CI/CD with Jenkins

Alessandro Bocci

name.surname@unipi.it

Advanced Software Engineering (Lab)

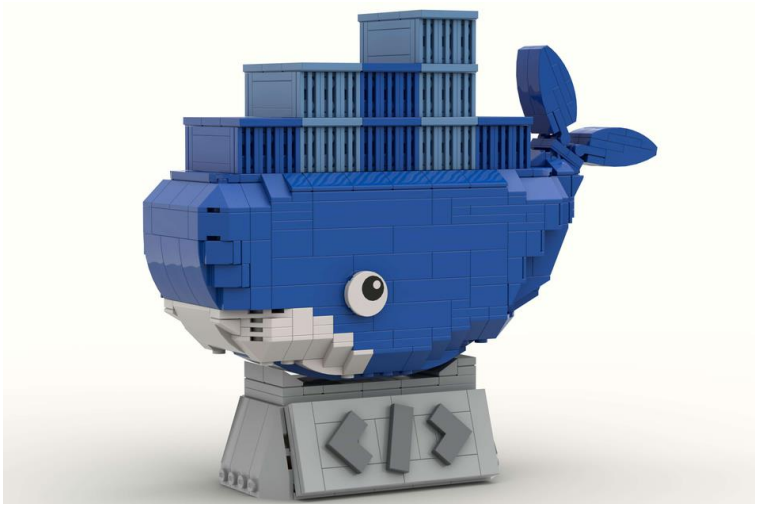
07/12/2023

Software Prerequisites

Docker images:

- `docker:dind` -> allows to run **d**ocker **in**side **d**ocker
- `jenkins/jenkins:2.361.4-jdk11` -> old version of jenkins

  (easy to use without doing an account, but it has vulnerabilities)



What will you do?

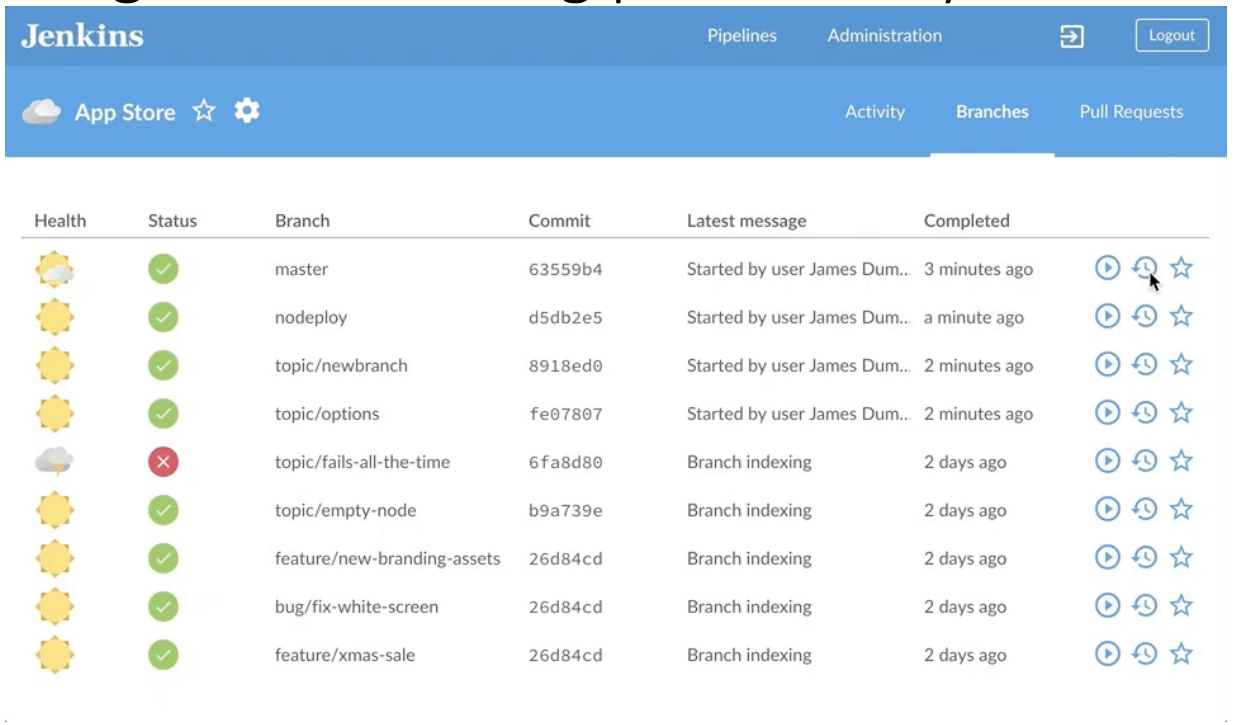
- Orchestrate building and testing a python application with PyInstaller.
- Get familiar with CI/CD with Jenkins.

Jenkins

- Jenkins is an open-source automation server.
- It helps automate the parts of software development related to building, testing, and deploying.
- Its aim is facilitating CI/CD.

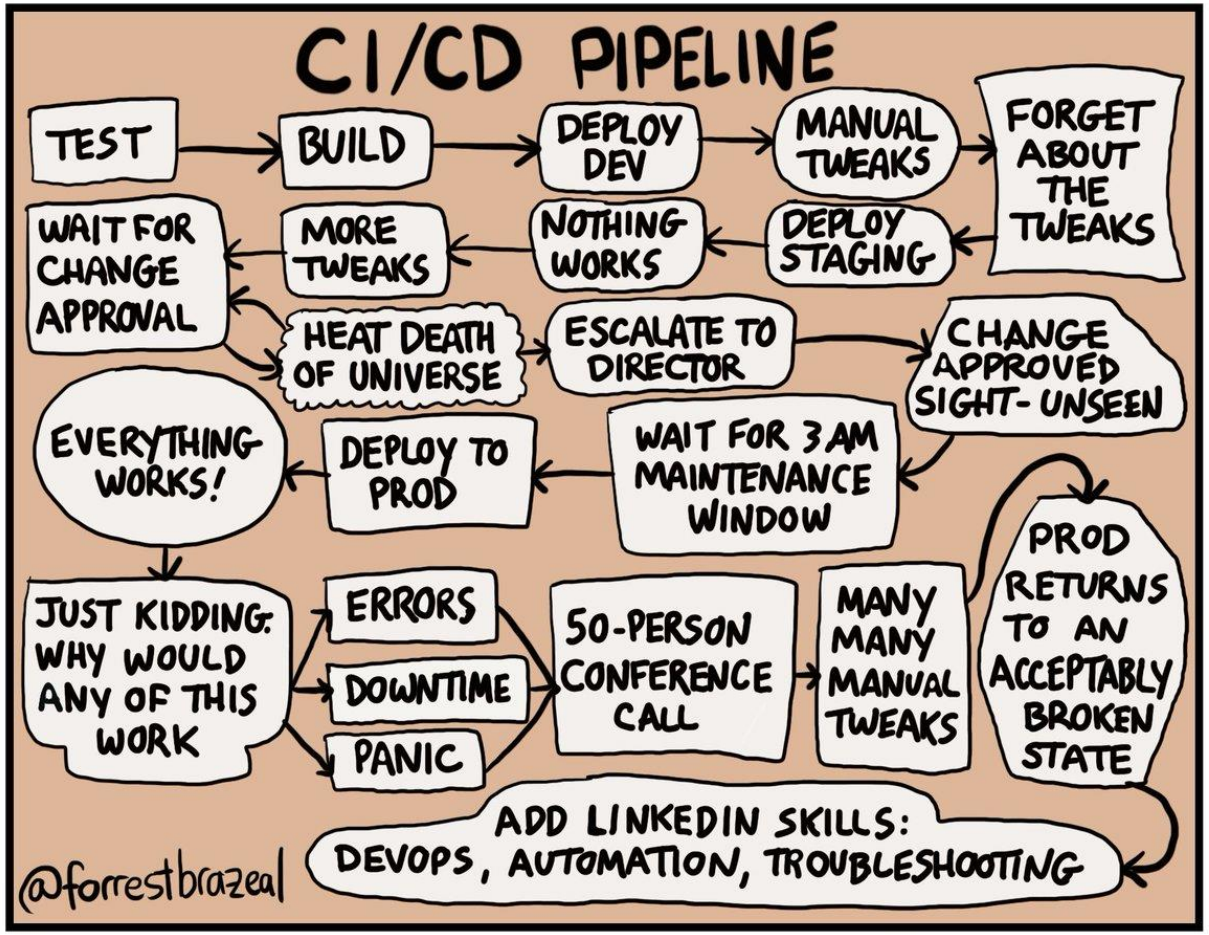
BlueOcean

- Jenkins' UI .
- It shows where in the pipeline is needed attention, facilitating exception handling and increasing productivity.



Pipelines

- Stages of CI/CD executed in sequence.
- Be careful to the complexity…



Today's Lab

Download the `Dockerfile` from the Moodle (or copy it from one of the slides).

LAB TODO

1. Set up `dind` and `Jenkins`.
2. Create a `Jenkins` pipeline.
3. Add stages and run the pipeline.
4. Obtain the resulting artifact.
5. Run it!

Create a bridge network

- It will be shared between Jenkins containers to communicate:

```
docker network create jenkins
```

Download and run docker:dind

- It will allow Jenkins to run docker containers (using **jenkins** network)

```
docker run \
  --name jenkins-docker \
  --rm \
  --detach \
  --privileged \
  --network jenkins \
  --network-alias docker \
  --env DOCKER_TLS_CERTDIR=/certs \
  --volume jenkins-docker-certs:/certs/client \
  --volume jenkins-data:/var/jenkins_home \
  docker:dind \
  --storage-driver overlay2
```

Create `Dockerfile`

- Customise the official Jenkins Docker image:

```
FROM jenkins/jenkins:2.361.4-jdk11
USER root
RUN apt-get update && apt-get install -y lsb-release
RUN curl -fsSLo /usr/share/keyrings/docker-archive-keyring.asc \
  https://download.docker.com/linux/debian/gpg
RUN echo "deb [arch=$(dpkg --print-architecture) \
  signed-by=/usr/share/keyrings/docker-archive-keyring.asc] \
  https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list
RUN apt-get update && apt-get install -y docker-ce-cli
USER jenkins
RUN jenkins-plugin-cli --plugins "blueocean:1.25.8 docker-workflow:521.v1a_a_dd2073b_2e"
```

Build it

- Build the image and tag it:

```
docker build -t myjenkins-blueocean:2.361.4-1 .
```

Run it!

```
docker run \
  --name jenkins-blueocean \
  --detach \
  --network jenkins \
  --env DOCKER_HOST=tcp://docker:2376 \
  --env DOCKER_CERT_PATH=/certs/client \
  --env DOCKER_TLS_VERIFY=1 \
  --publish 8080:8080 \
  --publish 50000:50000 \
  --volume jenkins-data:/var/jenkins_home \
  --volume jenkins-docker-certs:/certs/client:ro \
  --volume "$HOME":/home \
  --restart=on-failure \
  --env JAVA_OPTS="-Dhudson.plugins.git.GitSCM.ALLOW_LOCAL_CHECKOUT=true" \
  myjenkins-blueocean:2.361.4-1
```

For Windows (**without WSL**) you must use this command instead:

```
docker run --name jenkins-blueocean --detach ^
  --network jenkins --env DOCKER_HOST=tcp://docker:2376 ^
  --env DOCKER_CERT_PATH=/certs/client --env DOCKER_TLS_VERIFY=1 ^
  --volume jenkins-data:/var/jenkins_home ^
  --volume jenkins-docker-certs:/certs/client:ro ^
  --volume "%HOMEDRIVE%%HOMEPATH%":/home ^
  --restart=on-failure ^
  --env JAVA_OPTS="-Dhudson.plugins.git.GitSCM.ALLOW_LOCAL_CHECKOUT=true" ^
  --publish 8080:8080 --publish 50000:50000 myjenkins-blueocean:2.361.4-1
```

Access Jenkins

- Connect to: `http://localhost:8080`
- Retrieve password through issuing the command:

```
docker logs jenkins-blueocean
```

- Click "**Install suggested plugins**."
- Create first Admin User and continue until "**Start using Jenkins**"
- Ignore the notifications of vulnerabilities ☺

Create and setup the github project

- Clone the test application;

```
git clone https://github.com/jenkins-docs/simple-python-pyinstaller-app.git
```

- Note the folder where you cloned:
  - For macOS - `/Users/<your-username>/Documents/GitHub/`
  - For Linux/WSL - `/home/ase/simple-python-pyinstaller-app`
  - For Windows - `C:\Users\<your-username>\Documents\ase\simple-python-pyinstaller-app`
- Your home directory, e.g. `username/...` will be mapped to `home/...`

Create pipeline project in Jenkins

- **New Element -> Pipeline** (insert a name for your pipeline)
- **Choose the definition "Pipeline script from SCM"** (in **Pipeline** section)
- From the **SCM** field, choose **Git**.
- In the Repository URL field, specify the directory path of your locally cloned repository above, which is from your user account/home directory on your host machine, mapped to the /home directory of the Jenkins container - i.e.
  - For macOS - **/home/Documents/ase/simple-python-pyinstaller-app**
  - For Linux/WSL - **/home/ase/simple-python-pyinstaller-app**
  - For Windows - **/home/Documents/ase/simple-python-pyinstaller-app**
- Click **Save**

Create Jenkinsfile (in the cloned folder)

```
pipeline {
    agent none
    stages {
        stage('Build') {
            agent {
                docker {
                    image 'python:2-alpine'
                }
            }
            steps {
                sh 'python -m py_compile sources/add2vals.py sources/calc.py'
                stash(name: 'compiled-results', includes: 'sources/*.py*')
            }
        }
    }
}
```

New stage

Stage's agent

Stage execution steps

- Then commit:

```
git add .
git commit –m "Add initial Jenkinsfile"
```

Run the job in Jenkins

- Open Blue Ocean with the button **Open Blue Ocean**
- In the **This job has not been run** message box, click **Run**
  - Or “Build Now” inside the pipeline dashboard (outside Blue Ocean)

It performs the steps of the Build stage

Add Tests stage

```
        stage('Test') {
            agent {
                docker {
                    image 'qnib/pytest'
                }
            }
            steps {
                sh 'py.test --junit-xml test-reports/results.xml sources/test_calc.py'
            }
            post {
                always {
                    junit 'test-reports/results.xml'
                }
            }
        }
```

- Then commit:

```
git add .
git commit –m "Add ‘Test’ stage"
```

Re-Run the job in Jenkins

- Open Blue Ocean with the button **Open Blue Ocean**
- In the **This job has not been run** message box, click **Run**
  - Or “Build Now” inside the pipeline dashboard (outside Blue Ocean)

Add final deliver stage

```
        stage('Deliver') {
            agent any
            environment {
                VOLUME = '$(pwd)/sources:/src'
                IMAGE = 'cdrx/pyinstaller-linux:python2'
            }
            steps {
                dir(path: env.BUILD_ID) {
                    unstash(name: 'compiled-results')
                    sh "docker run --rm -v ${VOLUME} ${IMAGE} 'pyinstaller -F add2vals.py'"
                }
            }
            post {
                success {
                    archiveArtifacts "${env.BUILD_ID}/sources/dist/add2vals"
                    sh "docker run --rm -v ${VOLUME} ${IMAGE} 'rm -rf build dist'"
                }
            }
        }
```

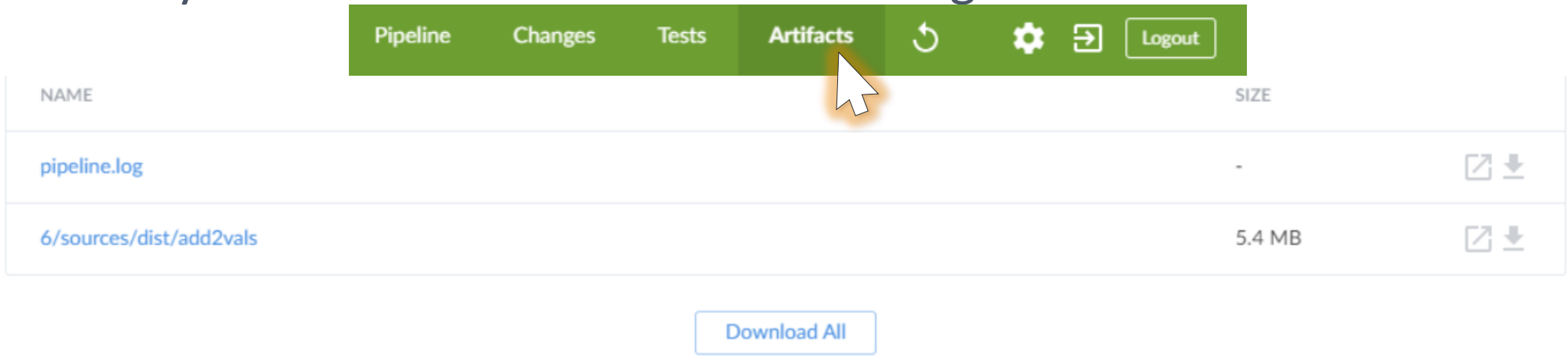
- Then commit:

```
git add .
git commit –m "Add ‘Deliver’ stage"
```

Re-Run the job in Jenkins

- Open Blue Ocean with the button **Open Blue Ocean**
- In the **This job has not been run** message box, click **Run**
  - Or “Build Now” inside the pipeline dashboard (outside Blue Ocean)

- Now you can download the executable generated inside Artifactis



Run the artifact

- Make the artifact executable

```
chmod +x add2vals
```

- Run it

```
./add2vals 2 3
```

Bonus stage


TIME
30"00
INSERT COIN(S)
JEFFRY
WIN
WIN
DURAL
SET 1
READY
BONUS STAGE
©SEGA

Bonus stage

- Write a Jenkins pipeline to build `microase232` (last lab version).

1. One stage for each build (gateway, math, string).
2. One stage using last lab unit test for math and string.

(Bonus of bonus)

3. One stage to deliver the code.

Lab take away

- ❑ Familiarise with CI/CD.
- ❑ Build a Jenkins pipeline.
- ❑ Run the result of the pipeline.