

We will create a simple flask app to serve the C Application

```
from subprocess import check_output

from flask import Flask
app = Flask(__name__)

@app.route('/<param>', methods=['GET', 'POST'])
def score(param=0):
    output = check_output(f"./calculate {param}")
    return f"Nilai anda {output}"
```

### Step 3 [Testing Script in Bash]

We will create a simple testing script using bash

```
#!/bin/sh -1

if [[ "$(./calculate 0)" == "Sangat Buruk" ]]
then
    echo "Test Passed"
    exit 0

else
    echo "Test Failed"
    exit 1
fi
```

### Step 4 [CI/CD using Github Actions]

We will create a simple testing script using bash

```
name: Student Scoring Testing and Deployment

on: [push]

jobs:
    build_and_test:
       runs-on: ubuntu-latest
       name: Build application using GCC and test it
       ...
```



QnA

Ask me anything and i will answer if i can... **Go ahead** 

## Thanks

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon and infographics & images by Freepik.