Software Development in modern days: DevOps methodology

Continuous Integration/Continuous Delivery/Continuous Deployment: A bit of theory and practice

Mario David
(david@lip.pt) - LIP and INCD

The INCD project
22153-01/SAICT/2016
is co-funded by:

Outline

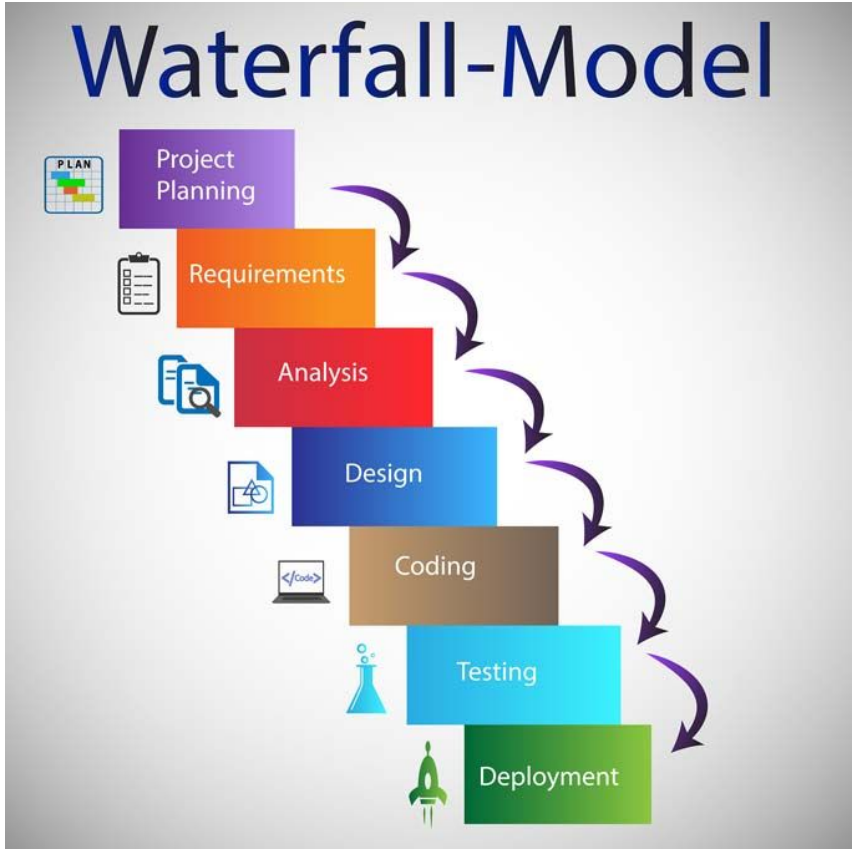
1. ***[Demo I]*** *IaC with Terraform and Ansible: deploy a kubernetes cluster*
   a. *https://github.com/LIP-Computing/tutorials/blob/main/iac/iac.md*
2. Software Development methodologies: past and present
3. A Walk on technologies and services: git, docker, terraform, ansible, virtual machines
4. **[Demo II]**: Examples of github and gitlab repositories
5. Interlude: Virtual Machines vs. Containers
6. Docker:
   a. ***[Demo III]***: Dockerfile
   b. ***[Demo IV]***: docker-compose
7. ***[Demo I]*** IaC (cont'd.): Terraform and Ansible
8. Continuous Integration/Continuous Delivery/Continuous Deployment: CI/CD
   a. Jenkins and Jenkins pipelines
   b. ***[Demo V]*** "udocker unit tests"

SW development: In the past

The project phases traditionally were:

Long periods, both between phases, as well as the overall process

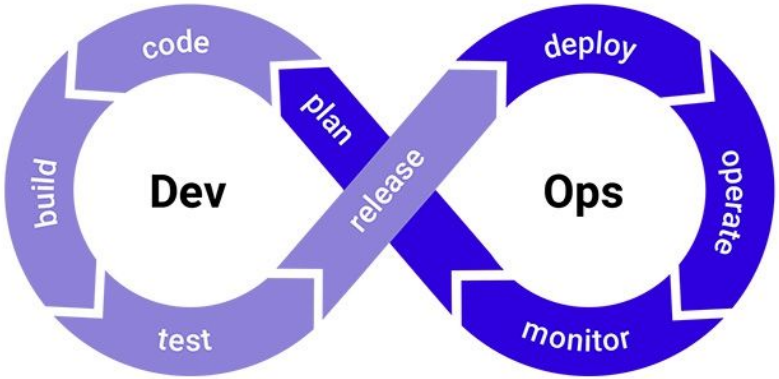
- Rigid and linear approach
- Each phase has distinct goals
- Once a phase is completed, there is no turning back
- Does not allow room to accommodate the inevitable changes



Source with other SW Dev. methodologies:
https://www.outsource2india.com/software/mobile-applications/project-methodologies.asp

DevOps methodology

- Also a set of practices.
- Enhances collaboration between the departments or groups.
- SW development life cycle:
  - Development.
  - Quality assurance.
  - Operations.



Short periods between phases ↠ **Fast**
Automation

Source: https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/

A Walk on technologies and services

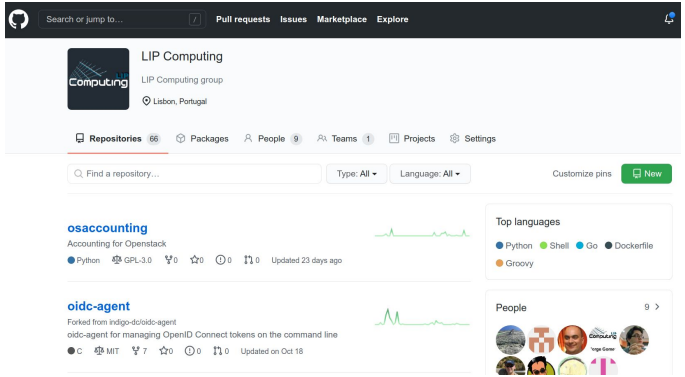
The “Code” - **you can “Code” almost anything**, not only SW, but also services, platforms, infrastructures, even documents (examples will be given later).

- Manage and track changes with a “**Version Control System**” (**VCS**):
  - Examples: **git** and **github** or **gitlab**
  - Hosted in repositories, public or private.
- To install, configure and run “services” and “platforms”, you can use:
  - **Docker** images
  - **Dockerfile** and/or **Docker-compose** files
  - **Ansible**
  - Manage sets of docker containers with: **docker-swarm** or kubernetes (**k8s** from now on).
- Instantiate Virtual Machines (VMs):
  - **Infrastructure as Code** (**IaC**): describe in a file how many VMs, which operating system, which networks, amount of storage and additional storage volumes, amount of RAM
  - Example: **Terraform**

Tools and services: Git - VCS

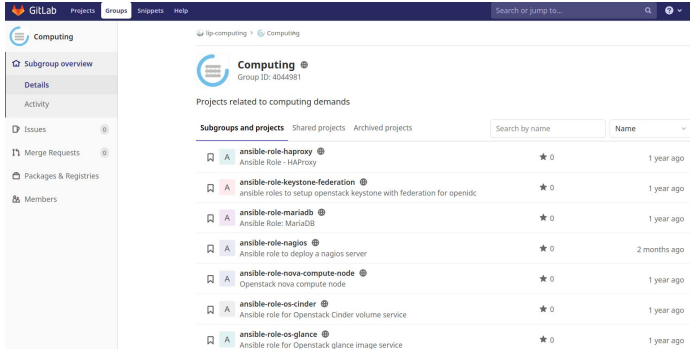
Git - Version Control System

Managing and tracking changes to your SW source code



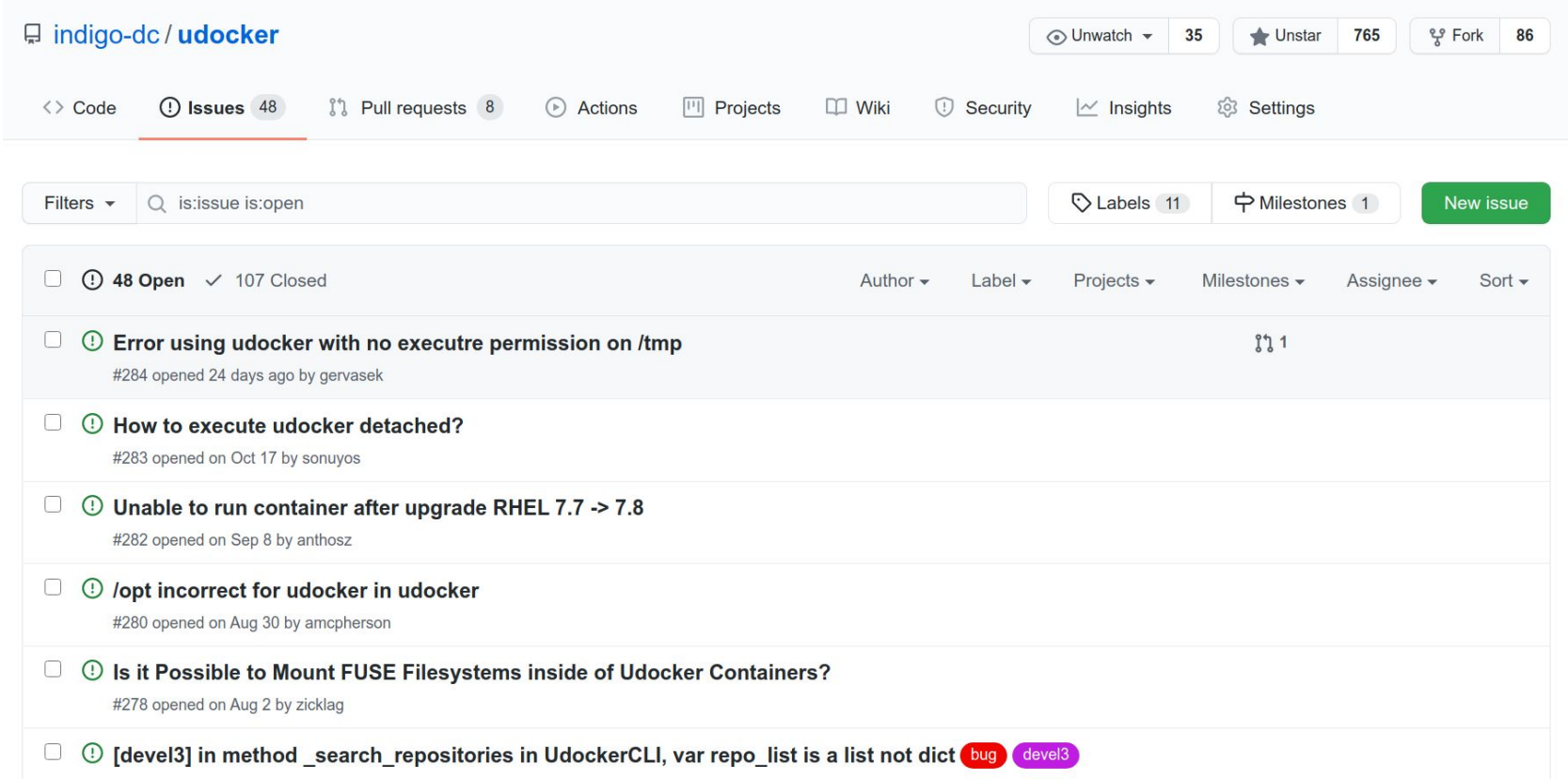
Github or Gitlab

Public repository services with Git as underlying management of files



Tools and services: Github - Issue tracker

Issue tracker: track and follow issues, SW problems and bugs, new features, even users questions



Tools and services: Github - Commit

After “git commit”

jorge-lip committed on Jan 8 Verified 2 parents d82d4a2 + 96

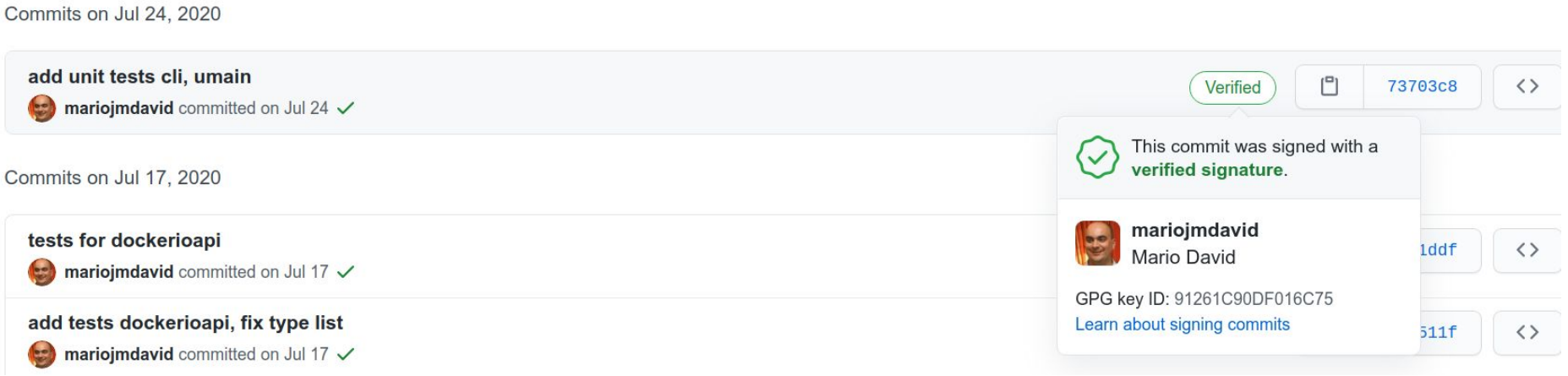
Showing 2 changed files with 33 additions and 12 deletions.

26 doc/installation_manual.md

```
@@ -47,48 +47,54 @@ From the *development* branch for the latest additions and fixes:
47 47     ./udocker install
48 48   ```
49 49
50    - To get a specific released version of udocker such as *v1.1.4*:
   50 + To get a specific released version of udocker such as v1.1.4:
51 51
52 52   ```
53 53     curl https://raw.githubusercontent.com/indigo-dc/udocker/v1.1.4/udocker.py > udocker
54 54     chmod u+rx ./udocker
55 55     ./udocker install
56 56   ```
57 57
58    - ### 2.2. INSTALL FROM REPOSITORIES
   58 + ### 2.2. INSTALL FROM UDOCKERTOOLS TARBALL
59 59
60    - This installation method uses the udocker tarball that contains statically compiled
   60 + This installation method uses the udockertools tarball that contains statically compiled
61 61   binaries and is built to be used across different hosts and OS distributions. Please
62 62   check the repositories for the latest release.
63 63
```

Tools and services: Github - Commits are signed

You can “sign” your commits - with your GPG key



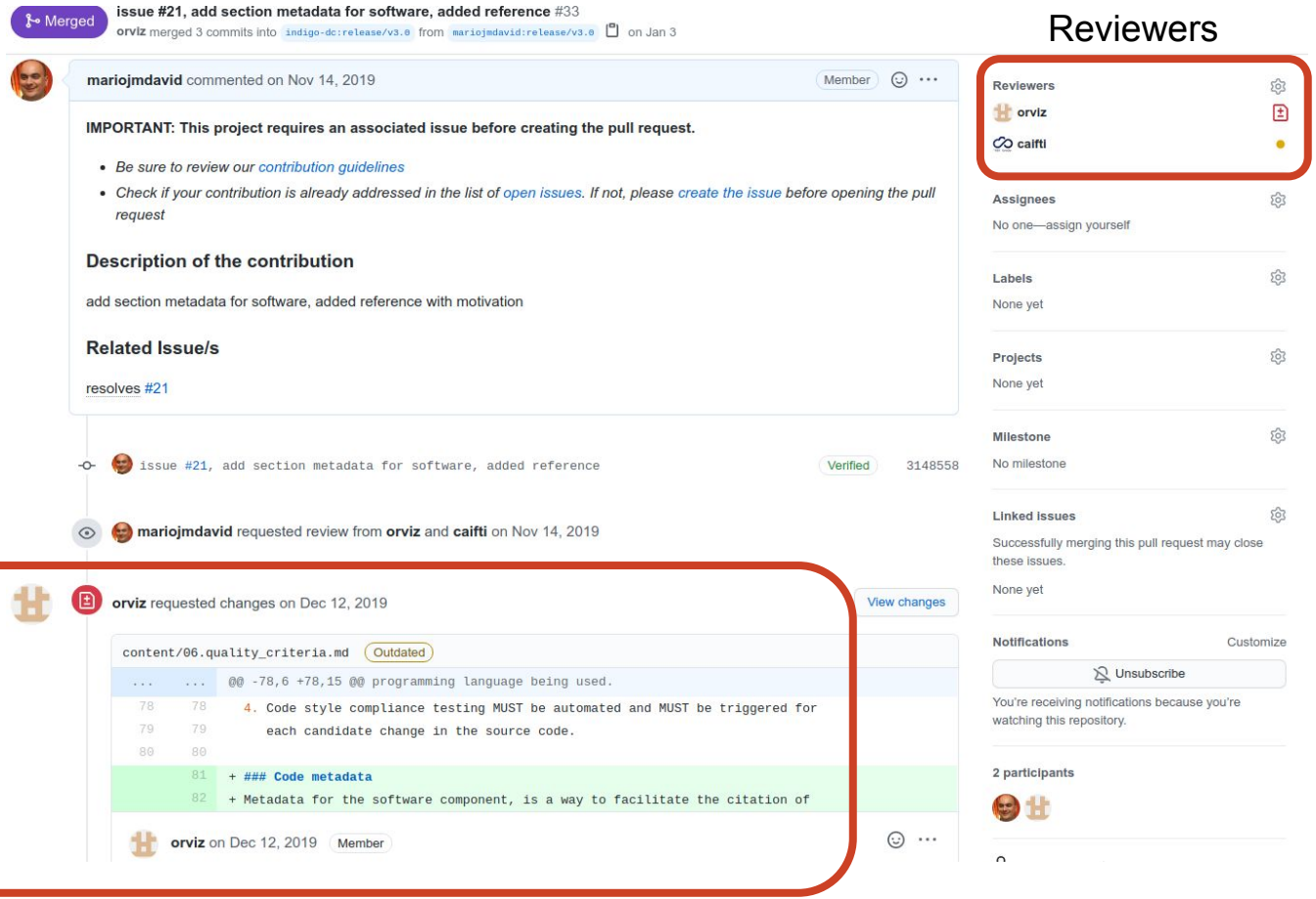
Tools and services: Github - Trigger CI tests

The “git commit” **triggers CI tests**

Tools and services: Github - Pull Request (PR)

Code Review: discussion and approval of 1 or more commits

Request changes



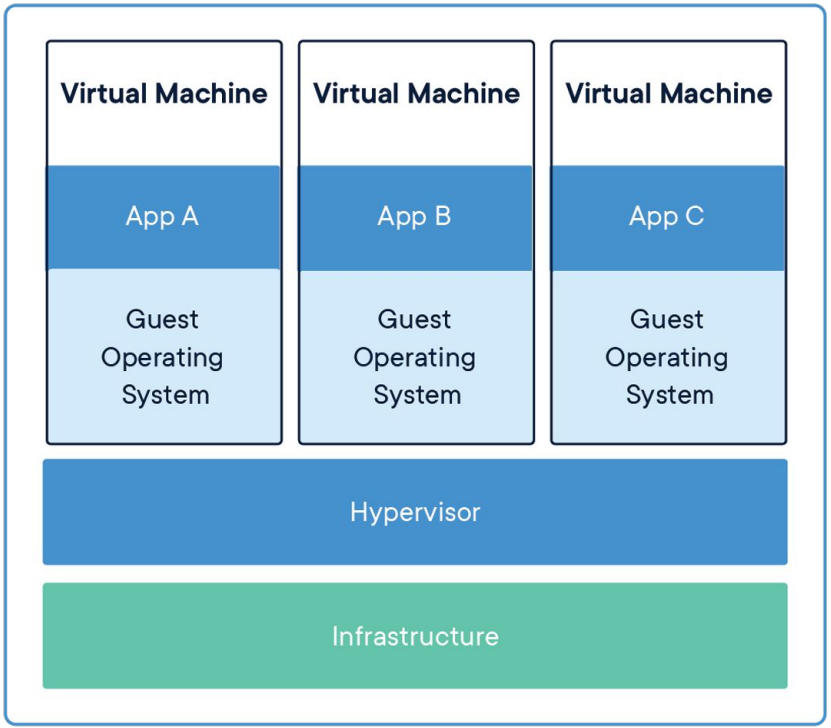
Reviewers

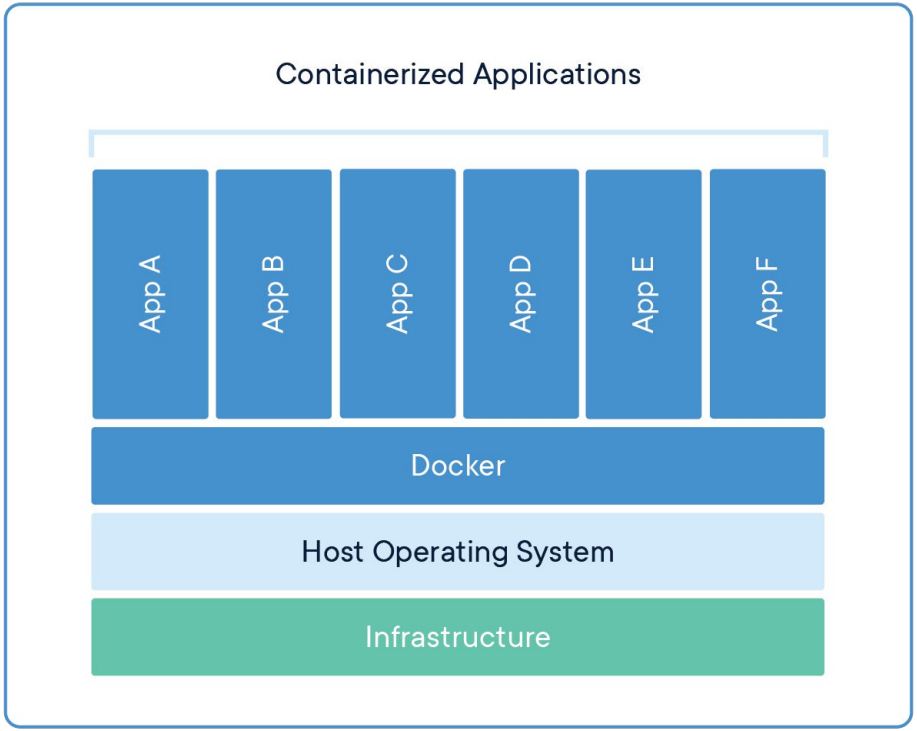
Github and gitlab: examples of repositories

Examples:

- SW source code (python): https://github.com/indigo-dc/udocker
- Document: https://github.com/indigo-dc/sqa-baseline
- Dockerfile and docker-compose: https://github.com/nginx-proxy/nginx-proxy
- Ansible: https://gitlab.com/lip-computing/computing/ansible-role-nagios

Interlude: Virtualization





Docker

Docker image: is the static package similar to a SW package, hosted in some repository, or local disk.

Docker container: is the actual execution of a given application or service.

Demo

- Dockerfile: https://github.com/nginx-proxy/nginx-proxy
  - git clone
  - docker build -t nginx .
- Docker-compose:
  - Copy paste into docker-compose.yml: https://github.com/nextcloud/docker
  - docker-compose up -d

Terraform + Ansible: the IaC

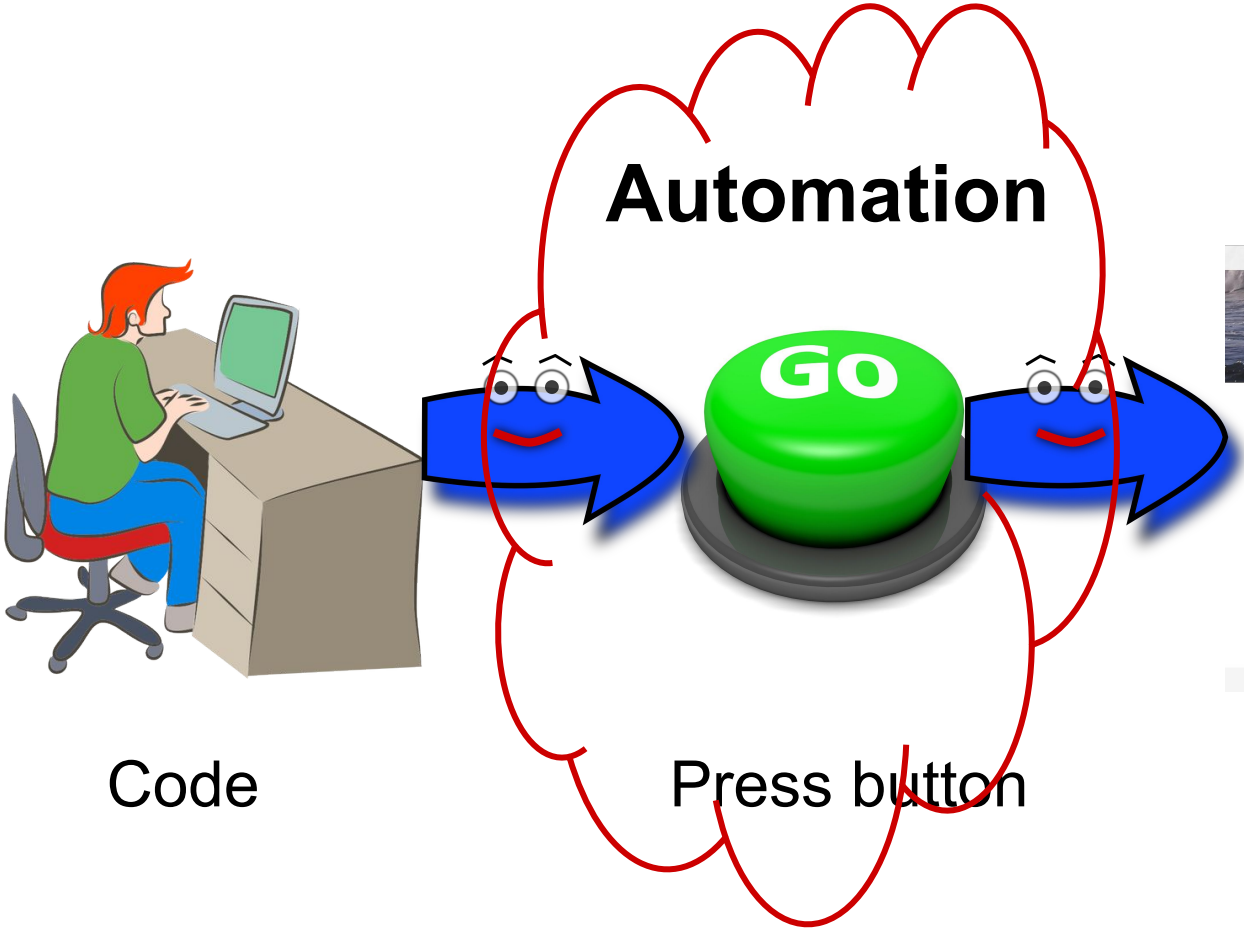
Coding your infrastructure:

- Terraform: Infrastructure as Code to provision and manage any cloud, infrastructure, or service.
- Ansible: Agentless IT automation.
- Example/Demo on how it works: A kubernetes cluster with 1 master and 2 nodes:
  - Terraform templates: Instantiates 3 VMs in our Openstack cloud infrastructure.
  - Ansible playbooks will install, configure and start the kubernetes cluster on those nodes.

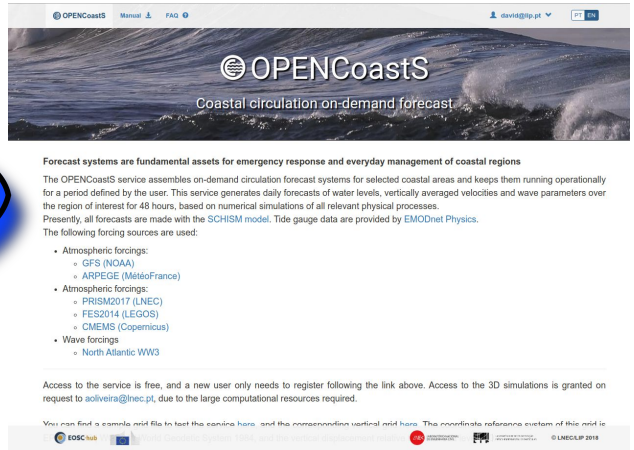
CI/CD - meaning and introduction

- **CI** ↠ Continuous Integration:
  - Coding
  - Building - includes **Automation** ↠ produces **artifacts**
  - Testing - includes Automation and **SW Quality Assurance** (of produced artefacts)
- **CD** ↠ Continuous Delivery ↠ Deployment:
  - **Delivery**: Artefacts are **released** - ready for usage (by users or system administrators) in production.
  - **Deployment**: refers mainly to services ↠ Installation, configuration, service (re)start

Some reflections and “What Is?”



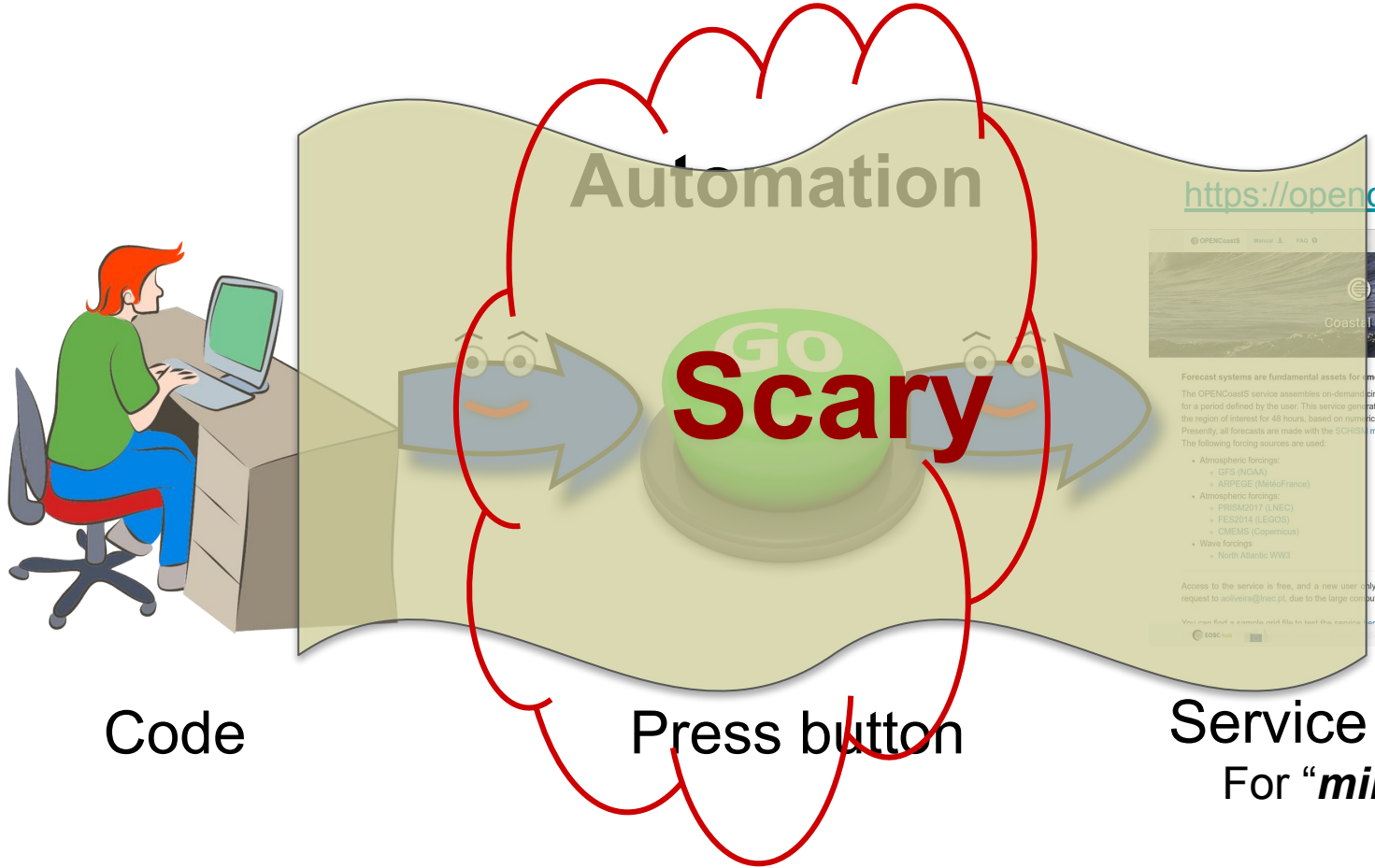
https://opencoasts.ncg.ingrid.pt/



Service in production

For “***millions***” of users

Some reflections and “What Is?”



Code

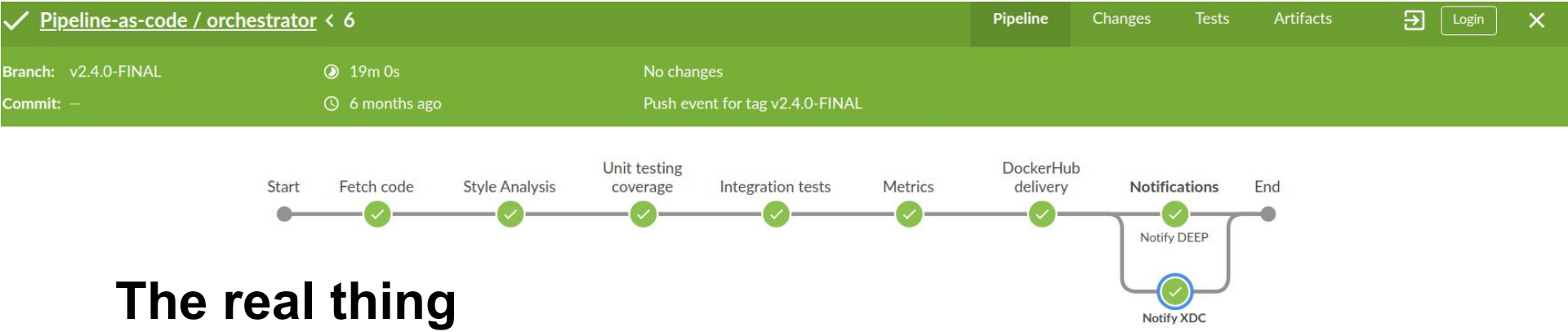
Press button

Service in production

For “***millions***” of users

On Automation

- Not all steps or phases are automatic: for example “**code review**” and approval/voting by partners/colleagues.
- Stop in the “**Delivery**” step.
- The Deployment is in a “***Staging***” or “***Pre-production***” or “***Preview***” infrastructure.
- Automation includes (for example) test for “**functionality**” and “**correctness**” of code execution and even “**security testing**”.

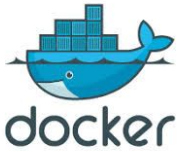


The real thing

What are ***Artifacts***

"They Live" in repositories - public or private

- "Deb": debian, ubuntu and derivatives
- "RPM": RedHat, Centos, Fedora and derivatives
- "EXE": Windows
- "PKG": Mac OSX
- Docker images
- Virtual Machine Images: many formats
- Pypi: python modules
- Maven: Java packages
- And: The "tarball"



The actual “Practical” and Technical Stuff

1. Code fetching - from Version Control System
2. Code style check
3. Single line of code count (SLOC) metrics gathering
4. Automated Deployment
5. Unit testing coverage
6. Functional and integration testing
7. Security linter/scanner
8. Vulnerability check on dependencies
9. **Code Review** - ***Not automated***
10. **Documentation** - ***Not automated***
11. Delivery
12. Notifications

CI

CD(elivery)

Software Quality Assurance - SQA

1. Code fetching - from Version Control System
2. Code style check
3. Single line of code count (SLOC) metrics gathering
4. Automated Deployment
5. Unit testing coverage
6. Functional and integration testing
7. Security linter/scanner
8. Vulnerability check on dependencies
9. **Code Review** - ***Not automated***
10. **Documentation** - ***Not automated***
11. Delivery
12. Notifications

SQA

http://hdl.handle.net/10261/160086

A set of Common Software Quality Assurance Baseline Criteria for Research Projects

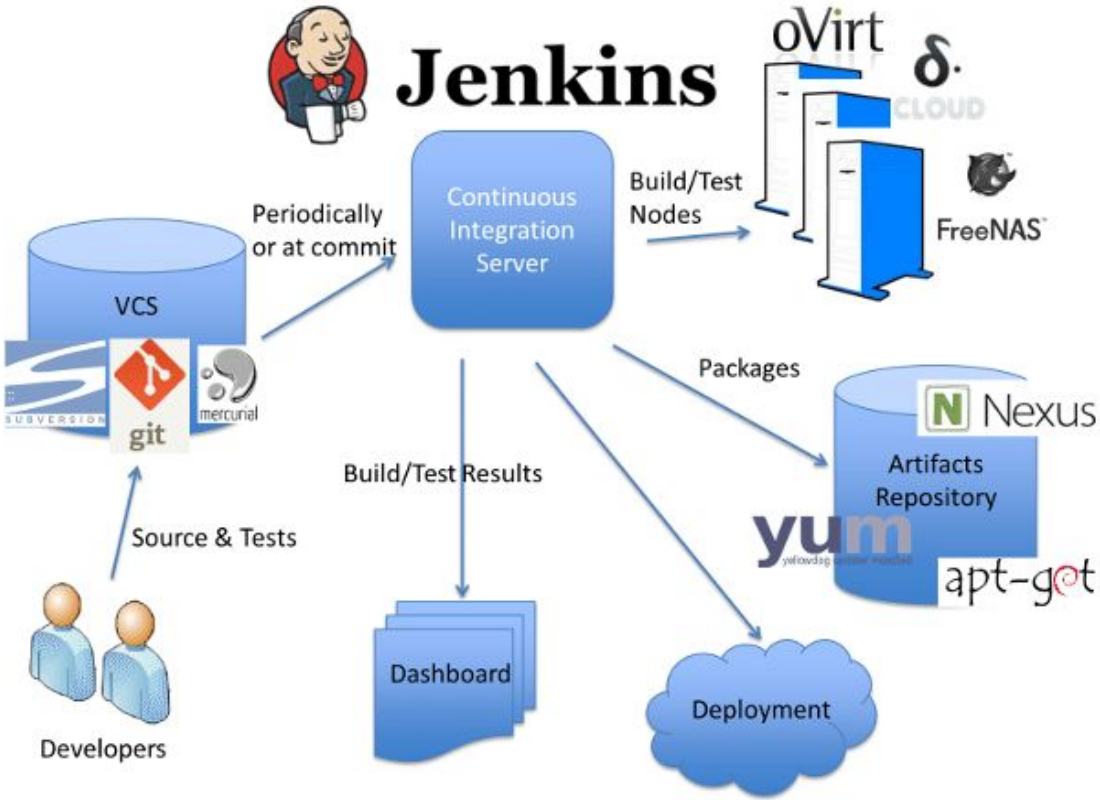
A DOI-citable version of this manuscript is available at http://hdl.handle.net/10261/160086.

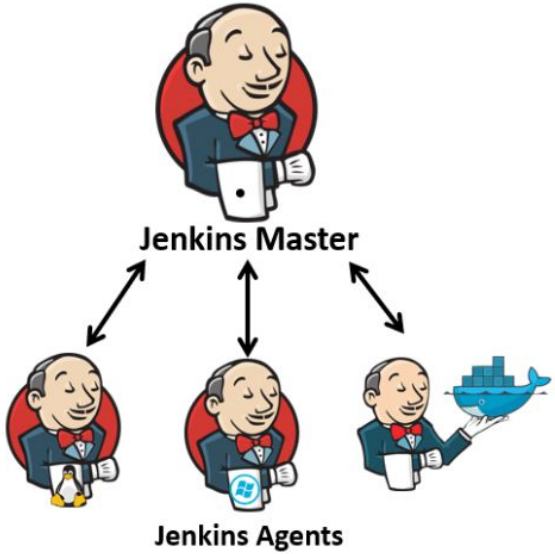
This manuscript (permalink) was automatically generated from indigo-dc/sqa-baseline@a9c34fa on April 29, 2020.

Authors

Tools and services: Jenkins CI/CD

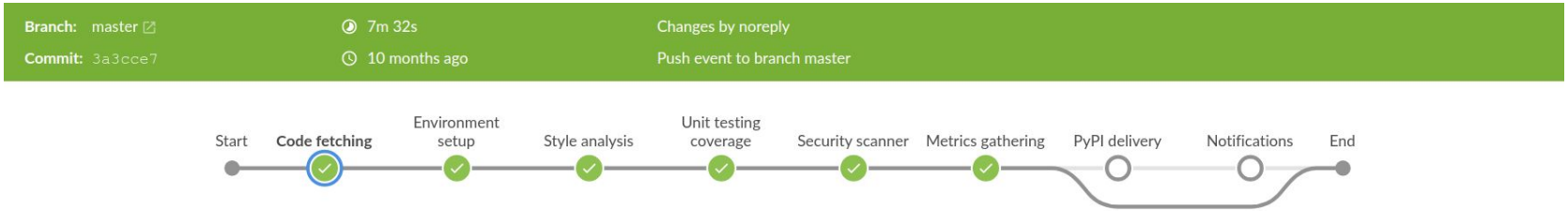
Jenkins - Automation service for CI/CD: https://jenkins.indigo-datacloud.eu/job/Pipeline-as-code/





Tools and services: Jenkins pipeline

Jenkins - pipeline

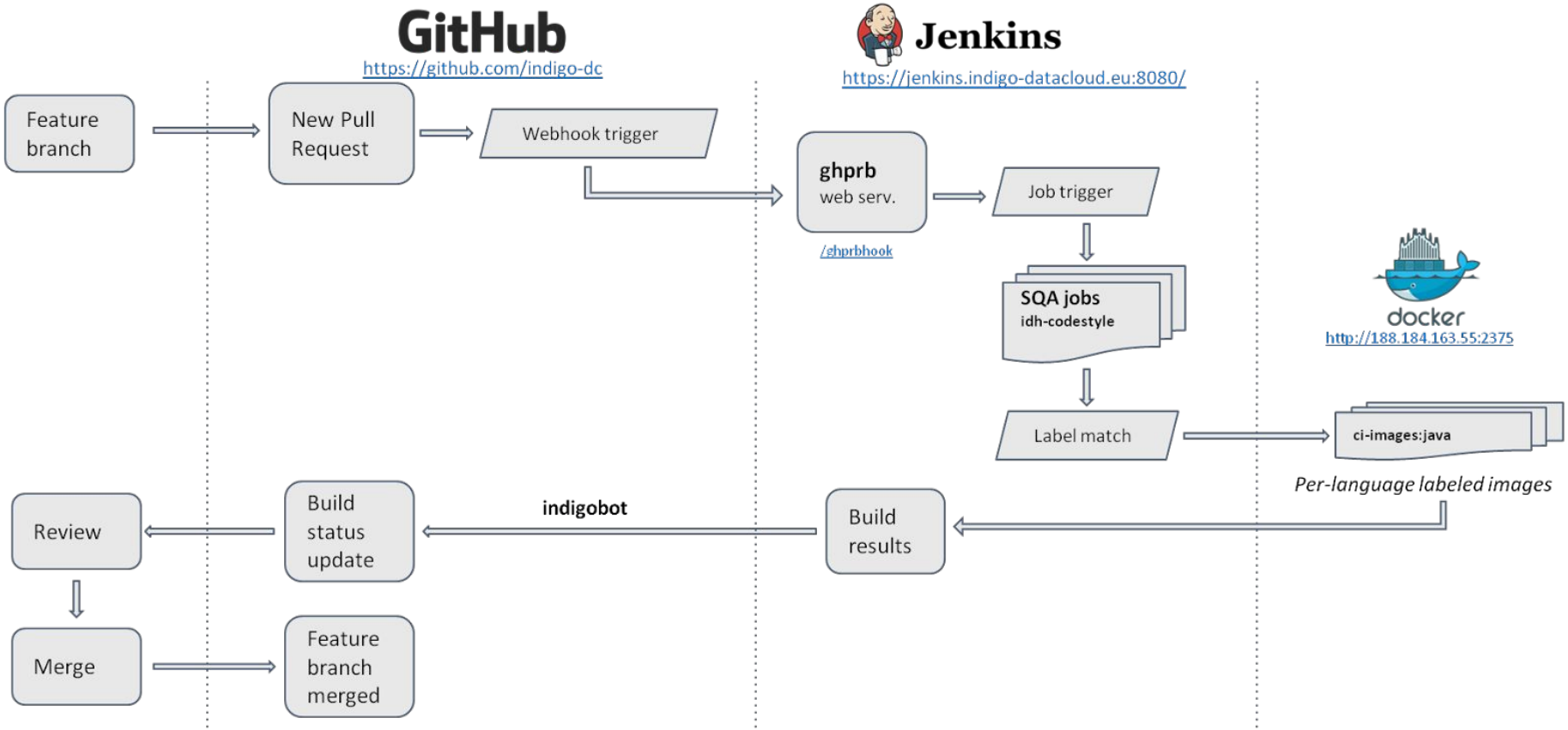


Code fetching - 4s

Check out from version control 22s

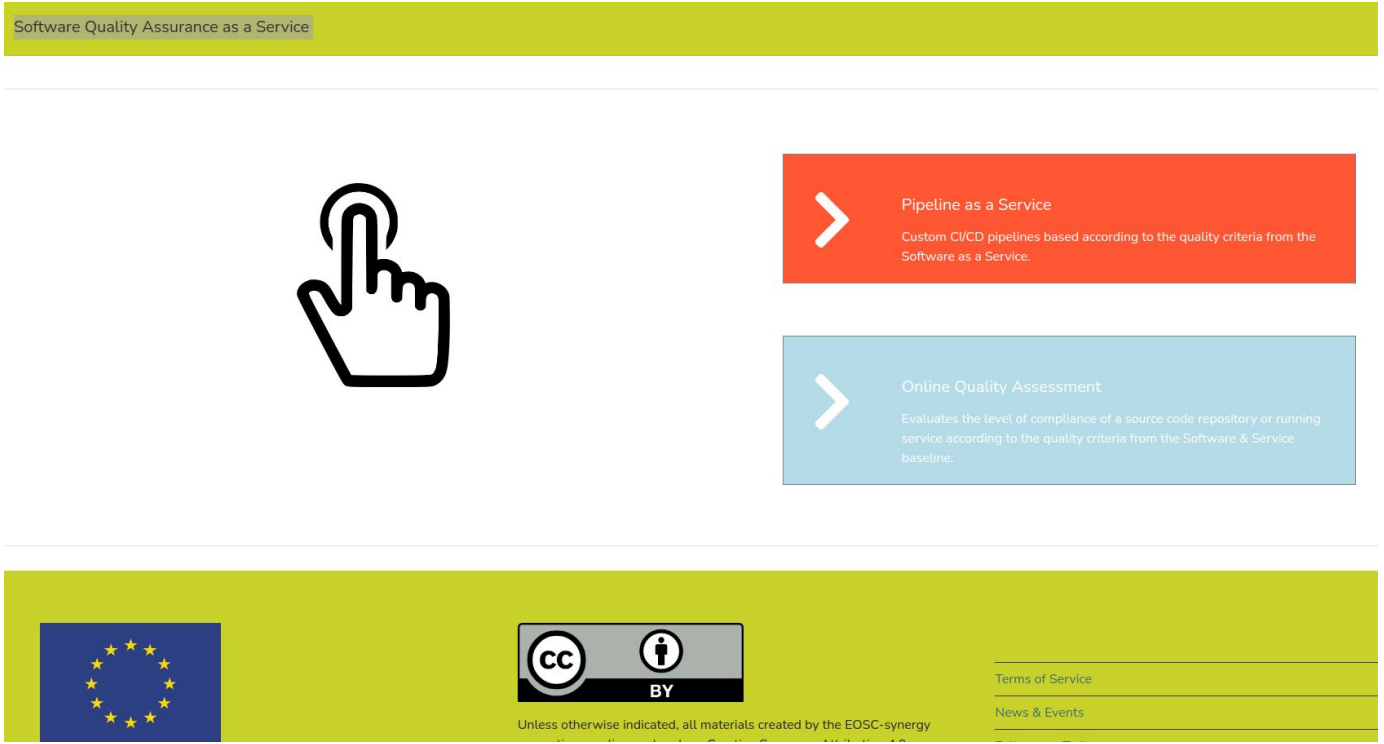
```
using credential indigobot-github
Cloning the remote Git repository
Cloning with configured refspecs honoured and without tags
Cloning repository https://github.com/indigo-dc/udocker.git
 > git init /tmp/workspace/Pipeline-as-code_udocker_master # timeout=10
Fetching upstream changes from https://github.com/indigo-dc/udocker.git
 > git --version # timeout=10
using GIT_ASKPASS to set credentials indigobot-github
 > git fetch --no-tags --progress -- https://github.com/indigo-dc/udocker.git +refs/heads/master:refs/remotes/origin/master
Fetching without tags
 > git config remote.origin.url https://github.com/indigo-dc/udocker.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/master:refs/remotes/origin/master # timeout=10
 > git config remote.origin.url https://github.com/indigo-dc/udocker.git # timeout=10
Fetching upstream changes from https://github.com/indigo-dc/udocker.git
using GIT_ASKPASS to set credentials indigobot-github
 > git fetch --no-tags --progress -- https://github.com/indigo-dc/udocker.git +refs/heads/master:refs/remotes/origin/master
Checking out Revision 3a3cce7124b0d420cecc938bfc7676324ed9f017 (master)
Commit message: "Update README.md"
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 3a3cce7124b0d420cecc938bfc7676324ed9f017
 > git rev-list --no-walk 975f3c660bf1b90ced77548470bfeda4070e52a7 # timeout=10
```

Pipeline workflow



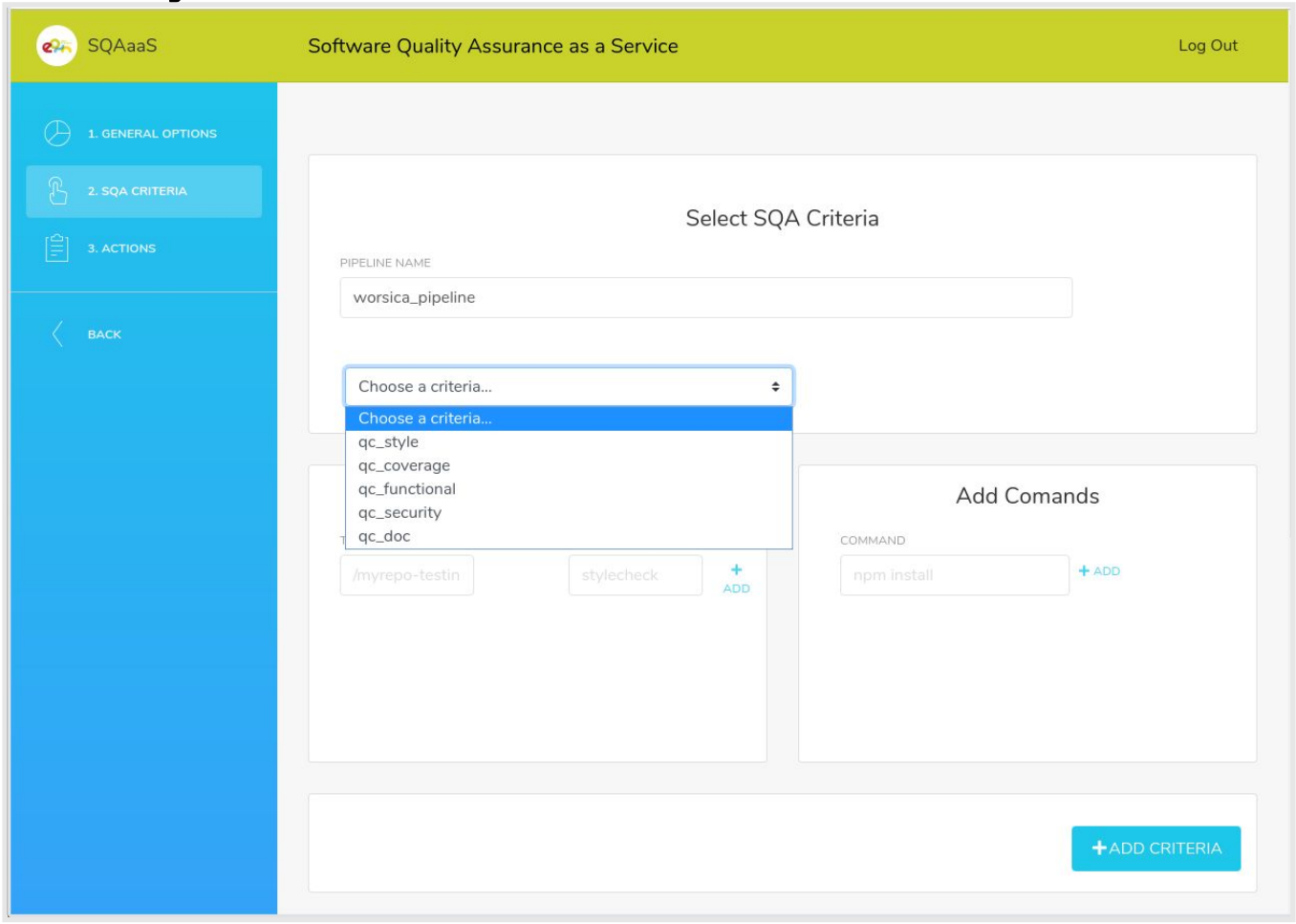
Software Quality Assurance as a Service

In the framework of a EU project: **EOSC-Synergy** - https://sqaaas.eosc-synergy.eu/#/auth/select-option



Software Quality Assurance as a Service

Compose a Jenkins pipeline for SQA



SQAaaS

Compose a Jenkins pipeline for SQA - The result: https://github.com/WORSICA/worsica-cicd/blob/development/.sqa/config.yml

```
...
sqa_criteria:
 qc_functional: #functional/unitary + coverage
   repos:
     worsica_processing:
       container: processing
       tox:
         testenv:
           - worsica-processing-unittest
         tox_file: '/usr/local/worsica_web_products/tox.ini'
     worsica-intermediate:
       container: intermediate
       tox:
         testenv:
           - worsica-intermediate-functional
         tox_file: '/usr/local/worsica_web_intermediate/tox.ini'
     worsica_frontend:
       container: frontend
       tox:
         testenv:
           - worsica-portal-functional
         tox_file: '/usr/local/worsica_web/tox.ini'
...
```

Further information and sources

- Software Development methodologies:
  - https://www.outsource2india.com/software/mobile-applications/project-methodologies.asp
  - https://xcelacore.com/how-software-development-has-changed-in-the-past-decade/
- Software Quality Assurance: https://github.com/indigo-dc/sqa-baseline
- Service Quality Assurance: https://github.com/EOSC-synergy/service-qa-baseline