

TURNKEY CONTINUOUS DELIVERY

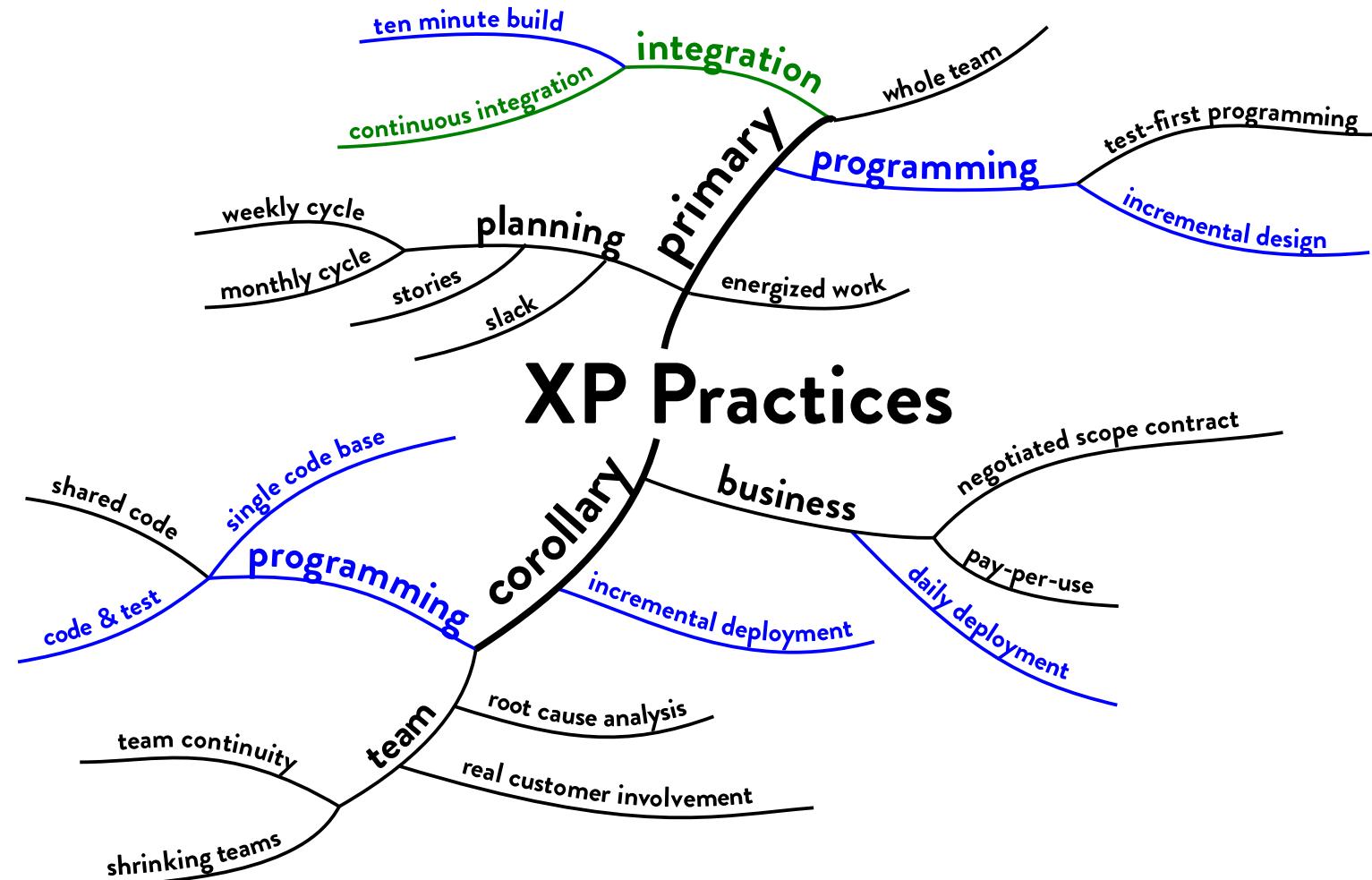
Gianni Bombelli

bombelli@intre.it

<https://github.com/bombix>

<https://www.linkedin.com/in/gianni-bombelli>

EXTREME PROGRAMMING



CONTINUOUS INTEGRATION



Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.

CONTINUOUS ...



Kofy
Senaya

Director of
Product @
Clearbridge
Mobile

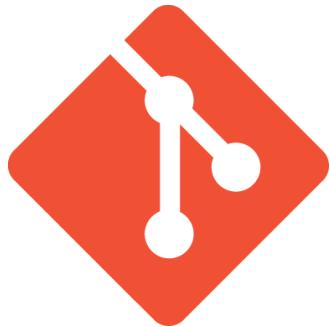
Continuous Delivery is similar to Continuous Integration. You are building a product that can be released to production at any time. Continuous Delivery requires building testing, and releasing faster and more frequently. Continuous Integration happens before you build as you are testing code. Delivery means you can release something to the staging environment or the pre-production environment.

Continuous Delivery is when your code is always ready to be released but isn't pushed to production unless you make the decision to do so. It is a manual step. With Continuous Deployment, any updated working version of the app is automatically pushed to production.

TOOLS



HashiCorp
Vagrant



GitLab



ANSIBLE

WHAT DO WE DO NOW?

- bootstrap our Continuous Delivery and test environment
- configure the tools necessary to manage and build our software
- create our ~~build~~ Continuous Delivery pipeline
- develop a simple micro-service
 - REST server
 - written in ES6
 - executed with Node.js
 - released as a docker image

REPOSITORIES

Build and Release Infrastructure

<https://github.com/bombix/workshop-turnkey>

Example Project

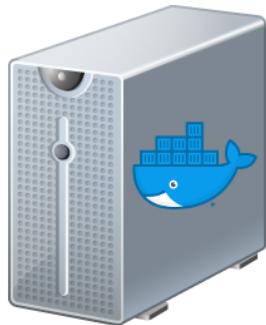
<https://github.com/intresrl/microservice-example-hex2hsl>

- Remove the remote named origin
- Checkout the tag start
- Reset master branch to the commit with the start tag

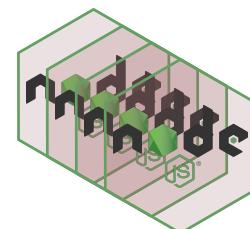
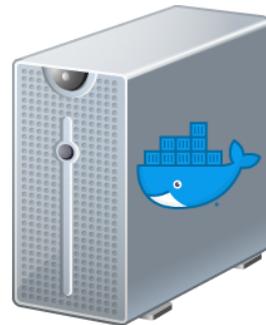
INFRASTRUCTURE



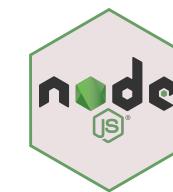
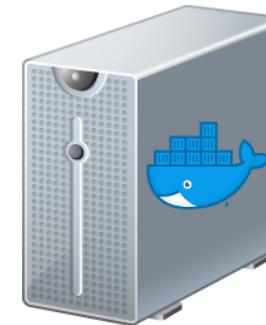
Turnkey CD



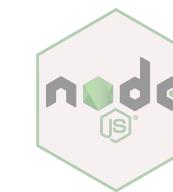
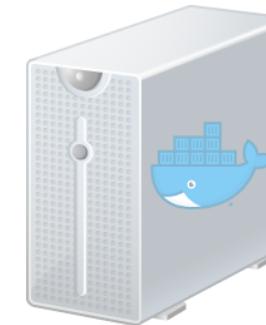
Testing



Staging



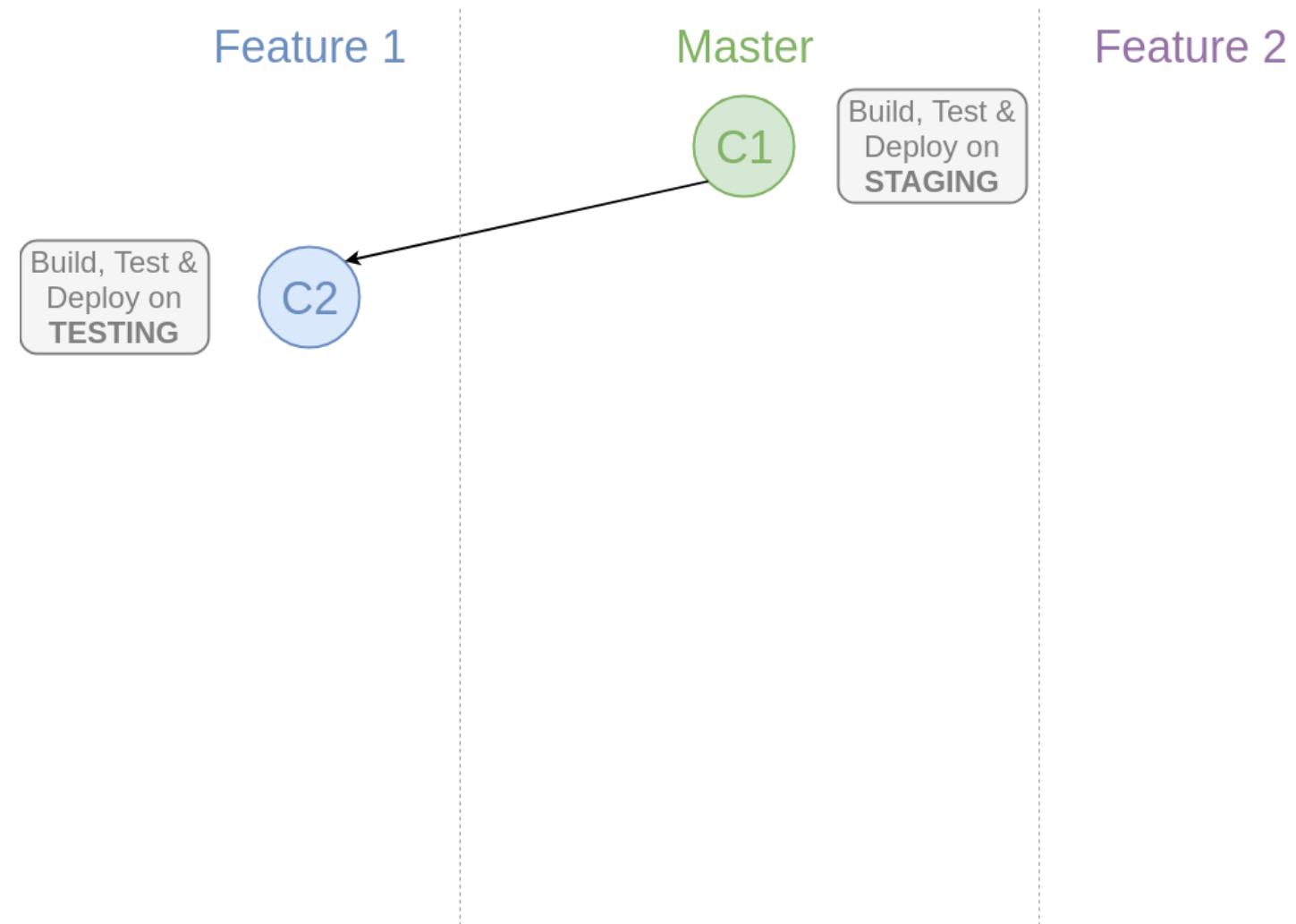
Production



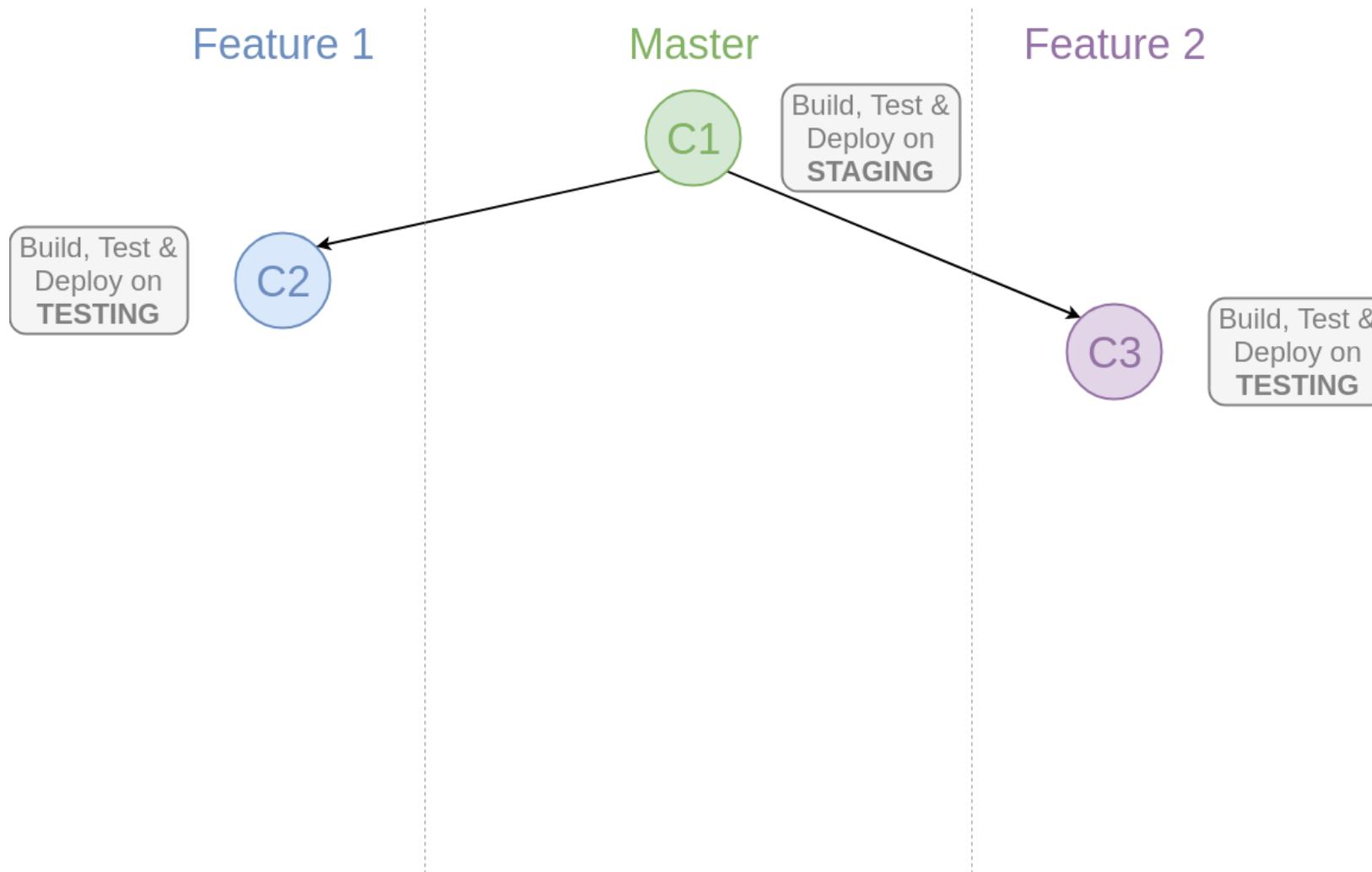
INFRASTRUCTURE AS CODE



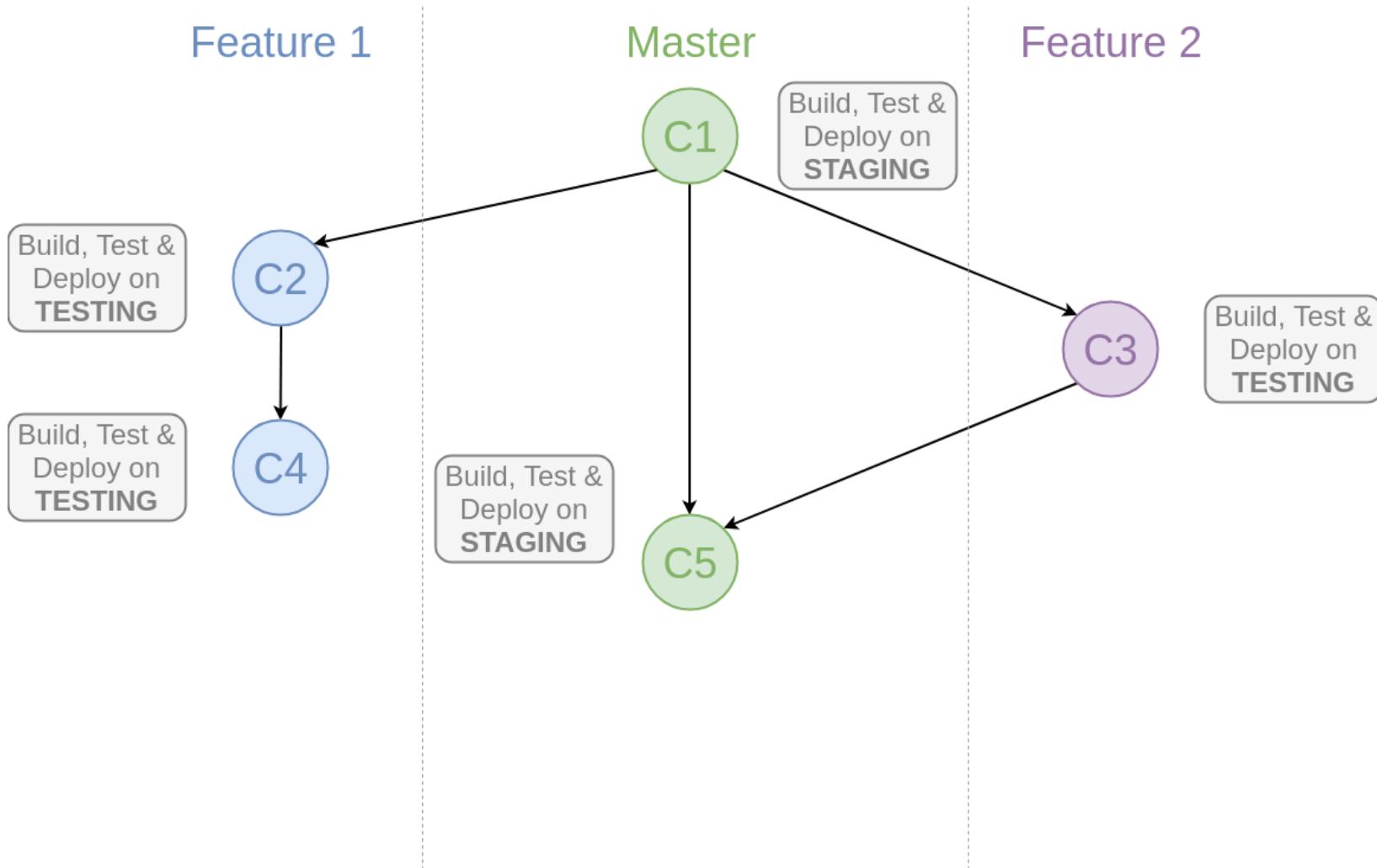
WORKFLOW - F.1



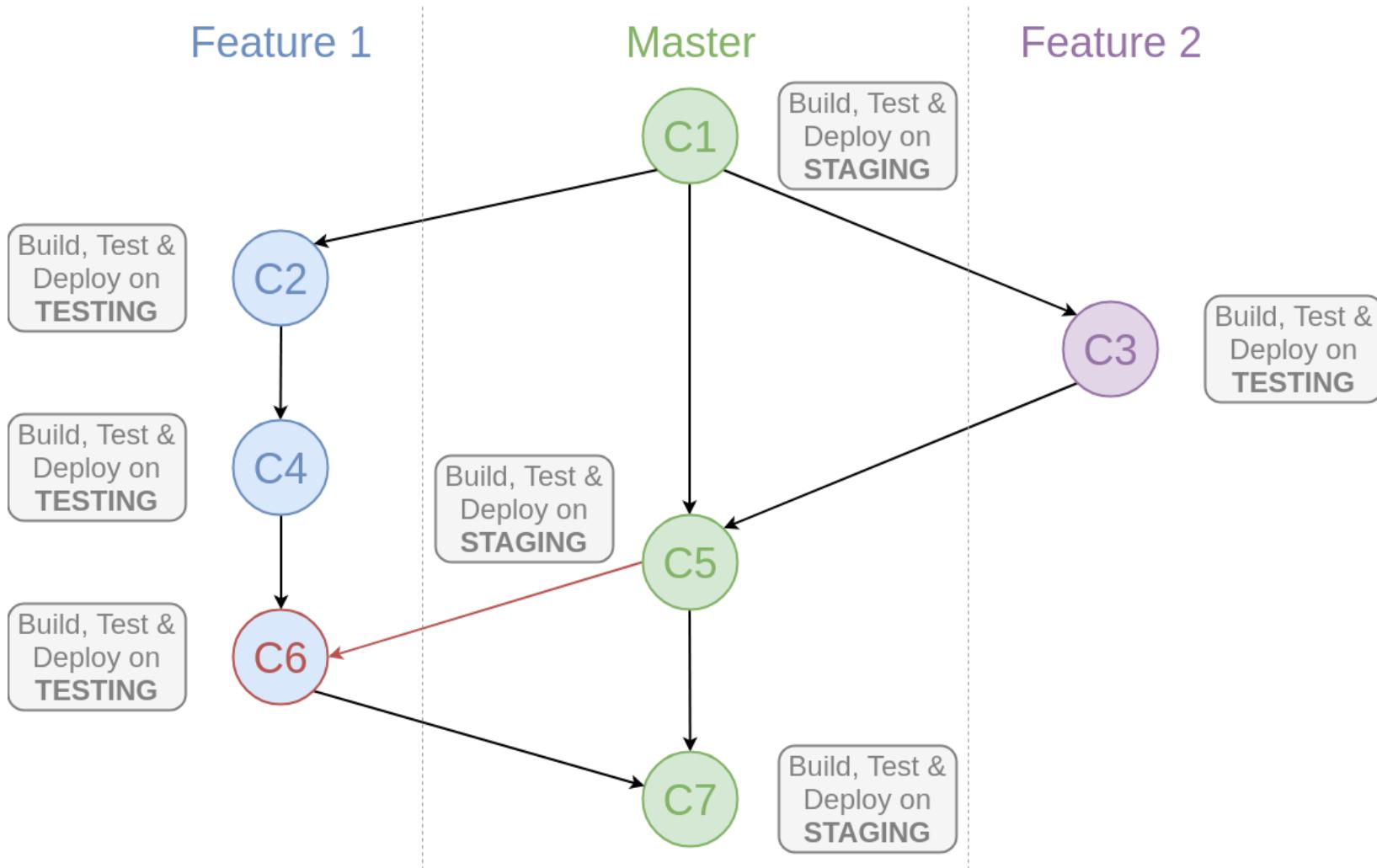
WORKFLOW - F.2



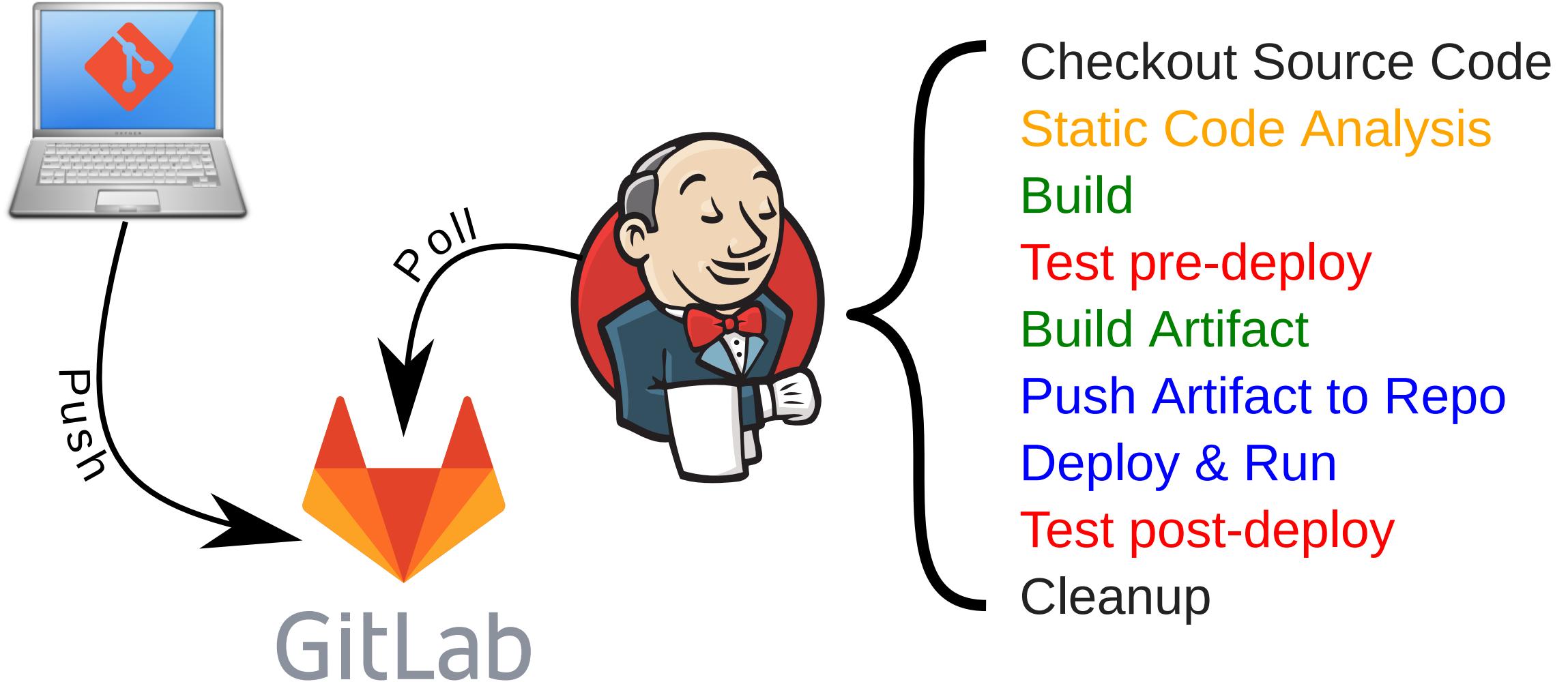
WORKFLOW - F.2 MERGE



WORKFLOW - F.1 MERGE

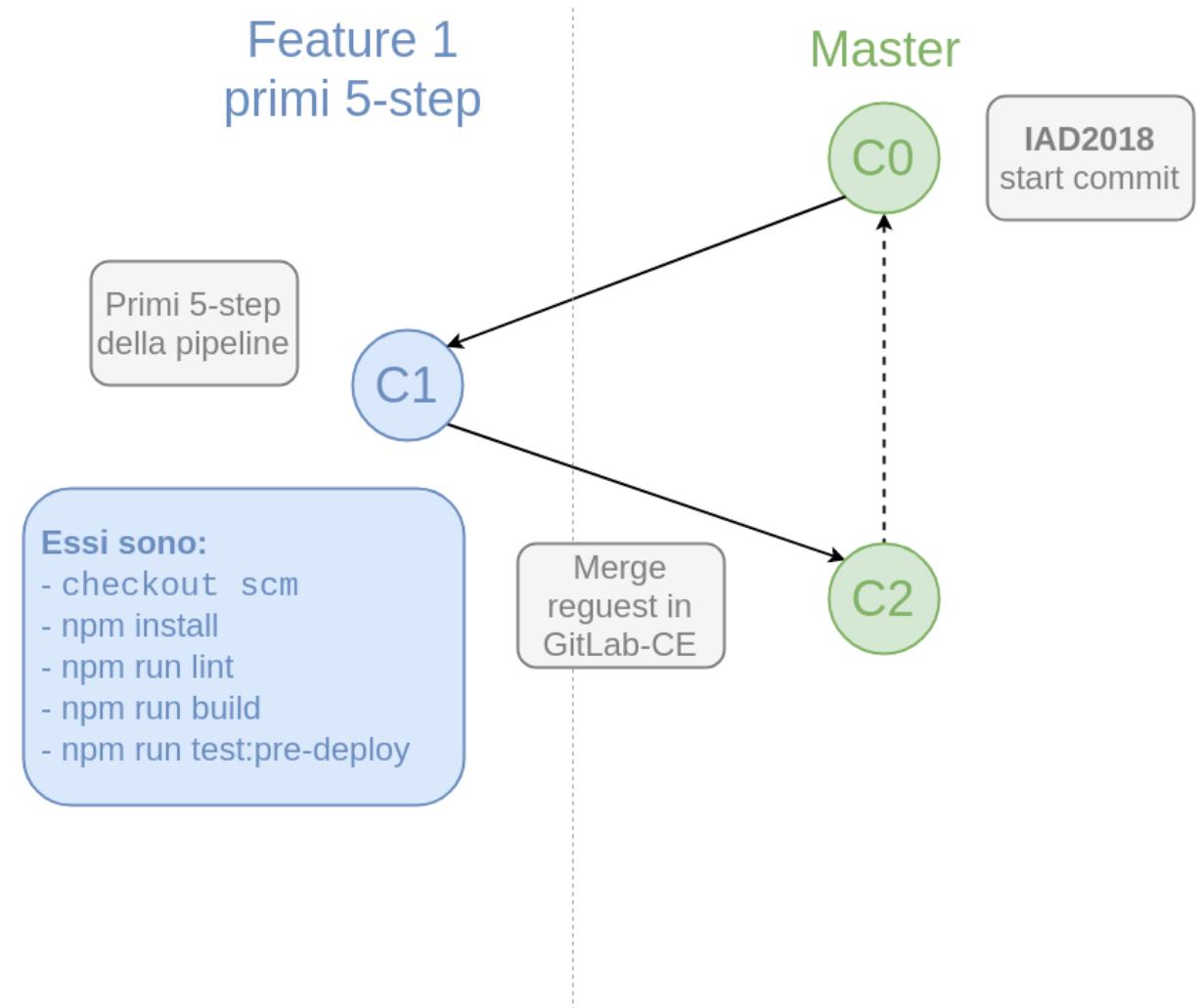


BUILD AUTOMATION



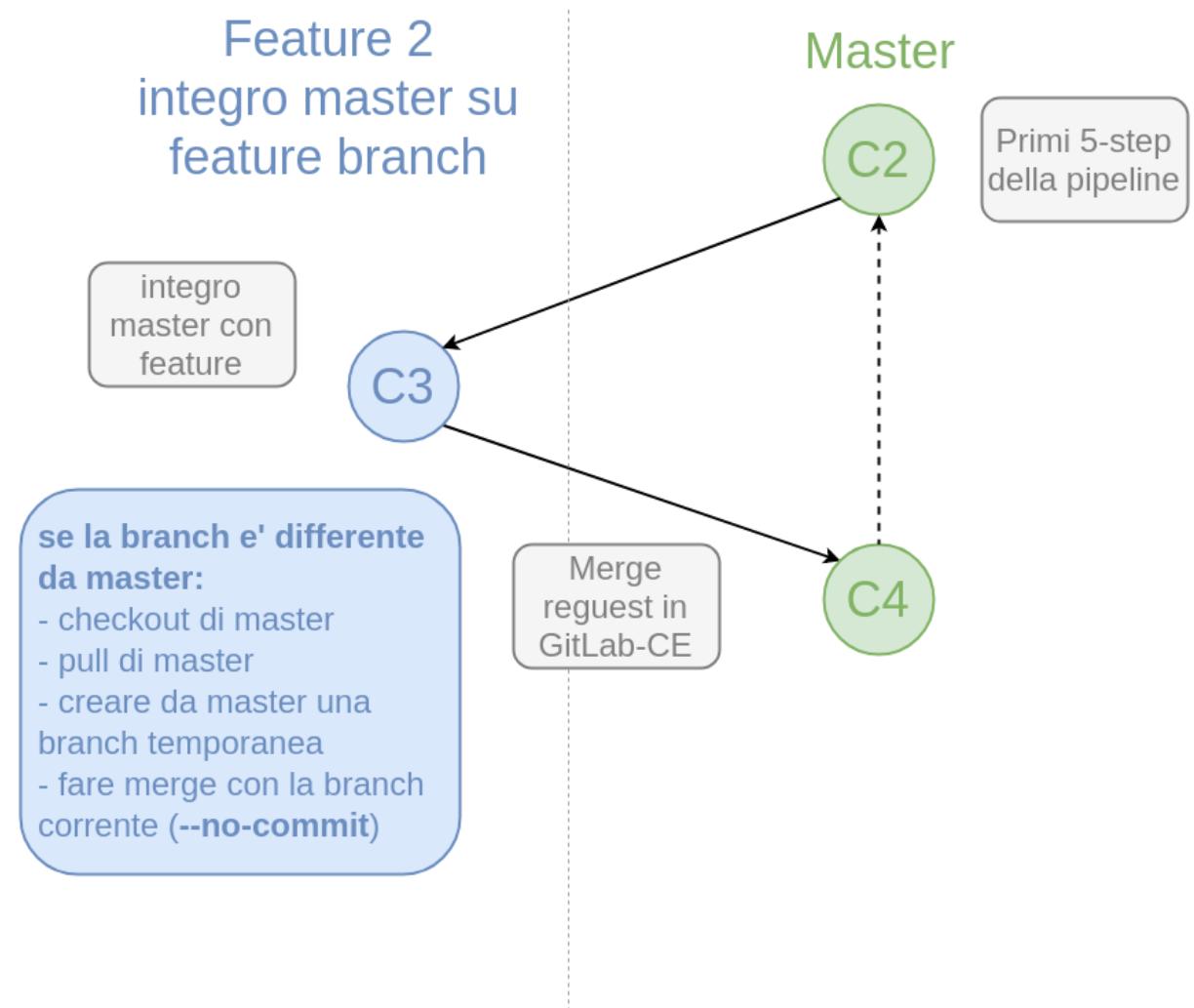
ESERCIZIO 0

ESERCIZIO 0



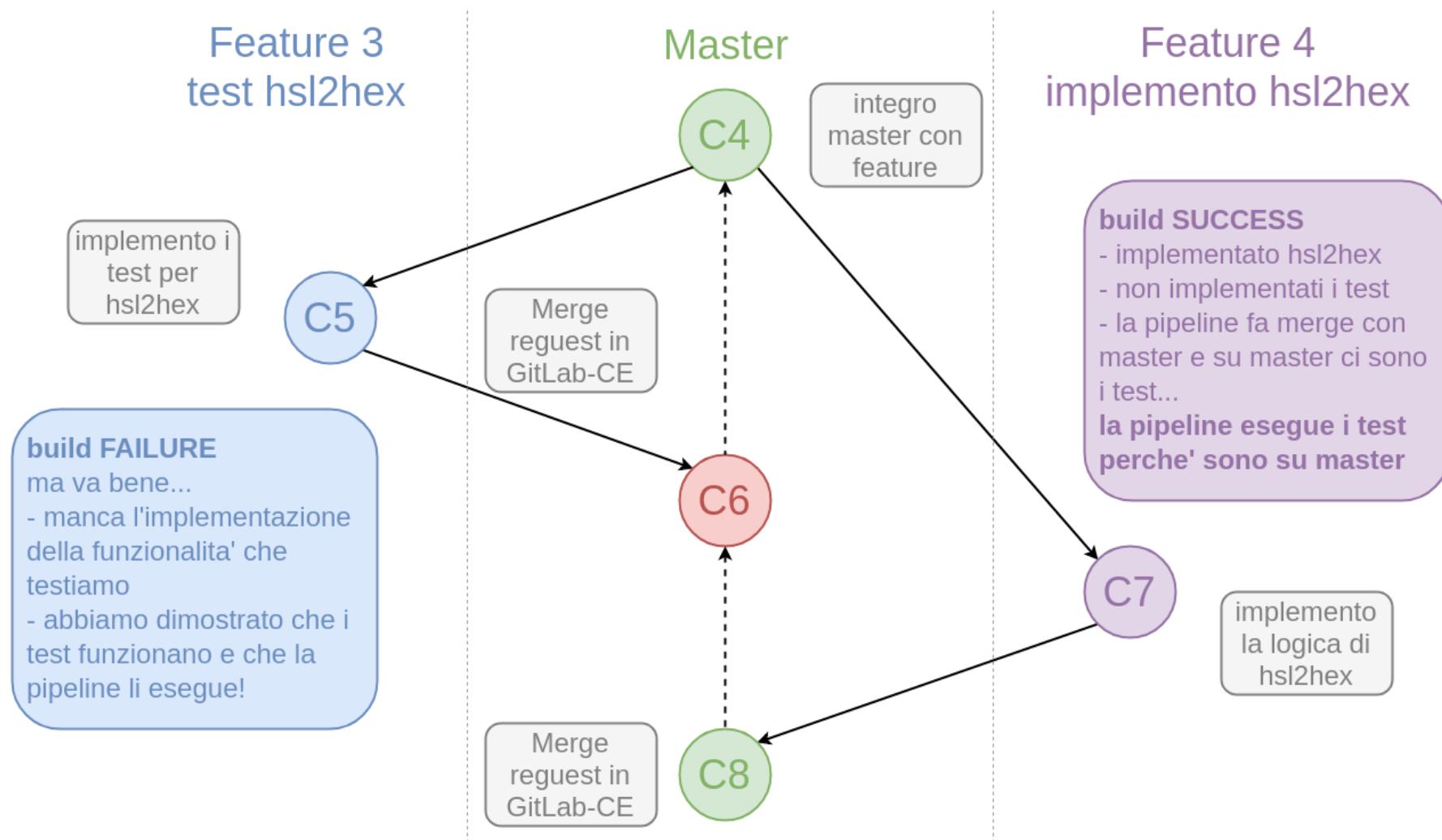
ESERCIZIO 1

ESERCIZIO 1

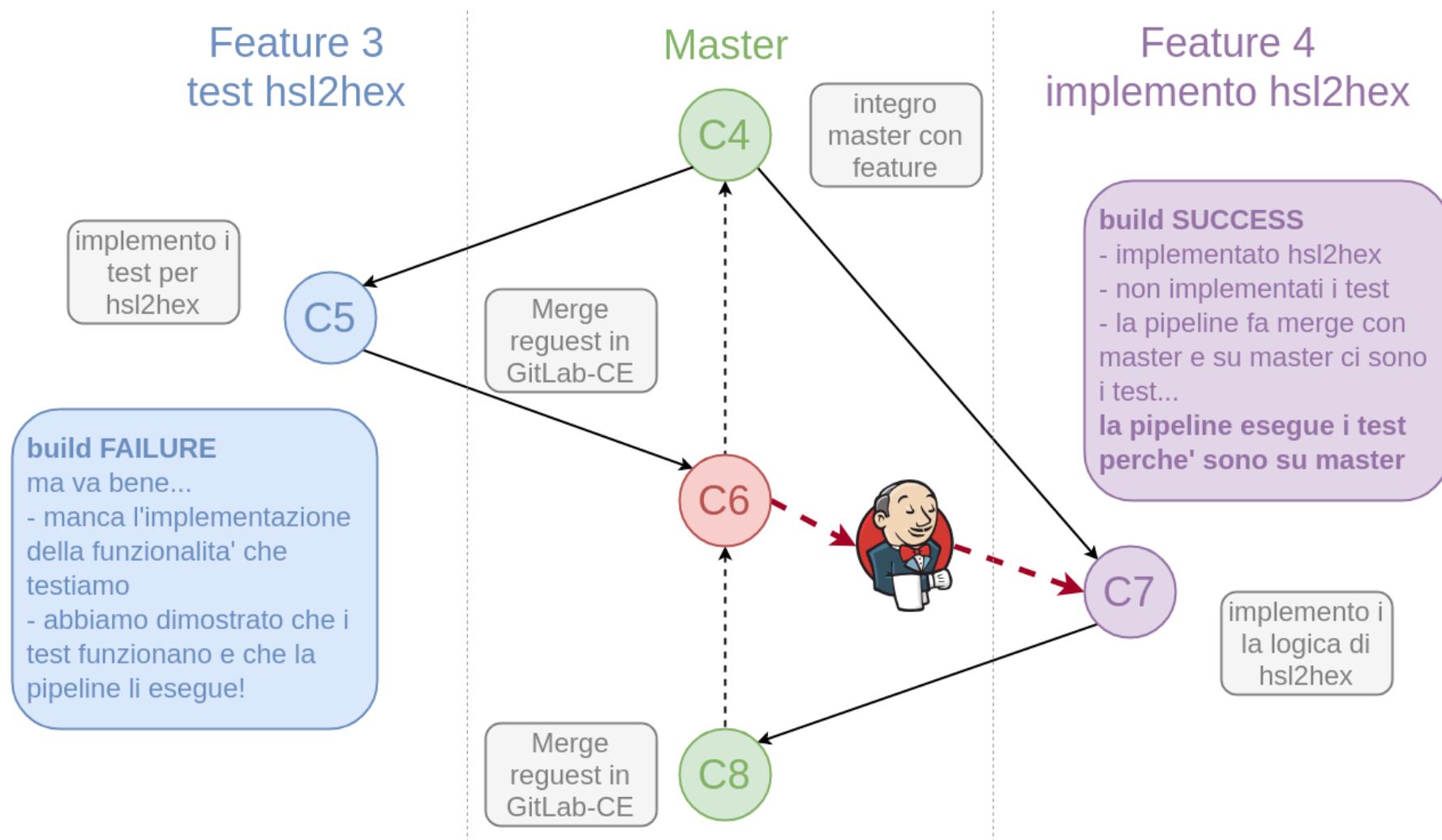


ESERCIZIO 2

ESERCIZIO 2

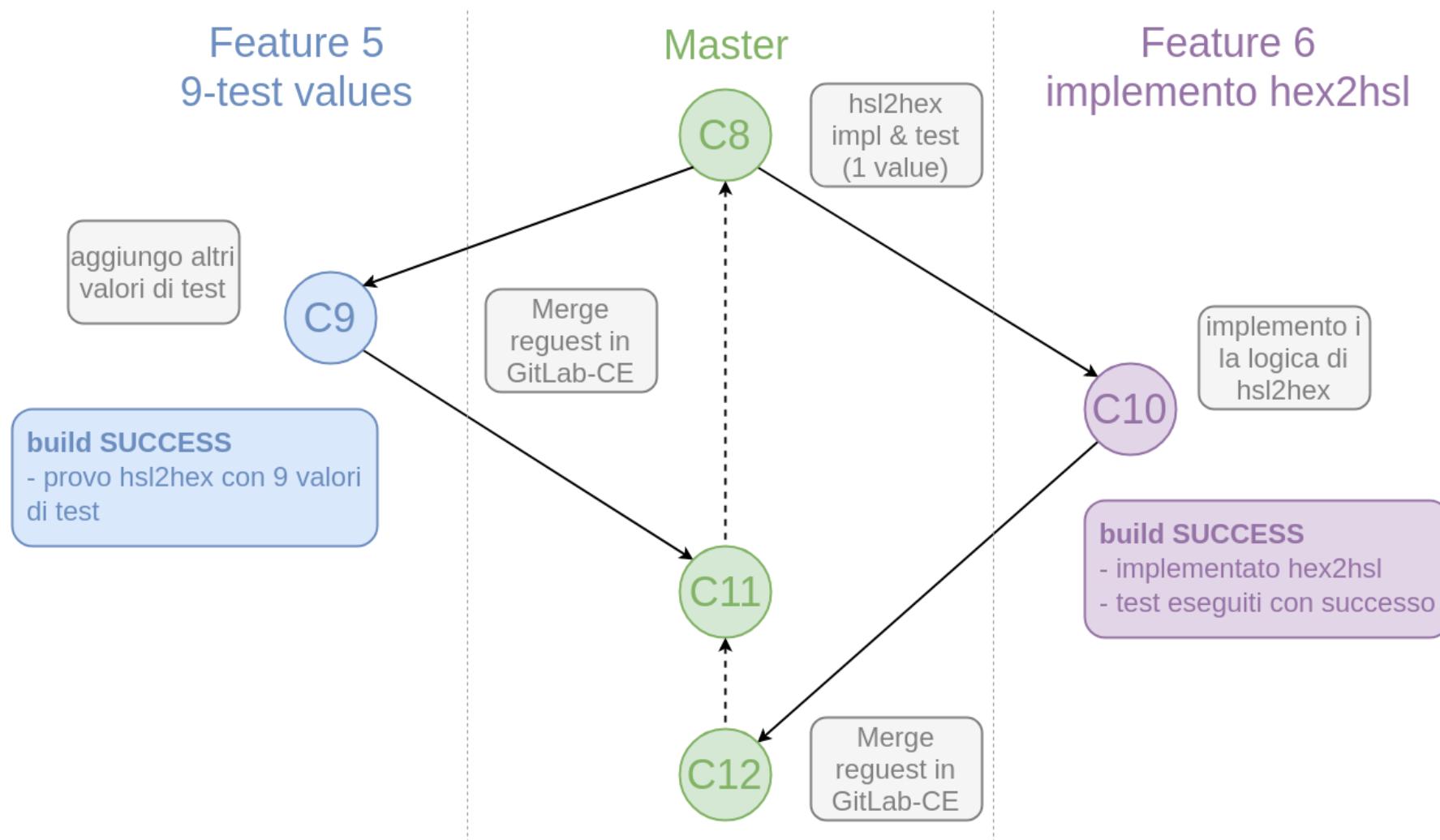


ESERCIZIO 2

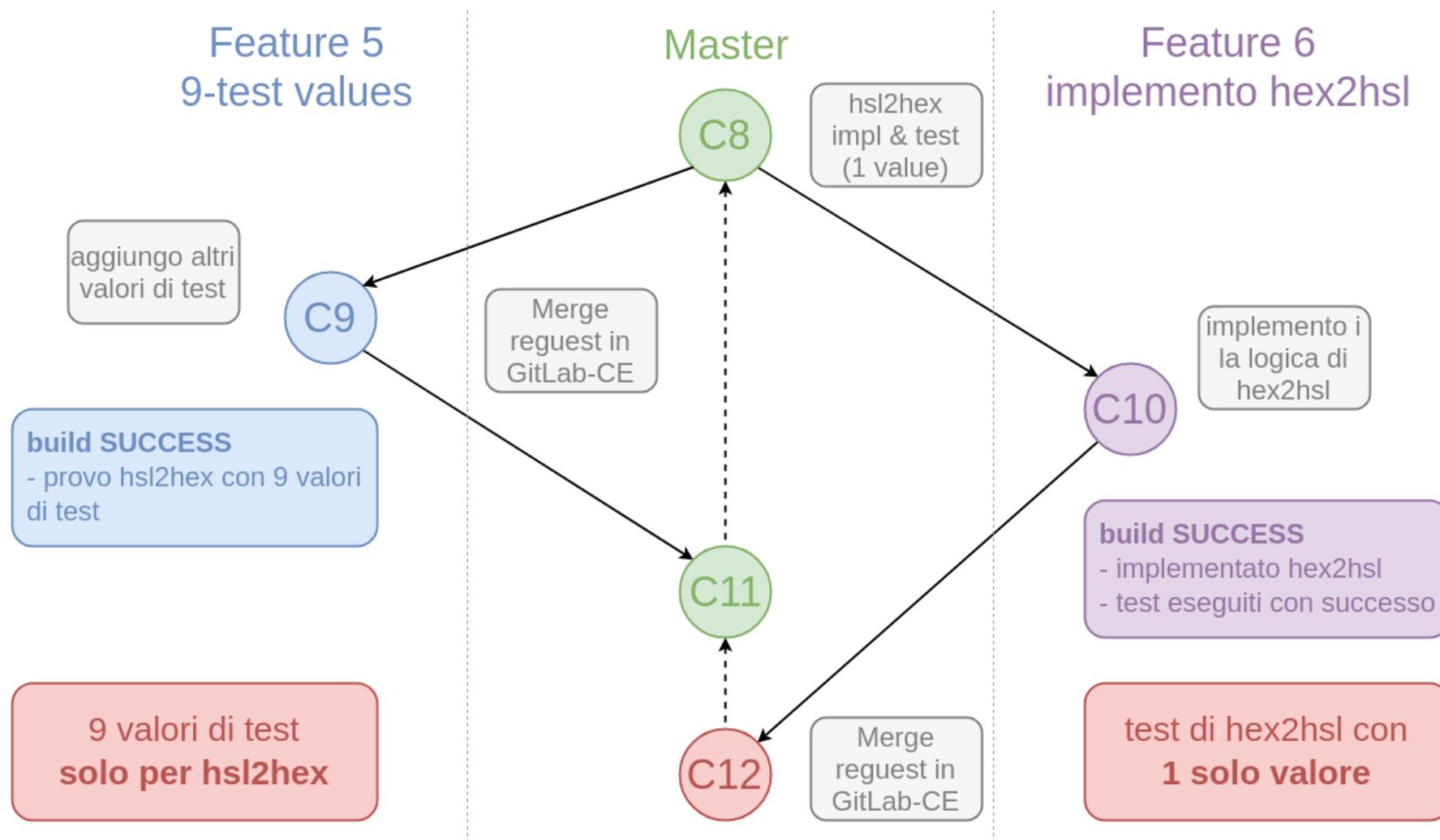


ESERCIZIO 3

ESERCIZIO 3

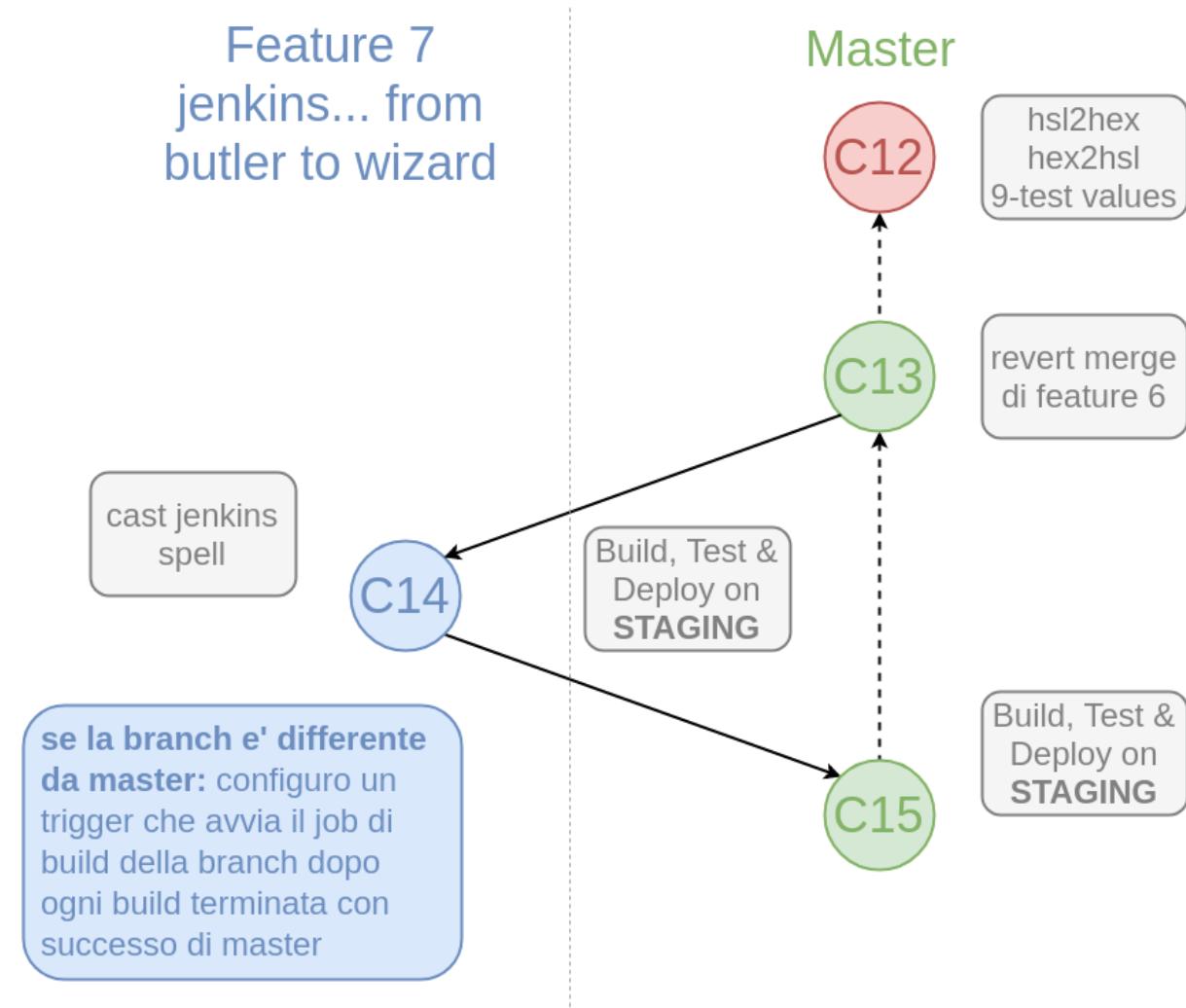


ESERCIZIO 3



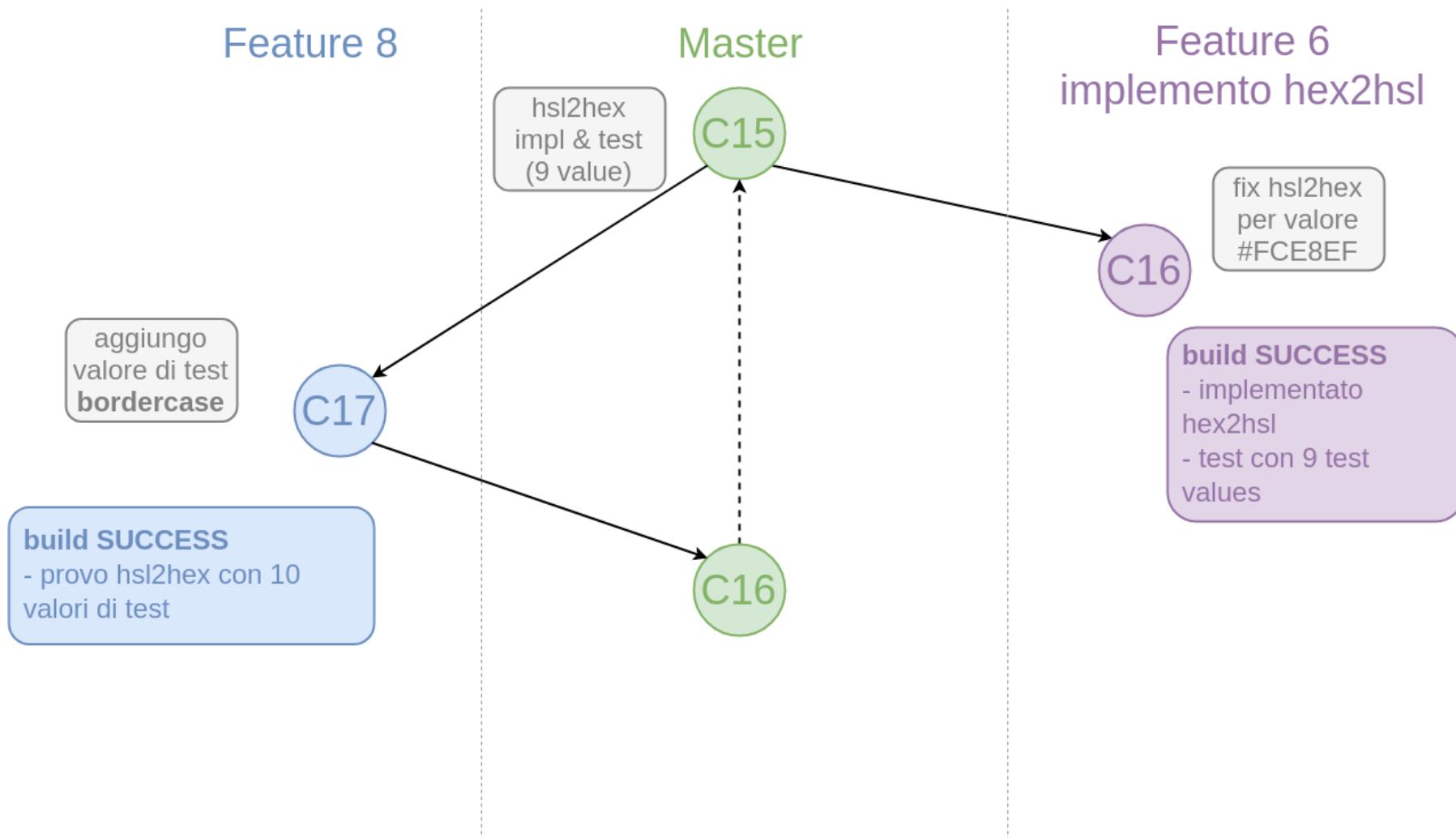
ESERCIZIO 4

ESERCIZIO 4

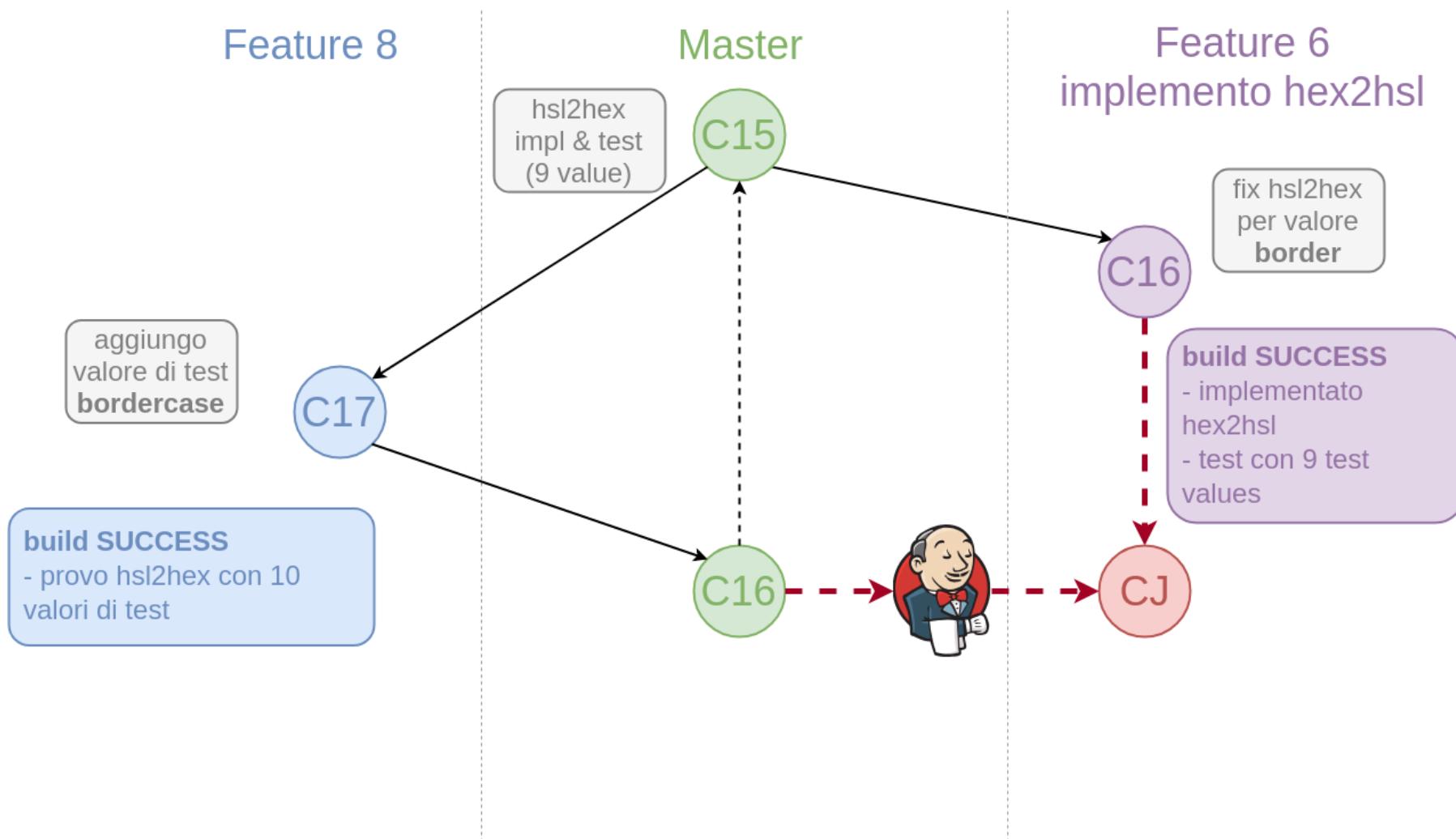


ESERCIZIO 5

ESERCIZIO 5



ESERCIZIO 5



IT'S ENOUGH ?

- single shared code-base
- automated build & merge different branches
- self-testing build
- integrate every commit
- ten-minutes build & fail fast!
- everyone can see what's happening
- end-to-end test
- automated deployment
- test the deployment
- test in a production-like environment

NEXT STEPS ...

- Build Artifact
- Push to Artifact Repository
- Deploy on a test or staging server
- Run it
- Execute the post-deployment tests

BUILD ARTIFACT AND PUSH

```
stage('Build Docker Image') {  
    def container-name = ${pipelineName}-${env.BRANCH_NAME}  
    sh "docker -H tcp://192.168.50.91:2375 build \  
        -t 192.168.50.91:5000/${container-name}:${env.BUILD_ID} \  
        -t 192.168.50.91:5000/${container-name}:latest ."  
}  
  
stage('Push to Docker Registry') {  
    def container-name = ${pipelineName}-${env.BRANCH_NAME}  
    sh "docker -H tcp://192.168.50.91:2375 \  
        push 192.168.50.91:5000/${container-name}:${env.BUILD_ID}"  
    sh "docker -H tcp://192.168.50.91:2375 \  
        push 192.168.50.91:5000/${container-name}:latest"  
}
```

DEPLOY & RUN

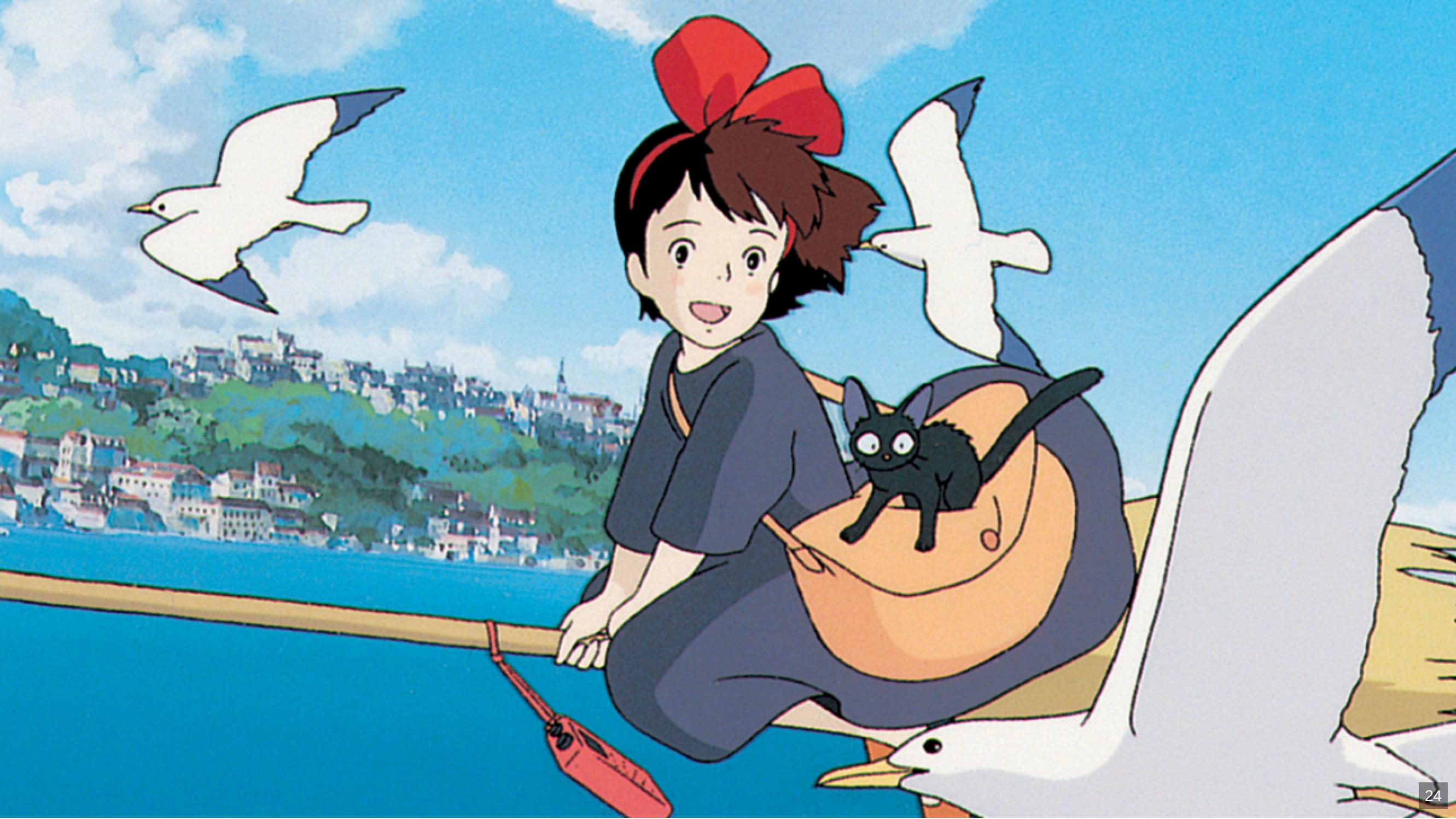
```
stage ('Deploy and Run') {
    if (env.BRANCH_NAME == 'master') {
        print "Deploy docker container to : staging environmet"
        sh "ansible-playbook -i /ansible/inventory/hosts.yml \
            /ansible/hex2hsl.yml --limit staging"
    } else {
        print "Deploy docker container to : testing environmet"
        sh "ansible-playbook -i /ansible/inventory/hosts.yml \
            --extra-vars 'hex2hsl_branch=${env.BRANCH_NAME} hex2hsl_port=${test_\
            /ansible/hex2hsl.yml --limit testing"
    }
}
```

TEST POST-DEPLOY

```
stage('Test post-deploy') {  
    def test_url = ''  
    if (env.BRANCH_NAME == 'master') {  
        test_url = 'http://192.168.50.93:3100'  
    } else {  
        test_url = 'http://192.168.50.92:' + test_port  
    }  
    sh "npm run test:post-deploy -- --test_url=${test_url}"  
  
    step([$class: 'XUnitBuilder',  
          thresholds: [[$class: 'FailedThreshold', unstableThreshold: '1']],  
          tools: [[[$class: 'JUnitType',  
                  pattern: 'test-report/test-post-deploy-report.xml']]])  
}  
}
```

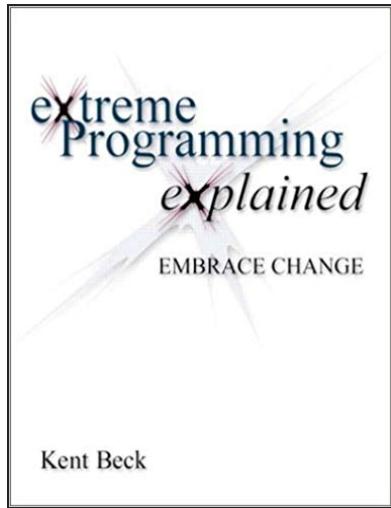
AND NOW, IT'S ENOUGH ?

- Regression tests: we always test everything
- e2e tests: yes, post-deploy tests runs on staging
- Security scans: only dependency audit performed by npm
- Performance test: nope!
- Deployment test: yes, deploy on staging environment



A photograph of a man with dark, curly hair and glasses, smiling warmly at the camera. He is wearing a dark suit jacket over a light-colored shirt. A small, fluffy kitten is nestled in his hands, looking up at him. The background consists of vertical wooden slats, possibly from a window or screen, with some light filtering through.

Special thanks to
Ferdinando Santacroce
[@jesuswasrasta](https://twitter.com/jesuswasrasta)



Continuous Integration

Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including tests) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly. This article is a quick overview of Continuous Integration summarizing the technique and its current usage.

01 May 2006 Martin Fowler

Translations: Portuguese · Chinese · Korean · French · Chinese · Czech

Find similar articles to this by looking at these tags: popular · agile · continuous delivery · extreme programming

For more information on this, and related topics, take a look at my [page](#) for delivery.

Throughput, my envoys, offers continuous delivery, and continuous integration. CruiseControl, the first continuous integration system, was originally developed at ThoughtWorks. ThoughtWorks Shakes, our products group, offers a range of services for continuous integration and delivery. It supports multiple projects, integration with multiple projects, a crazy-looking dashboard, and support for automated deployment. You can see how it is here to use for small setups.



GRAZi3

Packt

THE
DEVOPS 2.0
TOOLKIT



AUTOMATING THE CONTINUOUS DEPLOYMENT PIPELINE WITH CONTAINERIZED MICROSERVICES

VIKTOR FARCIĆ

The
Pragmatic
Programmers

Release It!
Second Edition

Design and Deploy
Production-Ready Software

