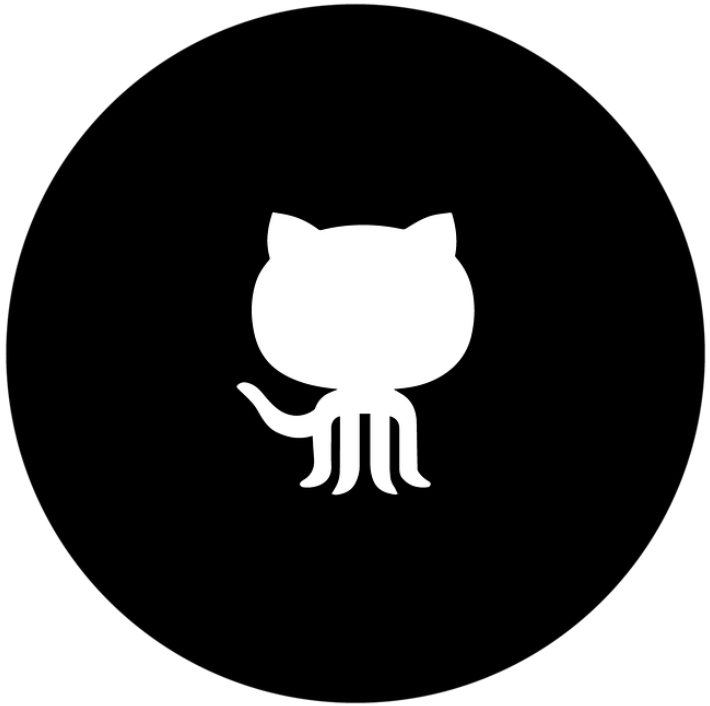
Github Actions

Boas van der Putten

PhD student Amsterdam UMC, the Netherlands

Goal + outline



Goal:

Show why and how you should Github Actions

Outline:

- GitHub Actions background
- Basic components of a Github action
- Real life examples
- Recap

What are Github Actions?

Automation of software development tasks, e.g.:

- Checking dependencies
- Updating version
- Verify pull requests
- Run functional tests

Part of continuous integration practices

Continuous Integration (CI)

Software development approach where small changes are incorporated often in your base code

Every integration needs automatic checks

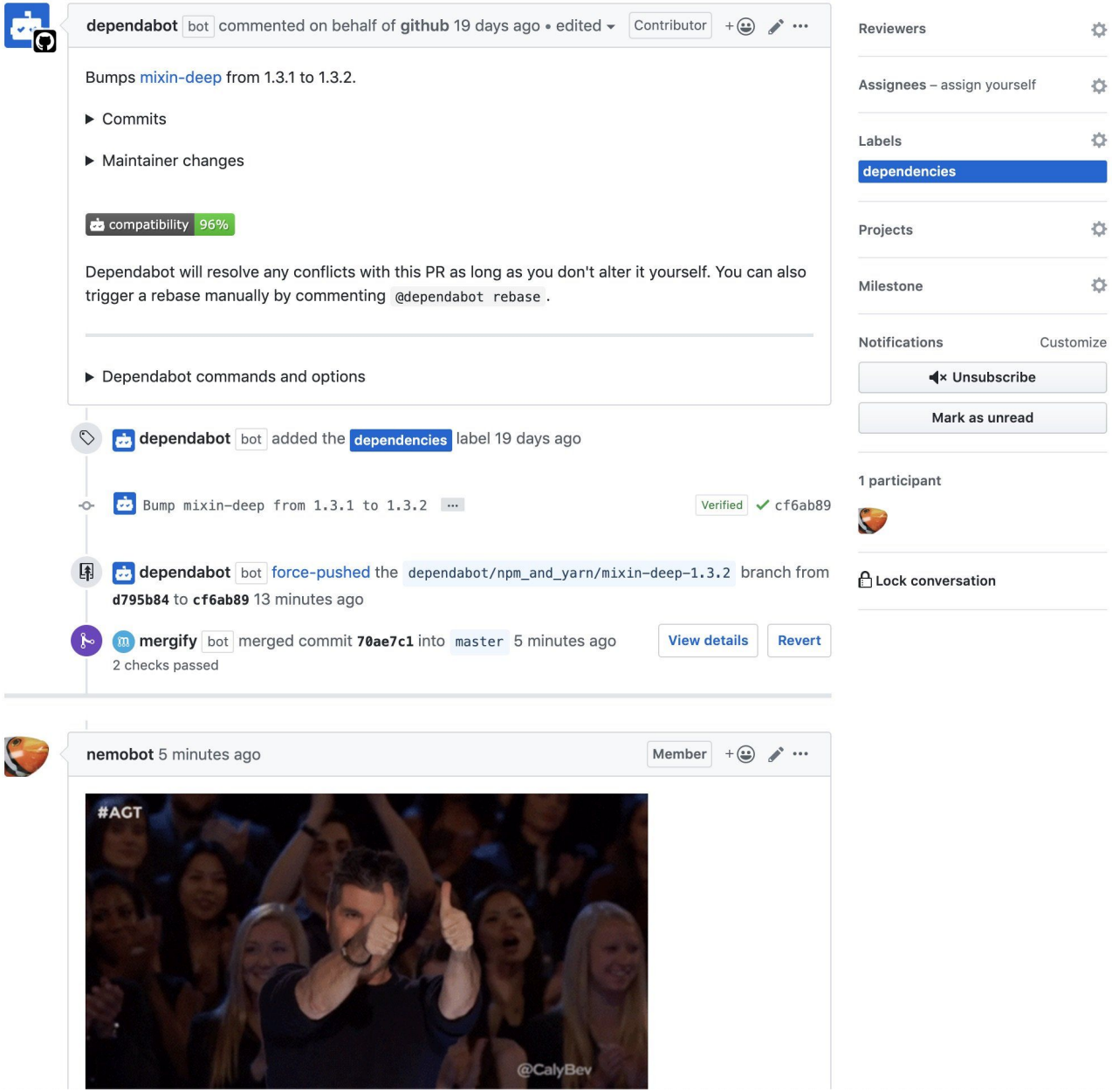
Software solutions to support this:

- CircleCI
- Travis CI
- Jenkins
- etc

Advantages of Github Actions

- ✔ Completely free for open source repos
- ✔ Many premade actions available on Github Marketplace
- ✔ Great integration with Github

Automation is everywhere



https://twitter.com/gabro27/status/1173547934132178944

Basic YAML syntax

YAML format used to store Github Action configurations

See also: https://en.wikipedia.org/wiki/YAML#Syntax

Lists, dictionaries

Basic YAML syntax

	Standard	Optional inline format

List	`- item1`
	`- item2`
	`- item3`
		`[ item1, item2, item3 ]`

Dictionary (key: value)	`name: boas` `job: phd`	`{name: boas, job: phd}`

Basic YAML syntax

	Standard	Optional inline format

List
(- item)
	`- item1`
	`- item2`
	`- item3`
		`[ item1, item2, item3 ]`

Dictionary (key: value)	`name: boas` `job: phd`	`{name: boas, job: phd}`

<u>Combinations possible</u>
e.g. dictionary containing dictionaries, one of which contains a list as value

```
- martin:
    name: Martin D'vloper
    job: Developer
    skills:
      - python
      - perl
```

Basic YAML syntax

Multiple lines (useful for e.g. multiple commands)

Preserve newlines ("|")

```
run: |
  which prokka
  prokka --cleandb
```

Fold newlines to spaces (">")
Force newline by including double newline

```
run: >
  which
  prokka

  prokka
  --cleandb
```

Same two commands

Basic outline of a Github Action

Name of the action: `name`

When should the action be activated: `on`

The action(s) itself: `jobs`, consisting of one or more `steps`

Follow along:
https://github.com/microbinfie-hackathon2020/CSIS/blob/main/.github/workflows/prokka.yml

`name` and `on` sections

Action name

```
name: prokka
```

Activate on pull request

```
on: [pull_request]
```

`name` and `on` sections

Action name

Activate on pull request

```
name: prokka

on: [pull_request]
```

Actions can be activated on certain events (push, pull request, release, etc.)

Activations can be further specified

Specifying the `on` section

Activate on pushes to `main` branch

```
on:
  push:
    branches: [ main ]
```

Activate on pull requests to `main` and `dev` branches

```
on:
  pull request:
    branches: [ main, dev ]
```

Activate when a release is created

```
on:
  release:
    types: [ created ]
```

Much more information:
https://docs.github.com/en/free-pro-team@latest/actions/reference/workflow-syntax-for-github-actions#on

Standard `on` section used in this hackathon

```
on:
  push:
    branches: [ main, dev ]
  pull request:
    branches: [ main, dev ]
```

Activate on push and pull request, to either `main` or `dev` branches

The action(s) itself: `jobs`

Multiple jobs can be specified within an action

In this case, three jobs will be run **in parallel**

(→ Dependencies on other jobs can be defined)

```
jobs:
  build:
    <some code>
  test:
    <some code>
  any-name:
    <some code>
```

Structure of a job

A minimal job consists of a `runs-on` section and one or multiple `steps`

`runs-on` describes the OS (“runner”) the job will use, e.g.:

- Ubuntu: ubuntu-latest (== ubuntu-18.04), ubuntu-20.04*, ubuntu-16.04
- macOS: macos-latest (== macos-10.15), macos-11.0
- Windows server 2019: windows-latest (== windows-2019)

*preview, Ubuntu 18.04 is currently used when ubuntu-latest is specified

Structure of a job

Job named `first-job` runs on Ubuntu 18.04

Two steps: `foo` and `bar`

```
jobs:
  first-job:
    runs-on: ubuntu-latest
    steps:
      - name: foo
        run: echo foo
      - name: bar
        run: echo bar
```

Structure of a job

Steps are (finally) where the real actions are specified

- here, two steps: `download` and `test`

(example, do not “git clone” IRL)

Check out the repo

Run basic tests

```
steps:
  - name: download
    run: |
     git clone https://github.com/tseemann/perl-biotool.git
     cd perl-biotool
  - name: test
    run: |
     which perl-biotool
     perl-biotool --version
     perl-biotool --help
     ! perl-biotool --doesnotexist
     perl-biotool --test
```

Putting it all together…

An action that:

- Has a name
  - Activates on every push and pull request to main and dev branches
  - Runs on Ubuntu 18.04
  - Downloads some code and runs basic tests

However: many more possibilities

Github Marketplace

Offers lots of premade apps and actions to automate things

Apps: travisCI, dependabot, codecov, etc.

>6000 “community” actions available

- Often to be used as modular `steps` within your workflow
- Can be used by including:

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

Uses version 2 of community action “checkout”

`actions/checkout@v2` example

Checks out Github repo under `$GITHUB_WORKSPACE`

For community actions, the `uses` command is needed

Parameters (e.g. `repository` or `path`) can be defined after `with`

```
- uses: actions/checkout@v2
  with:
    repository: tseemann/prokka
    path: prokka
```

`actions/checkout@v2` example

Checks out Github repo under `$GITHUB WORKSPACE`

For community actions, the `uses` command is needed

Parameters (e.g. `repository` or `path`) can be defined after `with`

```
- uses: actions/checkout@v2
  with:
    repository: tseemann/prokka
    path: prokka
```

Checks out prokka repo into `$GITHUB WORKSPACE/prokka`

Example prokka

```yaml
name: prokka

on:
  push:
    branches: [ main, dev ]
  pull_request:
    branches: [ main, dev ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
```

Example prokka

```
name: prokka

on:
  push:
    branches: [ main, dev ]
  pull_request:
    branches: [ main, dev ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
```

Install dependencies

```
      - name: apt
        run:  sudo apt-get install --no-install-recommends tree libdb-dev
libbio-perl-perl libxml-simple-perl
      # Checks-out prokka under $GITHUB_WORKSPACE, so your job can access
```

Example prokka

```
name: prokka

on:
  push:
    branches: [ main, dev ]
  pull_request:
    branches: [ main, dev ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: apt
        run:  sudo apt-get install --no-install-recommends tree libdb-dev
libbio-perl-perl libxml-simple-perl
      # Checks-out prokka under $GITHUB_WORKSPACE, so your job can access
      - uses: actions/checkout@v2
        with:
          repository: tseemann/prokka
          path: prokka
```

Install dependencies

Use ‘checkout’ community action to check out prokka repo

Example prokka

```
name: prokka

on:
  push:
    branches: [ main, dev ]
  pull_request:
    branches: [ main, dev ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: apt
        run:  sudo apt-get install --no-install-recommends tree libdb-dev
libbio-perl-perl libxml-simple-perl
      # Checks-out prokka under $GITHUB_WORKSPACE, so your job can access
      - uses: actions/checkout@v2
        with:
          repository: tseemann/prokka
          path: prokka
      - name: directory structure
        run: tree -d $GITHUB_WORKSPACE
```

Install dependencies

Use ‘checkout’ community action to check out prokka repo

Check directory structure

Example prokka

```
name: prokka

on:
  push:
    branches: [ main, dev ]
  pull_request:
    branches: [ main, dev ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: apt
        run:  sudo apt-get install --no-install-recommends tree libdb-dev
libbio-perl-perl libxml-simple-perl
      # Checks-out prokka under $GITHUB_WORKSPACE, so your job can access
      - uses: actions/checkout@v2
        with:
          repository: tseemann/prokka
          path: prokka
      - name: directory structure
        run: tree -d $GITHUB_WORKSPACE
      - name: ls bin dir
        run: ls -lh $GITHUB_WORKSPACE/prokka/bin
$GITHUB_WORKSPACE/prokka/binaries/linux
```

Install dependencies

Use ‘checkout’ community action to check out prokka repo

Check directory structure

Check bin directory contents

Example prokka

Fix PATH

```
- name: test prokka
        run: |
         export PATH=$GITHUB_WORKSPACE/prokka/bin:$PATH
         export PATH=$GITHUB_WORKSPACE/prokka/binaries/linux:$PATH
         cd $GITHUB_WORKSPACE/prokka
```

Example prokka

Fix PATH

Check if executables are in PATH

```
- name: test prokka
        run: |
         export PATH=$GITHUB_WORKSPACE/prokka/bin:$PATH
         export PATH=$GITHUB_WORKSPACE/prokka/binaries/linux:$PATH
         cd $GITHUB_WORKSPACE/prokka
         which prokka
         which makeblastdb
```

Example prokka

Fix PATH

Check if executables are in PATH

Check prokka flags

```
- name: test prokka
        run: |
         export PATH=$GITHUB_WORKSPACE/prokka/bin:$PATH
         export PATH=$GITHUB_WORKSPACE/prokka/binaries/linux:$PATH
         cd $GITHUB_WORKSPACE/prokka
         which prokka
         which makeblastdb
         prokka --version
         prokka --help
         ! prokka --doesnotexist
         prokka --depends
         prokka --setupdb
         prokka --listdb
```

Example prokka

Fix PATH

Check if executables are in PATH

Check prokka flags

Small functional test

```
- name: test prokka
        run: |
         export PATH=$GITHUB_WORKSPACE/prokka/bin:$PATH
         export PATH=$GITHUB_WORKSPACE/prokka/binaries/linux:$PATH
         cd $GITHUB_WORKSPACE/prokka
         which prokka
         which makeblastdb
         prokka --version
         prokka --help
         ! prokka --doesnotexist
         prokka --depends
         prokka --setupdb
         prokka --listdb
         prokka --cpus 2 --outdir asm --prefix asm test/plasmid.fna
         grep '>' asm/asm.fna
         prokka --cleandb
```

Example Quast

Job `build` also has a `strategy` section

- Defines a matrix of Python versions
- `setup-python` community actions then installs all versions of Python
- 4 parallel jobs with different Pythons

```
name: quast

on:
  push:
    branches: [ main, dev ]
  pull request:
    branches: [ main, dev ]

jobs:
  build:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        python-version: [3.5, 3.6, 3.7, 3.8]

    steps:
    - uses: actions/checkout@v2
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v2
      with:
        python-version: ${{ matrix.python-version }}
```

Example
Quast

Install tree

```
    - name: apt
      run: |
        sudo apt-get install tree
```

Example
Quast

Install tree

```
- name: apt
  run: |
    sudo apt-get install tree
```

Checkout Quast

```
- uses: actions/checkout@v2
  with:
    repository: ablab/quast
    path: quast
```

Example
Quast

Install tree

```
- name: apt
  run: |
    sudo apt-get install tree
```

Checkout Quast

```
- uses: actions/checkout@v2
  with:
    repository: ablab/quast
    path: quast
```

Install Quast

```
- name: Install quast
  run: |
    cd $GITHUB_WORKSPACE/quast
    ./setup.py install
```

Example
Quast

Install tree

```
- name: apt
  run: |
    sudo apt-get install tree
```

Checkout Quast

```
- uses: actions/checkout@v2
  with:
    repository: ablab/quast
    path: quast
```

Install Quast

```
- name: Install quast
  run: |
    cd $GITHUB_WORKSPACE/quast
    ./setup.py install
```

Download test data and run test

```
- name: test 1
  run: |
    wget quast.sf.net/test_data.tar.gz && tar xzf test_data.tar.gz
    tree test_data
    which quast.py
    quast.py --threads 1 --test
```

Recap

You've learned:

- Why Github Actions are useful
- What a minimal Github Action YAML looks like
- What two real life examples look like (Prokka & Quast)
- A glimpse of extra functionalities Github Actions offer

next up

Forming teams and project ideas

Discord: https://discord.gg/uUupE7V84m