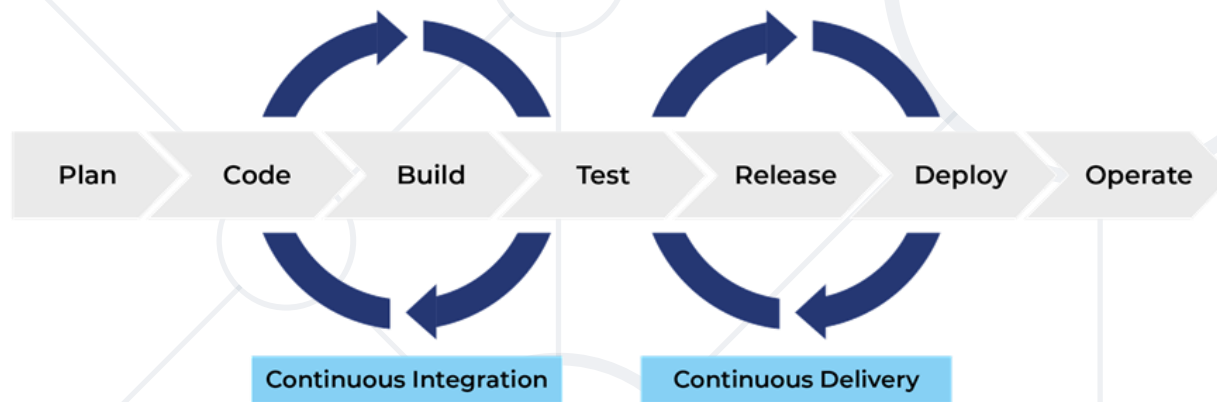


# Continuous Integration, CI / CD Pipelines

Continuous Integration, Continuous Delivery,  
Continuous Deployment, GitHub Actions, Render.com



**SoftUni Team**  
Technical Trainers



**SoftUni**



Software University

<https://softuni.bg>

# Have a Question?

**sli.do**

**#Dev-Ops**

# Table of Contents

1. The CI/CD Pipelines
2. CI/CD Tools
3. GitHub Actions





# The CI/CD Pipelines

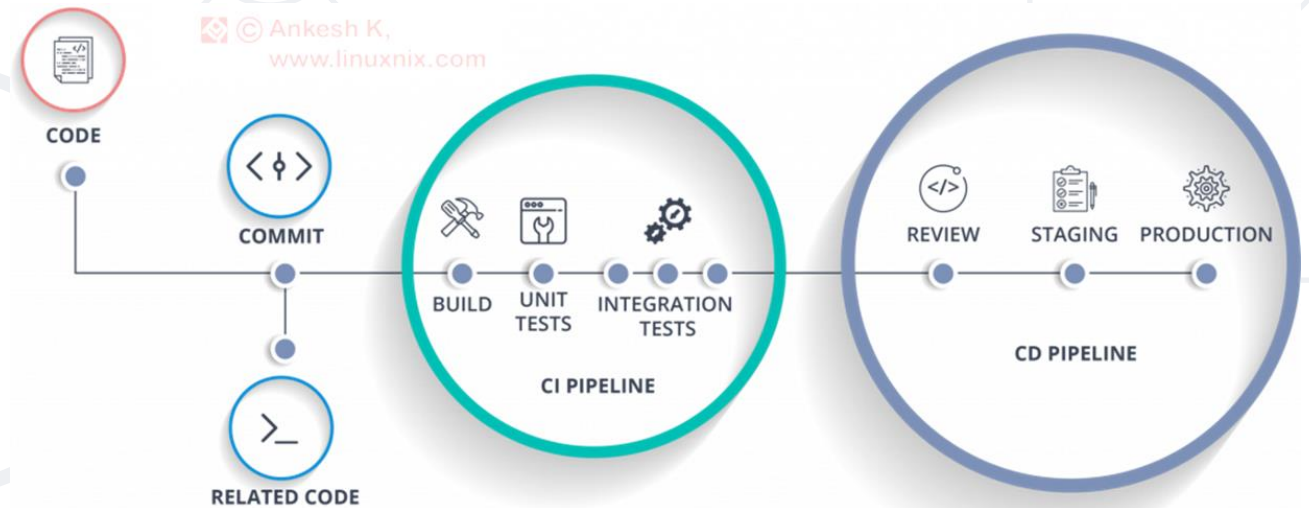
# What is CI/CD?

- **CI/CD = Continuous Integration + Continuous Delivery**  
( + Continuous Deployment )



- **Automates** much of the process to get new code from a **commit** into **production**
  - Developers **regularly** merge their code changes into a **central** repository, which is then **automatically tested** and **deployed** to **production** to ensure frequent and reliable software updates

- **CI/CD pipeline**
  - Continuously **integrate** and **release** new features
- **Continuous integration (CI)**
  - Write code, test it and **integrate** it in the product
- **Continuous delivery (CD)**
  - Continuously **release** new features
- **QAs** monitor and sometimes maintain the CI/CD pipeline



- Integrating the code from different developers frequently (at least once a day)
- **Automated building** and **testing** the code
  - Typically, at **Git push** in certain branch
- **Finding integration problems** and bugs early
  - Continuously maintain software quality
- CI is implemented by a **CI system** (like Jenkins, GitHub Actions, TeamCity, Azure Pipelines)

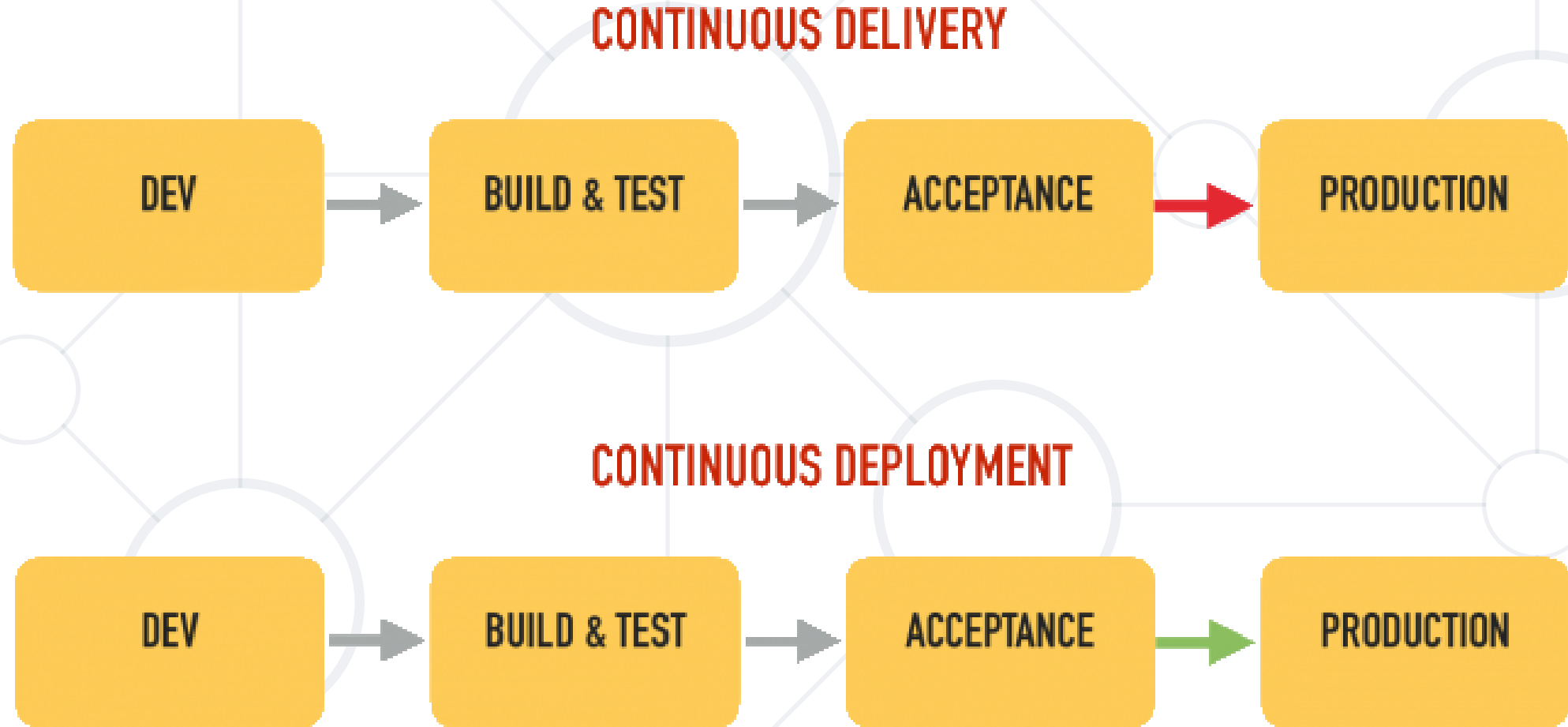
- Regularly execute **automated tests** as part of the software delivery pipeline
  - Ensures consistent software quality
- Implemented with a **CI system**
  - **Unit tests** executed at each commit / push
  - **Integration tests** executed at each major commit / push
  - **End-to-end tests** executed every night (execution takes hours)



- Keeping your codebase **deployable at any point**
- **CD** continuously verifies that
  - Software **builds** correctly
  - Passes the **automated tests**
  - Has all the necessary **configuration** and assets for **deployment in production**
- E.g., build an **.apk** package for Android apps

- Continuous **automated deployment**
- E.g., after each **git push** in certain branch
  - The software is **built**, the **tests** are executed, and binaries are **deployed** and configured correctly
- Automated deployment typically uses a **testing environment**
  - Sometimes directly to the **production** servers
- Deployment should be done by script (not by hand)

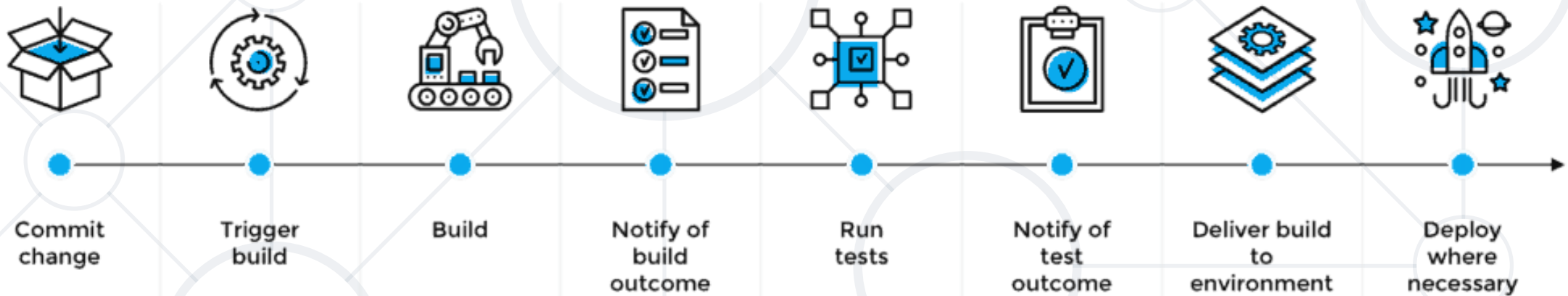
# Continuous Delivery vs. Continuous Deployment



- **CI/CD pipeline == CI + CD**
  - Continuously integrate, test and release new features
- On **git push**, the CI/CD pipeline does automatically
  - **Build** the software (compile, package, sign, etc.)
  - Run the **automated tests** (unit & integration)
  - **Deploy** in the testing environment & run E2E tests
    - Or only prepare for deployment
    - Or deploy directly on production

# CI/CD Pipeline View

## CI/CD Pipeline



- **Development** environment
  - Code commit
- **Testing** environment
  - Continuous integration, automated testing
- **Staging** environment
  - Continuous delivery, user acceptance test
- **Production** environment
  - Continuous deployment, monitoring

- A **single source repository**, which contains everything needed for the build
  - Source code, database structure, libraries, scripts, etc.
- **Frequent iterations and check-ins** to the main branch
  - Use small segments of code and merge them into the branch often
- **Automated and self-testing builds**

- Higher efficiency of web deployment
- Reduced risk of defects
- Faster product delivery
- Exclusive log generation
- Easier rollback of code changes
- More test liability
- Customer satisfaction







**CI/CD Tools**

# GitHub Actions

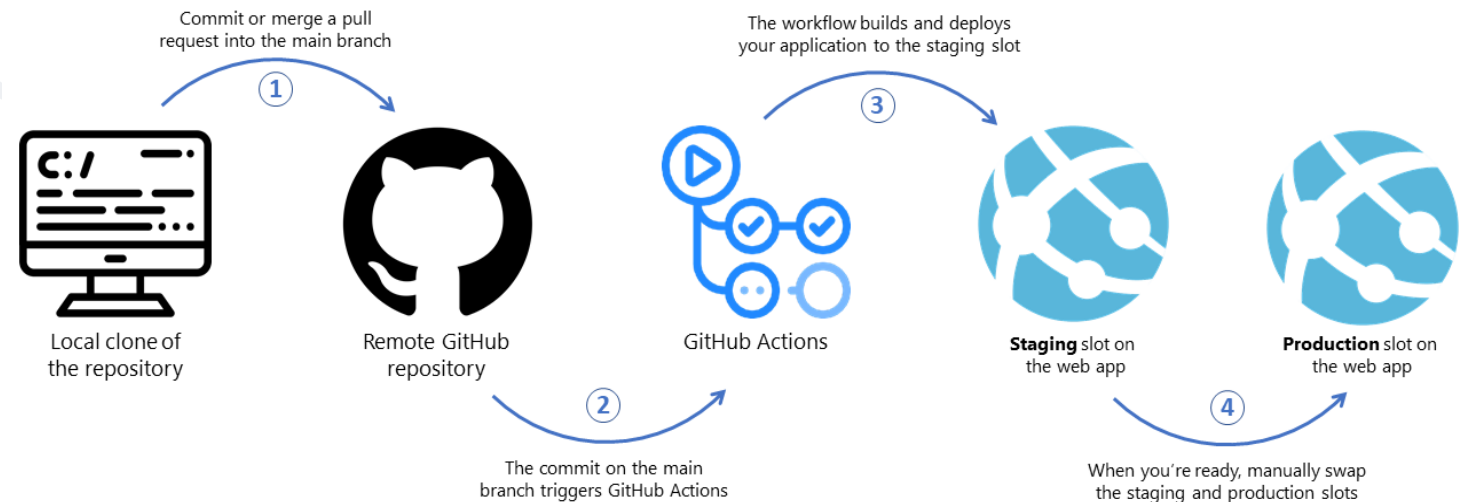
- GitHub Actions == powerful **CI/CD platform**
  - Integrated directly into GitHub repos
- Enables developers to **automate** workflows, build, test and code deployment
- Large **library** of pre-built actions and custom workflows
- Free for public repos + 2000 mins per month for private repos with the **free plan**



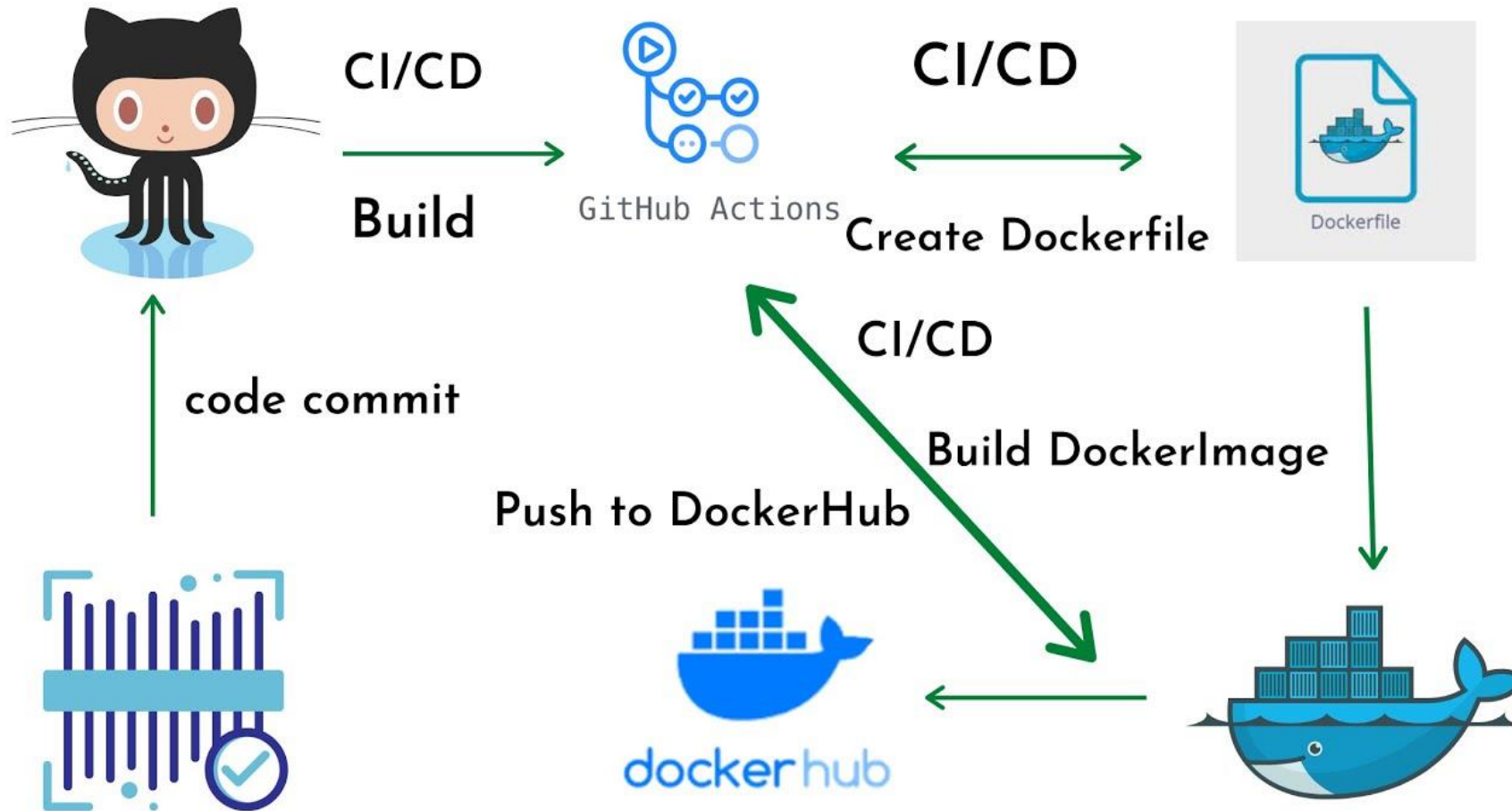
- Flexible environment
  - Supports **various** programming languages
- Allows developers to **trigger workflows**
  - Based on events like code commits, pull requests, issue updates
- Allows defining **custom** workflows
- YAML syntax

# GitHub Actions and Other Platforms (1)

- You can use it to integrate and deploy code changes to a **cloud application platform** and test, track, and manage these changes
- With **GitHub Actions for Azure**, you can deploy to Azure
- GitHub Actions also supports other **CI/CD tools, Docker, and automation platforms**



# GitHub Actions and Other Platforms



# CircleCI

- CircleCI == CI/CD platform
- Supports a wide range of programming languages
- Integrates with **version control systems**
  - e.g., GitHub and Bitbucket
- User-friendly configuration file
  - Allows custom workflows



# Azure DevOps

- Azure DevOps == set of deployment tools that enables
  - Planning
  - Developing
  - Testing
  - Deployment
- Facilitates creation of CI/CD pipelines





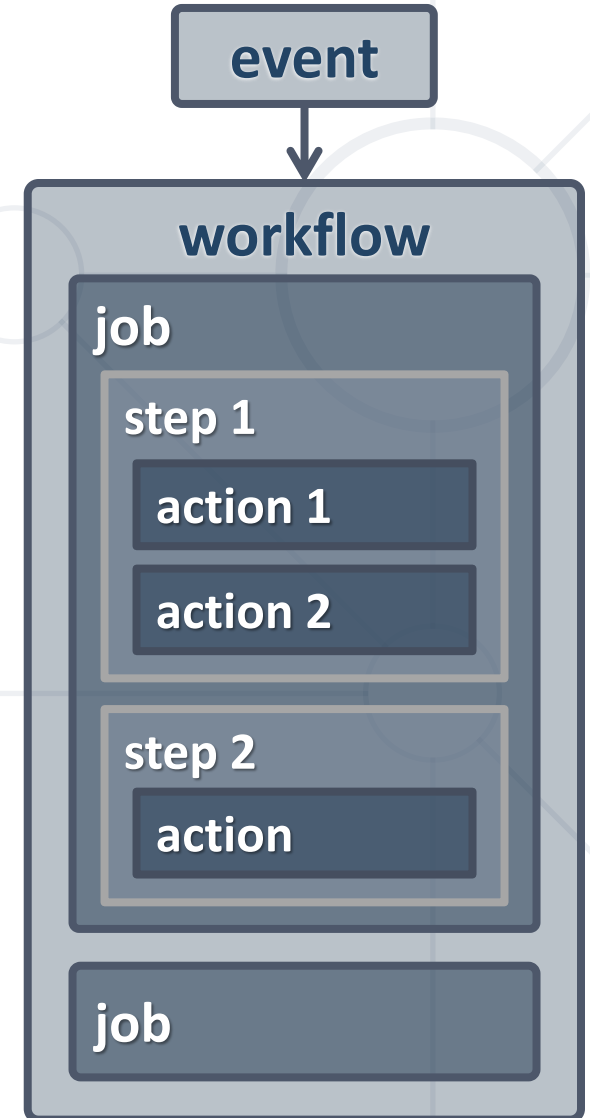
- Offers **services** for
  - Version control
  - Agile project management
  - CI/CD
  - Application monitoring
  - and many more...
- Supports **integration** with **version control systems**, e.g., Git
- Provides **code repository** for managing source code



# GitHub Actions

## Introduction

- **Events** execute **workflows**  
(one or several jobs, running in parallel)
- **Workflows** hold **jobs**  
(e.g., build, check security, deploy)
- **Jobs** hold **steps** (e.g., "checkout the code", "install .NET", "run tests", ...)
- **Steps** hold **actions**  
(commands like `dotnet test`)

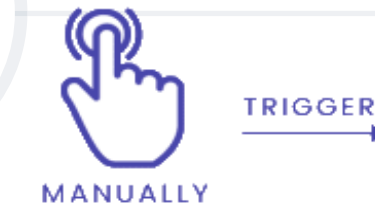


- Specific **triggers** that can activate workflows in a repository
- Allow **automation** of various **tasks** and **actions** based on different types of events that occur within the repository
- Each event can be used to **start a workflow** that performs **specific action**, e.g.,
  - Running tests
  - Deploy code
  - Sending notifications

- **Repository**
  - Specific to the repository and are triggered by actions like code pushes, pull requests, etc.
- **Workflow**
  - Related to the workflows themselves and are triggered by workflow-specific events
- **Webhook**
  - Triggered by external services integrated with GitHub using webhooks
- **External**
  - Specific to actions taken by external services
- **Internal**
  - Related to actions within the GitHub repository or organization

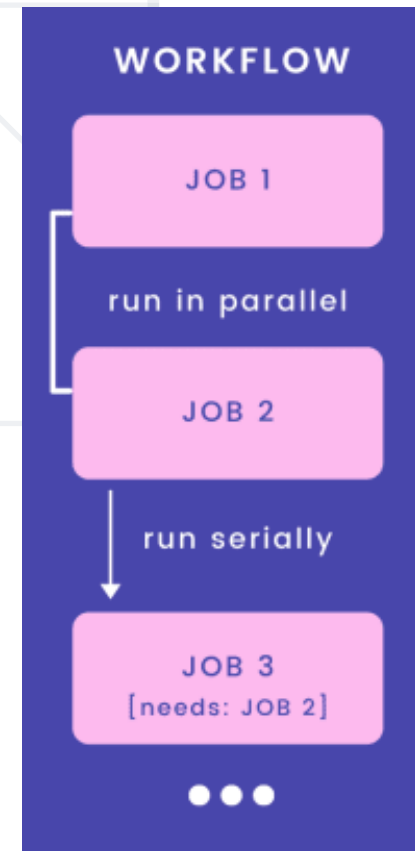
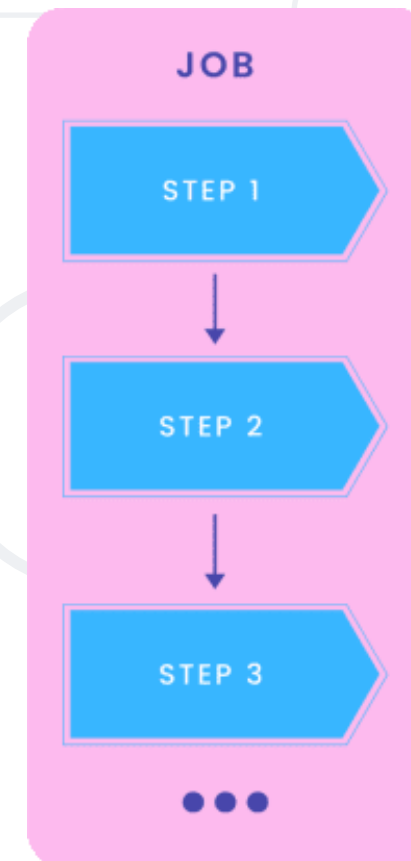
- **GitHub Actions workflow** is a configurable automated procedure
- Made of one or many **jobs**
- Defined by a **YAML file** in **.github/workflows** folder in your repo
- Can be triggered by **events** in the repo, on **schedule** or **manually**
- A **GitHub repository** can have **multiple workflows**

```
.github > workflows > my-workflow.yaml
1  name: learn-github-actions
2  on: [push]
3  jobs:
4    check-bats-version:
5      runs-on: ubuntu-latest
6      steps:
7 >   - name: Check out repository ...
9 >   - name: Install Node.js ...
13 >  - name: Install bats ...
15 >  - name: Run bats ...
```



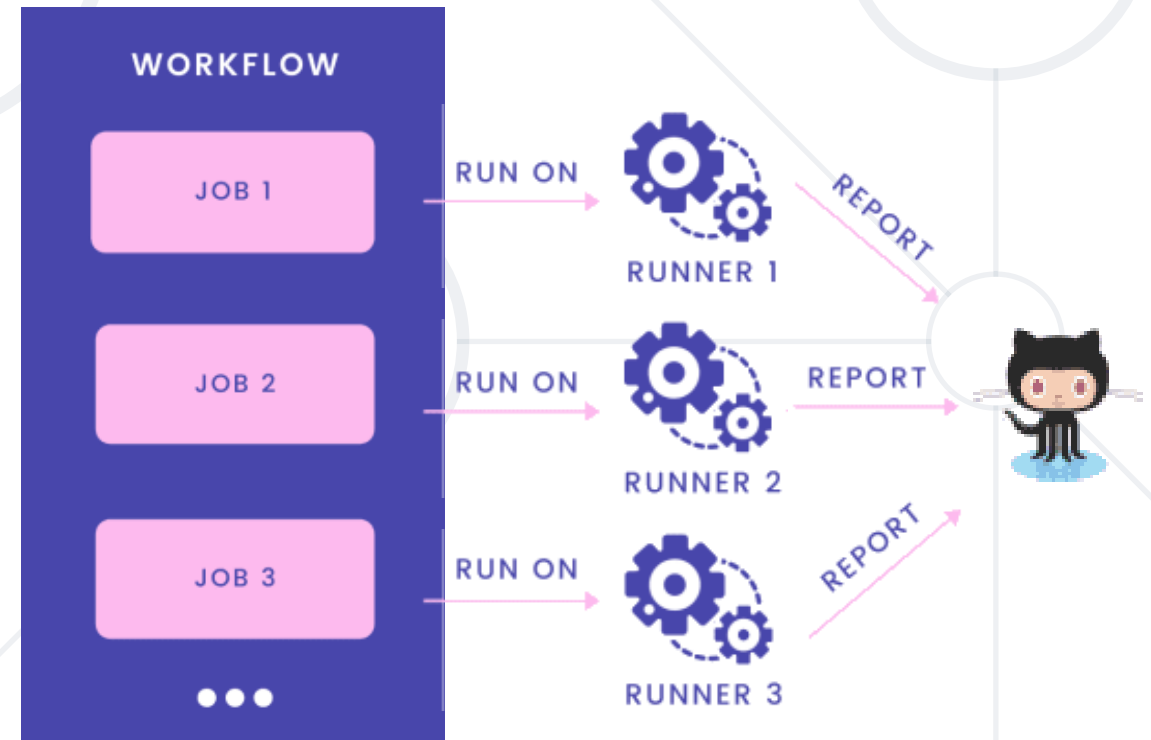
- **Job** == a **set of steps** that will be executed on **the same runner**
- All **jobs in the workflow** normally run in **parallel**
- When you have **jobs that depend on each other**, they run **serially**

```
github > workflows > my-workflow.yaml
1  name: learn-github-actions
2  on: [push]
3  jobs:
4    check-bats-version:
5      runs-on: ubuntu-latest
6      steps:
7 >     - name: Check out repository ...
9 >     - name: Install Node.js ...
13 >    - name: Install bats ...
15 >    - name: Run bats ...
```



- To **run jobs**, we must specify a **runner** for each of them
- A **runner** is a **server that runs jobs**
- Runs only **1 job at a time**
- Reports **job progress, logs,** and **results** back to GitHub
  - We can look at them in the UI of the repository
- Two types: **GitHub hosted** or **self-hosted**

```
3 jobs:  
4   check-bats-version:  
5     runs-on: ubuntu-latest
```

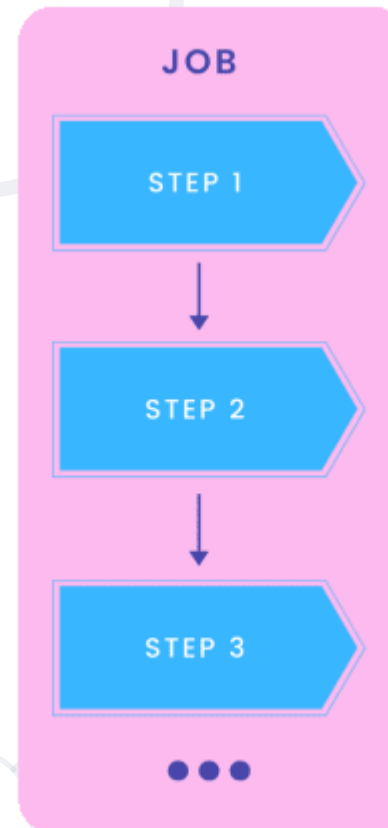




# Steps and Actions

- **Steps** are **individual tasks** within a **job**
- They run serially, one after another
- Each **step** is either a **shell script** that will be executed, or an **action** that will be run
- An **action** is basically a **standalone command**
- **Actions** run serially within a step
- **Actions** can be reused

```
3 jobs:
4   check-bats-version:
5     runs-on: ubuntu-latest
6     steps:
7       - name: Check out repository
8         uses: actions/checkout@v3
9       - name: Install Node.js ...
13      - name: Install bats ...
15      - name: Run bats ...
```



# Workflow Syntax Keywords

- **name**
  - for **names** of workflows, steps, which GitHub Action displays
- **on**
  - used to define which events can cause the workflow to run (**triggers**)
- **jobs**
  - used to **list jobs**
- **runs-on**
  - specify **runner environment**

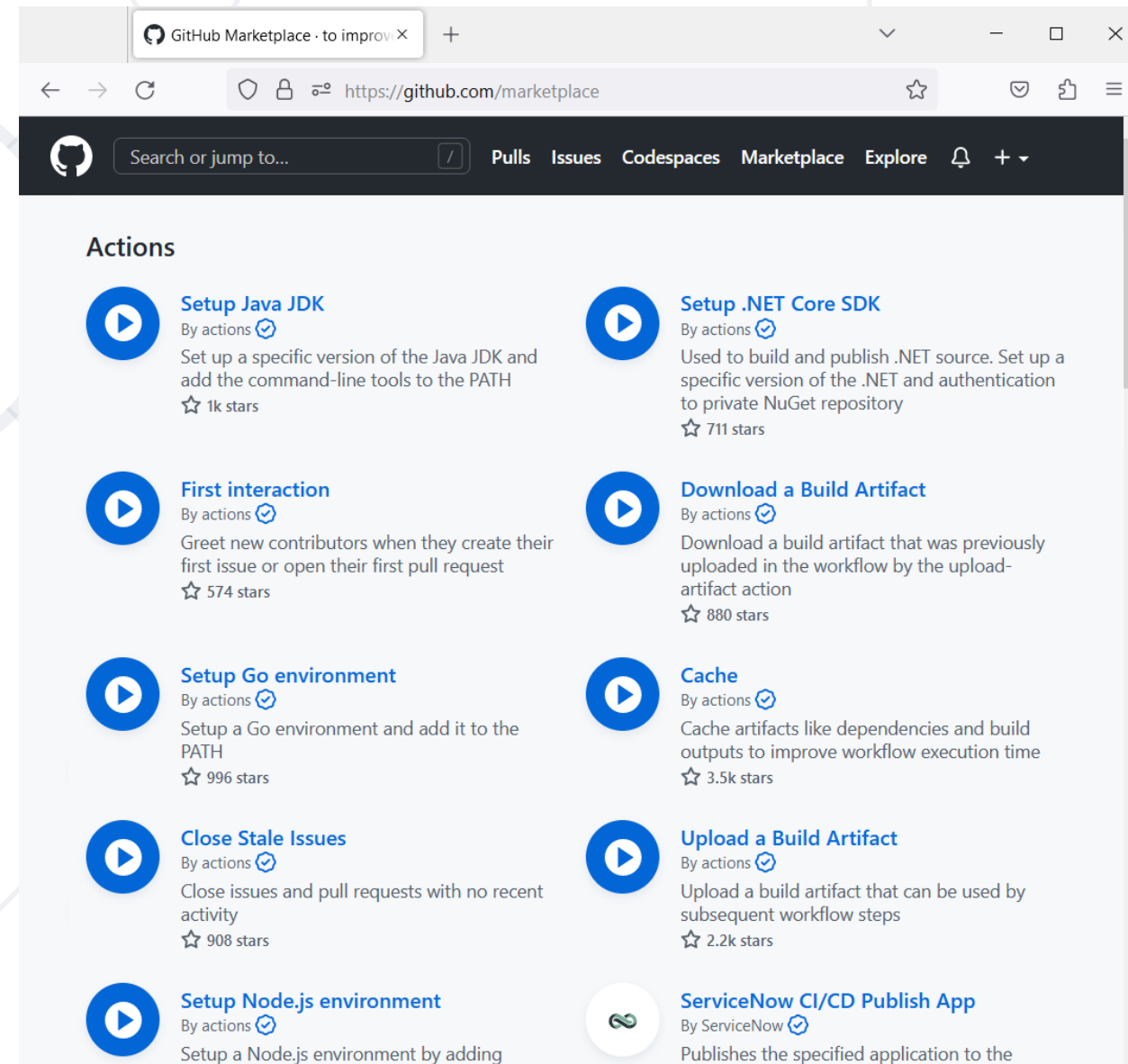
```
.github > workflows > my-workflow.yaml
1  name: learn-github-actions
2  on: [push]
3  jobs:
4      check-bats-version:
5          runs-on: ubuntu-latest
6          steps:
7              - name: Check out repository
8                uses: actions/checkout@v3
9              - name: Install Node.js
10               uses: actions/setup-node@v3
11               with:
12                   node-version: '14'
13              - name: Install bats
14                run: npm install -g bats
15              - name: Run bats
16                run: bats -v
```

# Workflow Syntax Keywords (2)

- **steps**
  - used to **list steps** to run in the job
- **uses**
  - **use an action** which is already defined with its version (v3)
- **with**
  - **input parameters** required by some actions
- **run**
  - tells the job to **execute a Shell command** on the runner

# GitHub Marketplace

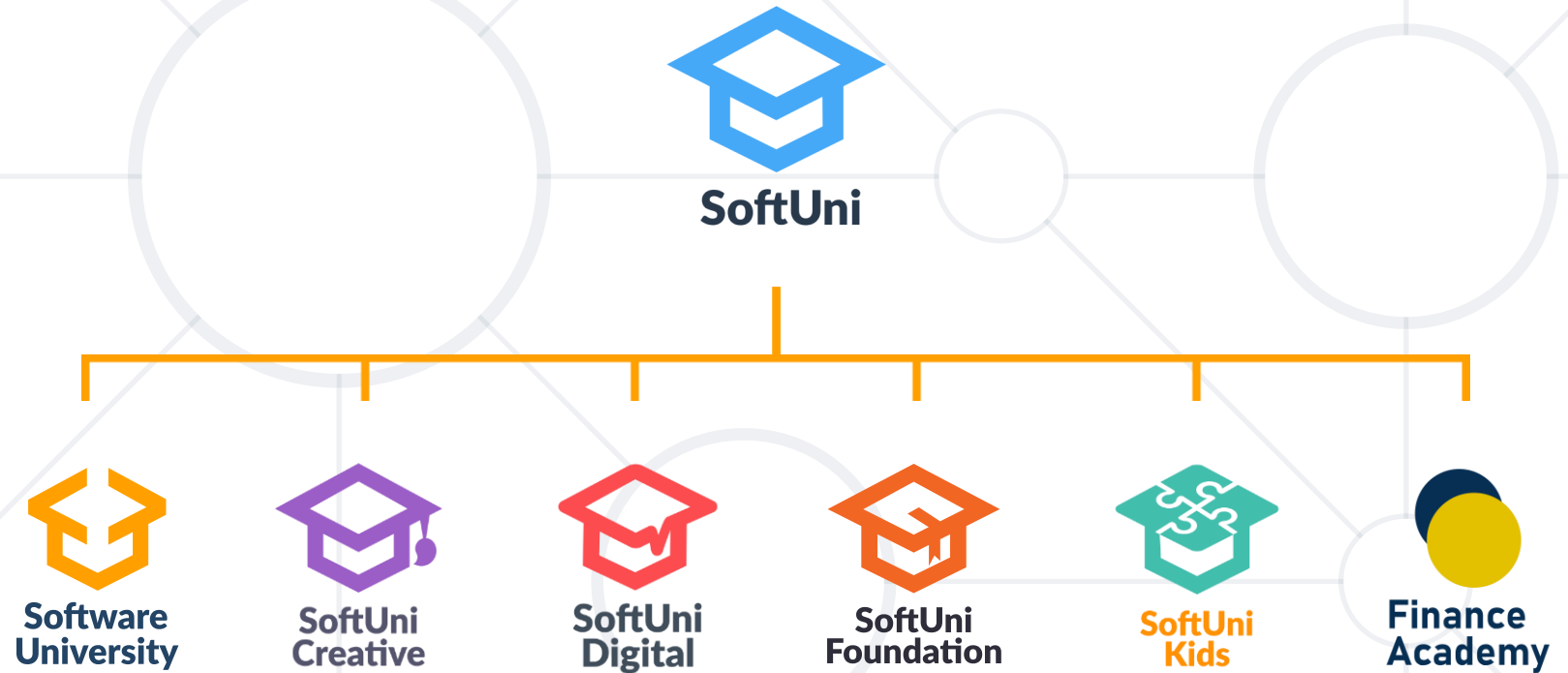
- GitHub Marketplace contains tools that add functionality and improve your workflow
- You can discover, browse, and install tools, including **GitHub Actions**
- GitHub uses it to suggest you **workflow templates** based on code in your repo



- **CI/CD** == a method to **frequently deliver apps** by introducing **automation** into **continuous delivery** and **continuous deployment**
- There are a lot of **CI/CD platforms**
  - e.g., **GitHub Actions**, in which you can create **workflows** to **automate** your **build, test** and **deployment pipeline**



# Questions?



# SoftUni Diamond Partners

**SUPER  
HOSTING  
.BG**



**Coca-Cola HBC  
Bulgaria**



**POKERSTARS**  
POKER | CASINO | SPORTS  
a Flutter International brand

**INDEAVR**  
Serving the high achievers



**AMBITIONED**

 **DRAFT  
KINGS**



**SOFTWARE  
GROUP**

createX



**Postbank**  
Решения за твоето утре

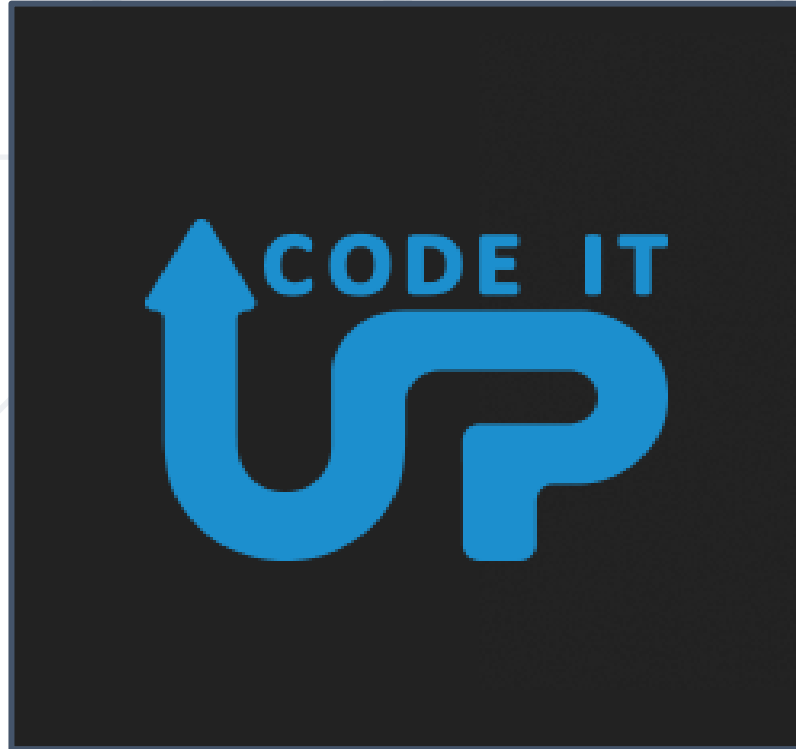


**BOSCH**

**DXC**  
TECHNOLOGY



**SmartIT**





- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>



- Software University – High-Quality Education, Profession and Job for Software Developers

- [softuni.bg](http://softuni.bg), [about.softuni.bg](http://about.softuni.bg)

- Software University Foundation

- [softuni.foundation](http://softuni.foundation)

- Software University @ Facebook

- [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)

- Software University Forums

- [forum.softuni.bg](http://forum.softuni.bg)



Software University

