
UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

DIPARTIMENTO DI INFORMATICA

CORSO DI LAUREA

IN

INFORMATICA E TECNOLOGIE PER LA PRODUZIONE DEL SOFTWARE

---

Tesi di laurea

in

Integrazione e test di sistemi software

# Analisi e sperimentazione dell'utilizzo dei containers Docker durante il ciclo di sviluppo del software

Relatori:

**Chiar.mo Prof. Michele SCALERA**

Correlatore:

**Dott. Giovanni BRUNO**

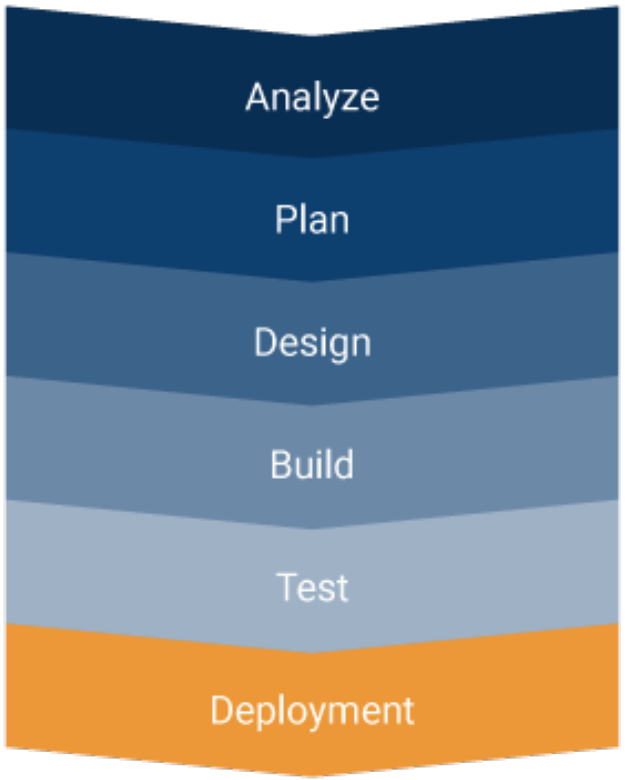
Laureando:

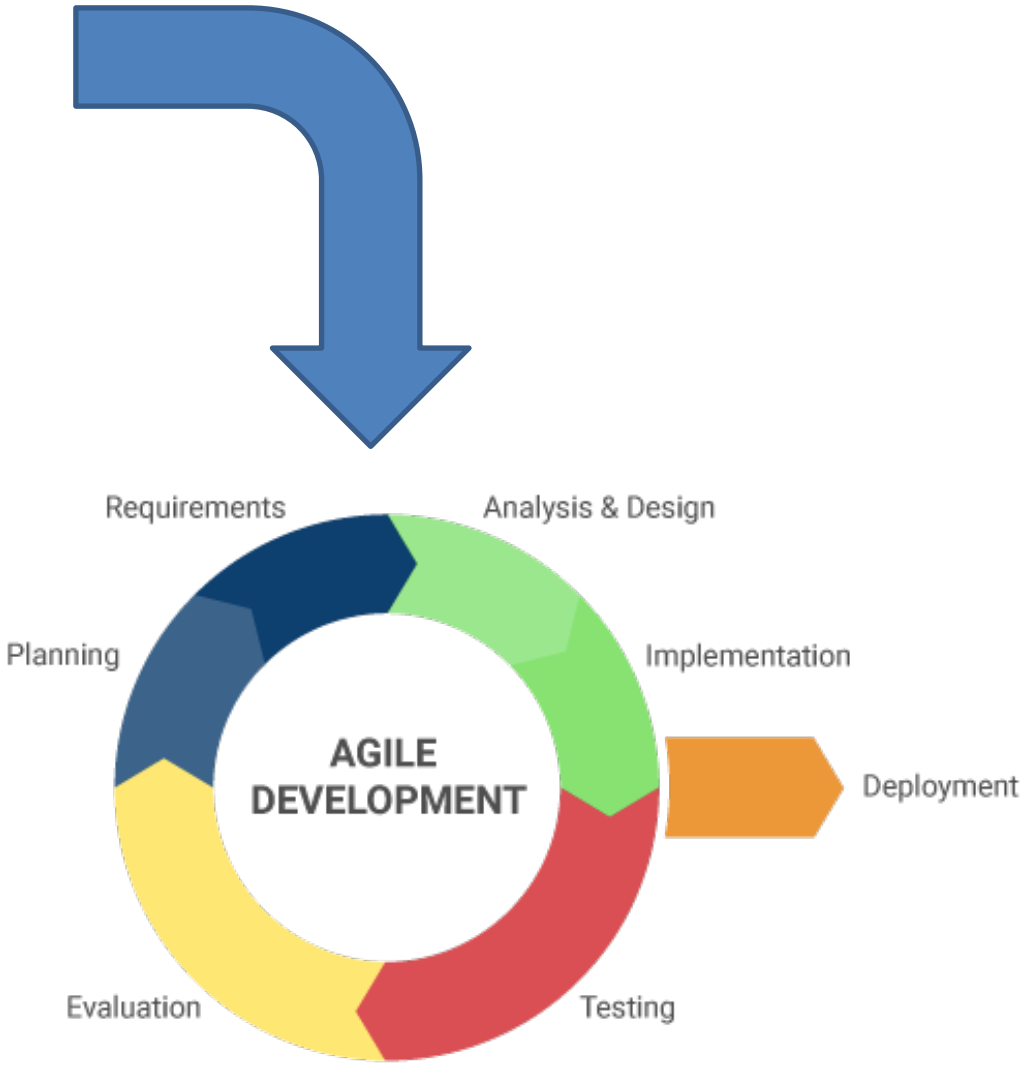
**Francesco CASSANO**

# OBIETTIVO

- Verificare come l'uso dei containers Docker possa supportare le varie fasi di un progetto di sviluppo software Agile.

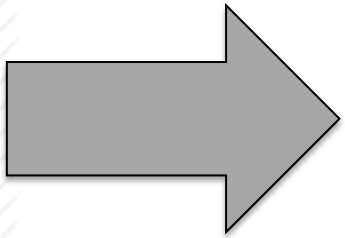
# DA WATERFALL AD AGILE

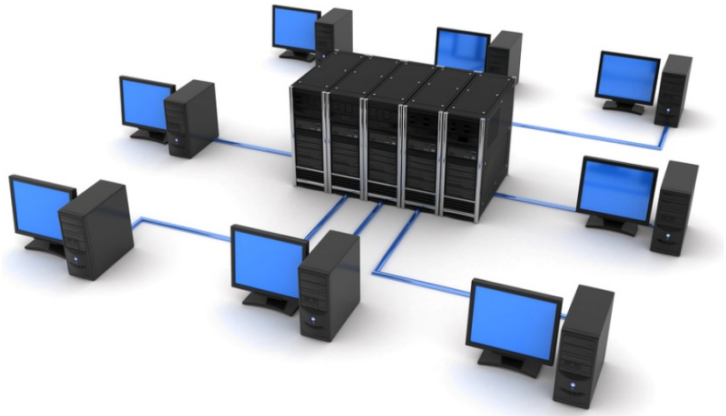


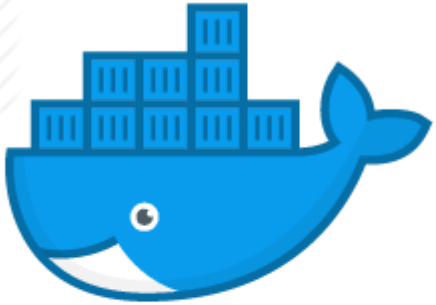



# DOCKER NELLO SVILUPPO

DEPLOY

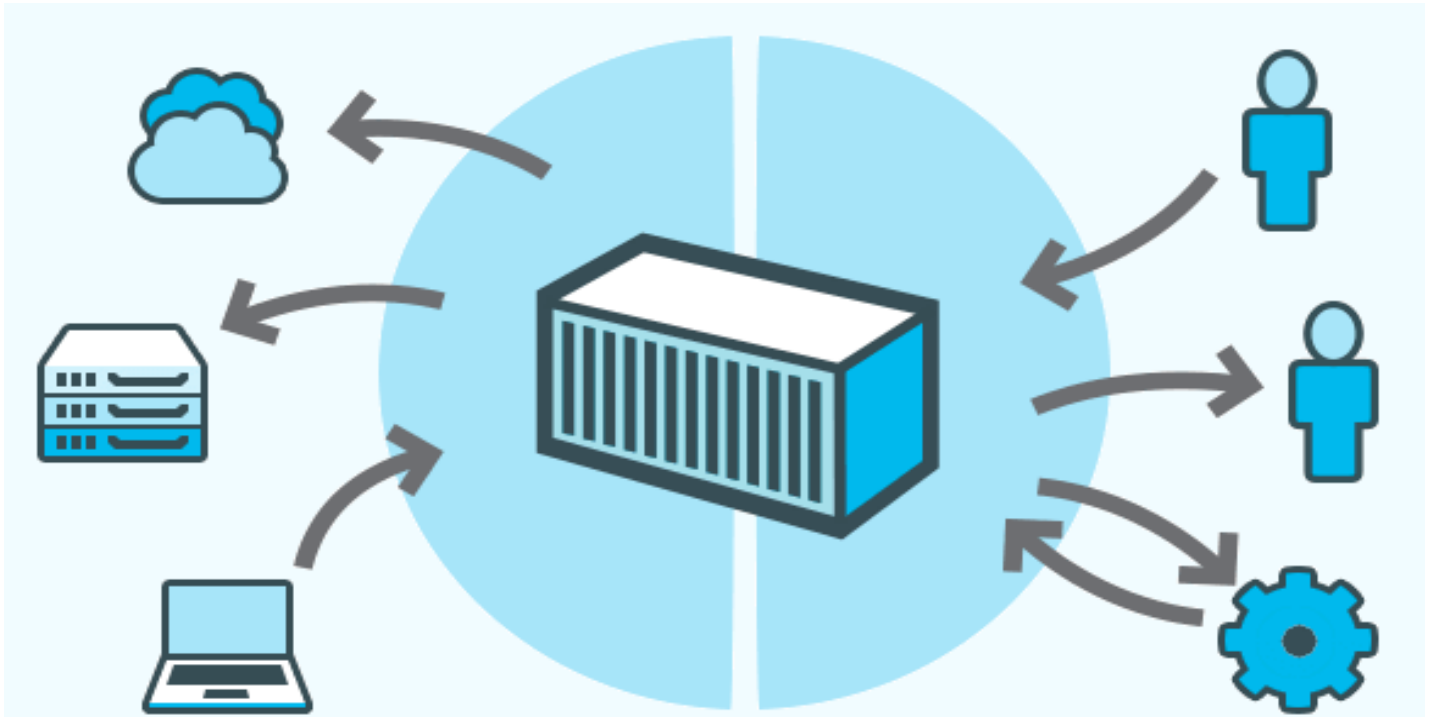






docker

# DOCKER



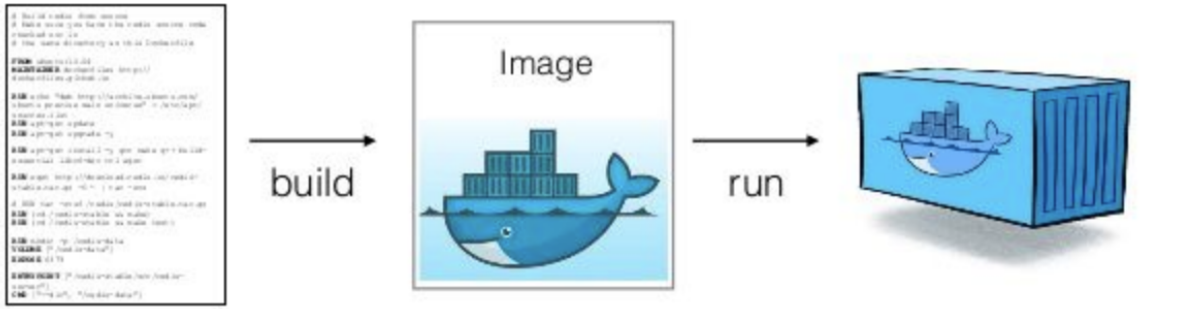
## Docker Engine

offre un'interfaccia veloce e comoda per l'esecuzione dei container

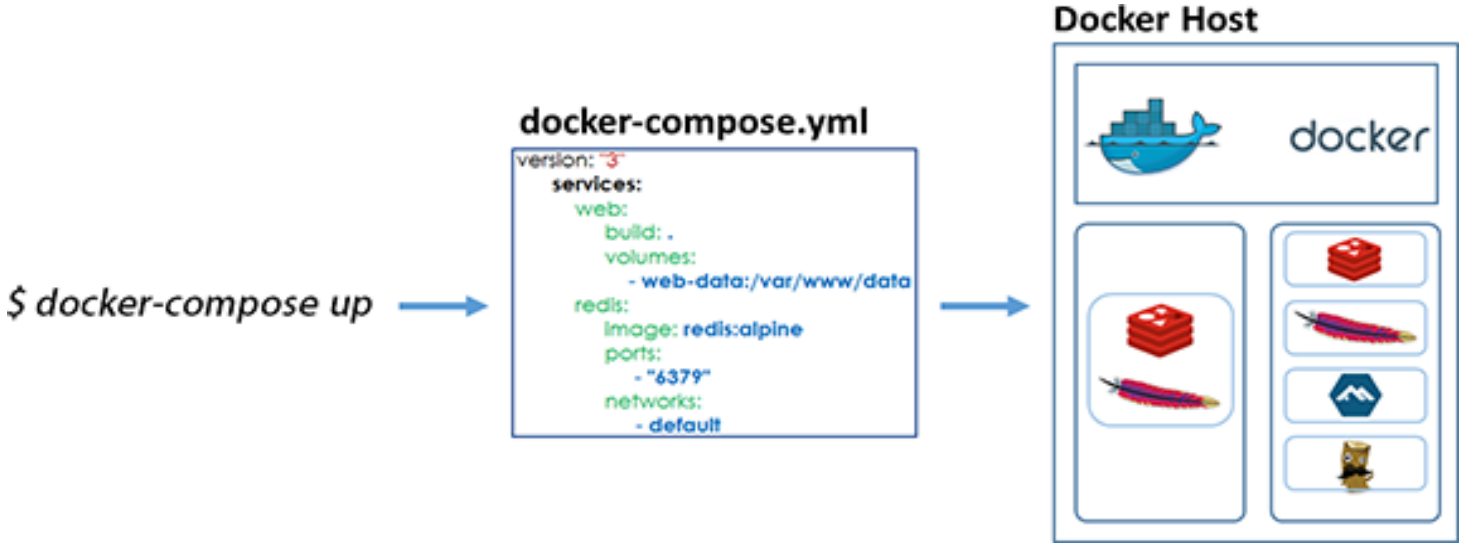
## Docker Hub

fornisce un gran numero di immagini del contenitore pronti per il download

# DOCKERFILE




# DOCKER COMPOSE



# PREPARAZIONE AMBIENTE

- Installazione Docker;

ubuntu

Windows 10

- Creazione directory;

```
/myproject
      /python
            Dockerfile
      /java
            Dockerfile
 docker-compose.common.yml
 docker-compose.dev.yml
 docker-compose.prod.yml
 Makefile
 Python-test.sh
 .gitlab-ci.yml
```

# Dockerfile

```
FROM python:3.6-slim

COPY . /code
WORKDIR /code

RUN pip install --no-cache-dir -r requirements.txt
RUN pip install -e .

ENTRYPOINT python ./mypackage/run.py
```


```
FROM maven:3.5-jdk-8

COPY . /usr/src/app
WORKDIR /usr/src/app

RUN apt-get update && apt-get install entr -y

RUN mvn clean package --batch-mode
ENTRYPOINT java -jar target/docker-compose-java-example-1.0-SNAPSHOT.jar
```

# docker-compose.common.yml

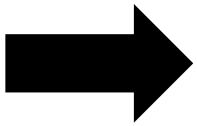
```
version: '2'

services:
  python:
    build: ./python
    environment:
      - POSTGRES_USER
      - POSTGRES_PASSWORD
      - POSTGRES_DB
      - POSTGRES_HOST

  java:
    build: ./java
    environment:
      - POSTGRES_USER
      - POSTGRES_PASSWORD
      - POSTGRES_DB
      - POSTGRES_HOST
```

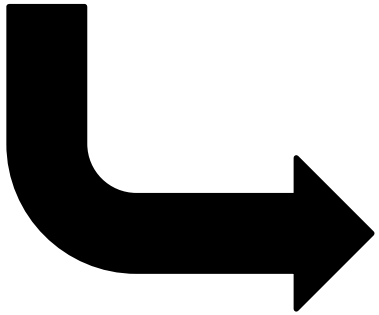
# docker-compose.dev.yml


```
volumes:
    - ./python/:/code
```

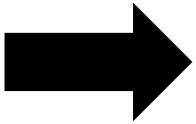


Riflette modifiche nel contenitore

```
entrypoint: watchmedo auto-restart --recursive --pattern="*.py" --
directory="." python mypackage/run.py
```



WATCHDOG



Esamina i file, riavvia il processo ed esegue il comando

# docker-compose.dev.yml


```
python-tests:
    image: registry.gitlab.com/f.cassano32/myproject/python:dev
    entrypoint: watchmedo auto-restart --recursive --pattern="*.py" --
directory="." pytest
    depends_on:
        - python
```

# python-tests.sh

```
#!/bin/bash
docker-compose -f docker-compose.common.yml -f docker-compose.dev.yml
run --rm --entrypoint pytest python $*
```

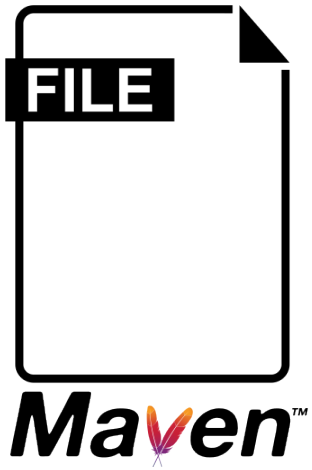
# docker-compose.dev.yml


**Dockerfile:**

```
RUN apt-get update && apt-get install entr –y
```

```
entrypoint: sh -c 'find src/ | entr mvn clean compile exec:java --batch-mode --quiet'
```

**pom.xml**



# docker-compose.dev.yml


```
postgres:
    image: postgres:9.6
    environment:
        - POSTGRES_USER=user
        - POSTGRES_PASSWORD=password
        - POSTGRES_DB=myproject
    volumes:
        -/data/myproject/postgres:/var/lib/postgresql/data
pgadminer:
    image: clue/adminer
    ports:
        -"99:80"
```

# CREAZIONE ED ESECUZIONE DEI CONTENITORI

**Costruzione:**

```
docker-compose -f docker-compose.common.yml -f docker-compose.dev.yml build
```

**Esecuzione:**

```
docker-compose -f docker-compose.common.yml -f docker-compose.dev.yml up
```

```
PS C:\Users\SER-03\myproject> docker-compose -f docker-compose.common.yml -f docker-compose.dev.yml up
Starting myproject_postgres_1  ... done
Starting myproject_pgadminer_1 ... done
Starting myproject_python_1    ... done
Starting myproject_java_1      ... done
Starting myproject_python-tests_1 ... done
Attaching to myproject_postgres_1, myproject_pgadminer_1, myproject_python_1, myproject_java_1, myproject_python-tests_1
postgres_1       | LOG:  database system was shut down at 2019-03-18 15:22:16 UTC
postgres_1       | LOG:  MultiXact member wraparound protections are now enabled
postgres_1       | LOG:  database system is ready to accept connections
postgres_1       | LOG:  autovacuum launcher started
pgadminer_1      | 2019-03-18 15:22:29,950 CRIT Supervisor running as root (no user in config file)
pgadminer_1      | 2019-03-18 15:22:29,952 INFO supervisord started with pid 1
pgadminer_1      | 2019-03-18 15:22:30,954 INFO spawned: 'nginx' with pid 8
pgadminer_1      | 2019-03-18 15:22:30,956 INFO spawned: 'php5-fpm' with pid 9
pgadminer_1      | 2019-03-18 15:22:31,987 INFO success: nginx entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
pgadminer_1      | 2019-03-18 15:22:31,987 INFO success: php5-fpm entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
python_1         | Sono nel container! PYTHON!!!
python-tests_1   | ============================= test session starts ==============================
python-tests_1   | platform linux -- Python 3.6.8, pytest-4.3.1, py-1.8.0, pluggy-0.9.0
python-tests_1   | rootdir: /code, inifile:
python-tests_1   | collected 1 item
python-tests_1   |
python-tests_1   | tests/my_test.py .                                                   [100%]
python-tests_1   |
python-tests_1   | =========================== 1 passed in 0.19 seconds ===========================
java_1           | Sono nel container!! JAVA!!!
```

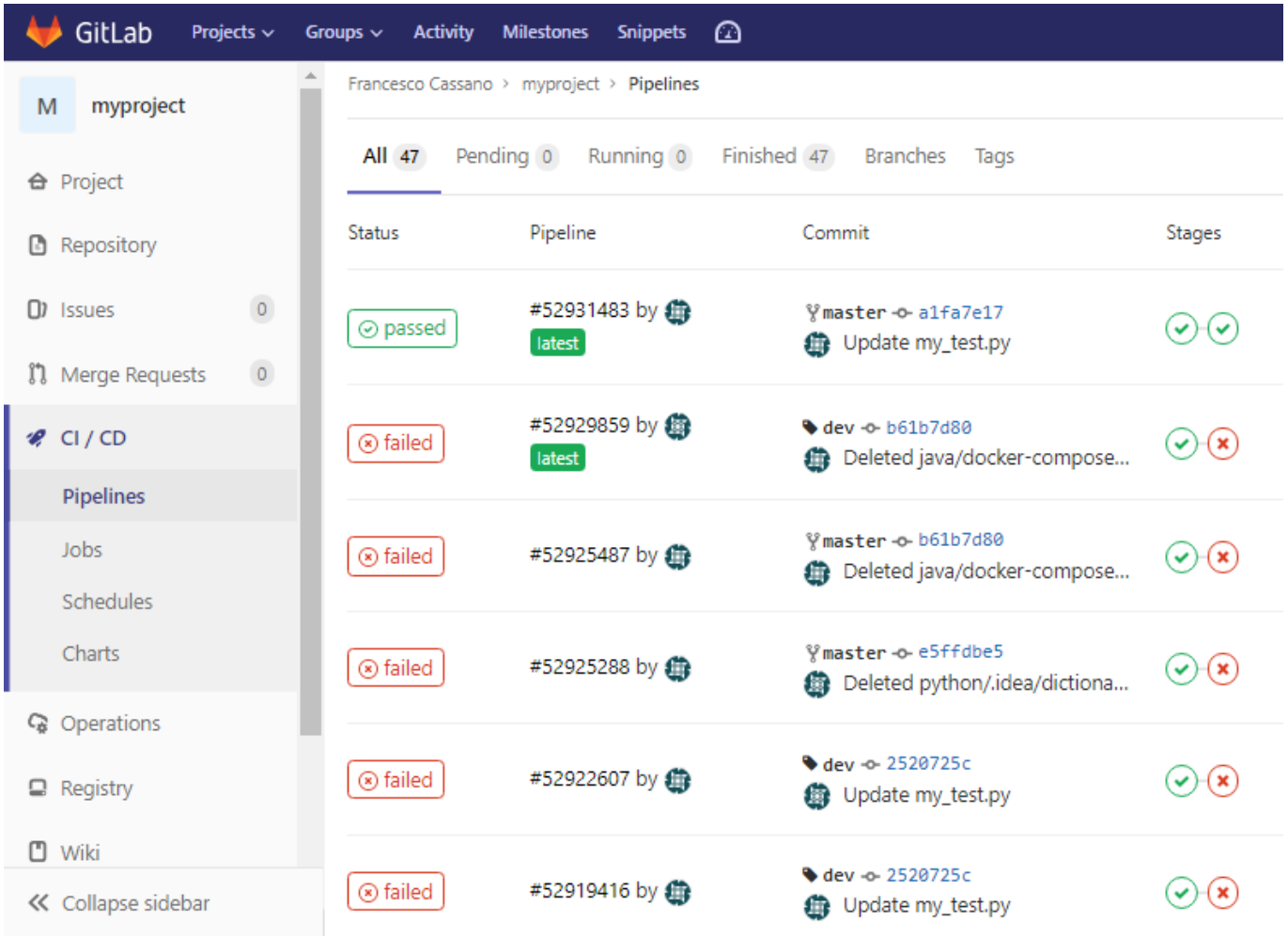

# docker-compose.prod.yml


```
version: '2'
services:
   python:
      image: $IMAGE/python:$TAG
      restart: always
   java:
      image: $IMAGE/java:$TAG
      restart: always
```


# .gitlab-ci.yml


```
stages:
  - build
  - test
variables:
  TAG: $CI_BUILD_REF
  IMAGE: $CI_REGISTRY_IMAGE
services:
  - docker:dind
image: docker
before_script:
  - apk add --update py-pip
  - pip install docker-compose
  - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN $CI_REGISTRY
build:
  stage: build
  script:
      - docker-compose -f docker-compose.common.yml -f docker-compose.prod.yml build
      - docker-compose -f docker-compose.common.yml -f docker-compose.prod.yml push
test-python:
   stage: test
   script:
      - docker-compose -f docker-compose.common.yml -f docker-compose.prod.yml pull python
      - docker-compose -f docker-compose.common.yml -f -docker-compose.prod.yml run --rm --
entrypoint pytest python
```

# PIPELINE GITLAB

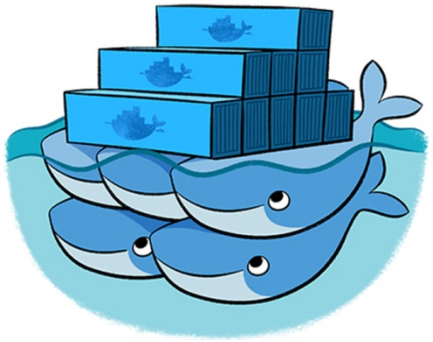


# Conclusioni

- Analisi e progettazione →Supporta principi sviluppo
- Codifica →Ambiente comodo e portabile
- Test →Integrazione continua
- Rilascio → Distribuzione continua

# Sviluppi futuri

- Docker Swarm
- Kubernetes



# GRAZIE PER L'ATTENZIONE