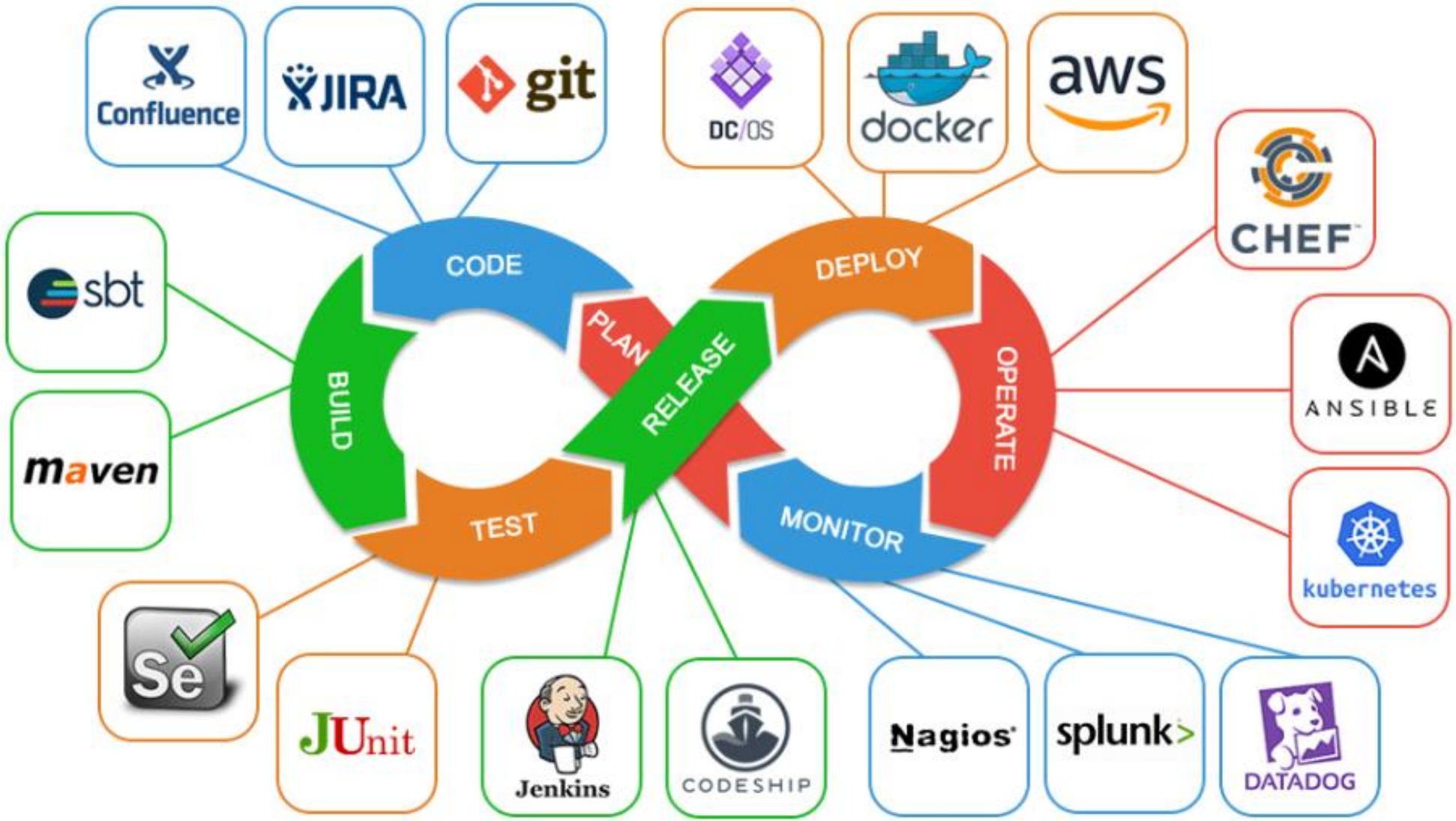

Dev
Ops

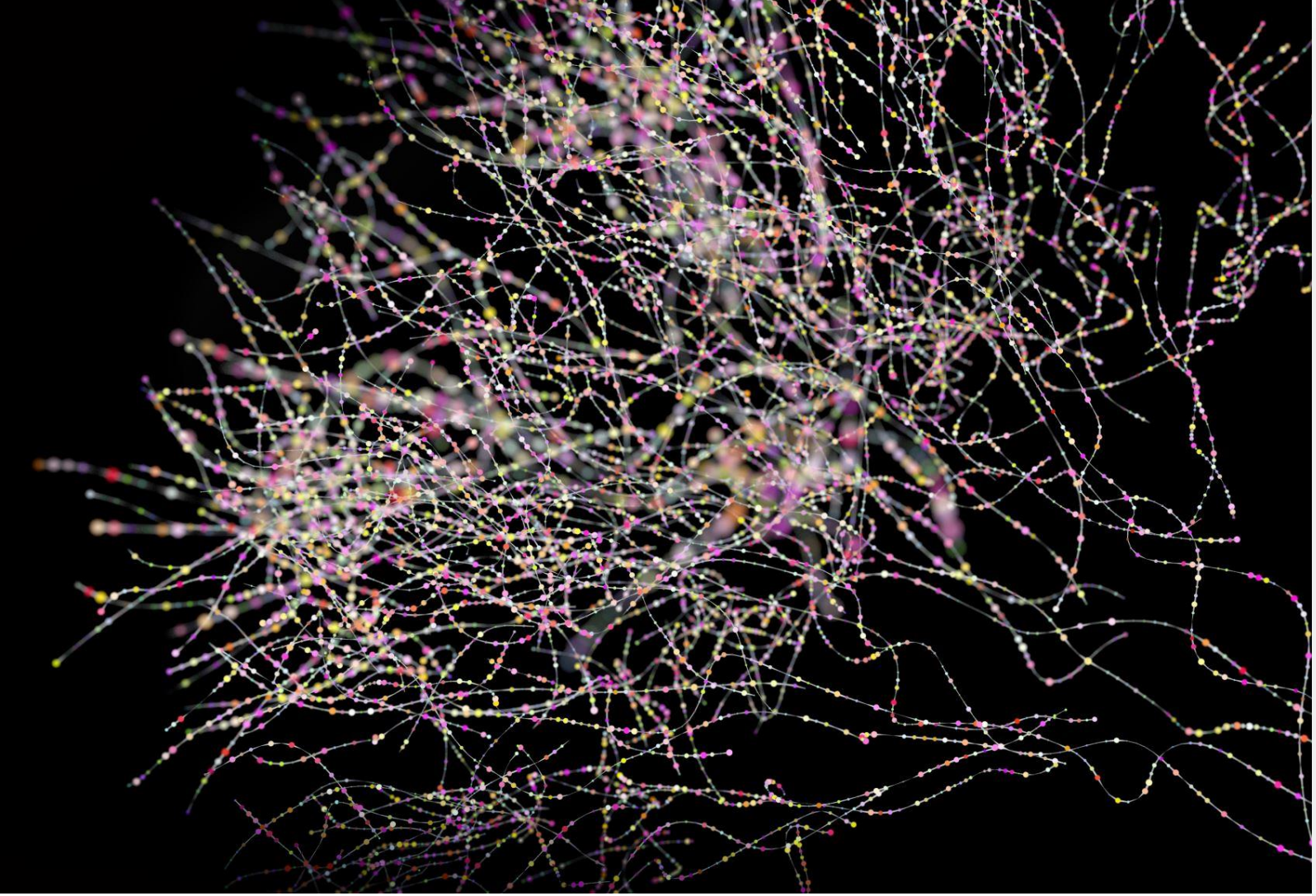

Confluence
JIRA
git
DC/OS
docker
aws
CHEF
sbt
Maven
CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR
ANSIBLE
kubernetes
Se
JUnit
Jenkins
CODESHIP
Nagios
splunk>
DATADOG

VENTAJAS DEVOPS

- Entrega más rápida y continua
- Mayor colaboración y transparencia
- Mayor calidad del software
- Optimización de recursos
- Mejora en la colaboración y la cultura
- Flexibilidad y adaptabilidad
- Mayor seguridad
- Retroalimentación continua y mejora continua

PRESENTACIONES

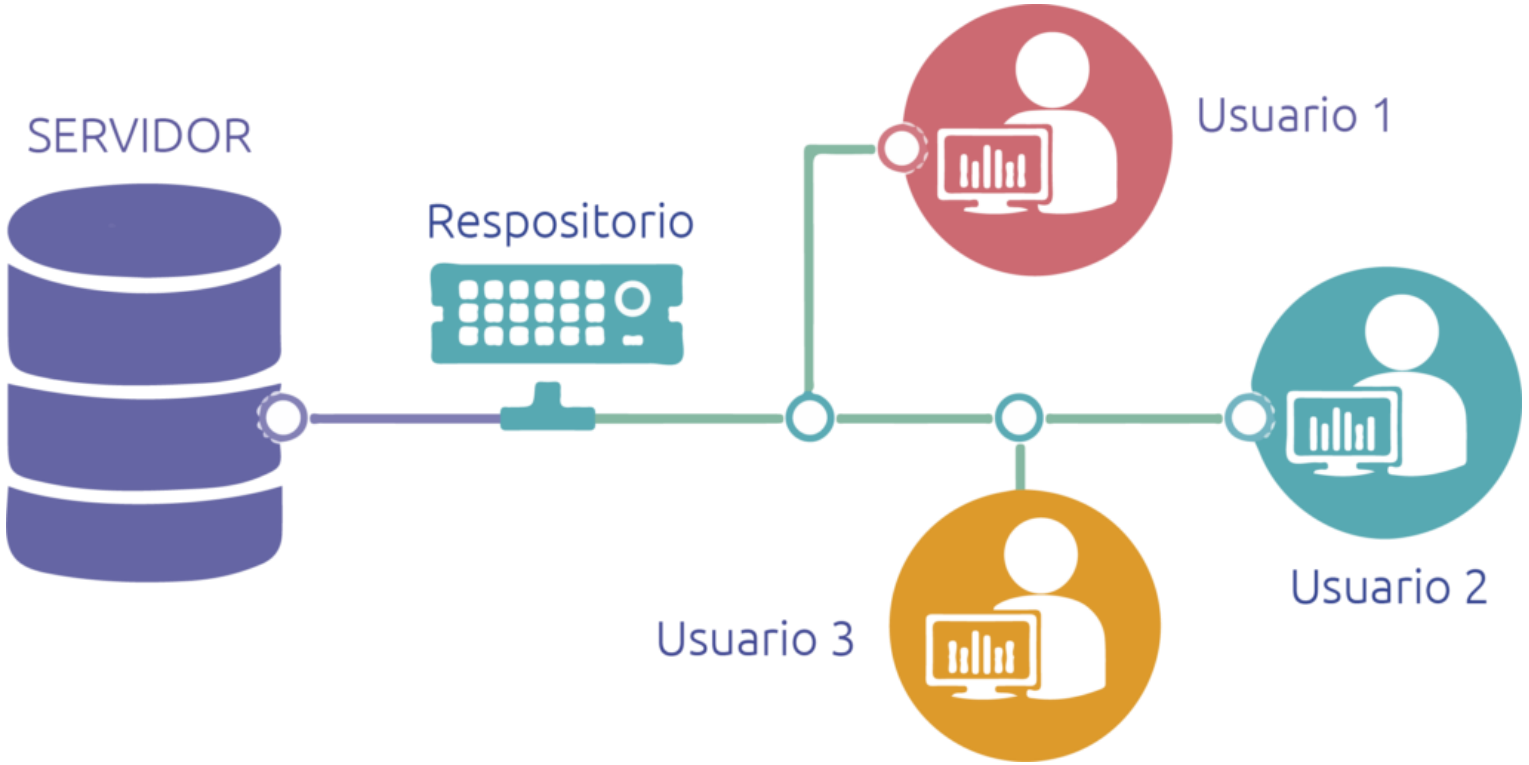
- Conocimientos de DevOps
- Conocimientos de Bitbucket
- Conocimientos de Jenkins
- Conocimientos de Testing

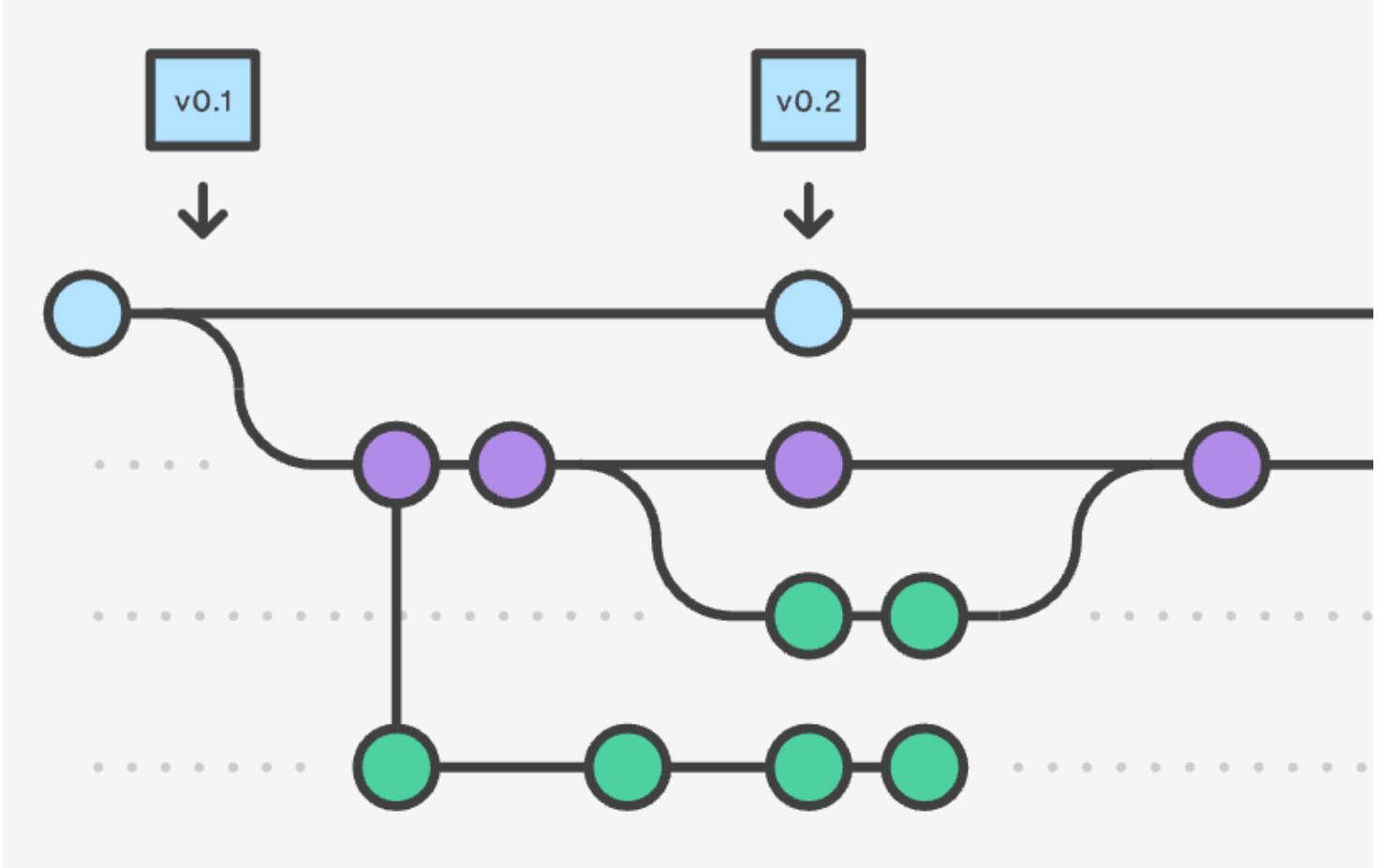


Modulo 1: Bitbucket

- Introducción a los CVS
- Gestión de un repositorio remoto
- Ramificaciones
- GitFlow
  - Desarrollo
  - Release
  - Hotfix
- Pull Request

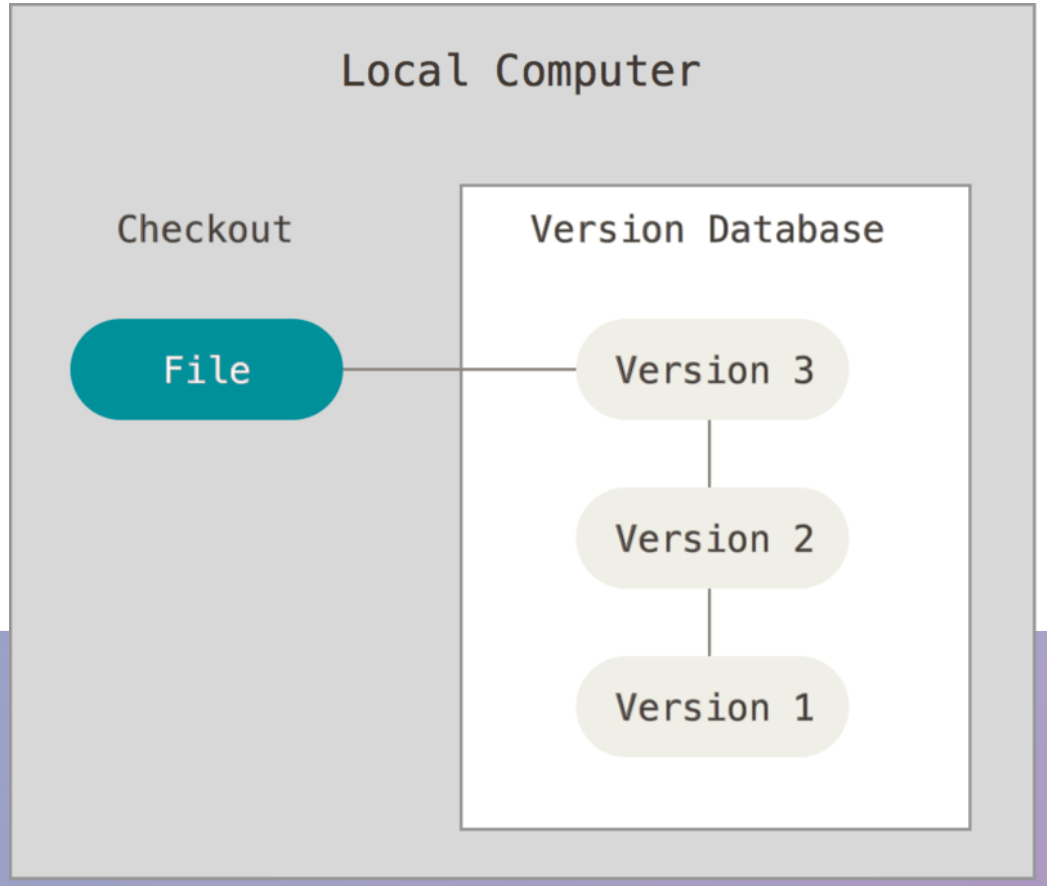
CONTROL DE VERSIONES



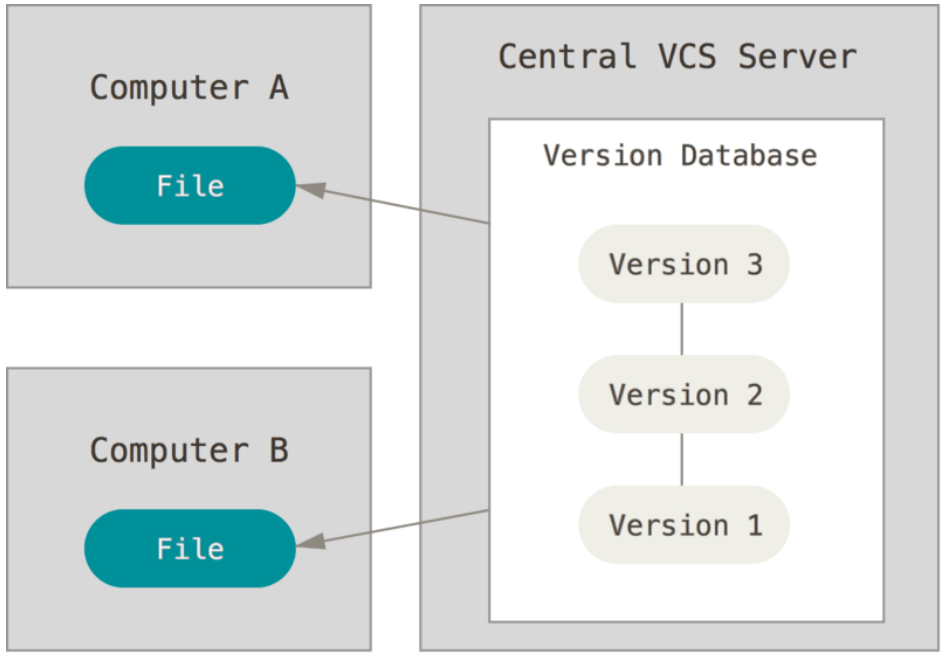


¿POR QUÉ USAR UN VCS?

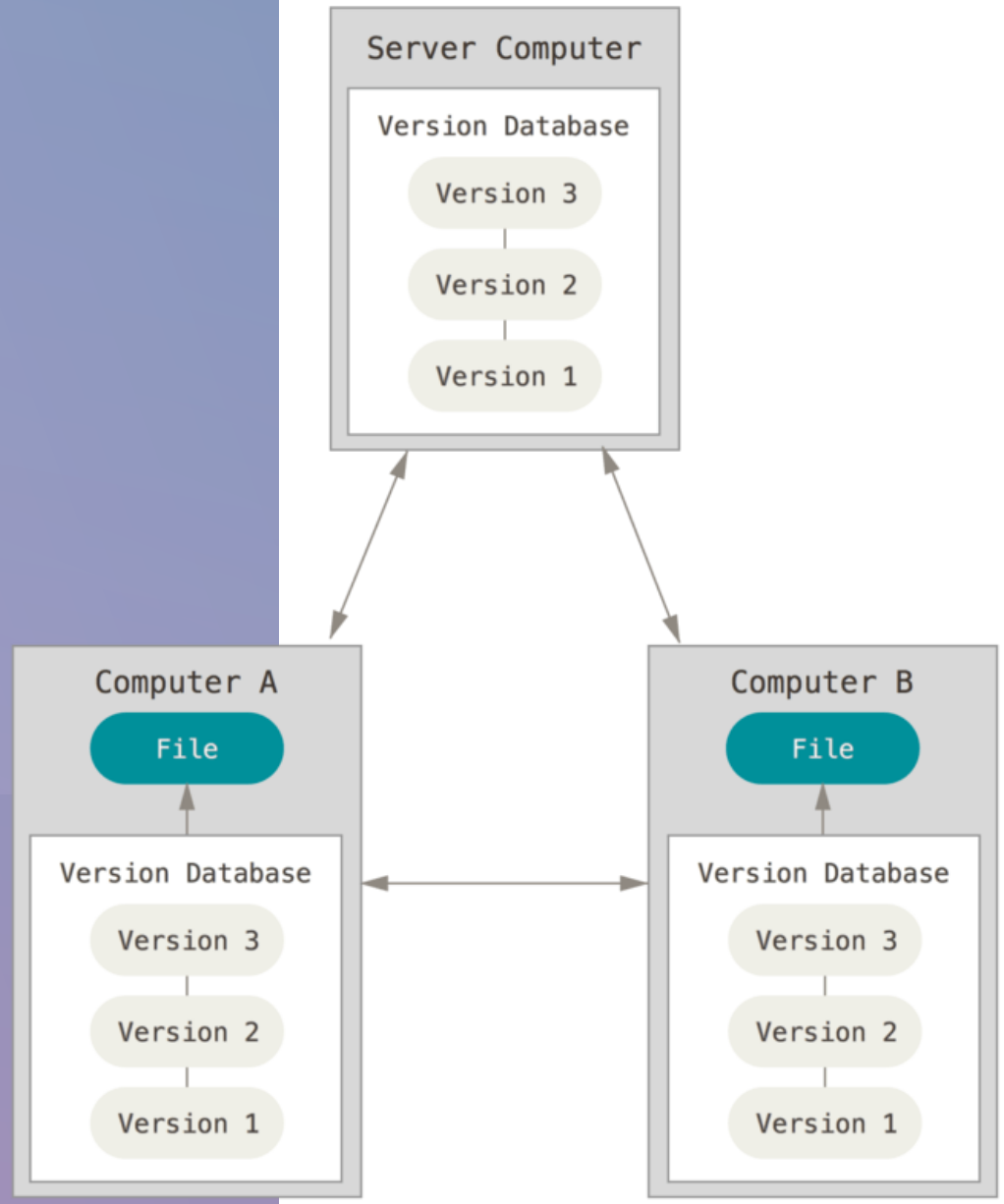
SISTEMAS DE CONTROL DE VERSIONES LOCALES



SISTEMAS DE CONTROL DE VERSIONES CENTRALIZADOS



SISTEMAS DE CONTROL DE VERSIONES DISTRIBUIDOS

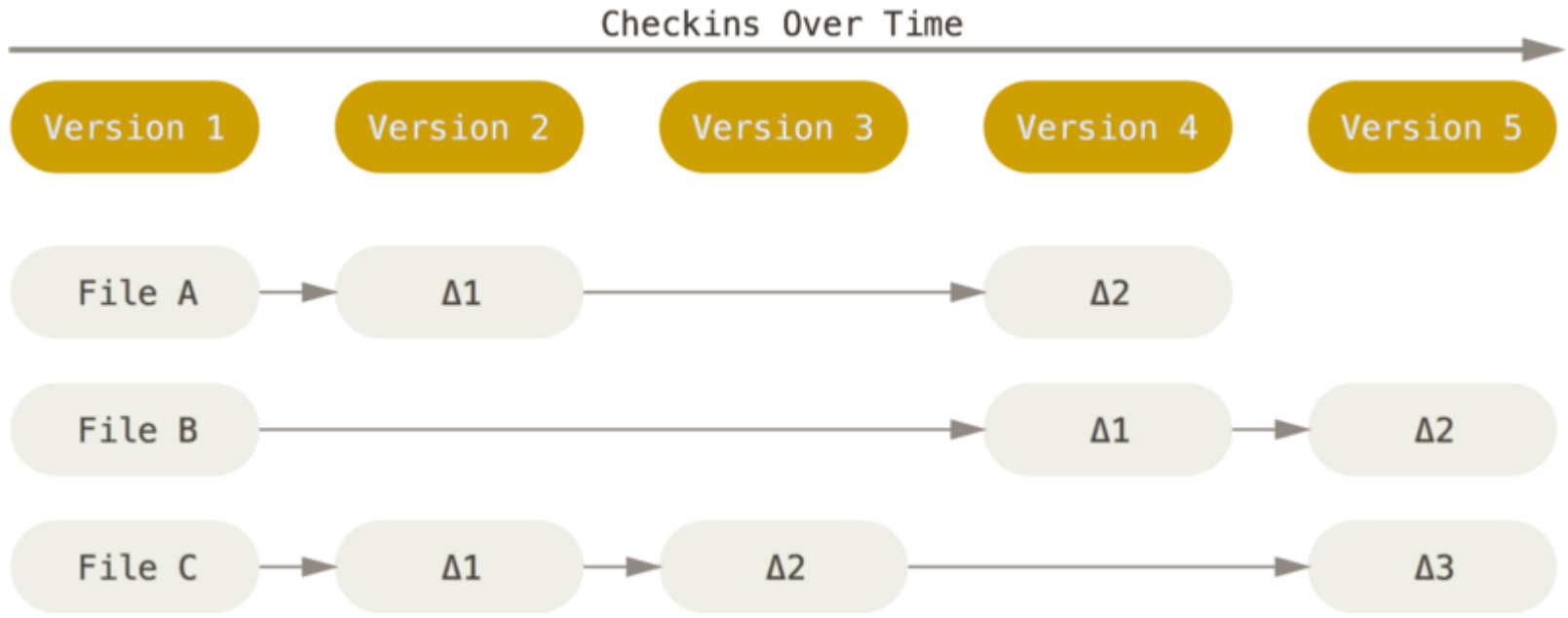


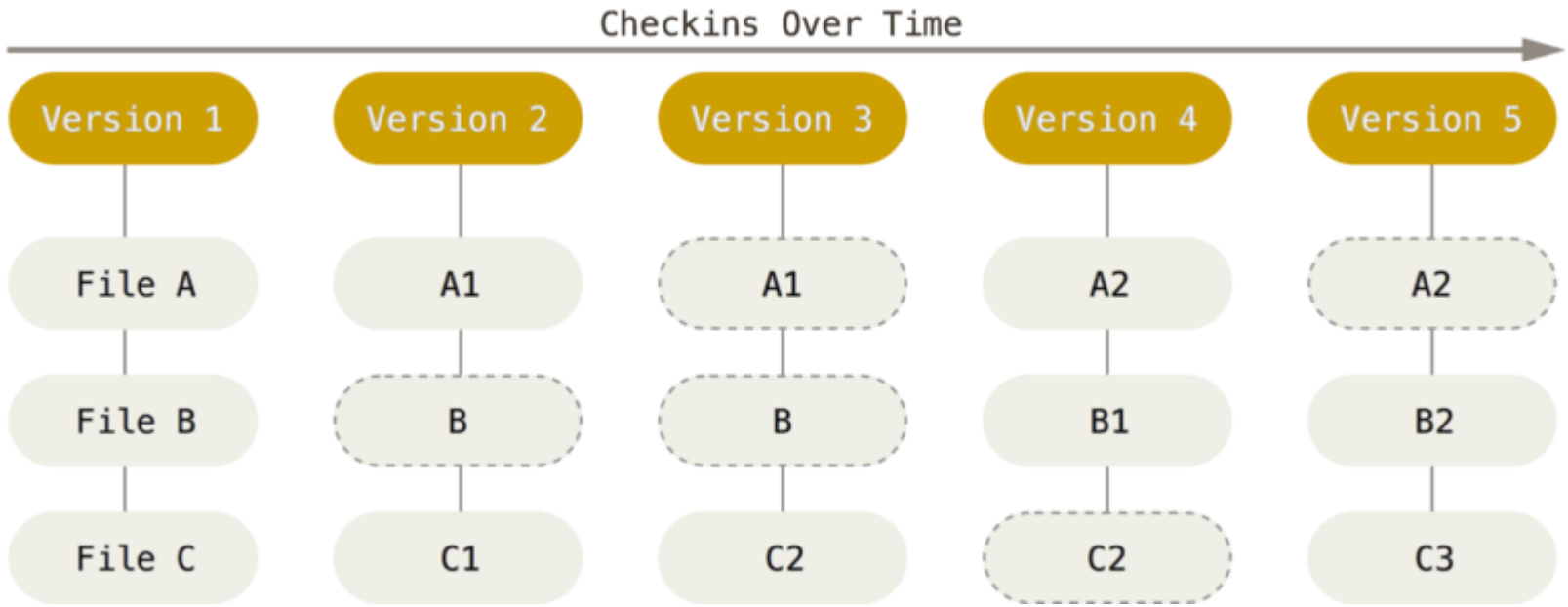
HISTORIA DE GIT

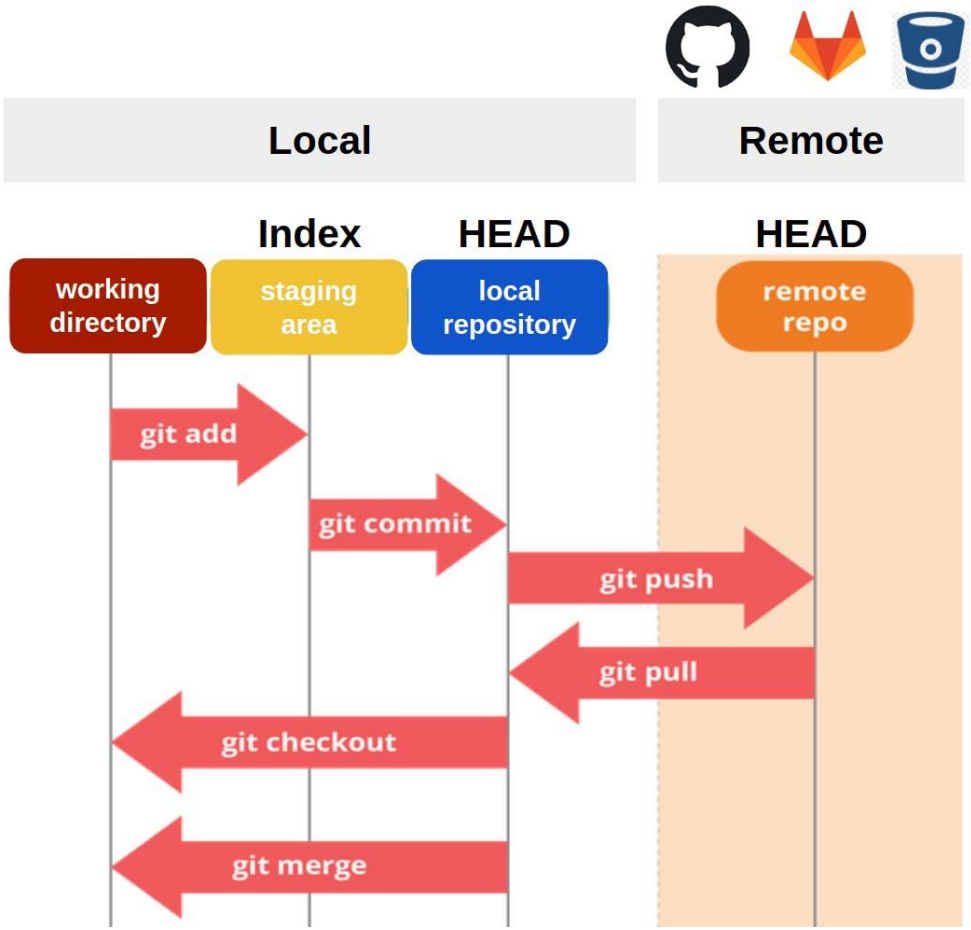
git

¿QUÉ ES GIT?

COPIAS INSTANTÁNEAS, NO DIFERENCIAS

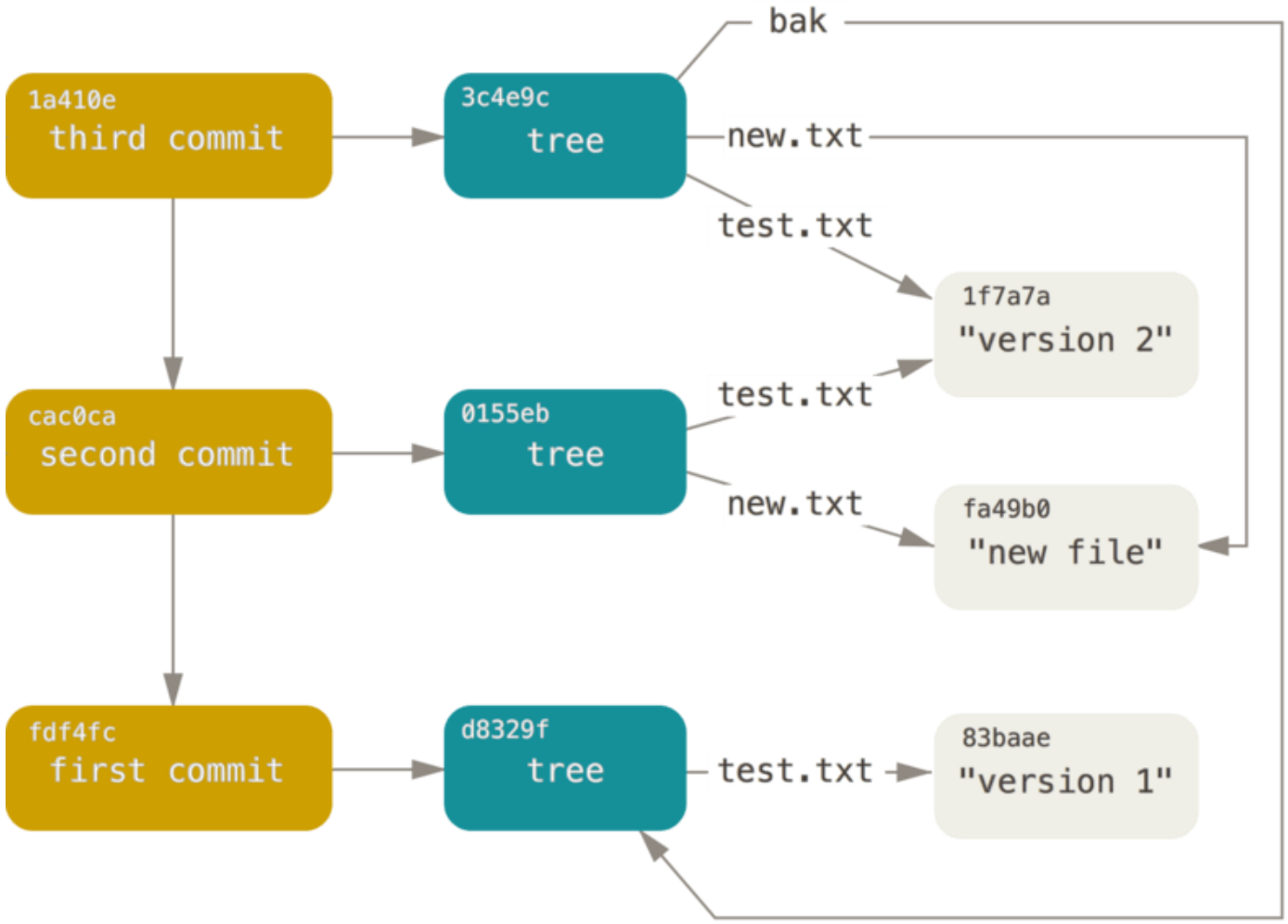




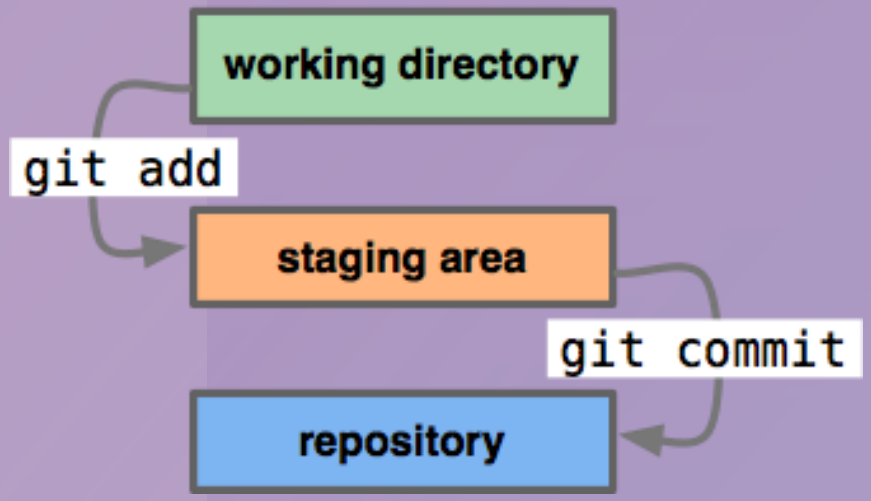


OPERACIONES LOCALES

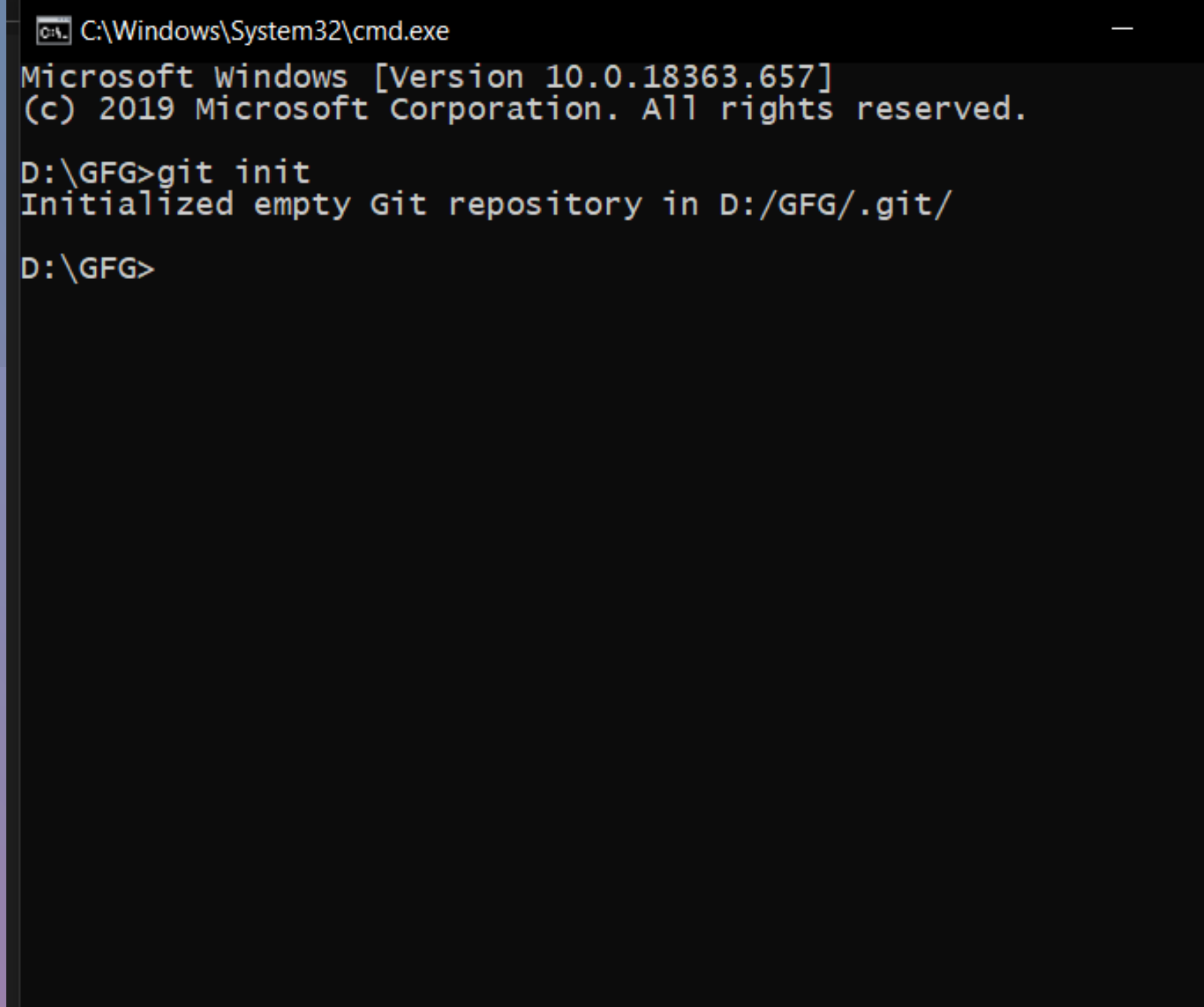
INTEGRIDAD


1a410e
third commit
3c4e9c
tree
bak
new.txt
test.txt
1f7a7a
"version 2"
cac0ca
second commit
0155eb
tree
test.txt
new.txt
fa49b0
"new file"
fdf4fc
first commit
d8329f
tree
test.txt
83baae
"version 1"

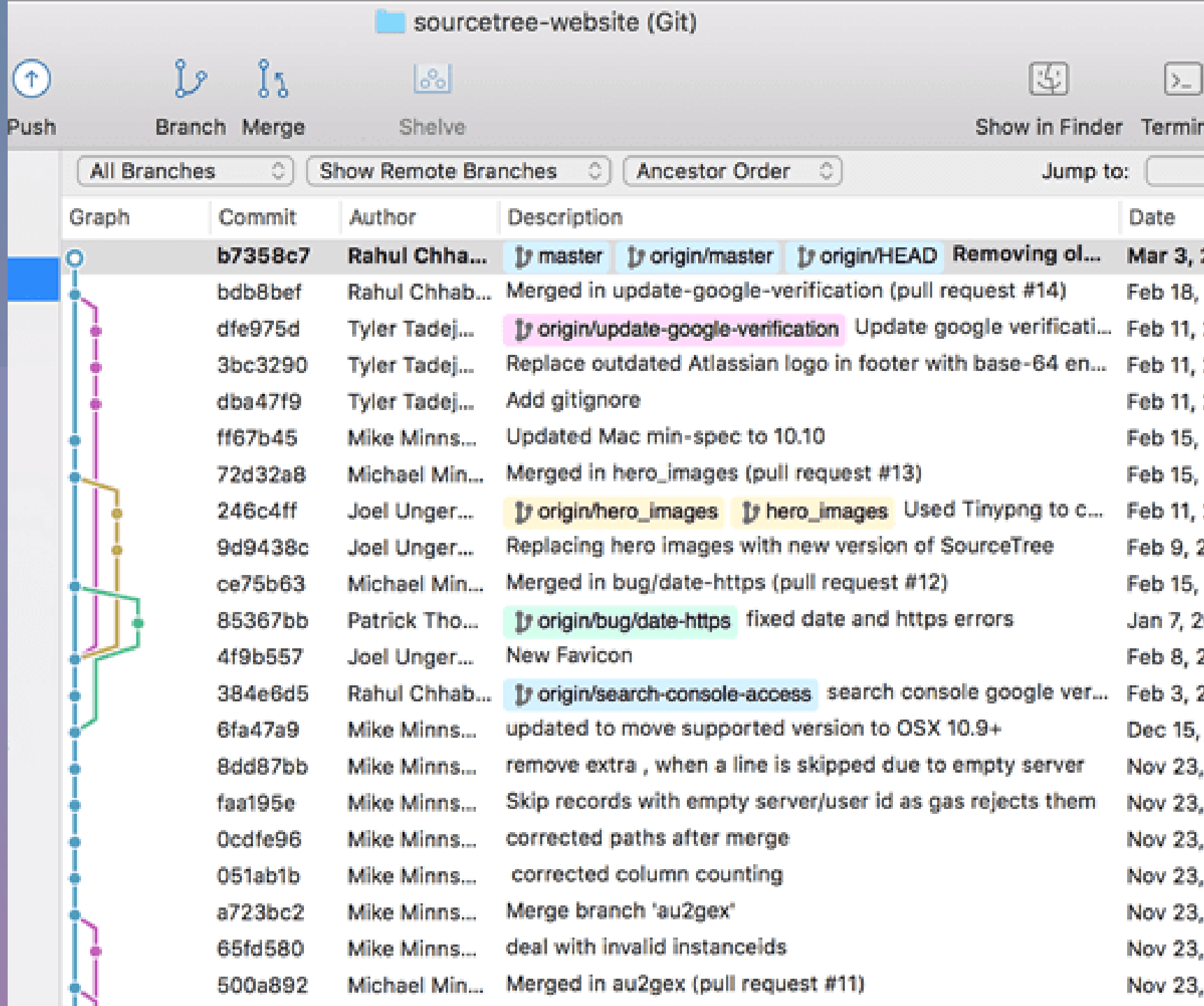
ESTADOS DE GIT



LINEA DE COMANDOS



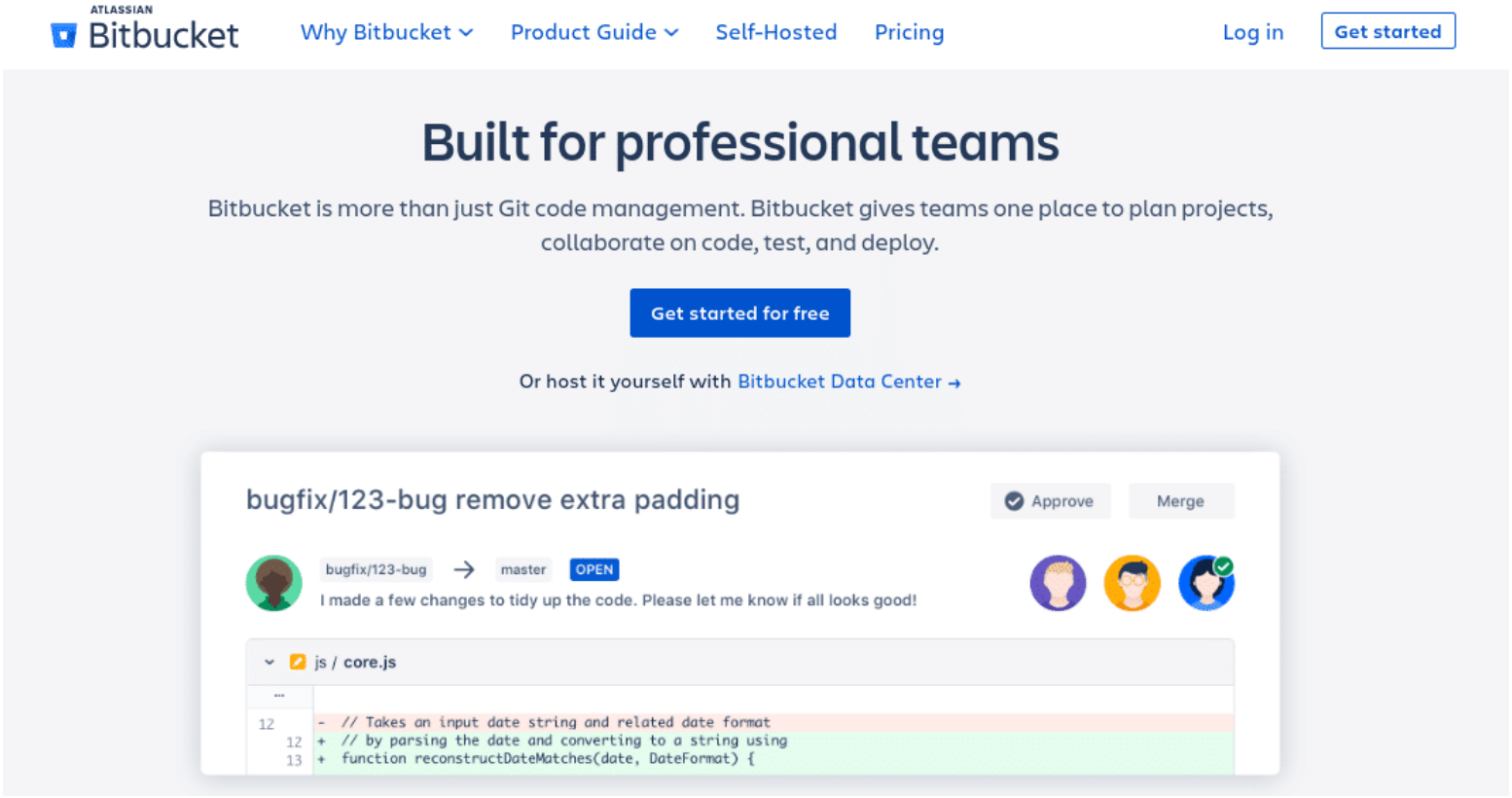
SOURCETREE

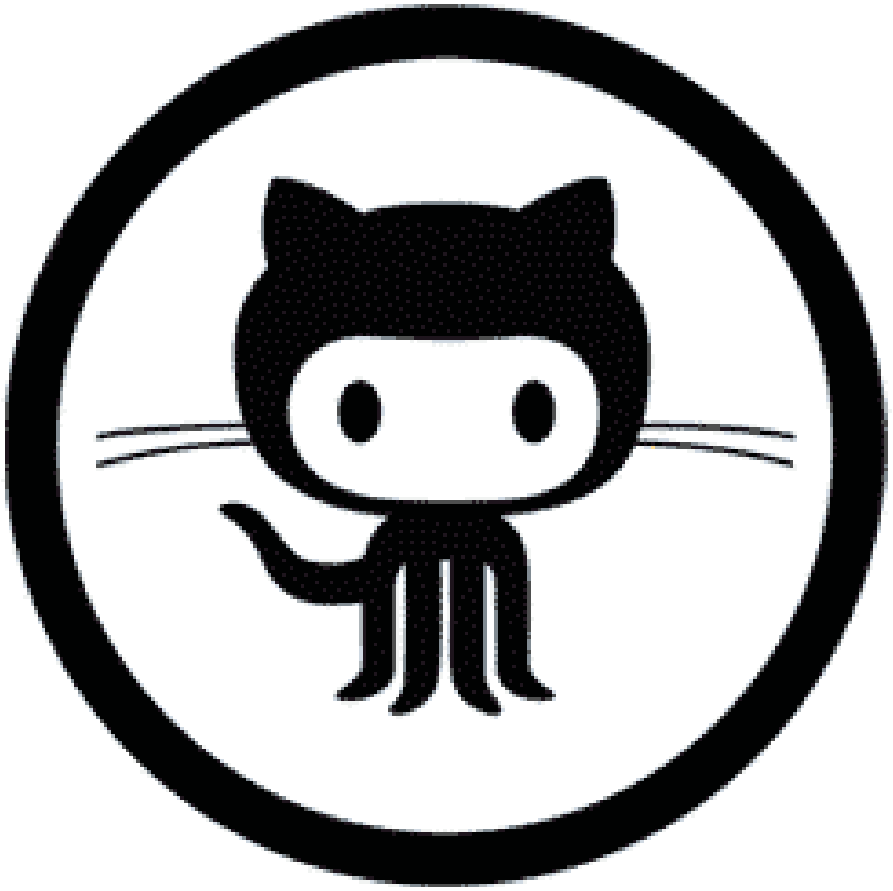

sourcetree-website (Git)
Push
Branch
Merge
Shelve
Show in Finder
All Branches
Show Remote Branches
Ancestor Order
Jump to:
Graph
Commit
Author
Description
Date
b7358c7 Rahul Chha... master origin/master origin/HEAD Removing ol... Mar 3,
bdb8bef Rahul Chhab... Merged in update-google-verification (pull request #14) Feb 18,
dfe975d Tyler Tadej... origin/update-google-verification Update google verificati... Feb 11,
3bc3290 Tyler Tadej... Replace outdated Atlassian logo in footer with base-64 en... Feb 11,
dba47f9 Tyler Tadej... Add gitignore Feb 11,
ff67b45 Mike Minns... Updated Mac min-spec to 10.10 Feb 15,
72d32a8 Michael Min... Merged in hero_images (pull request #13) Feb 15,
246c4ff Joel Unger... origin/hero_images hero_images Used Tinypng to c... Feb 11,
9d9438c Joel Unger... Replacing hero images with new version of SourceTree Feb 9,
ce75b63 Michael Min... Merged in bug/date-https (pull request #12) Feb 15,
85367bb Patrick Tho... origin/bug/date-https fixed date and https errors Jan 7,
4f9b557 Joel Unger... New Favicon Feb 8,
384e6d5 Rahul Chhab... origin/search-console-access search console google ver... Feb 3,
6fa47a9 Mike Minns... updated to move supported version to OSX 10.9+ Dec 15,
8dd87bb Mike Minns... remove extra , when a line is skipped due to empty server Nov 23,
faa195e Mike Minns... Skip records with empty server/user id as gas rejects them Nov 23,
0cdfe96 Mike Minns... corrected paths after merge Nov 23,
051ab1b Mike Minns... corrected column counting Nov 23,
a723bc2 Mike Minns... Merge branch 'au2gex' Nov 23,
65fd580 Mike Minns... deal with invalid instanceids Nov 23,
500a892 Michael Min... Merged in au2gex (pull request #11) Nov 23,

GITHUB

	
1. It is a software	1. It is a service
2. It is installed locally on the system	2. It is hosted on Web
3. It is a command line tool	3. It provides a graphical interface
4. It is a tool to manage different versions of edits, made to files in a git repository	4. It is a space to upload a copy of the **Git** repository
5. It provides functionalities like Version Control System Source Code Management	5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features

GIT VS GITHUB

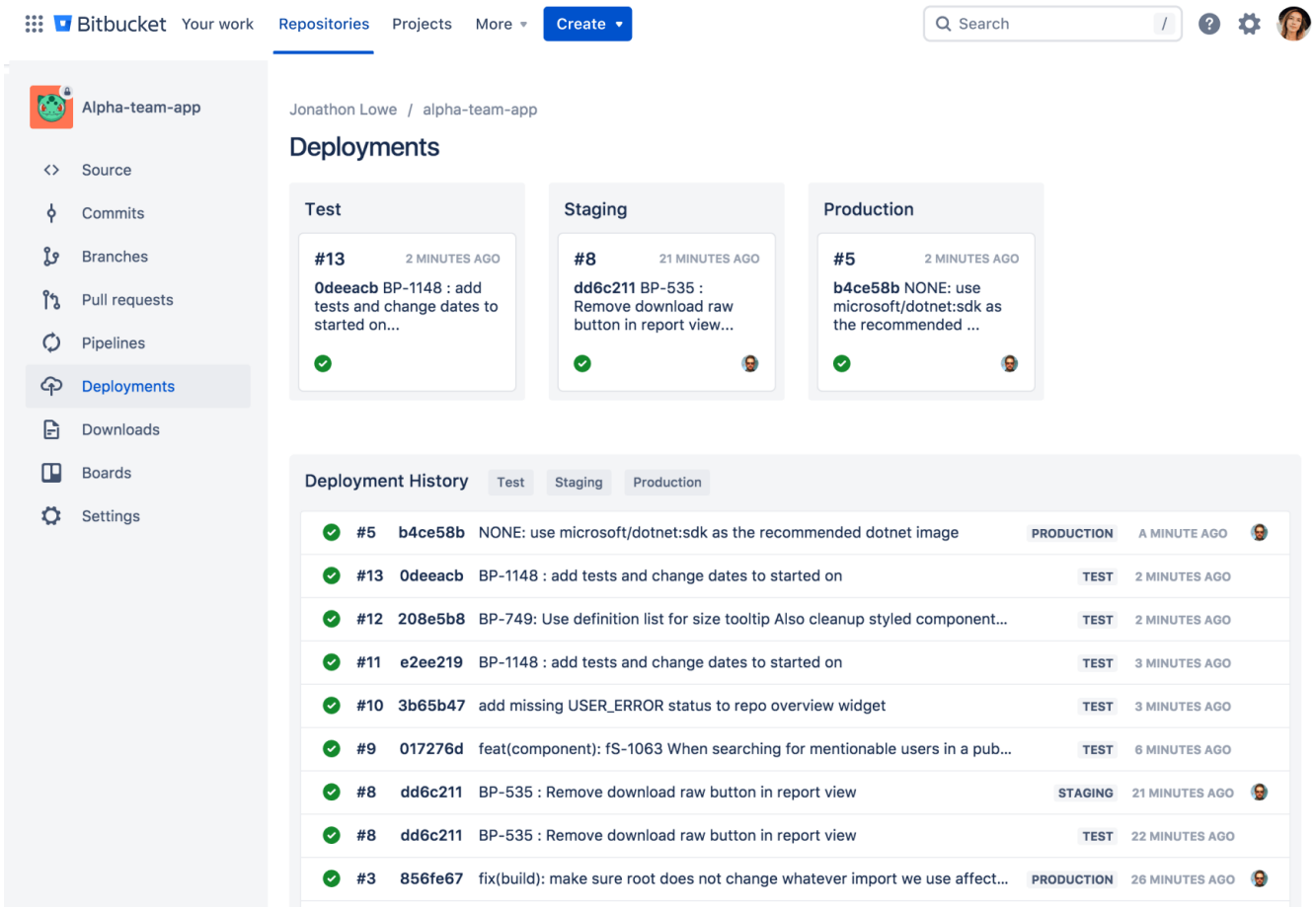

ATLASSIAN
Bitbucket
Why Bitbucket
Product Guide
Self-Hosted
Pricing
Log in
Get started
Built for professional teams
Bitbucket is more than just Git code management. Bitbucket gives teams one place to plan projects, collaborate on code, test, and deploy.
Get started for free
Or host it yourself with Bitbucket Data Center →
bugfix/123-bug remove extra padding
Approve
Merge
bugfix/123-bug
master
OPEN
I made a few changes to tidy up the code. Please let me know if all looks good!
js / core.js
12
- // Takes an input date string and related date format
12 + // by parsing the date and converting to a string using
13 + function reconstructDateMatches(date, DateFormat) {



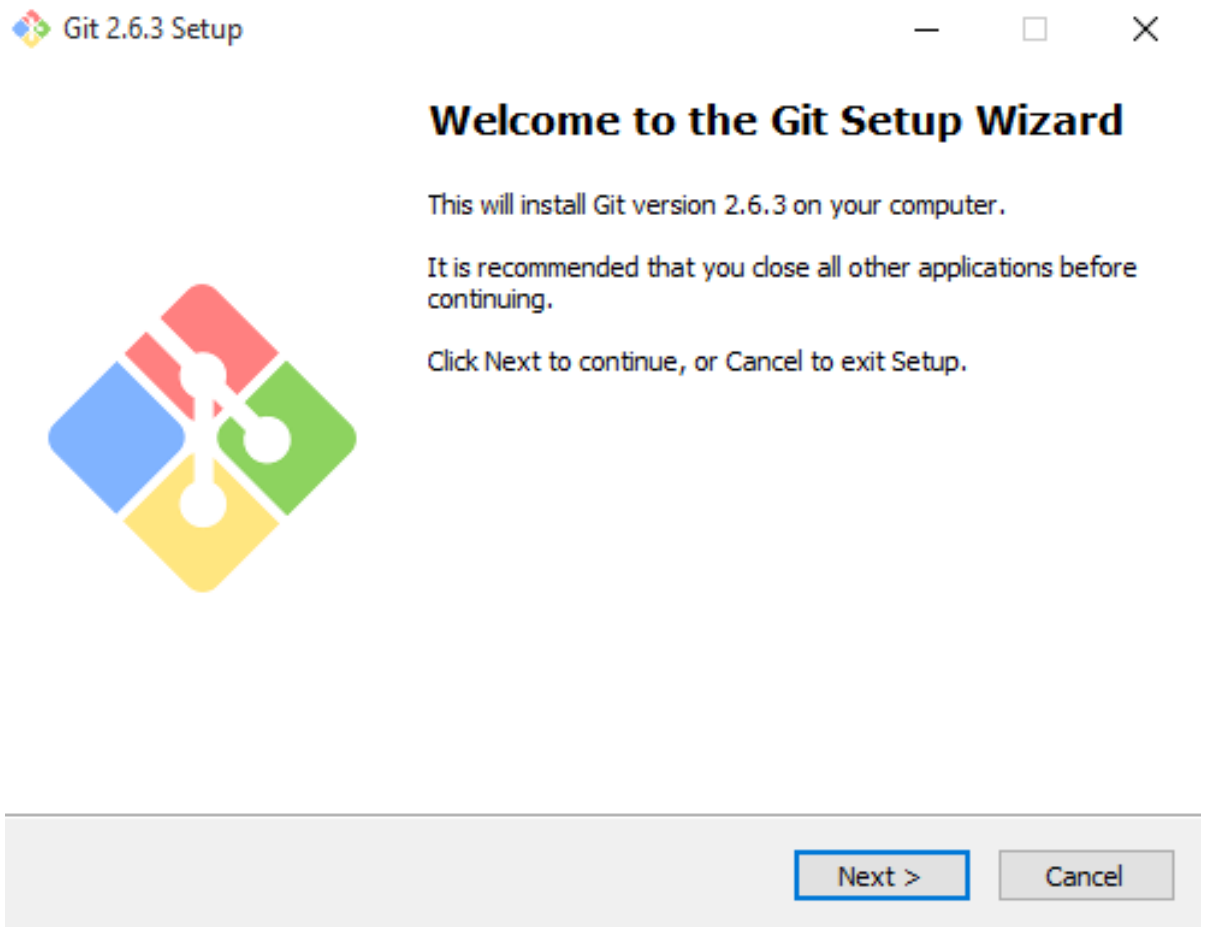
VS

Github

Bitbucket

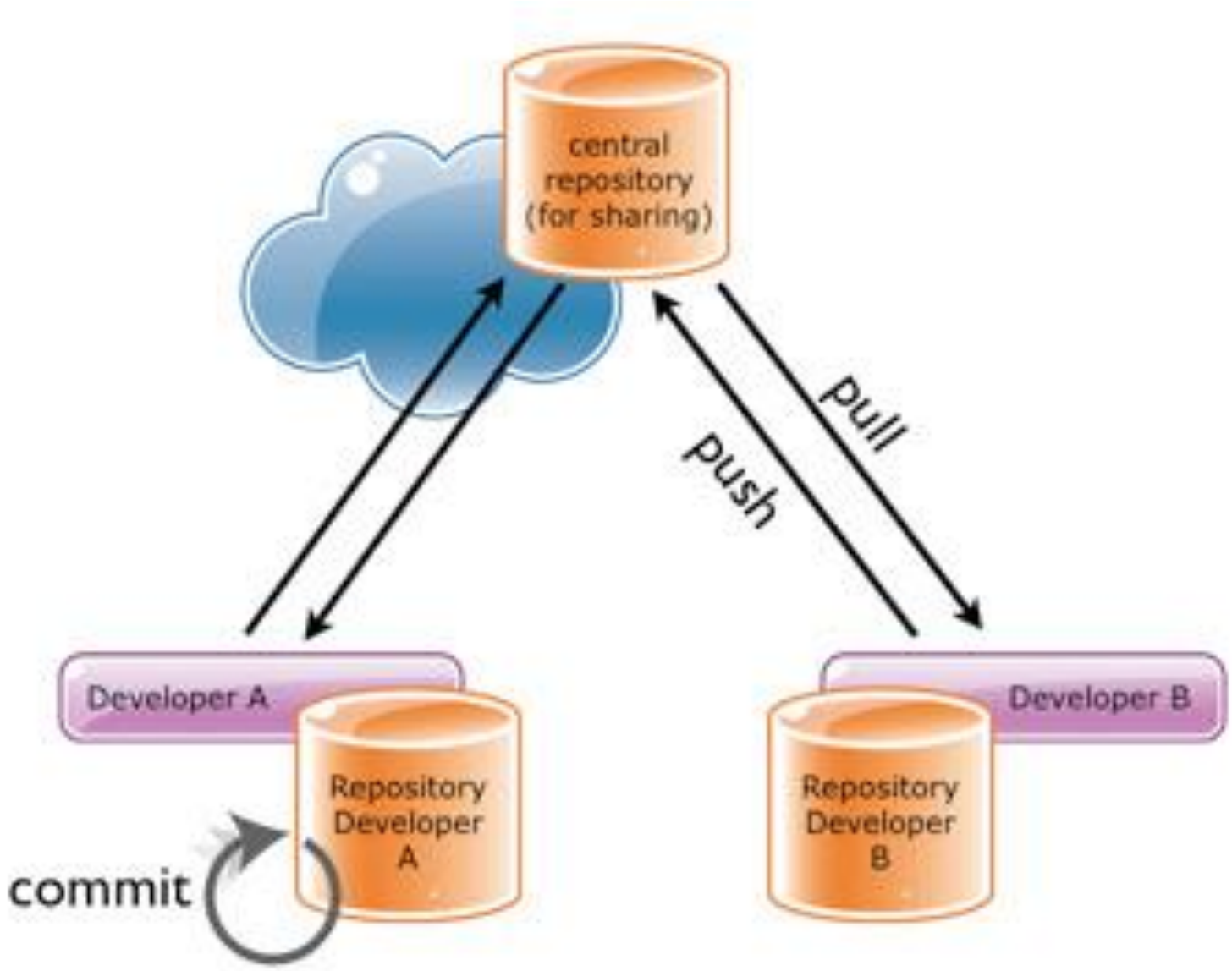


CREAR UNA CUENTA DE BITBUCKET



INSTALACIÓN DE GIT EN WINDOWS

REPOSITORIO DE GIT



Git Cheat Sheet

Setup

Set the name and email that will be attached to your commits and tags

```
$ git config --global user.name "Danny Adams"
$ git config --global user.email "my-email@gmail.com"
```

Start a Project

Create a local repo (omit <directory> to initialise the current directory as a git repo

```
$ git init <directory>
```

Download a remote repo

```
$ git clone <url>
```

Make a Change

Add a file to staging

```
$ git add <file>
```

Stage all files

```
$ git add .
```

Commit all staged files to git

```
$ git commit -m "commit message"
```

Add all changes made to tracked files & commit

```
$ git commit -am "commit message"
```

Basic Concepts

main: default development branch
origin: default upstream repo
HEAD: current branch
HEAD^: parent of HEAD
HEAD~4: great-great grandparent of HEAD

By @DoableDanny

Branches

List all local branches. Add -r flag to show all remote branches. -a flag for all branches.

```
$ git branch
```

Create a new branch

```
$ git branch <new-branch>
```

Switch to a branch & update the working directory

```
$ git checkout <branch>
```

Create a new branch and switch to it

```
$ git checkout -b <new-branch>
```

Delete a merged branch

```
$ git branch -d <branch>
```

Delete a branch, whether merged or not

```
$ git branch -D <branch>
```

Add a tag to current commit (often used for new version releases)

```
$ git tag <tag-name>
```

Merging

Merge branch a into branch b. Add --no-ff option for no-fast-forward merge


```
$ git checkout b
$ git merge a
```

Merge & squash all commits into one new commit

```
$ git merge --squash a
```

Rebasing

Rebase feature branch onto main (to incorporate new changes made to main). Prevents unnecessary merge commits into feature, keeping history clean


```
$ git checkout feature
$ git rebase main
```

Interatively clean up a branches commits before rebasing onto main

```
$ git rebase -i main
```

Interatively rebase the last 3 commits on current branch

```
$ git rebase -i Head~3
```

Undoing Things

Move (&/or rename) a file & stage move

```
$ git mv <existing_path> <new_path>
```

Remove a file from working directory & staging area, then stage the removal

```
$ git rm <file>
```

Remove from staging area only

```
$ git rm --cached <file>
```

View a previous commit (READ only)

```
$ git checkout <commit_ID>
```

Create a new commit, reverting the changes from a specified commit

```
$ git revert <commit_ID>
```

Go back to a previous commit & delete all commits ahead of it (revert is safer). Add --hard flag to also delete workspace changes (BE VERY CAREFUL)

```
$ git reset <commit_ID>
```

Review your Repo

List new or modified files not yet committed

```
$ git status
```

List commit history, with respective IDs

```
$ git log --oneline
```

Show changes to unstaged files. For changes to staged files, add --cached option

```
$ git diff
```

Show changes between two commits

```
$ git diff commit1_ID commit2_ID
```

Stashing

Store modified & staged changes. To include untracked files, add -u flag. For untracked & ignored files, add -a flag.

```
$ git stash
```

As above, but add a comment.

```
$ git stash save "comment"
```

Partial stash. Stash just a single file, a collection of files, or individual changes from within files

```
$ git stash -p
```

List all stashes

```
$ git stash list
```

Re-apply the stash without deleting it

```
$ git stash apply
```

Re-apply the stash at index 2, then delete it from the stash list. Omit stash@{n} to pop the most recent stash.

```
$ git stash pop stash@{2}
```

Show the diff summary of stash 1. Pass the -p flag to see the full diff.

```
$ git stash show stash@{1}
```

Delete stash at index 1. Omit stash@{n} to delete last stash made

```
$ git stash drop stash@{1}
```

Delete all stashes

```
$ git stash clear
```

Synchronizing

Add a remote repo

```
$ git remote add <alias> <url>
```

View all remote connections. Add -v flag to view urls.

```
$ git remote
```

Remove a connection

```
$ git remote remove <alias>
```

Rename a connection

```
$ git remote rename <old> <new>
```

Fetch all branches from remote repo (no merge)

```
$ git fetch <alias>
```

Fetch a specific branch

```
$ git fetch <alias> <branch>
```

Fetch the remote repo's copy of the current branch, then merge

```
$ git pull
```

Move (rebase) your local changes onto the top of new changes made to the remote repo (for clean, linear history)

```
$ git pull --rebase <alias>
```

Upload local content to remote repo

```
$ git push <alias>
```

Upload to a branch (can then pull request)

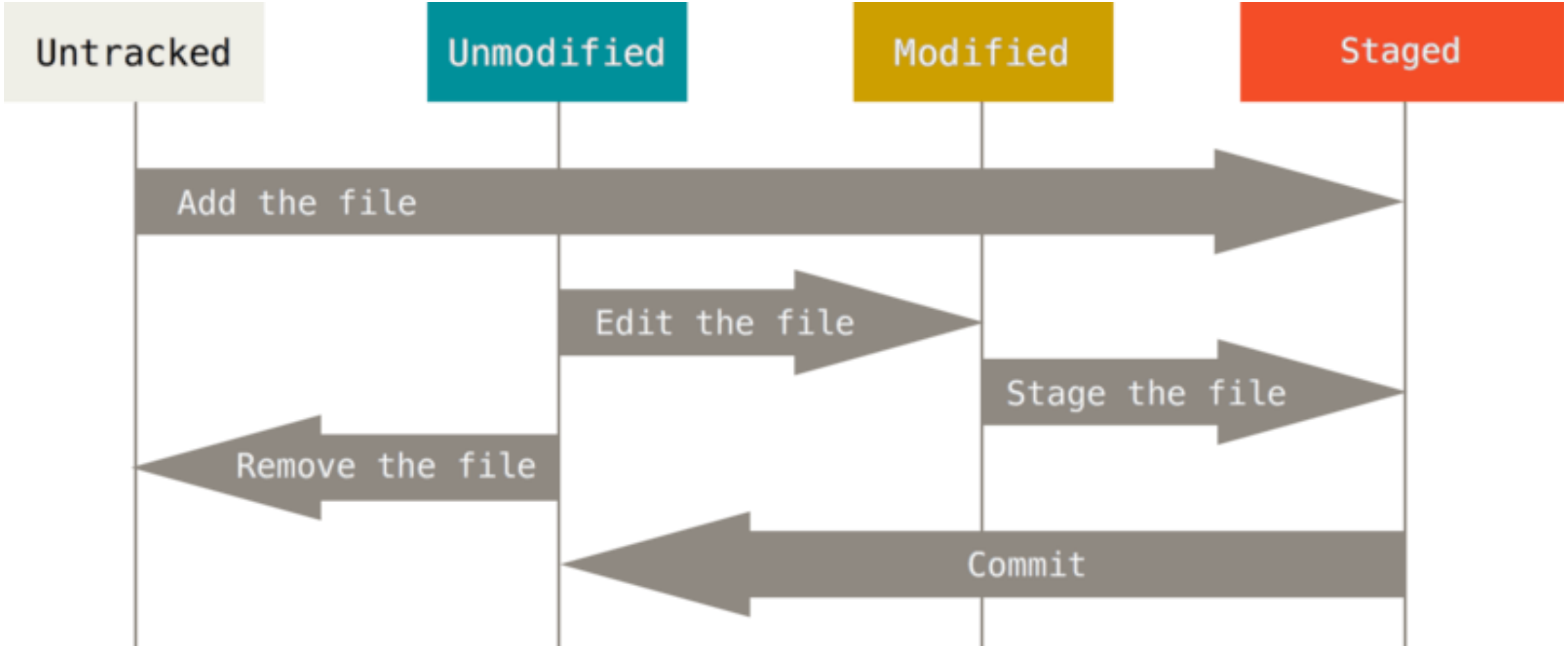
```
$ git push <alias> <branch>
```

INICIALIZAR REPOSITORIO

```
> git init
```

CLONACIÓN

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop (master)
$ cd "new folder"

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/new folder (master)
$ git clone https://github.com/ImDwivedi1/Git-Example.git
Cloning into 'Git-Example'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/new folder (master)
$
```



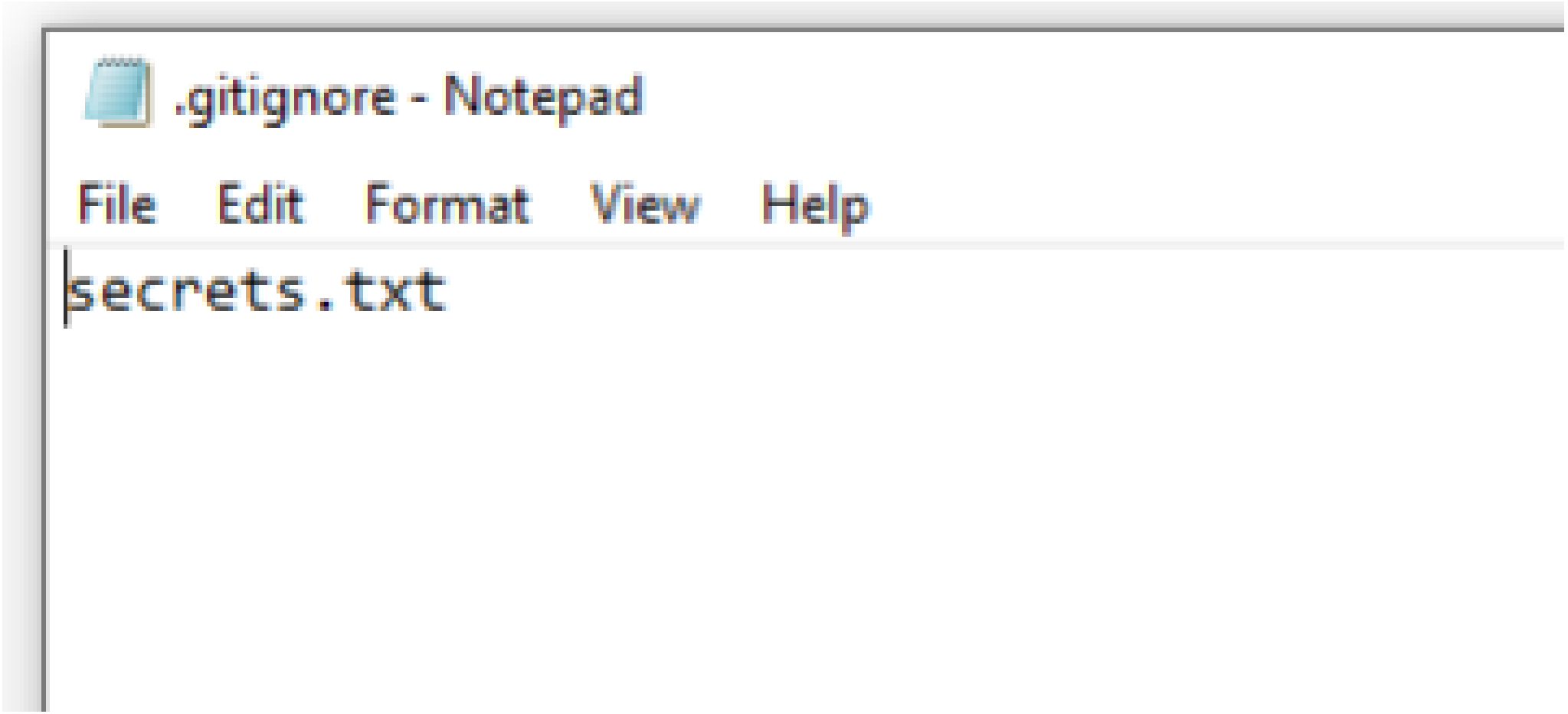
GUARDANDO CAMBIOS EN EL REPOSITORIO

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ touch demofile

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        demofile

nothing added to commit but untracked files present (use "git add" to
track)
```

GIT STATUS

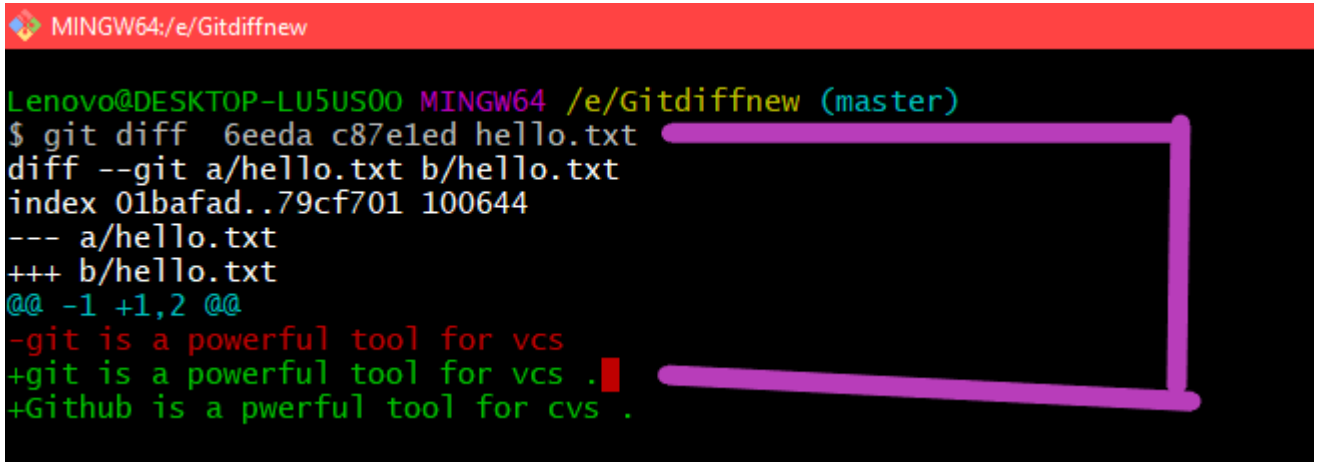
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ touch newfile.txt

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committ
        newfile.txt
```

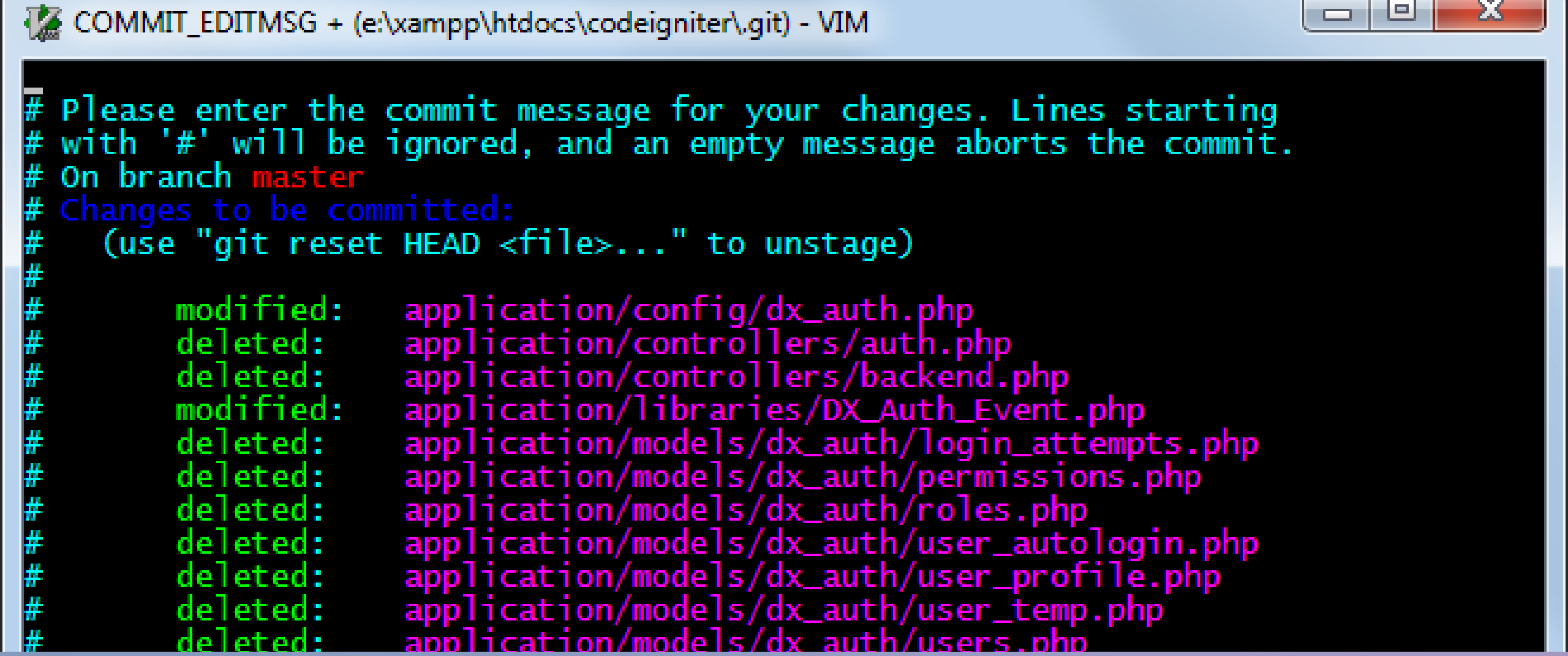
GIT ADD



GIT IGNORE

GIT DIFF


MINGW64:/e/Gitdiffnew
Lenovo@DESKTOP-LU5US0O MINGW64 /e/Gitdiffnew (master)
$ git diff 6eeda c87e1ed hello.txt
diff --git a/hello.txt b/hello.txt
index 01bafad..79cf701 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1 +1,2 @@
-git is a powerful tool for vcs
+git is a powerful tool for vcs .
+Github is a pwerful tool for cvs .


COMMIT_EDITMSG + (e:\xampp\htdocs\codeigniter\.git) - VIM
Please enter the commit message for your changes. Lines starting
with '#' will be ignored, and an empty message aborts the commit.
On branch master
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)
#
modified: application/config/dx_auth.php
deleted: application/controllers/auth.php
deleted: application/controllers/backend.php
modified: application/libraries/DX_Auth_Event.php
deleted: application/models/dx_auth/login_attempts.php
deleted: application/models/dx_auth/permissions.php
deleted: application/models/dx_auth/roles.php
deleted: application/models/dx_auth/user_autologin.php
deleted: application/models/dx_auth/user_profile.php
deleted: application/models/dx_auth/user_temp.php
deleted: application/models/dx_auth/users.php

GIT COMMIT

OPCIONES

Confirma la instantánea preparada

```
git commit
```

Confirma una instantánea de todos los cambios del directorio de trabajo.

```
git commit -a
```

Si se usa la opción **-m**, se omitirá la solicitud de editor de texto a favor de un mensaje insertado.

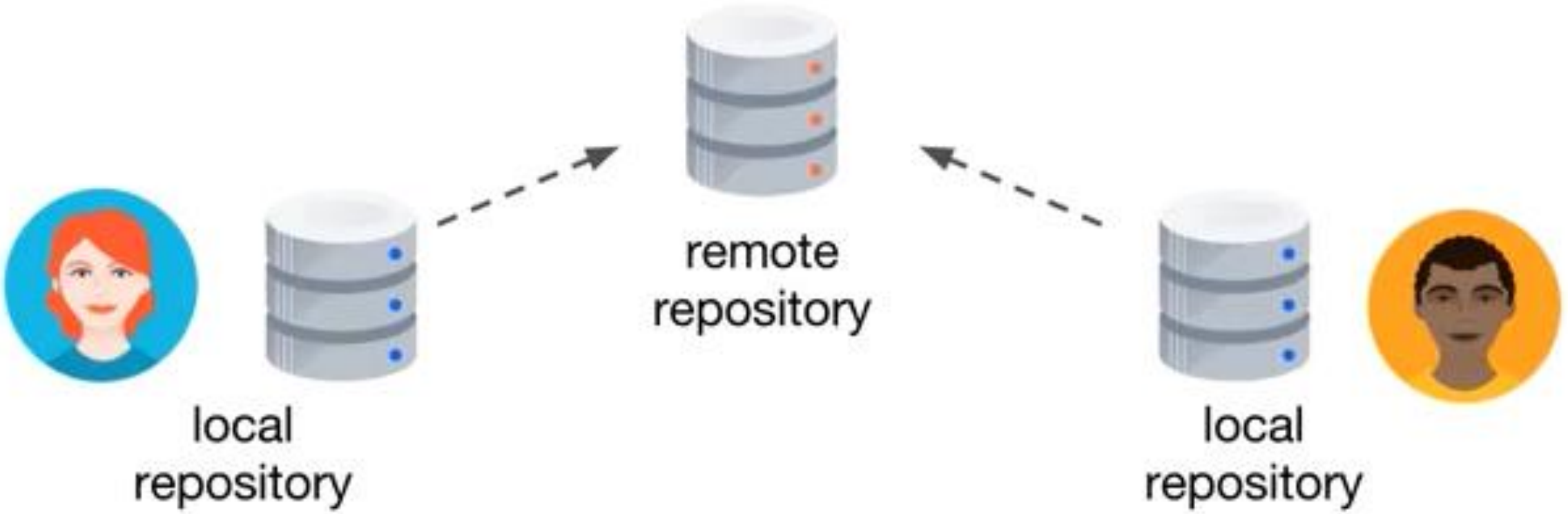
```
git commit -m "commit message"
```

Esta combinación crea inmediatamente una confirmación de todos los cambios preparados y aplica un mensaje de confirmación insertado.

```
git commit -am "commit message"
```

Al pasar esta opción, se modificará la última confirmación. En vez de crear una nueva confirmación, los cambios preparados se añadirán a la confirmación anterior

```
git commit --amend
```



REPOSITORIOS REMOTOS

```
$ cd grit
$ git remote -v
bakkdoor  https://github.com/bakkdoor/grit (fetch)
bakkdoor  https://github.com/bakkdoor/grit (push)
cho45     https://github.com/cho45/grit (fetch)
cho45     https://github.com/cho45/grit (push)
defunkt   https://github.com/defunkt/grit (fetch)
defunkt   https://github.com/defunkt/grit (push)
koke      git://github.com/koke/grit.git (fetch)
koke      git://github.com/koke/grit.git (push)
origin    git@github.com:mojombo/grit.git (fetch)
origin    git@github.com:mojombo/grit.git (push)
```

GIT REMOTE

```
Welcome@Welcome-PC MINGW64 /e/git-demos/pull-tst (master)
$ git remote add my_bootstrap https://github.com/git-test-jaz/bootstrap.git

Welcome@Welcome-PC MINGW64 /e/git-demos/pull-tst (master)
$ git remote -v
my_bootstrap    https://github.com/git-test-jaz/bootstrap.git (fetch)
my_bootstrap    https://github.com/git-test-jaz/bootstrap.git (push)
origin  https://github.com/git-test-jaz/tst-pull-2.git (fetch)
origin  https://github.com/git-test-jaz/tst-pull-2.git (push)

Welcome@Welcome-PC MINGW64 /e/git-demos/pull-tst (master)
$
```

GIT REMOTE ADD

```
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/schacon/ticgit
  Push  URL: https://github.com/schacon/ticgit
  HEAD branch: master
  Remote branches:
    master                               tracked
    dev-branch                           tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)
```

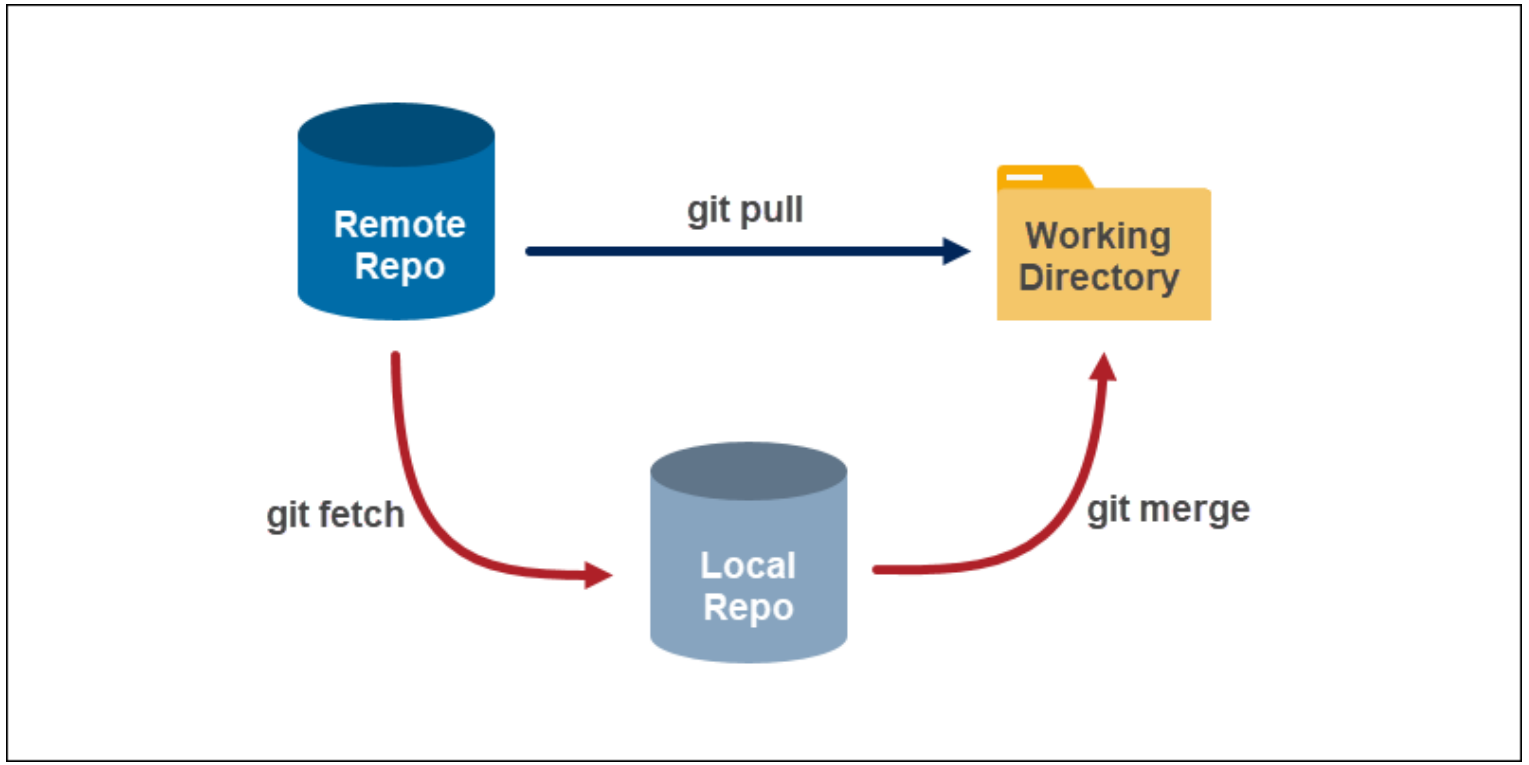
GIT REMOTE SHOW

MODIFICACIONES SOBRE CONEXIÓN

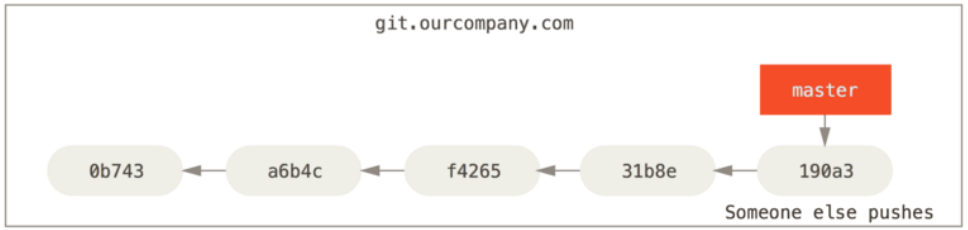
```
git remote rm <name>
```

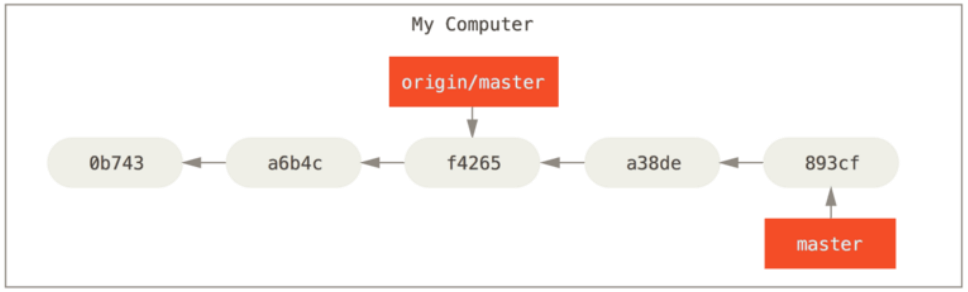
```
git remote rename <old-name> <new-name>
```

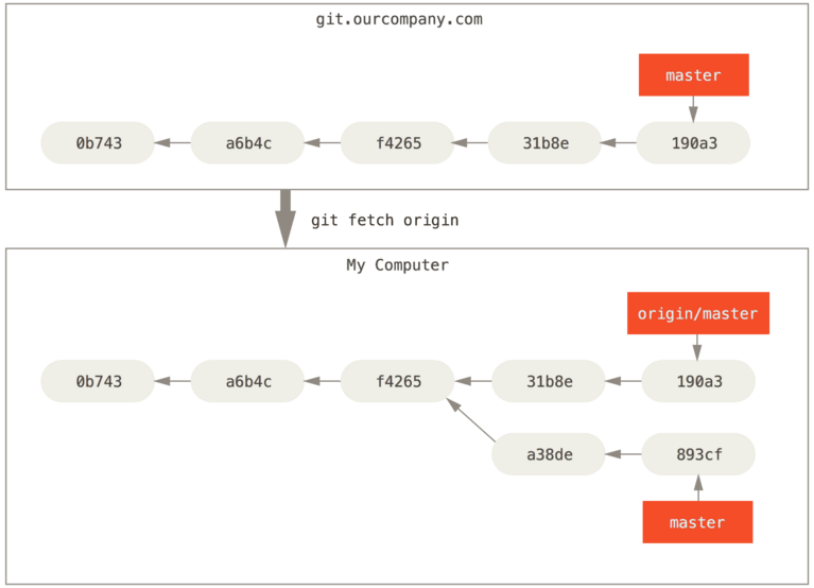
GIT FETCH


Remote Repo
git pull
Working Directory
git fetch
Local Repo
git merge

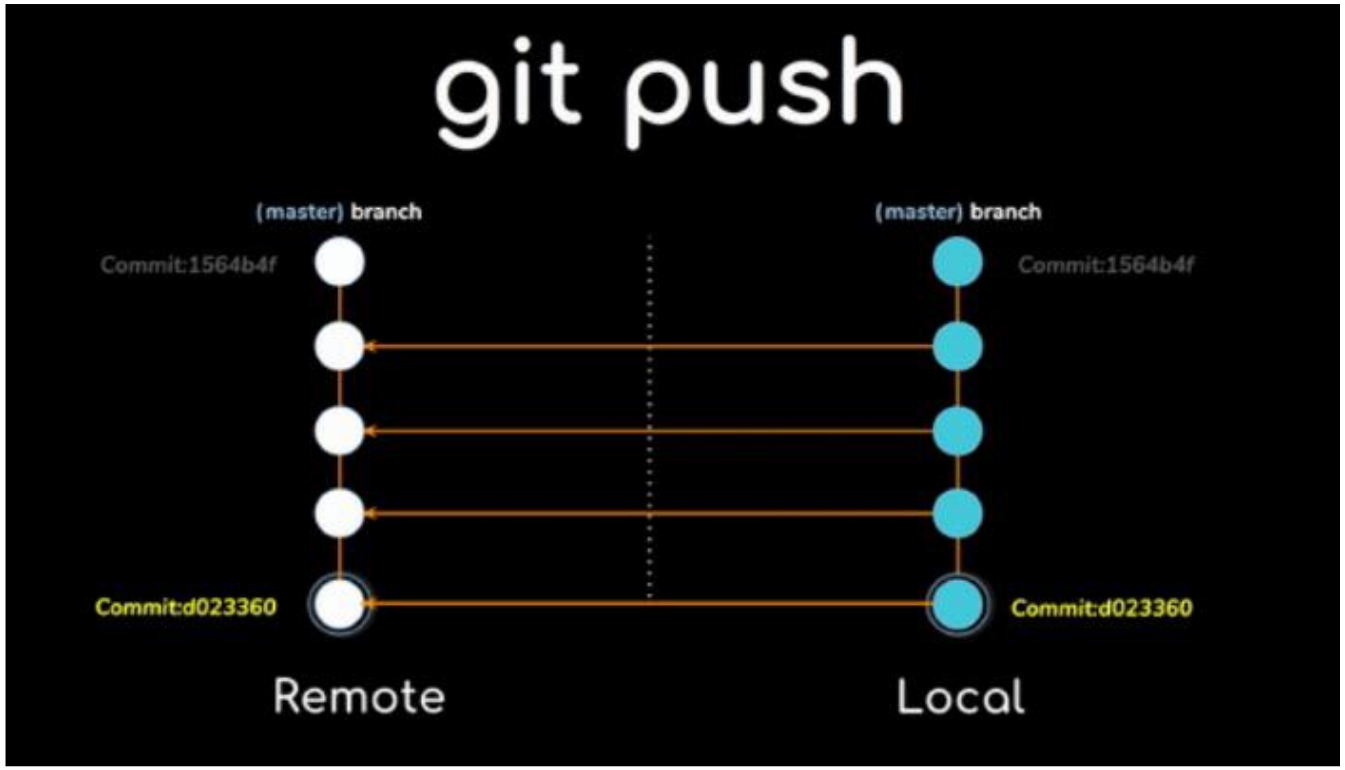
GIT FETCH


git.ourcompany.com
master
0b743
a6b4c
f4265
31b8e
190a3
Someone else pushes

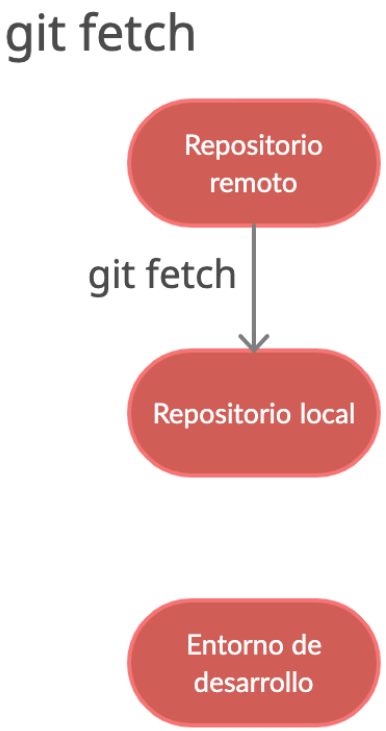

My Computer
origin/master
0b743
a6b4c
f4265
a38de
893cf
master

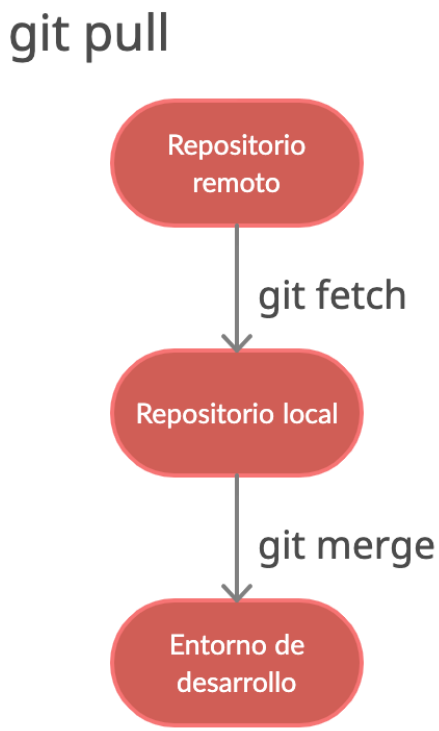

git.ourcompany.com
master
0b743
a6b4c
f4265
31b8e
190a3
git fetch origin
My Computer
origin/master
0b743
a6b4c
f4265
31b8e
190a3
a38de
893cf
master

GIT PUSH


git push
(master) branch
Commit:1564b4f
(master) branch
Commit:1564b4f
Commit:d023360
Commit:d023360
Remote
Local

GIT PULL

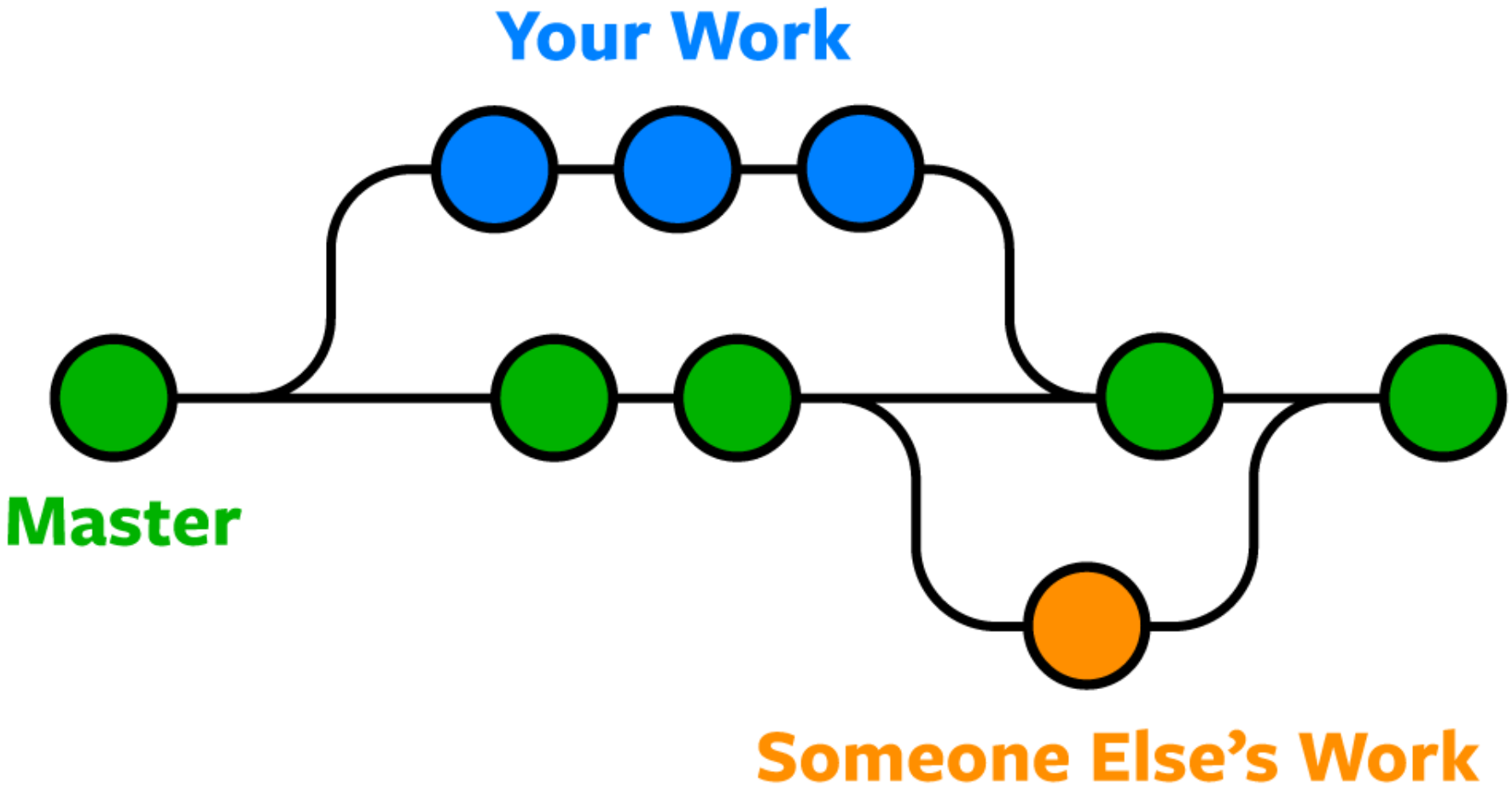

git fetch
Repositorio remoto
git fetch
Repositorio local
Entorno de desarrollo


git pull
Repositorio remoto
git fetch
Repositorio local
git merge
Entorno de desarrollo

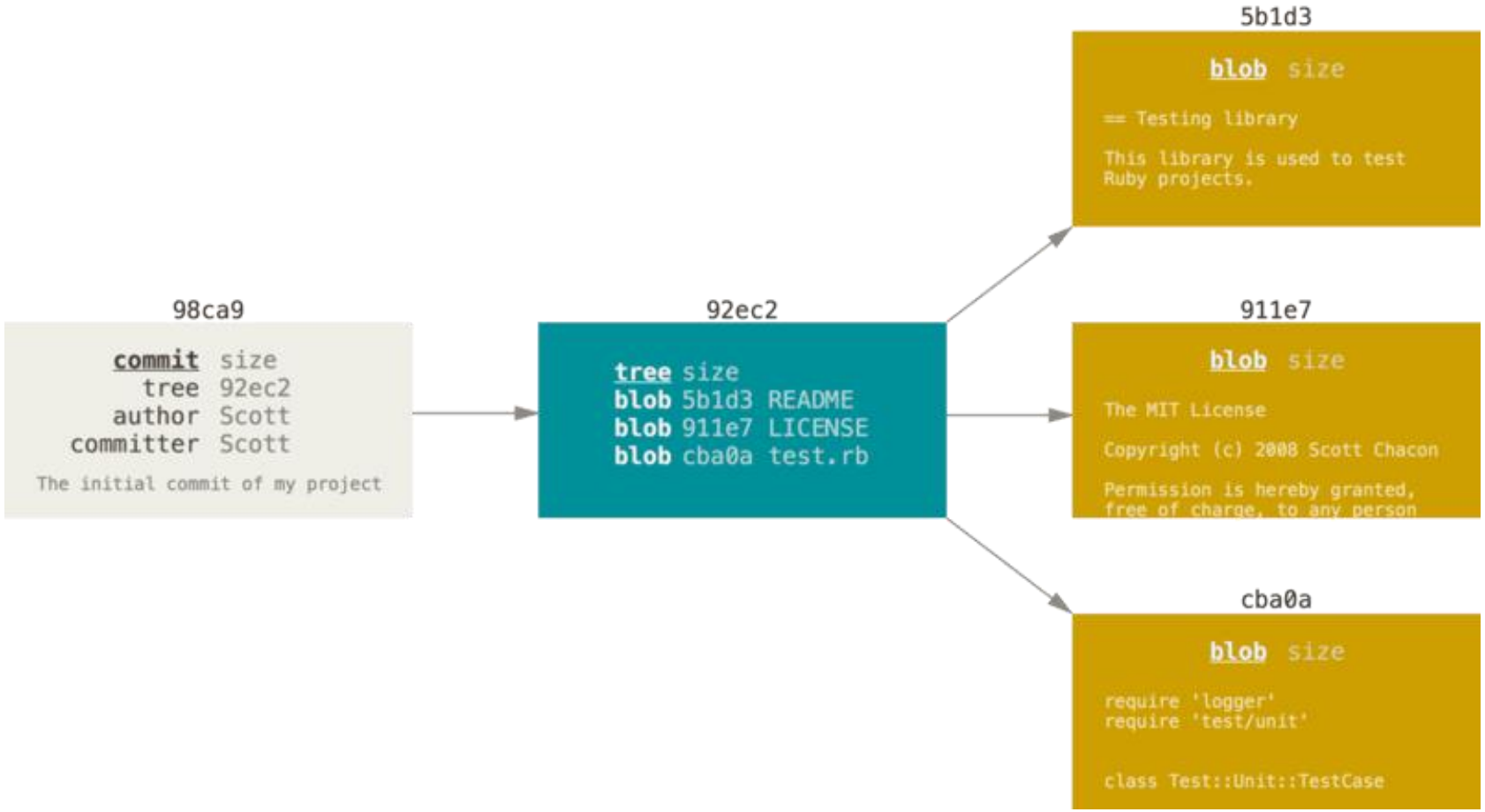
ELIMINAR RAMAS REMOTAS

```
git push origin --delete fix/authentication
```

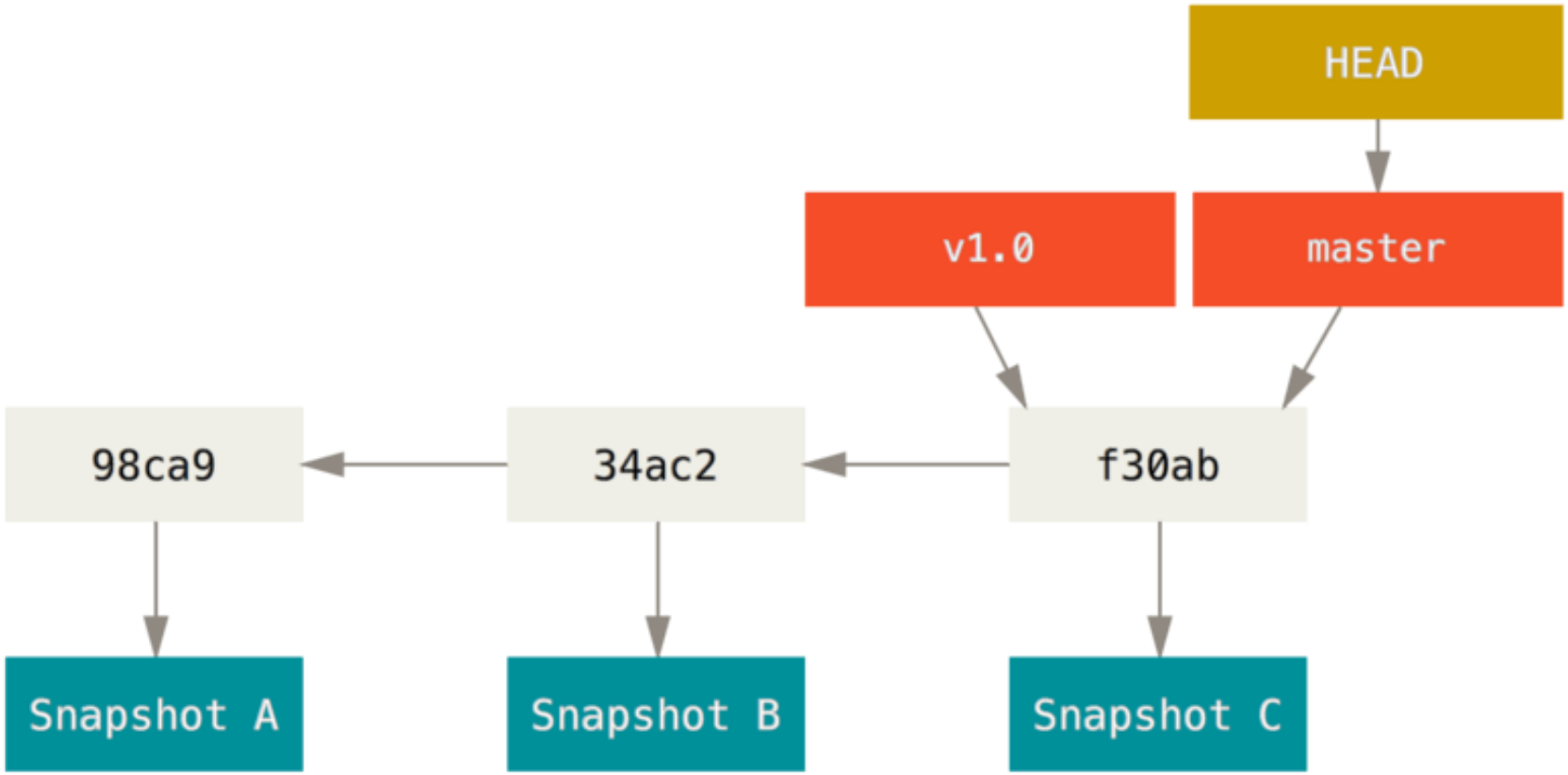

LABORATORIO 1
v2.6.0-beta.2
build: release 2.6.0-beta.2
build: build 2.6.0-beta.2
feat: dynamic directive arguments for v-on, v-bind and custom
origin/dynamic-directive-arguments
feat: dynamic args for custom
perf: improve scoped slots change detection accuracy (#9371)
test: test cases for v-on/v-bind dynamic arguments
refactor: v-bind dynamic arguments use bind helper
test: fix tests, resolve helper conflict
fix: fix middle modifier
feat: handle dynamic argument for v-bind.sync
origin/slot-optimization
perf: improve scoped slots change
feat: dynamic directive arguments for v-bind and v-on
refactor: extend dom-props update skip to more
fix: fix checkbox event edge case in Firefox
test: fix tests in IE/Edge
packages
scripts
src
test
types
.babelrc.js
.editorconfig
.eslintignore
.eslintrc.js
.flowconfig
.gitignore
BACKERS.md



RAMIFICACIONES

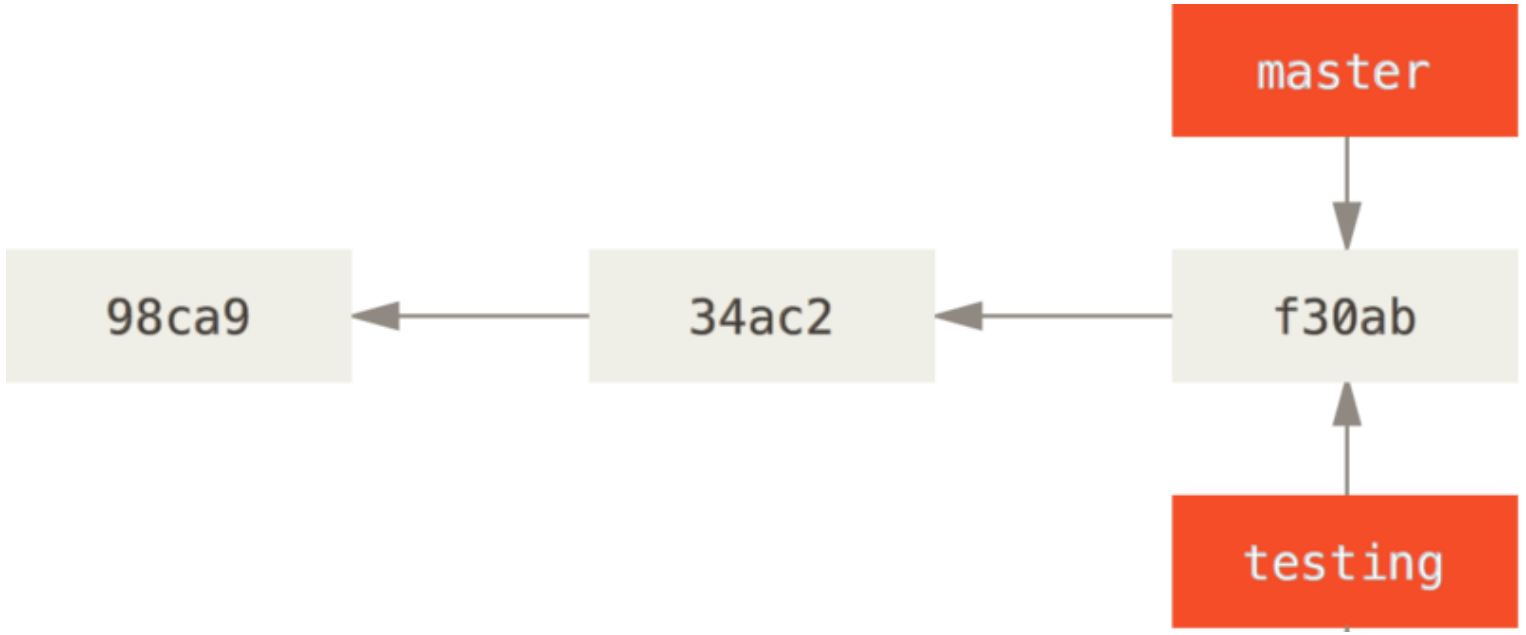


¿QUE ES UNA RAMA?

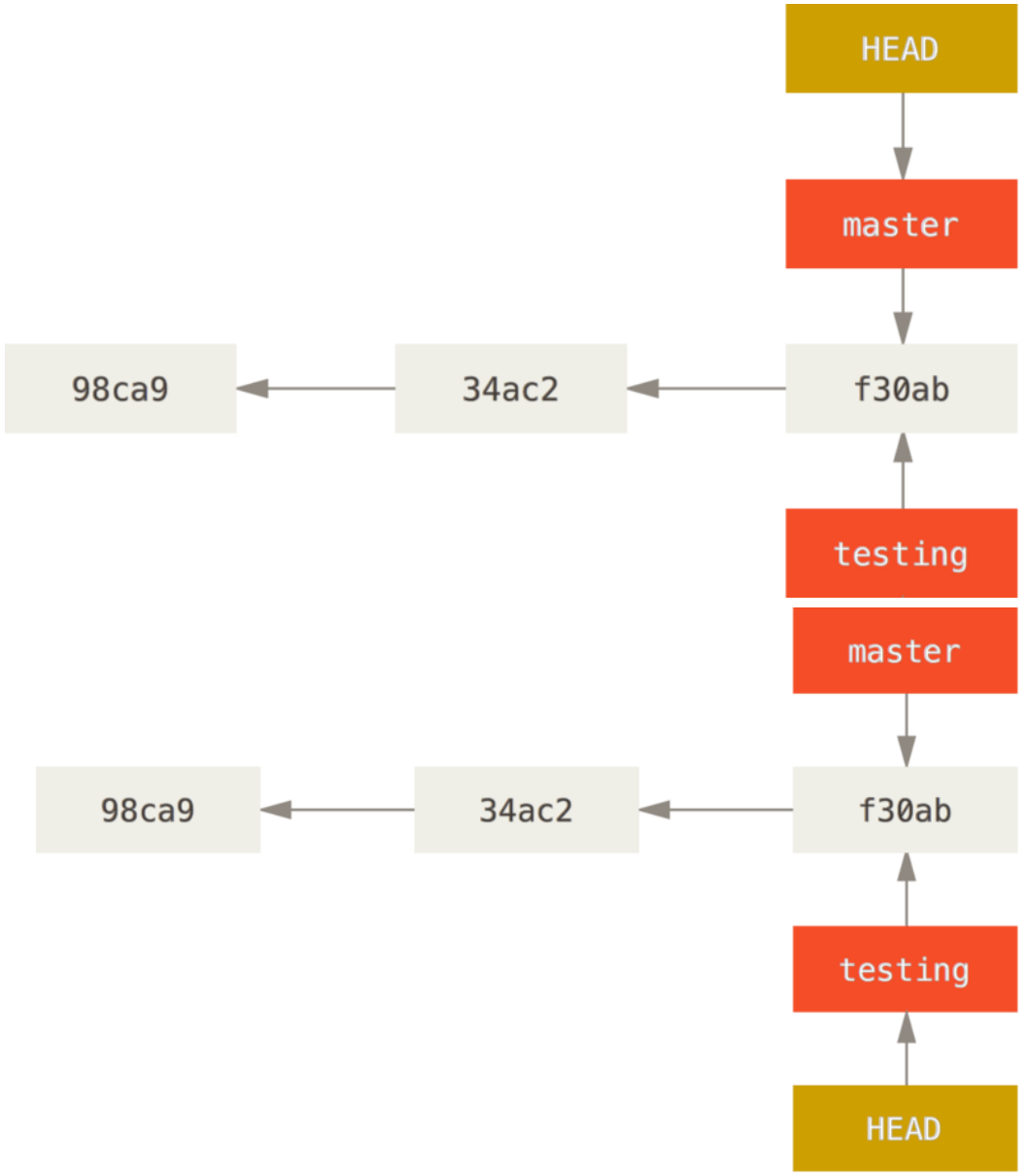


APUNTADOR MOVIL

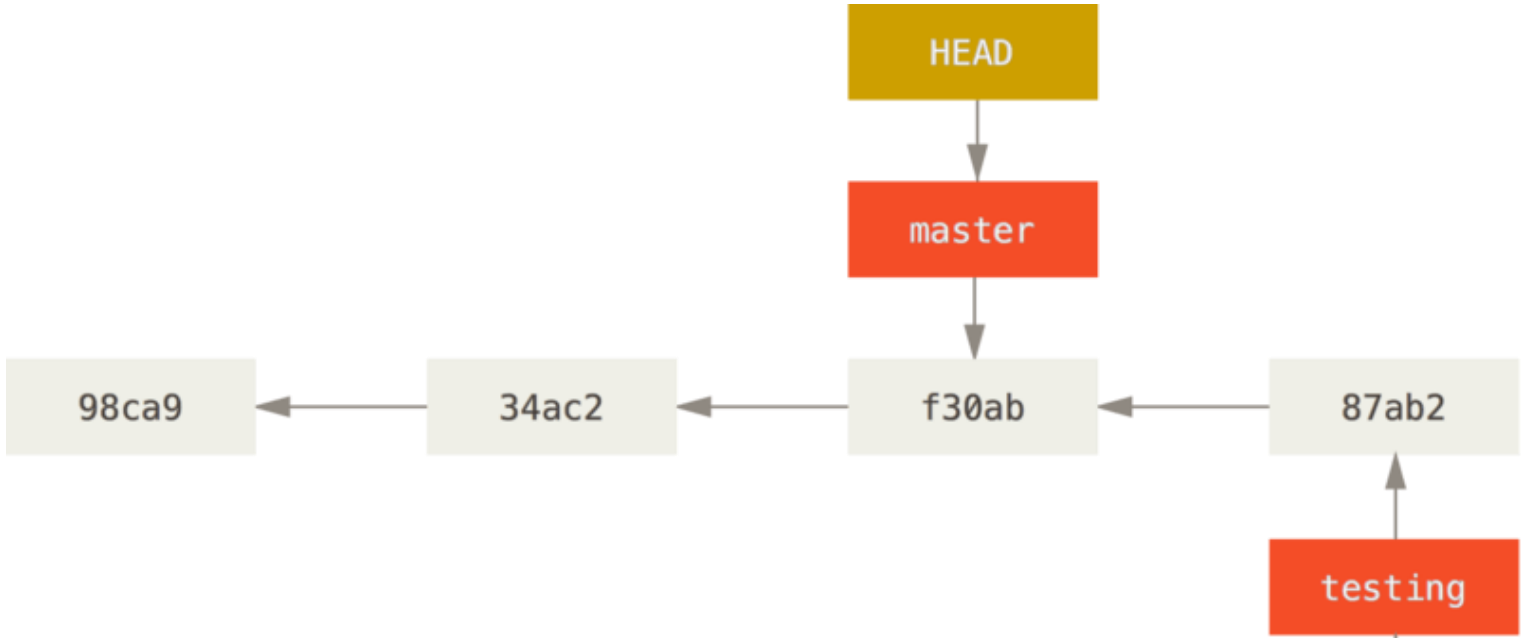
NUEVAS RAMAS



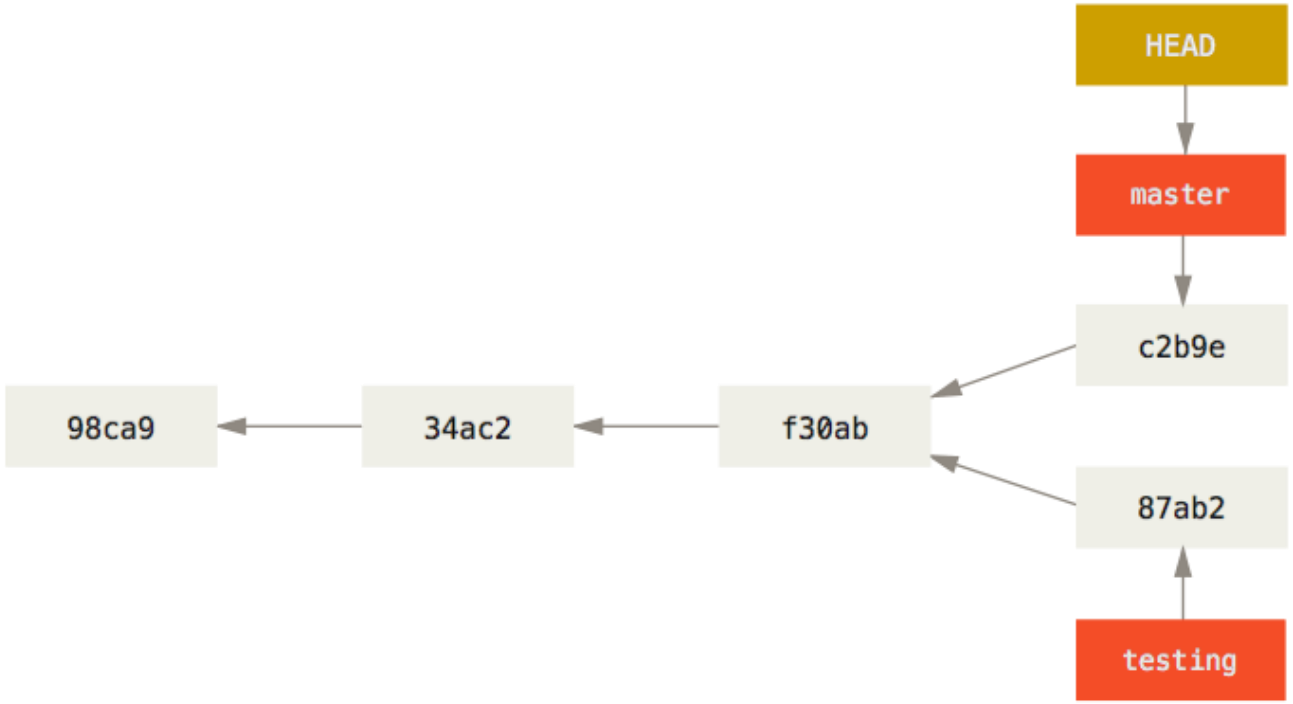
MOVERNOS DE UNA RAMA A OTRA



MOVERNOS DE UNA RAMA A OTRA



¿QUE NOS QUEDA FINALMENTE?



EJEMPLO

Trabajas en un sitio web.

Creas una rama para un nuevo tema sobre el que quieres trabajar.

Realizas algo de trabajo en esa rama.

En este momento, recibes una llamada avisándote de un problema crítico que has de resolver. Y sigues los siguientes pasos:

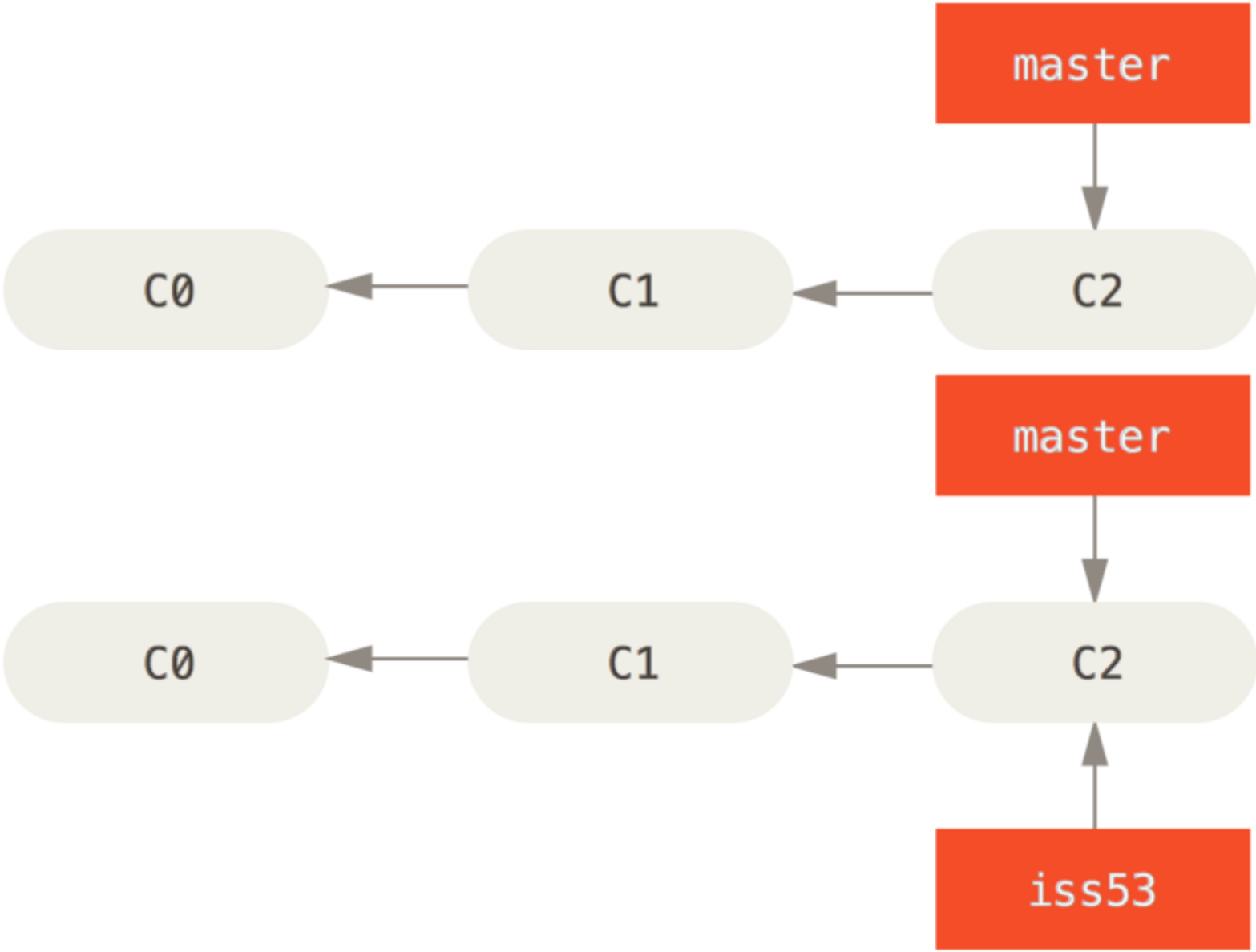
Vuelves a la rama de producción original.

Creas una nueva rama para el problema crítico y lo resuelves trabajando en ella.

Tras las pertinentes pruebas, fusionas (merge) esa rama y la envías (push) a la rama de producción.

Vuelves a la rama del tema en que andabas antes de la llamada y continuas tu trabajo

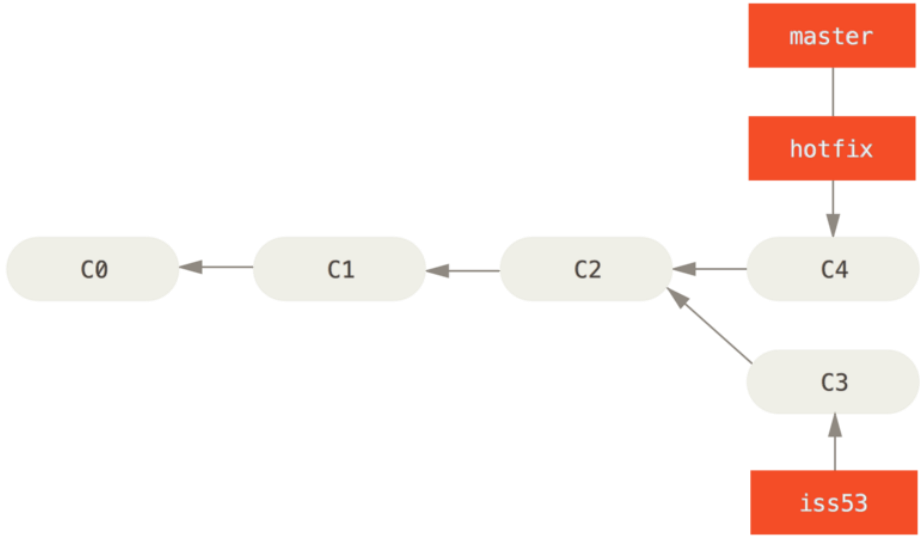
PASO 1



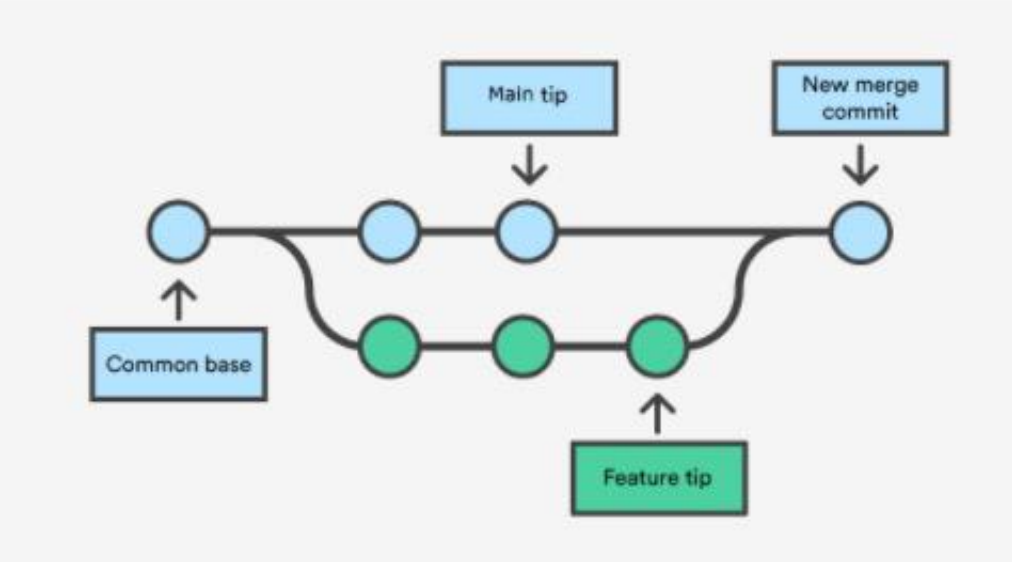
RECIBES LA LLAMADA

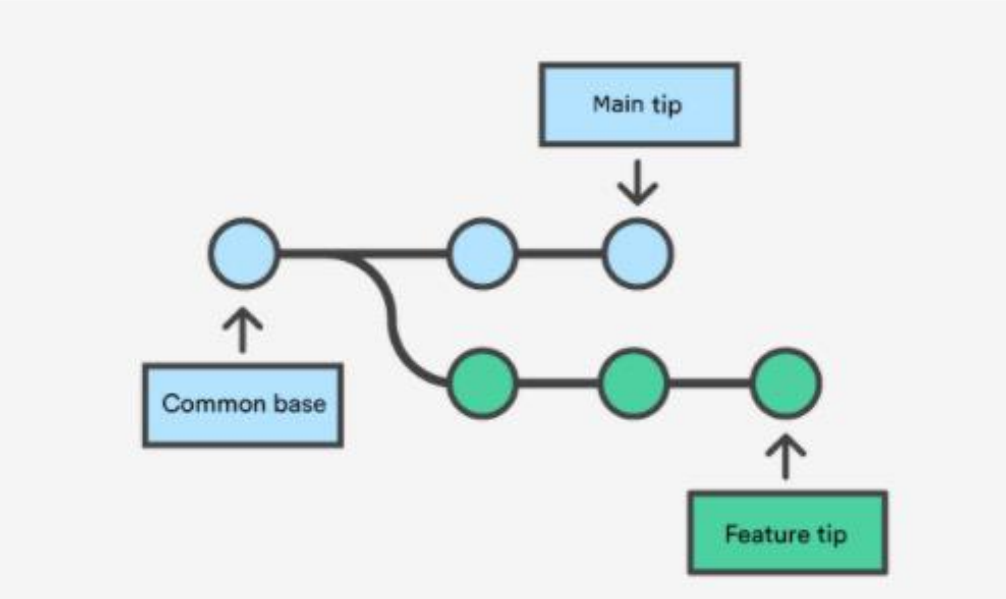
```
$ git checkout master
Switched to branch 'master'
```

```
$ git checkout -b hotfix
Switched to a new branch 'hotfix'
$ vim index.html
$ git commit -a -m 'fixed the broken email address'
[hotfix 1fb7853] fixed the broken email address
 1 file changed, 2 insertions(+)
```



MERGE


Main tip
New merge commit
Common base
Feature tip


Main tip
Common base
Feature tip

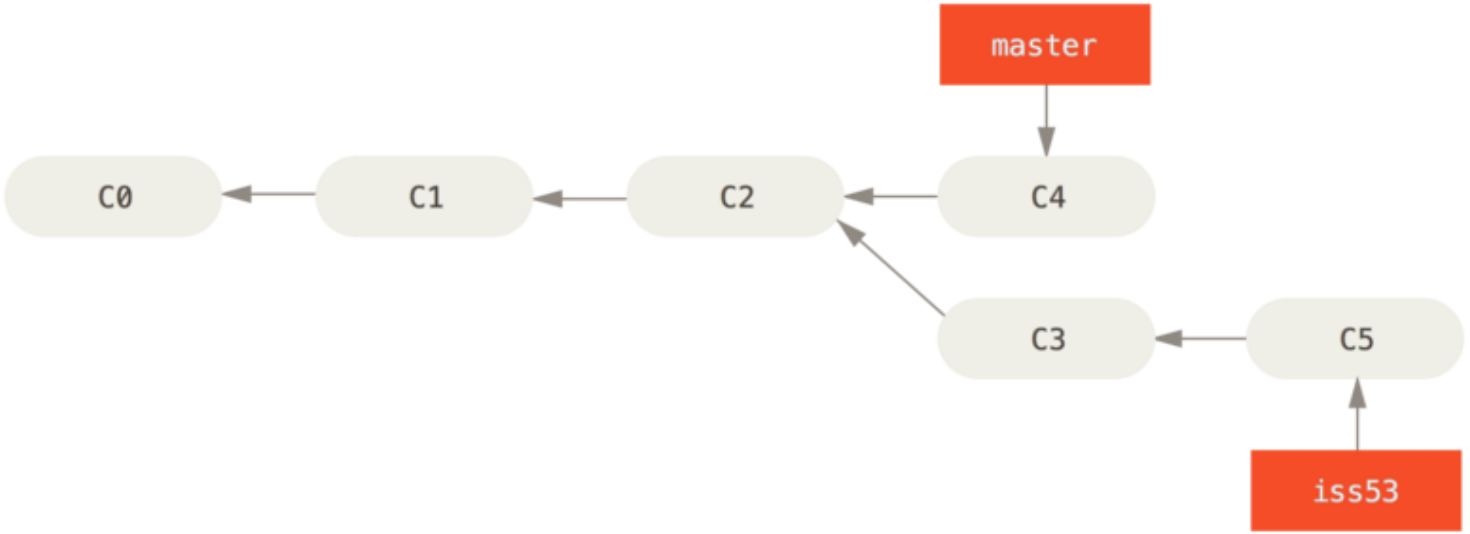
SOLUCIONAMOS EL PROBLEMA URGENTE

```
$ git branch -d hotfix
Deleted branch hotfix (3a0874c).
```

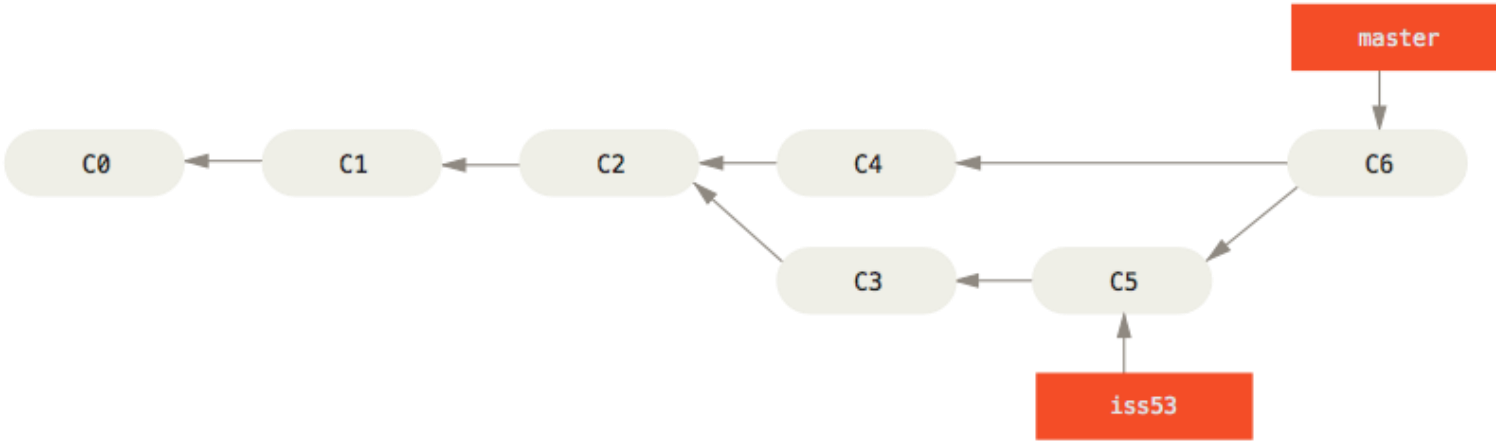
```
$ git checkout iss53
Switched to branch "iss53"
$ vim index.html
$ git commit -a -m 'finished the new footer [issue 53]'
[iss53 ad82d7a] finished the new footer [issue 53]
1 file changed, 1 insertion(+)
```

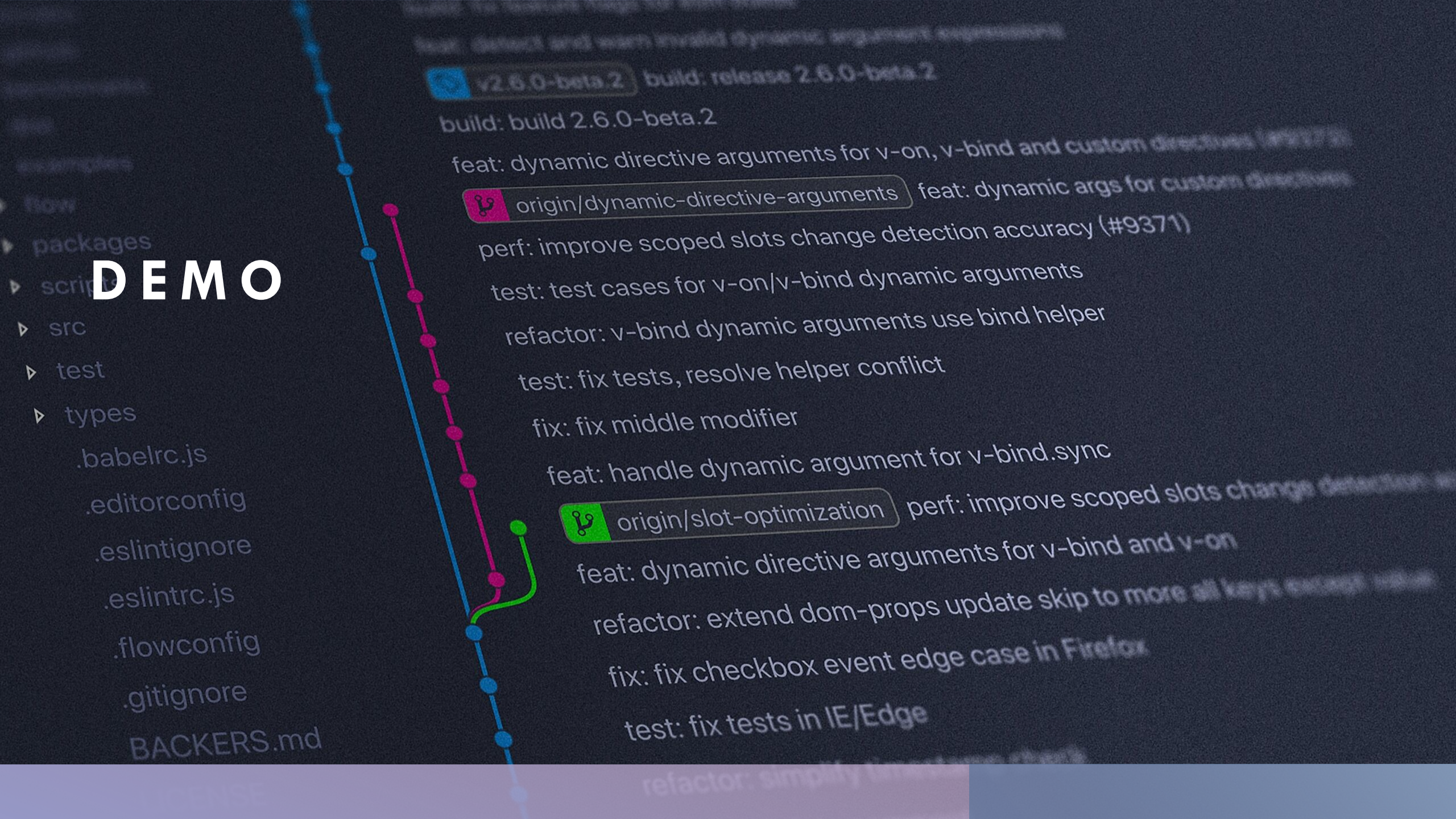
```
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast-forward
 index.html | 2 ++
 1 file changed, 2 insertions(+)
```

SITUACIÓN ACTUAL

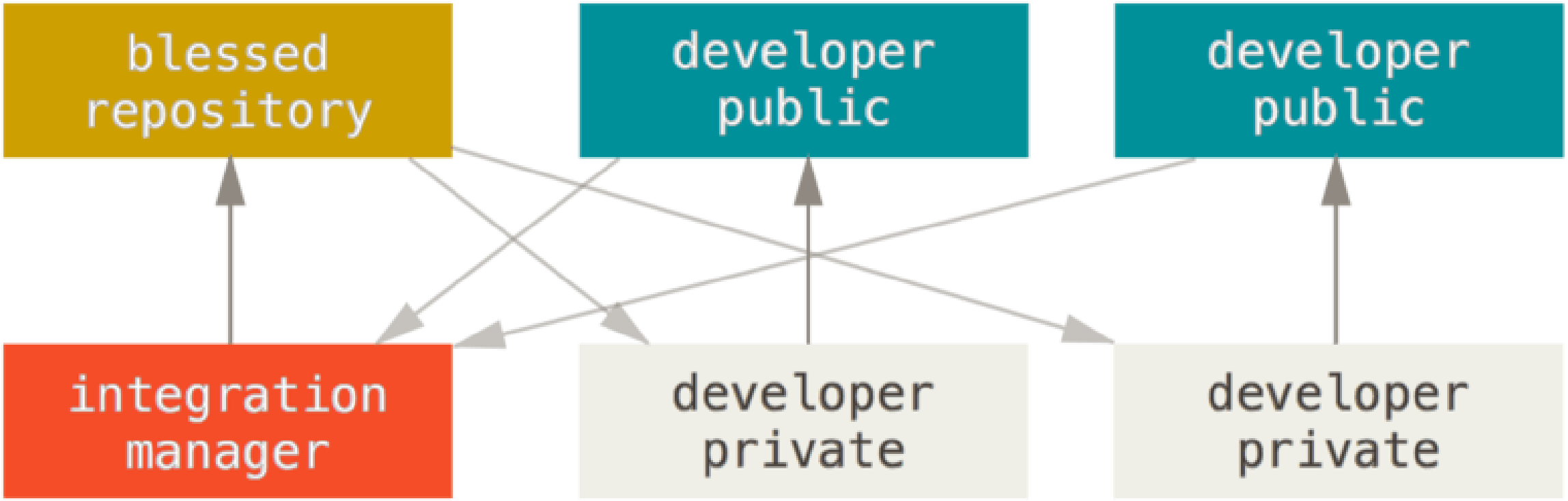


RESOLVIENDO EL PROBLEMA INICIAL




DEMO
v2.6.0-beta.2
build: release 2.6.0-beta.2
build: build 2.6.0-beta.2
feat: dynamic directive arguments for v-on, v-bind and custom
origin/dynamic-directive-arguments
feat: dynamic args for custom
perf: improve scoped slots change detection accuracy (#9371)
test: test cases for v-on/v-bind dynamic arguments
refactor: v-bind dynamic arguments use bind helper
test: fix tests, resolve helper conflict
fix: fix middle modifier
feat: handle dynamic argument for v-bind.sync
origin/slot-optimization
perf: improve scoped slots change
feat: dynamic directive arguments for v-bind and v-on
refactor: extend dom-props update skip to more
fix: fix checkbox event edge case in Firefox
test: fix tests in IE/Edge
packages
src
test
types
.babelrc.js
.editorconfig
.eslintignore
.eslintrc.js
.flowconfig
.gitignore
BACKERS.md

FORK

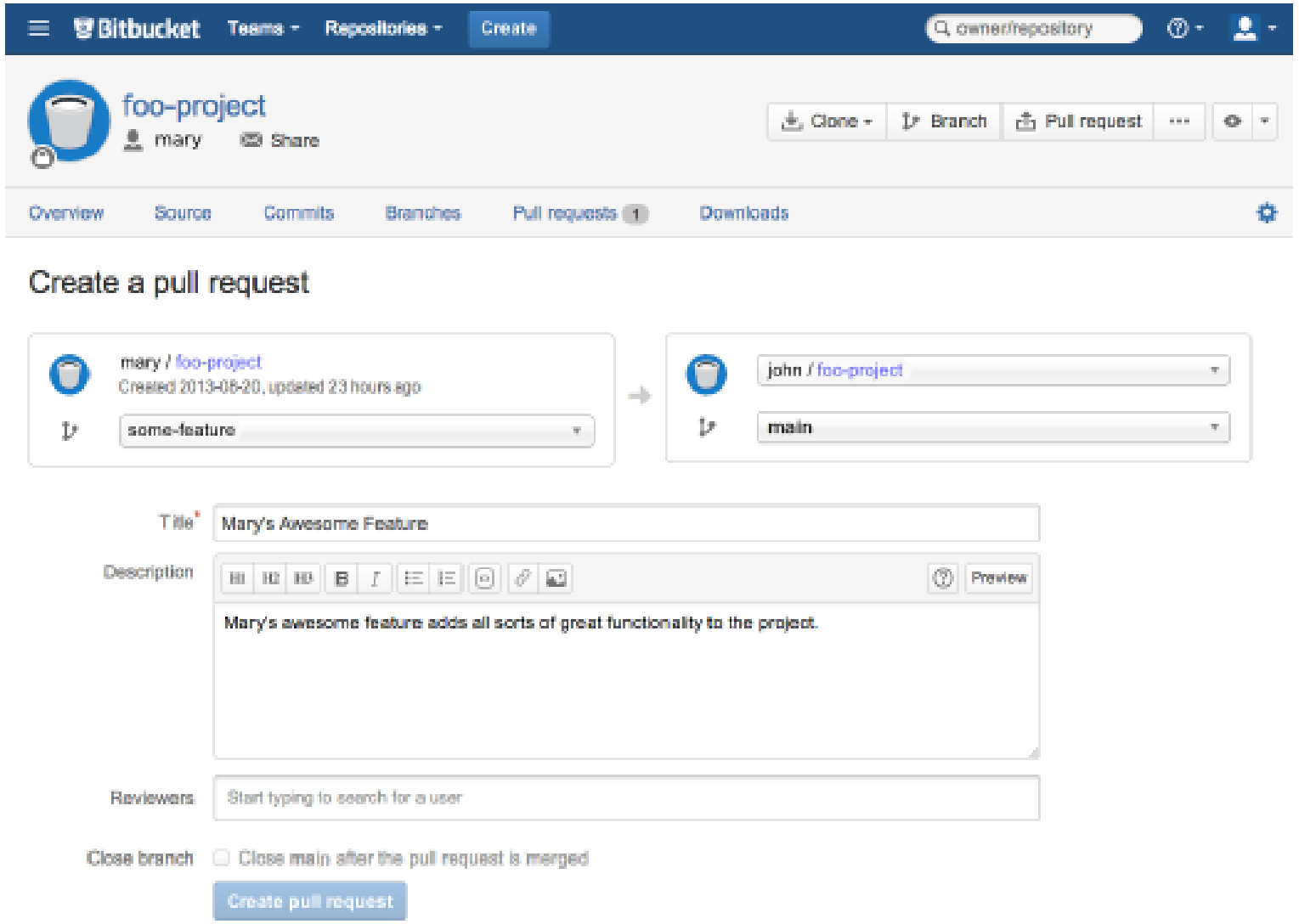


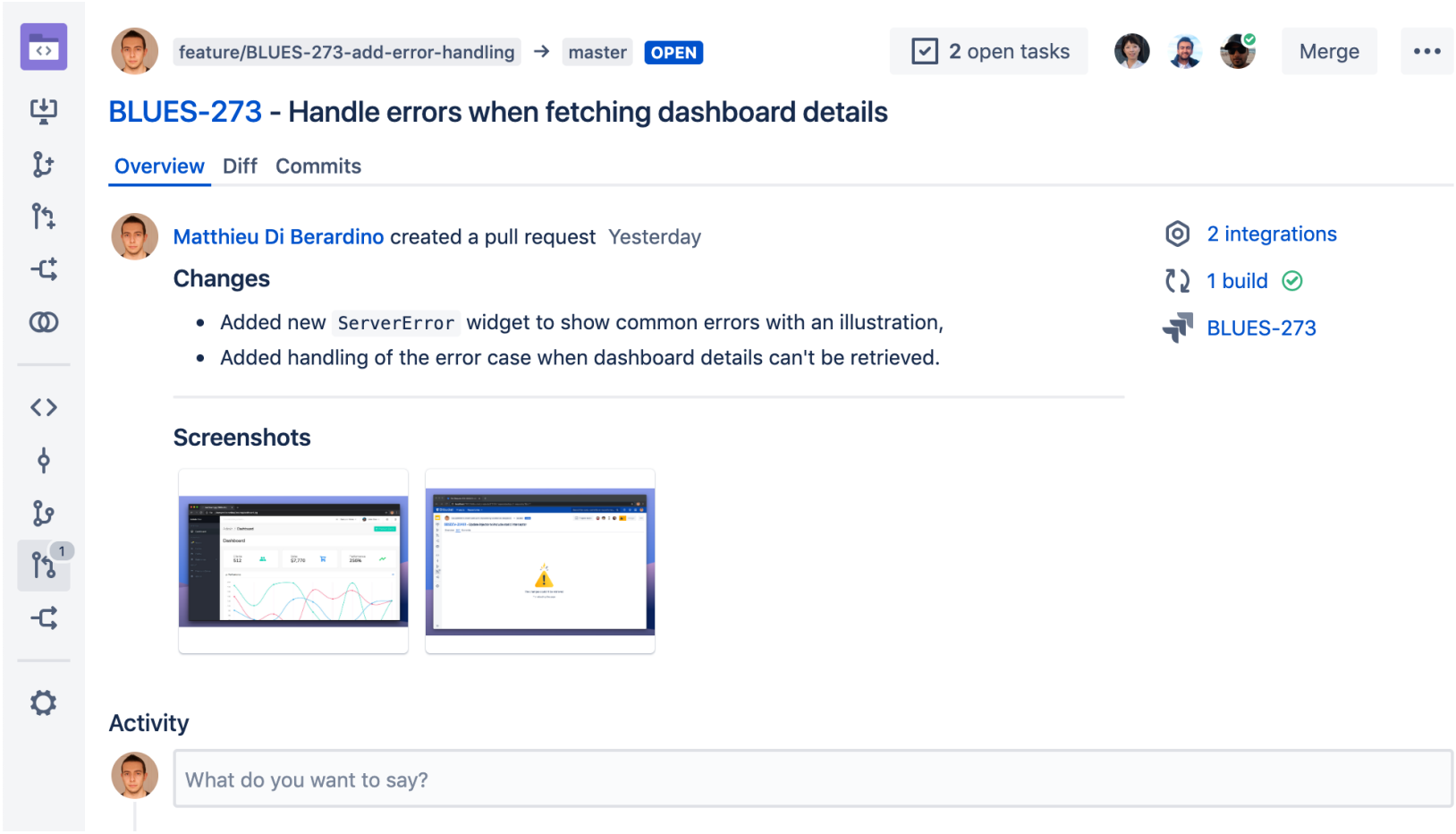
FLUJO DE TRABAJO

FLUJO DE TRABAJO

1. Se crea una rama a partir de master.
2. Se realizan algunos commits hacia esa rama.
3. Se envía esa rama hacia tu copia (fork) del proyecto.
4. Abres un Pull Request.
5. Se participa en la discusión asociada y, opcionalmente, se realizan nuevos commits.
6. El propietario del proyecto original cierra el Pull Request, bien fusionando la rama con tus cambios o bien rechazándolos.

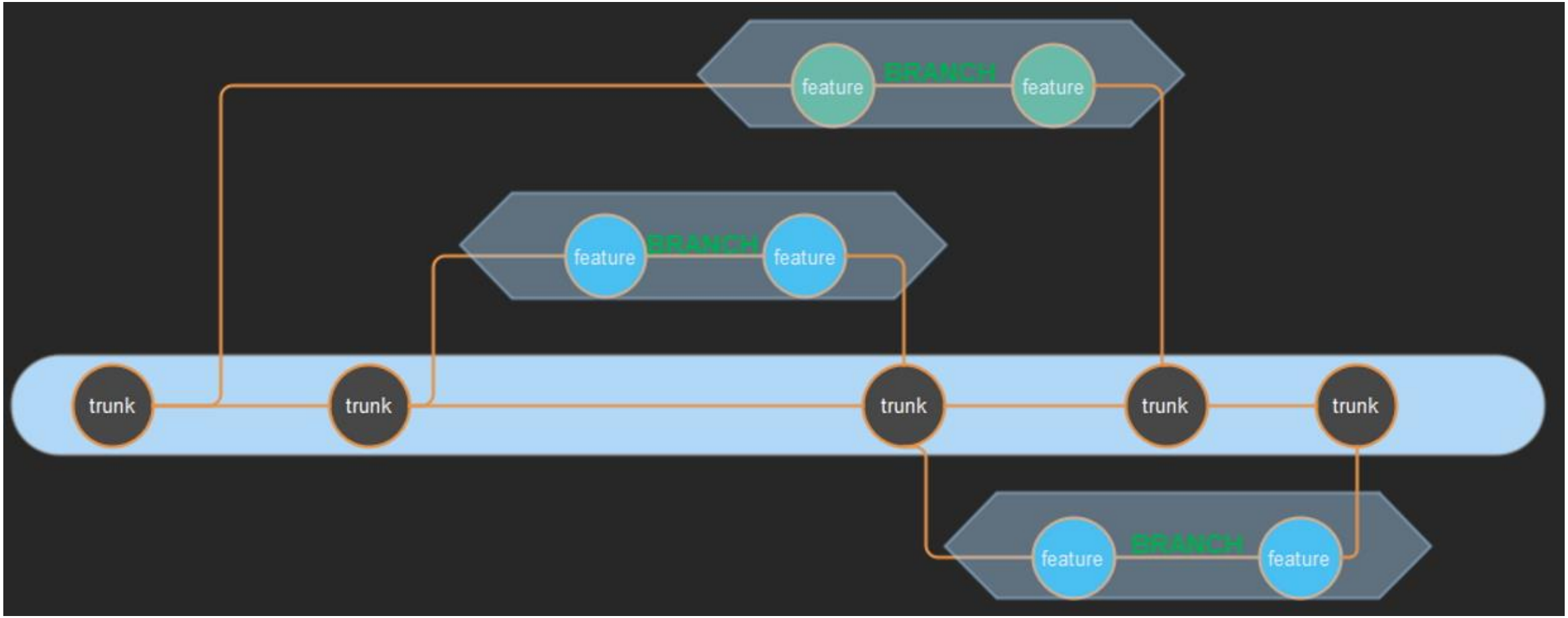
PULL REQUEST



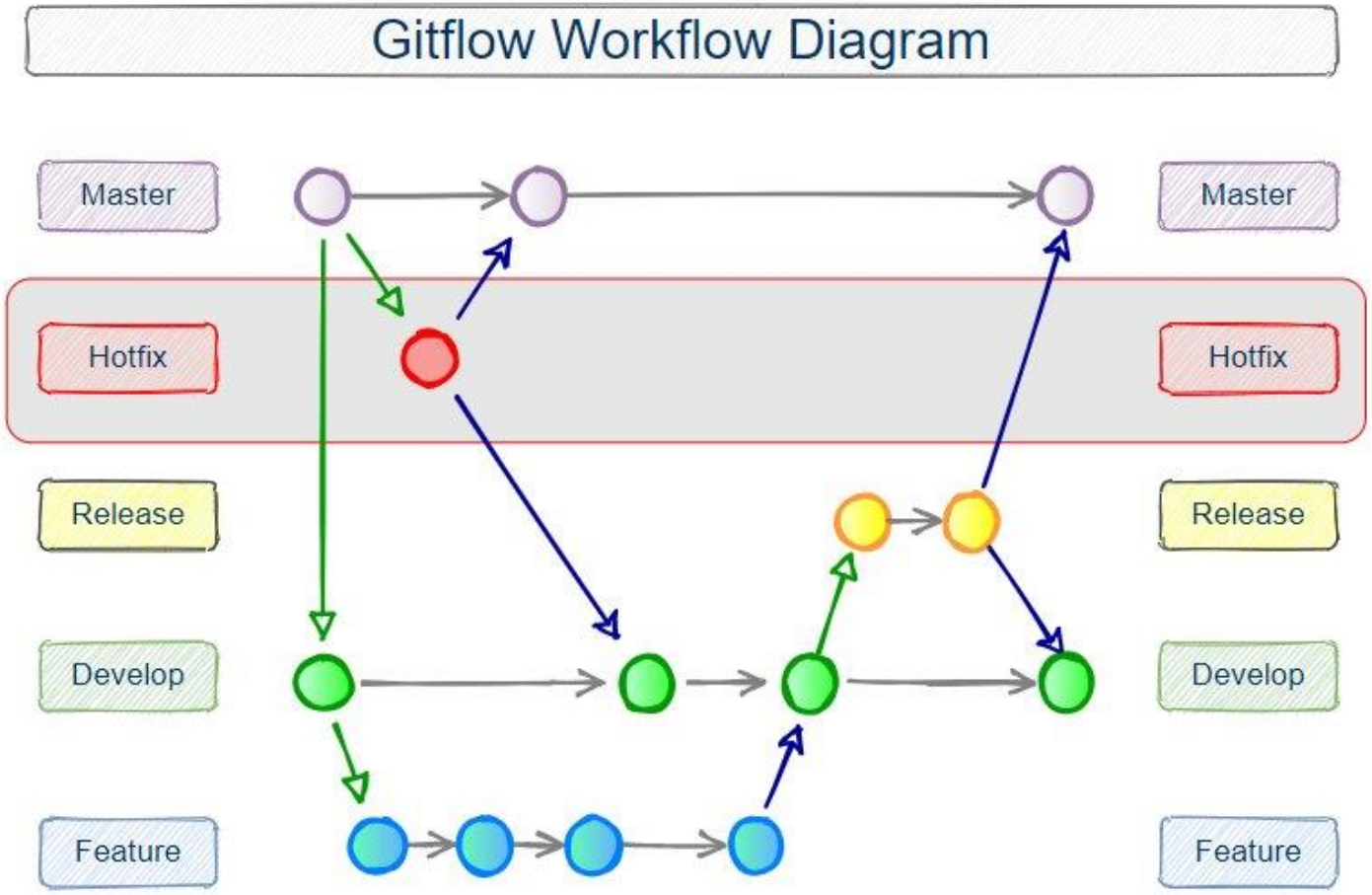


PULL REQUEST

LABORATORIO 2

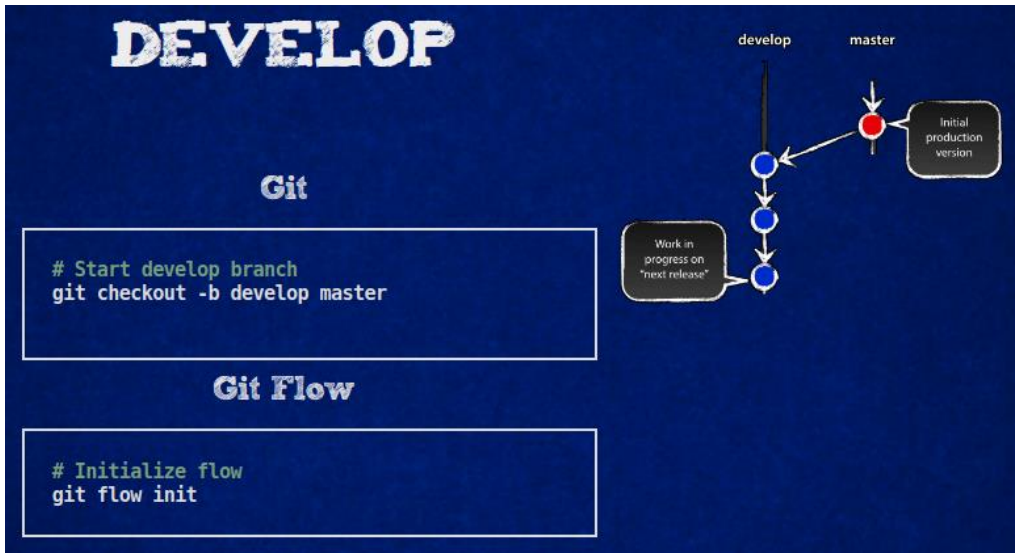


DESARROLLO BASADO EN TRONCOS


Gitflow Workflow Diagram
Master
Hotfix
Release
Develop
Feature
Master
Hotfix
Release
Develop
Feature

GITFLOW

RAMAS PRINCIPALES



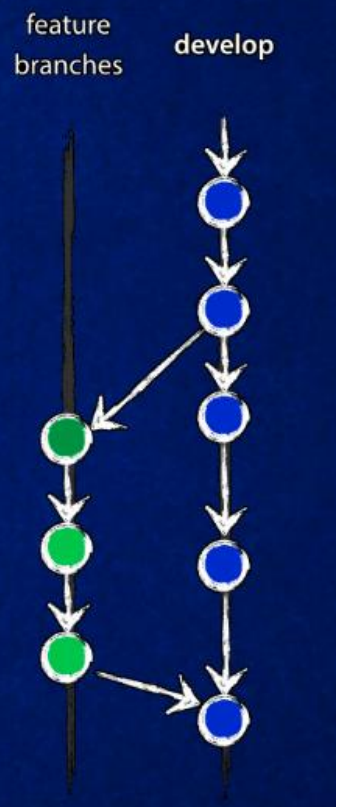
FEATURE/*

Git

```
# Creación
git checkout -b feature/lorem-ipsum develop

# Finalización
git checkout develop
git merge --no-ff feature/lorem-ipsum
git branch -d feature/lorem-ipsum
git push origin develop
```

Git Flow

```
# Creación
git flow feature start lorem-ipsum

# Finalización
git flow feature finish lorem-ipsum
```



FEATURE

RELEASE

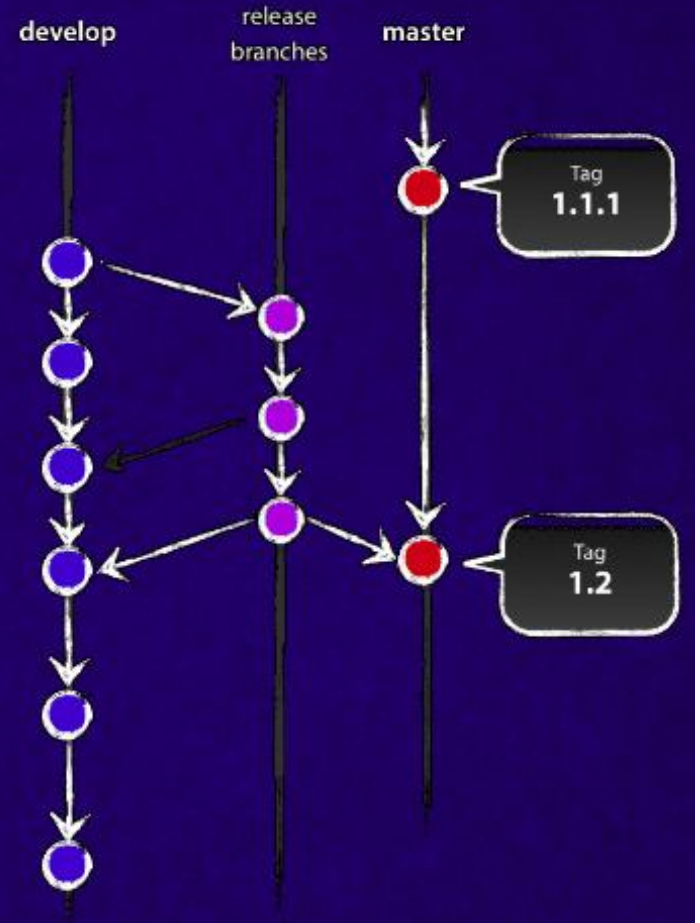
RELEASE/‹VERSION›

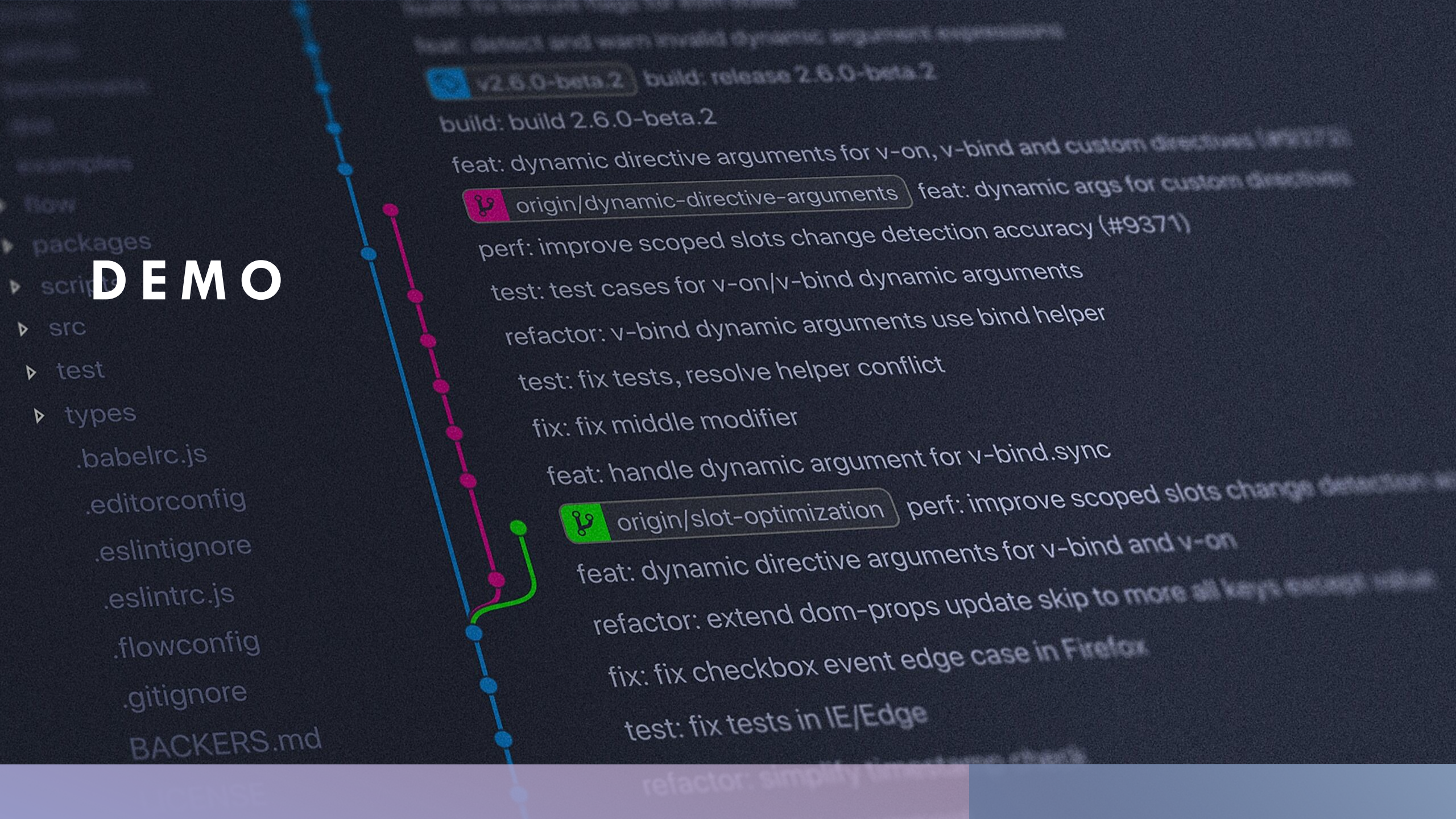
Git

```
# Creación
git checkout -b release/v2.3.0 develop

# Finalización
git checkout master
git merge --no-ff release/v2.3.0
git tag -a v2.3.0

git checkout develop
git merge --no-ff release/v2.3.0

git branch -d release/v2.3.0
```

Git Flow

```
# Creación
git flow release start v2.3.0

# Finalización
git flow release finish v2.3.0
```




DEMO
v2.6.0-beta.2
build: release 2.6.0-beta.2
build: build 2.6.0-beta.2
feat: dynamic directive arguments for v-on, v-bind and custom
origin/dynamic-directive-arguments
feat: dynamic args for custom
perf: improve scoped slots change detection accuracy (#9371)
test: test cases for v-on/v-bind dynamic arguments
refactor: v-bind dynamic arguments use bind helper
test: fix tests, resolve helper conflict
fix: fix middle modifier
feat: handle dynamic argument for v-bind.sync
origin/slot-optimization
perf: improve scoped slots change
feat: dynamic directive arguments for v-bind and v-on
refactor: extend dom-props update skip to more
fix: fix checkbox event edge case in Firefox
test: fix tests in IE/Edge
packages
src
test
types
.babelrc.js
.editorconfig
.eslintignore
.eslintrc.js
.flowconfig
.gitignore
BACKERS.md

HOTFIX

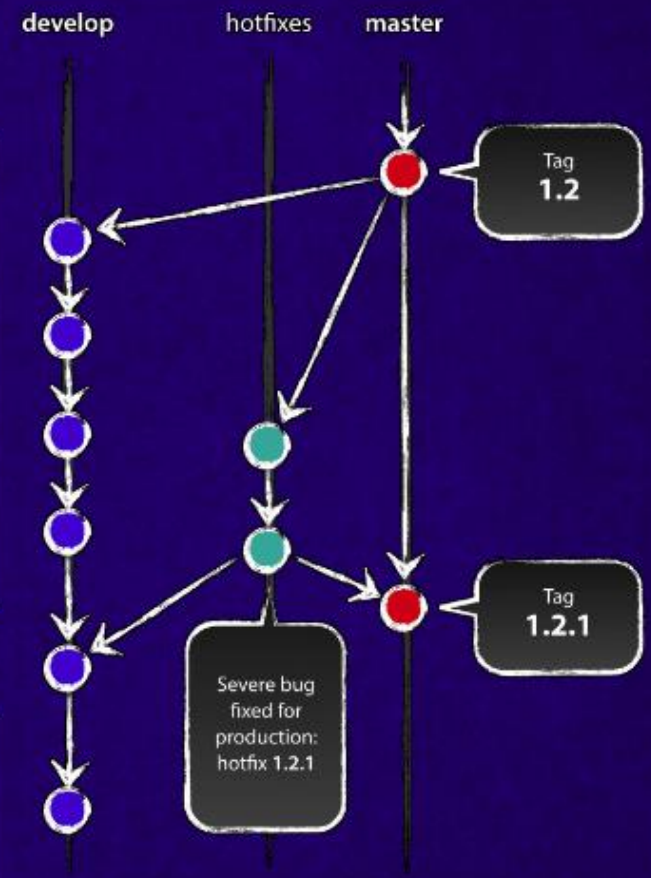
HOTFIX/‹VERSION›

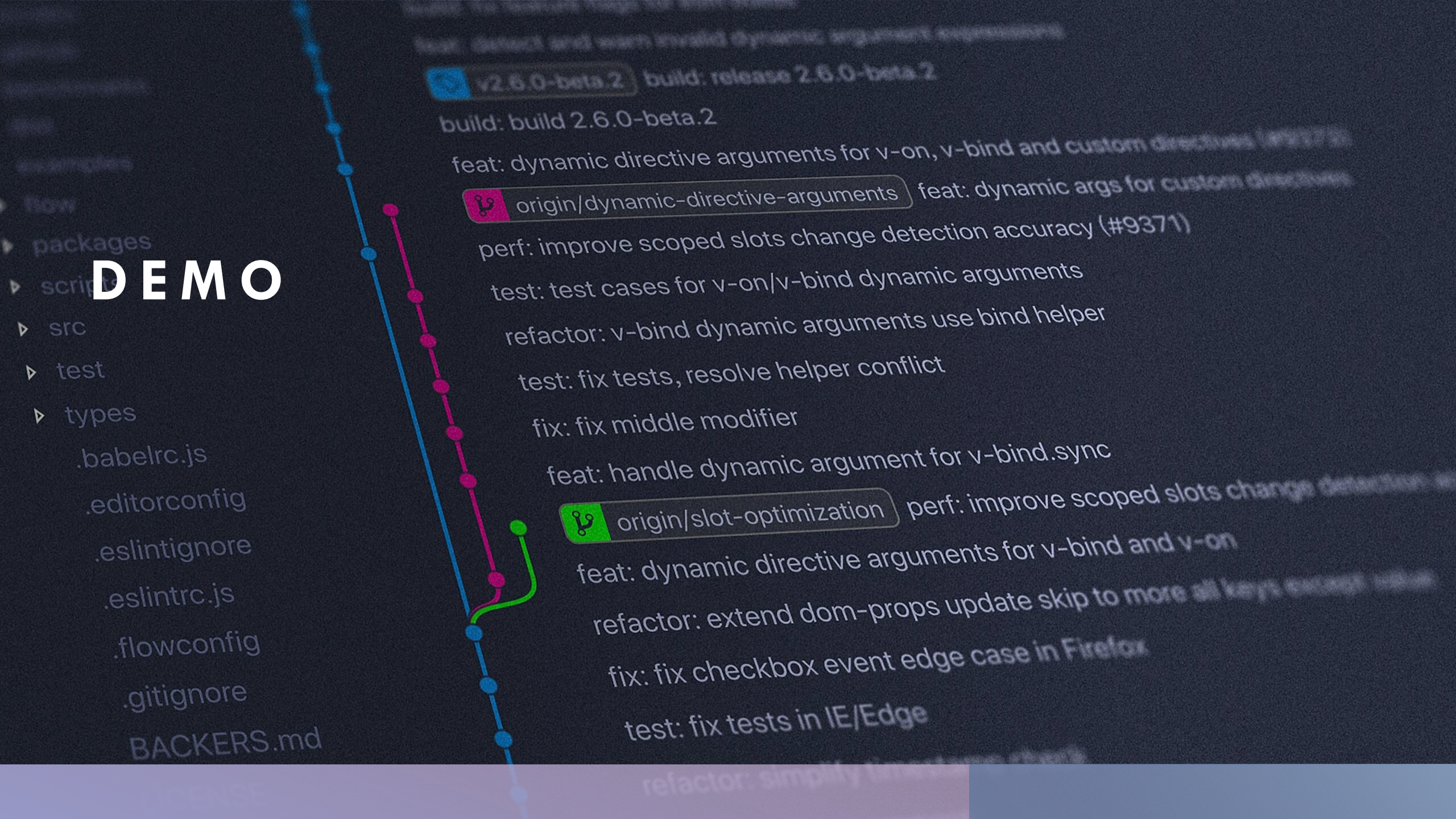
Git

```
# Creación
git checkout -b hotfix/v2.3.7 master

# Finalización
git checkout master
git merge --no-ff hotfix/v2.3.7
git tag -a v2.3.7

git checkout develop
git merge --no-ff hotfix/v2.3.7

git branch -d hotfix/v2.3.7
```

Git Flow

```
# Creación
git flow hotfix start v2.3.7

# Finalización
git flow hotfix finish v2.3.7
```




DEMO
v2.6.0-beta.2
build: release 2.6.0-beta.2
build: build 2.6.0-beta.2
feat: dynamic directive arguments for v-on, v-bind and custom
origin/dynamic-directive-arguments
feat: dynamic args for custom
perf: improve scoped slots change detection accuracy (#9371)
test: test cases for v-on/v-bind dynamic arguments
refactor: v-bind dynamic arguments use bind helper
test: fix tests, resolve helper conflict
fix: fix middle modifier
feat: handle dynamic argument for v-bind.sync
origin/slot-optimization
perf: improve scoped slots change
feat: dynamic directive arguments for v-bind and v-on
refactor: extend dom-props update skip to more
fix: fix checkbox event edge case in Firefox
test: fix tests in IE/Edge
packages
src
test
types
.babelrc.js
.editorconfig
.eslintignore
.eslintrc.js
.flowconfig
.gitignore
BACKERS.md

EJERCICIO 1


packages
src
test
types
.babelrc.js
.editorconfig
.eslintignore
.eslintrc.js
.flowconfig
.gitignore
BACKERS.md
v2.6.0-beta.2
build: release 2.6.0-beta.2
build: build 2.6.0-beta.2
feat: dynamic directive arguments for v-on, v-bind and custom
origin/dynamic-directive-arguments
feat: dynamic args for custom
perf: improve scoped slots change detection accuracy (#9371)
test: test cases for v-on/v-bind dynamic arguments
refactor: v-bind dynamic arguments use bind helper
test: fix tests, resolve helper conflict
fix: fix middle modifier
feat: handle dynamic argument for v-bind.sync
origin/slot-optimization
perf: improve scoped slots
feat: dynamic directive arguments for v-bind and v-on
refactor: extend dom-props update skip to more
fix: fix checkbox event edge case in Firefox
test: fix tests in IE/Edge

Modulo 2: Pruebas automatizadas

- Pruebas de unidad: Junit
- Pruebas de integración: Mockito
- Pruebas de aceptación Frontnend: Selenium
- Pruebas API Rest Backend
- Pruebas de rendimiento: Jmeter/Gatling/Blazemeter

INTRODUCCIÓN A PRUEBAS

vs.

Pruebas dinámicas

Pruebas estáticas

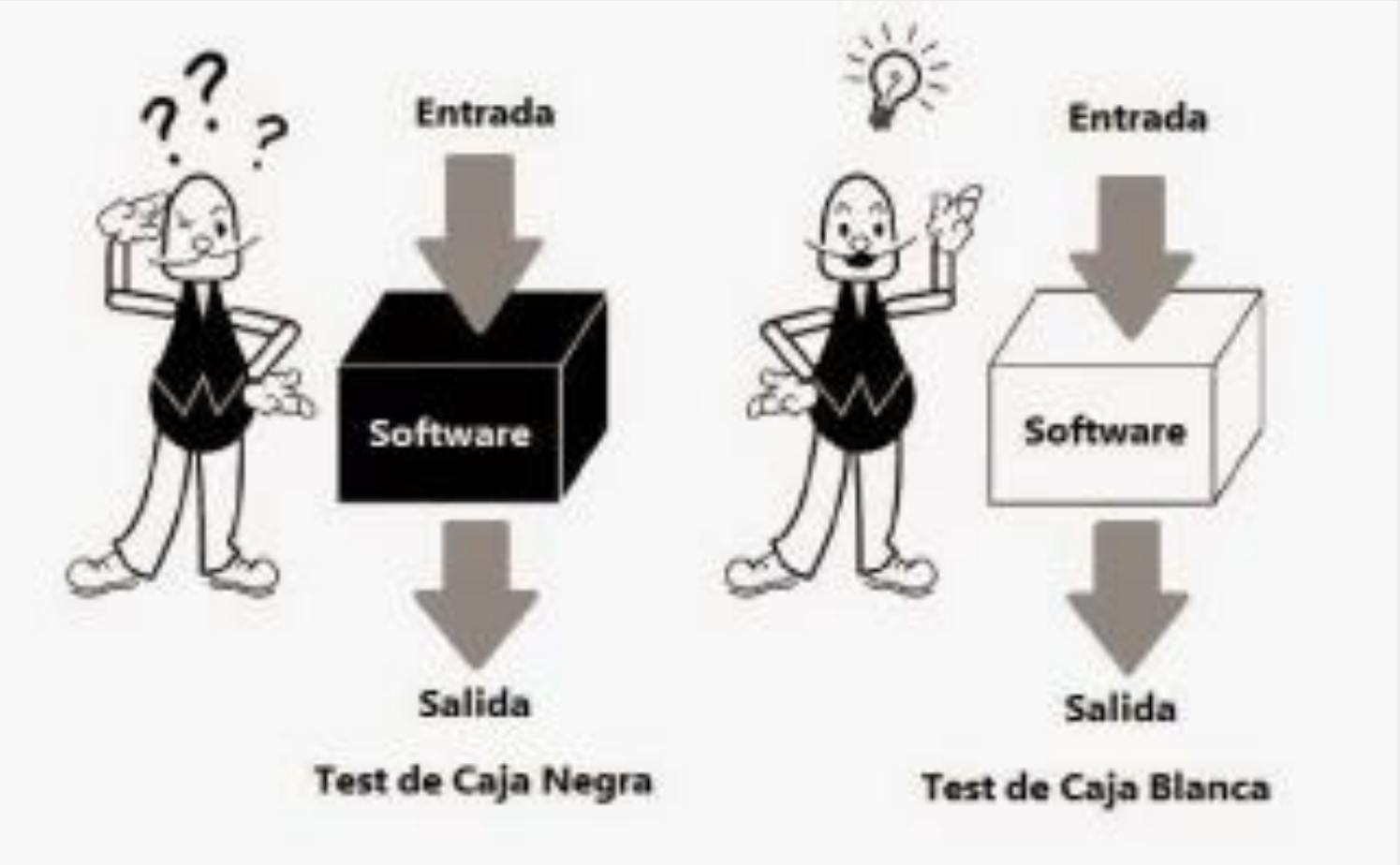
TIPOS DE SOFTWARE TESTING

Pruebas funcionales

- Pruebas unitarias
- Pruebas de integración
- Pruebas de sistema
- Pruebas de sanidad
- Pruebas de humo
- Pruebas de interfaz
- Pruebas de regresión
- Pruebas de aceptación

Pruebas no funcionales

- Pruebas de rendimiento
- Prueba de carga
- Pruebas de estrés
- Pruebas de volumen
- Pruebas de seguridad
- Pruebas de compatibilidad
- Pruebas de instalación
- Pruebas de recuperación
- Pruebas de confiabilidad
- Pruebas de usabilidad
- Pruebas de conformidad
- Pruebas de localización


Entrada
Software
Salida
Test de Caja Negra
Entrada
Software
Salida
Test de Caja Blanca

ESTRATEGIA DE PRUEBAS

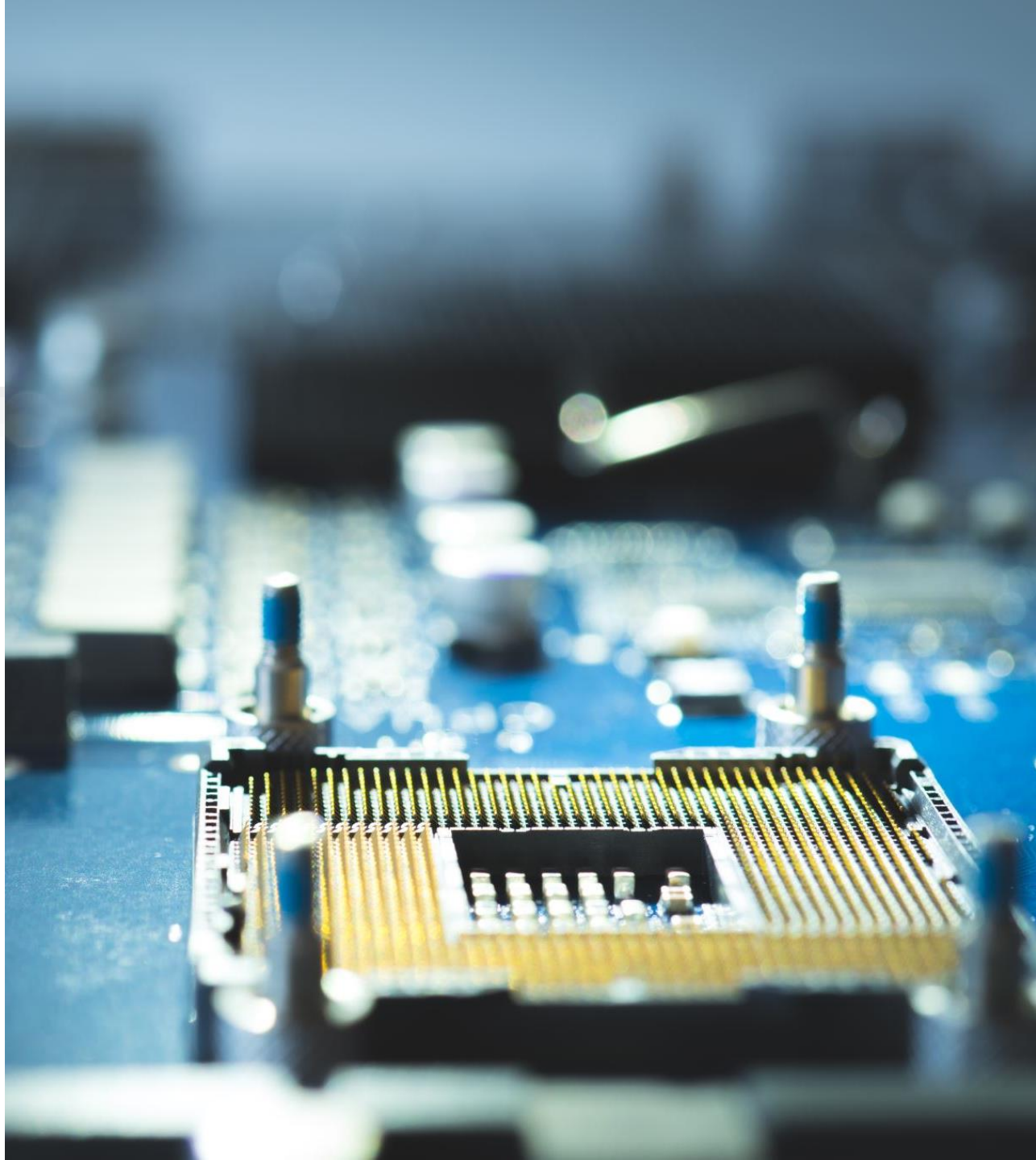
- Planificación: Definición de estrategias, alcance, recursos y calendario de pruebas.
- Diseño de Casos de Prueba: Creación de escenarios y casos de prueba basados en requisitos.
- Ejecución de Pruebas: Ejecución de casos de prueba y registro de resultados.
- Análisis de Resultados: Evaluación de los resultados de las pruebas y seguimiento de los defectos encontrados.
- Informe y Retest: Creación de informes de pruebas y reevaluación tras la corrección de defectos.

DISEÑO DE PRUEBAS

- Conocimiento de los Requisitos
- Reducción de la funcionalidad a probar
- Definición de Casos de Prueba
- Relacionar con los Casos de Usuario (historias)
- Pensar bien los costes de las pruebas
- Enfocarse en las funcionalidades más críticas
- Cada prueba requiere su herramienta

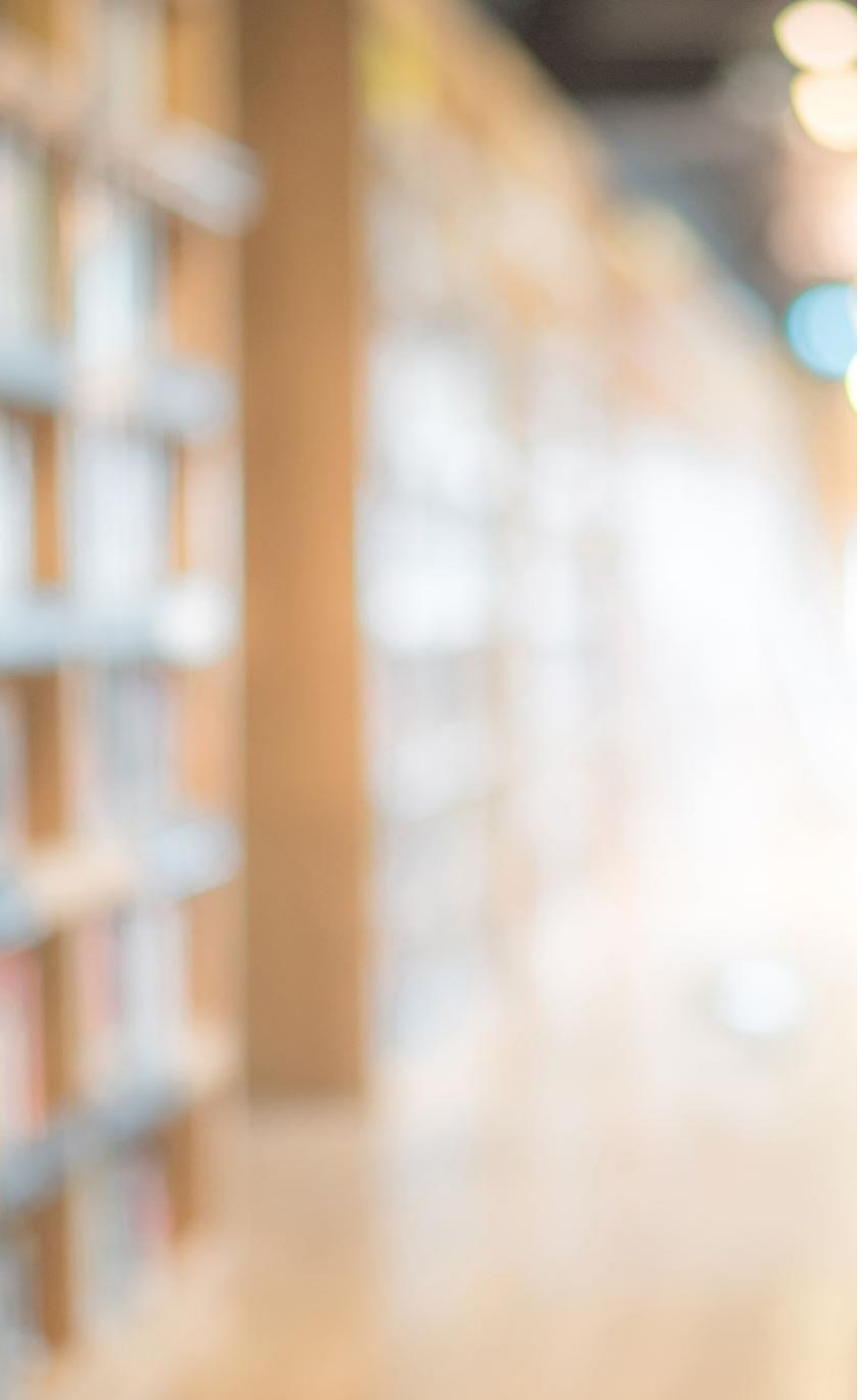
HERRAMIENTAS

- Pruebas Unitarias: Junit
- Pruebas de Integración: Mockito
- Pruebas de Aceptación/Funcionalidad Web: Selenium
- Bases de Datos/Web Rendimiento: Jmeter/Blazemeter/Gatling
- Servidor de Integración Contínua: Jenkins/TravisCI/CircleCI
- Herramienta de Seguimiento de Proyectos e Incidencias: Jira/Mantis
- Automatizador de la Compilación, Despliegue y Ejecución de proyectos: Maven/Gradle/NPM
- Despliegue de aplicaciones: Docker/Kubernetes



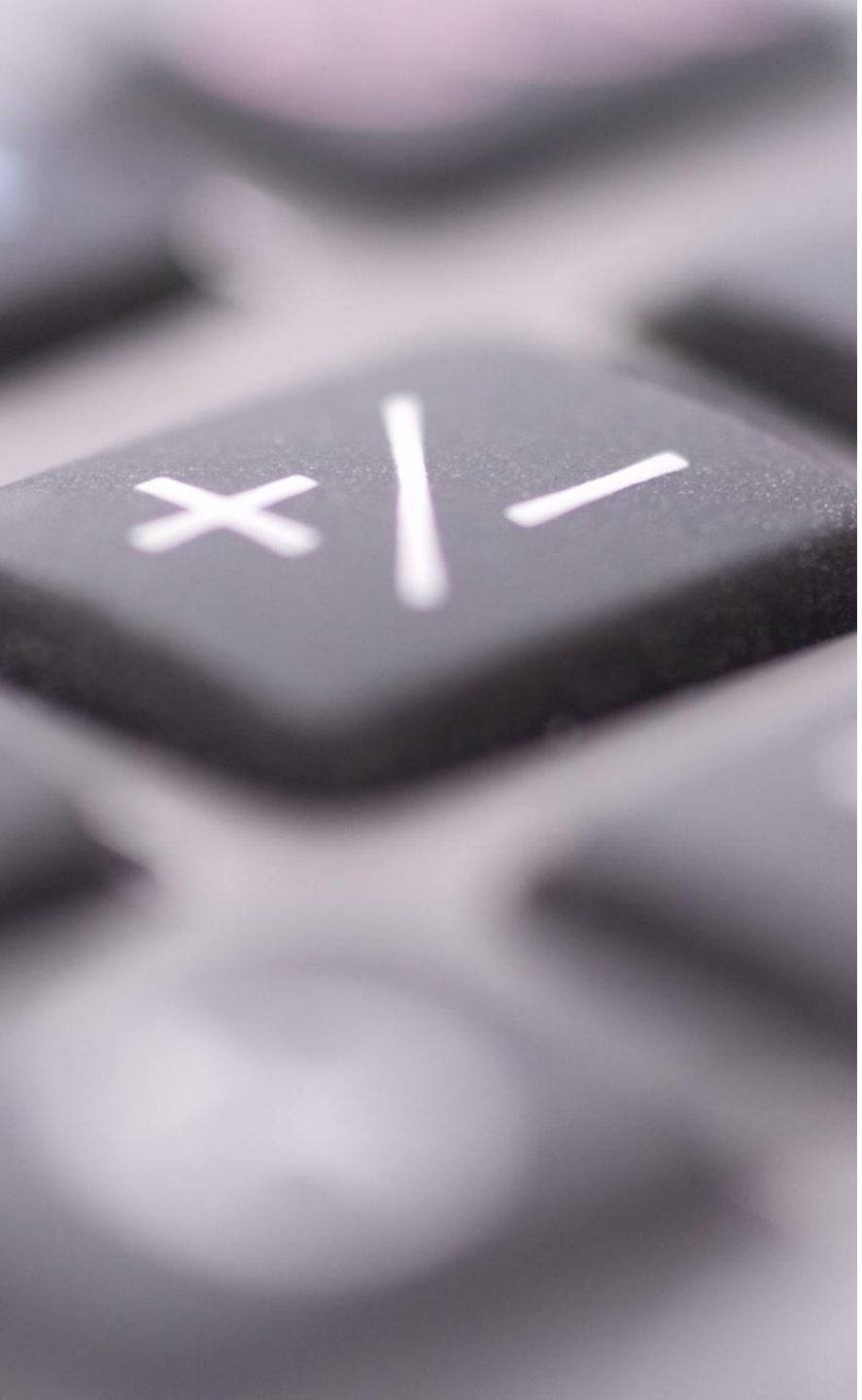
AUTOMATIZACIÓN DE LAS PRUEBAS


5
5



EJEMPLO

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class MiClaseTest {

    @Test
    public void pruebaSuma() {
        int resultado = MiClase.suma(3, 5);
        assertEquals(8, resultado);
    }
}
```



MENSAJES PERSONALIZADOS

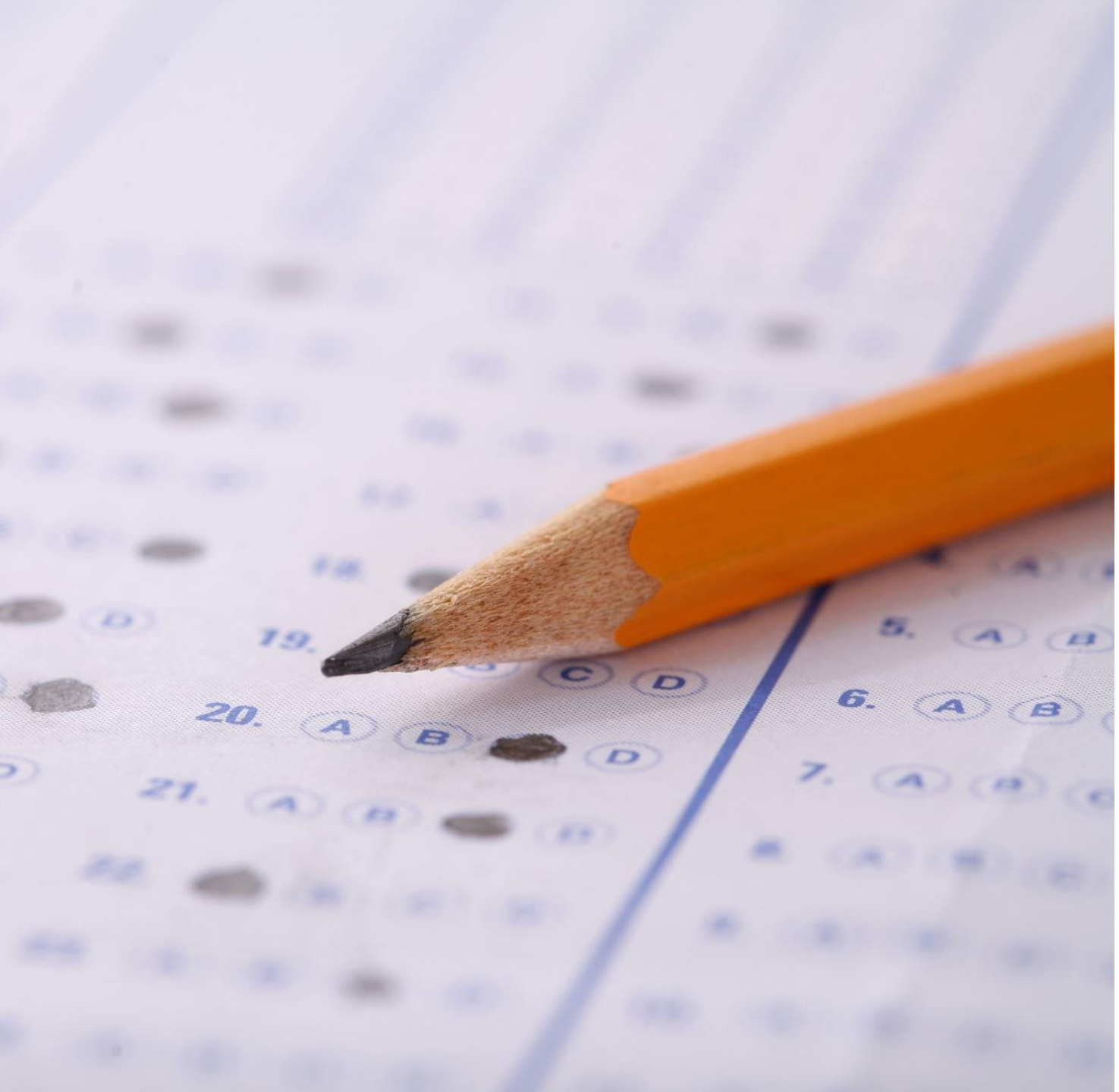
assertEquals(2, calculadora.suma(1, 1), “La suma debería ser 2”);

assertEquals(2, calculadora.suma(1, 1), () -> “La suma debería ser 2”)

LA ANATOMIA DE JUNIT

```java
package examples.nbank;

public class Conversion {

    public double tempConversion (double temperature, String unit) {
        if (unit.equals("F"))
            return (temperature - 32) * (5.0/9.0);
        else
            return (temperature * (9.0/5.0)) + 32;
    }

}
```

```java
package examples.nbank;

1 import static org.junit.Assert.assertEquals;

2 import org.junit.*;

3 public class ConversionTest {

    4 @Test
      public void testTempCoversion() throws Throwable {
          // Given
        6 Conversion underTest = new Conversion();

          // When
        7 double temperature = 80.0d;
          String unit = "";
        8 double result = underTest.tempConversion(temperature, unit);

          // Then - assertions for result of method tempConversion(double, String)
        9 assertEquals(176.0d, result, 0.0);
      }
}
```



TRABAJANDO CON TEST

EXCEPCIONES CONTROLADAS

- JUnit nos permite comprobar que un método lanza una excepción controlada.
  - Deben extender de Throwable. Por ejemplo RuntimeException.
  - assertThrows(Exception.class, () -> {});
  - assertThrows(Exception.class, () -> {}, message);

ASSERT ALL

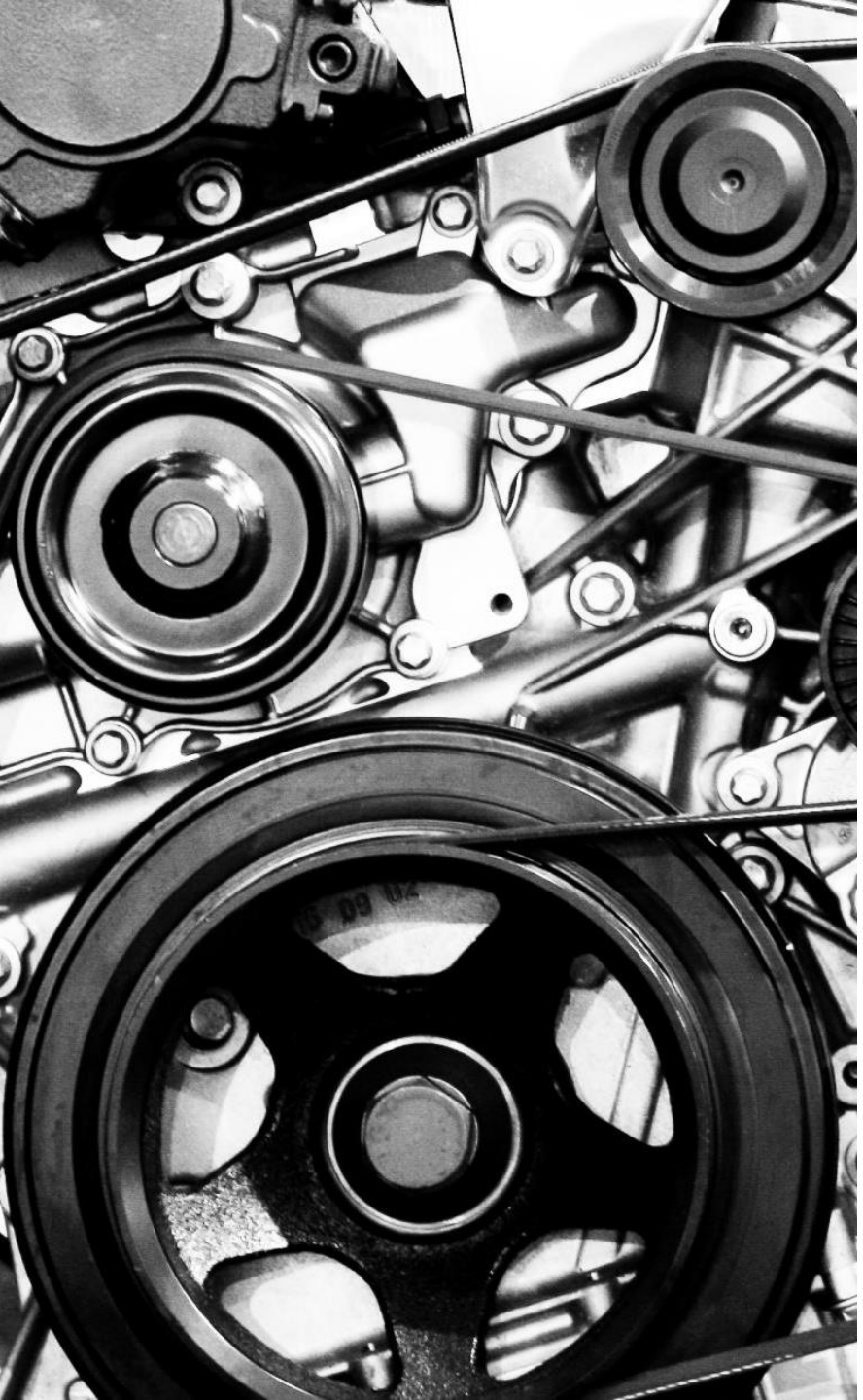
- Difícil seguimiento de asserts cuando hay muchos en un test.
- Si un assert falla, no se ejecutan los siguientes y no sabemos su evaluación.
- assertAll ejecuta todos los assert independientemente del posible fallo de uno de ellos.
- Reporta todos los fallos. Dónde se han producido y por qué.

EJEMPLO DE USO

- assertAll(String message, () -> {});
- Pueden incluir varias expresiones lambda.
- assertAll(“probando compras”, () -> assertNotNull(store.getProducts()), () -> assertEquals(2, store.getProducts().size()), () -> assertTrue(new BigDecimal(“300.00”).compareTo(actual Amount) == 0));

DAR NOMBRE A LOS TESTS

- JUnit 5 nombra a los tests con el nombre del método.
- Permite especificar un nombre personalizado anotando el método con @DisplayName(“Nombre del test”);
  - @Test
  - @DisplayName(“Comprobación suma calculadora”)
  - void sumaCalculadora() { … }



CICLO DE VIDA

- Proceso por el cual se crea, se ejecuta y se destruye una instancia encargada de la realización de las pruebas
- Se encarga el motor de JUnit 5.
- Se crea una nueva instancia con cada test que se ejecuta.
- JUnit 5 permite ejecutar hooks en diferentes momentos del ciclo de vida.
- Hooks de JUnit 5: @BeforeAll / @AfterAll @BeforeEach/ @AfterEach

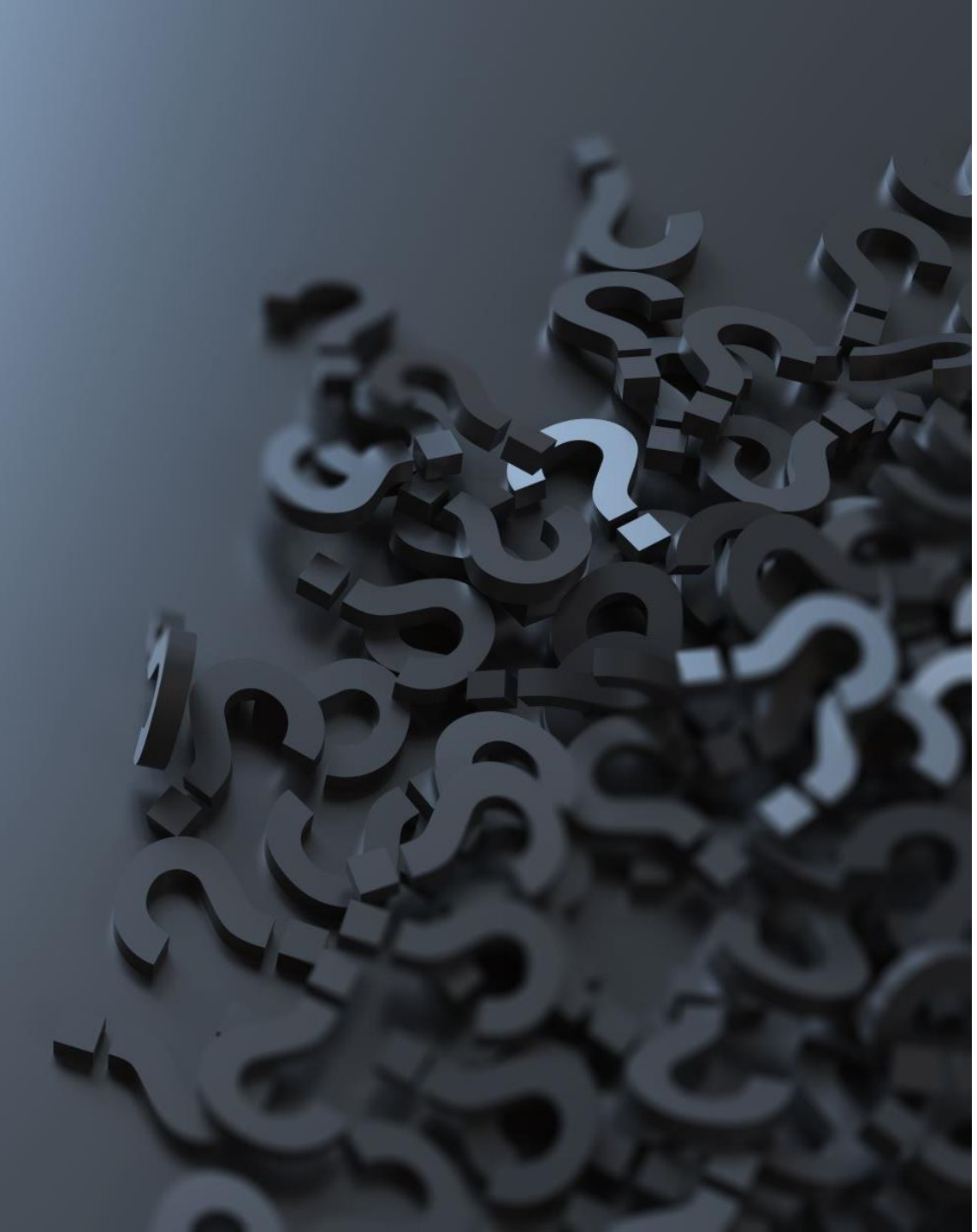
@BEFOREEACH / @AFTEREACH

- Se ejecuta una vez que se crea una nueva instancia, es decir, cada vez que se ejecuta un test.
- @BeforeEach - Se ejecuta antes de la ejecución del test.
- @AfterEach - Se ejecuta después de la ejecución del test.

```
public class TiendaTest {
      private List<String> products;
  @BeforeEach
  void setup() {
              products = Arrays.asList("product1",
"product2");
  }

  @AfterEach
  void teardown() {
      products.clear();
  }
```

@BEFOREALL/@AFTERALL

- Se ejecuta antes de crear/después de destruir la instancia, por lo que se implementa en un método estático.
- Si se anota en un método no estático, éste fallará, ya que la instancia no existe.
- Se puede forzar que la instancia sólo se cree una vez, aunque es mala práctica, pues compartes el estado de la clase entre tests.
- @Creando una instancia por clase, nos permite quitar el static a @BeforeAll/@AfterAll



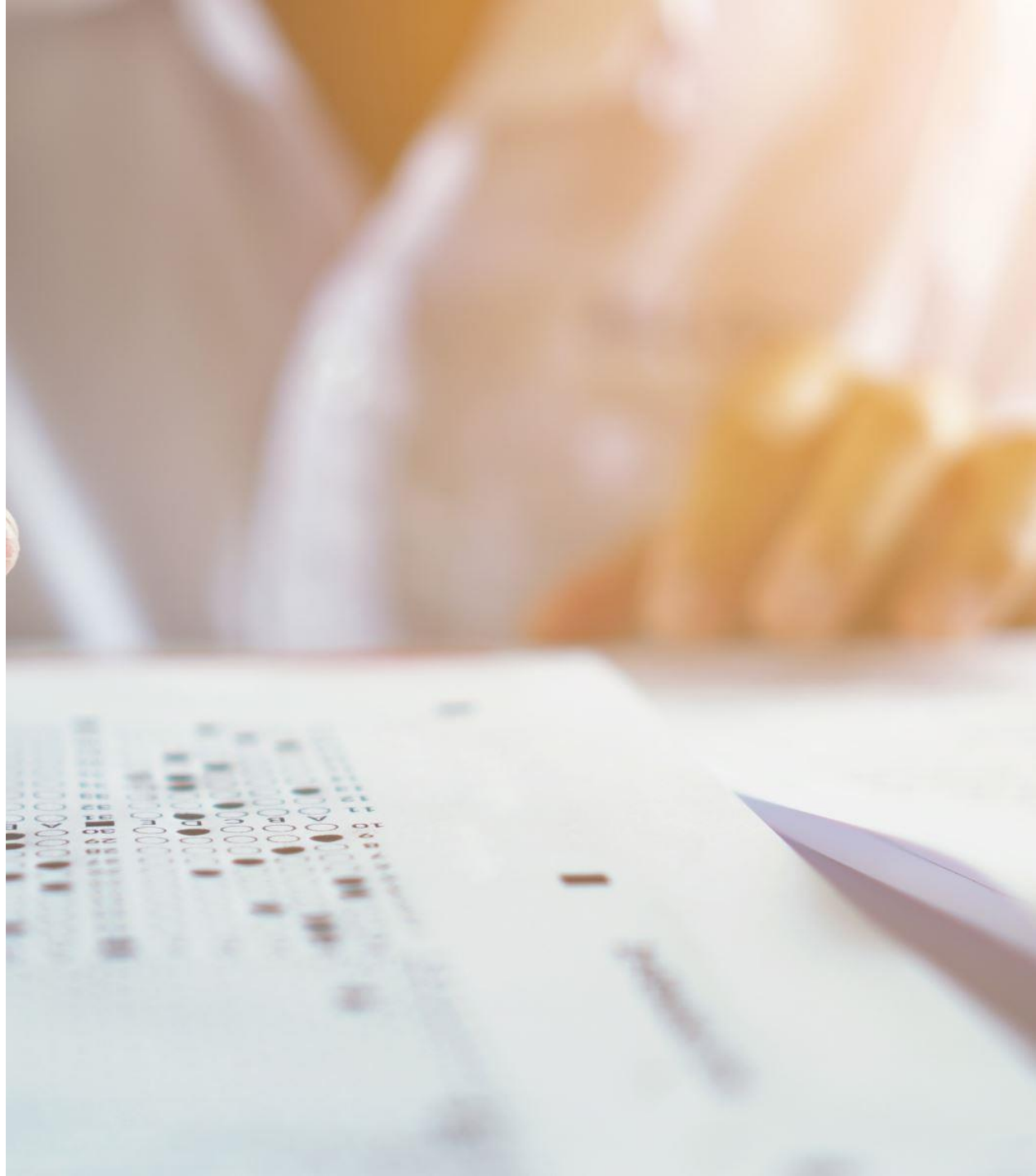
DESHABILITANDO TESTS UNITARIOS

- Los tests pueden deshabilitarse para evitar su ejecución.
- Anotamos el test con @Disabled ○ JUnit 5 recomienda especificar un motivo:
- @Disabled(“Se deshabilita este test hasta que el bug @B54 se resuelva”)

TESTS CONDICIONALES - ANOTACIONES

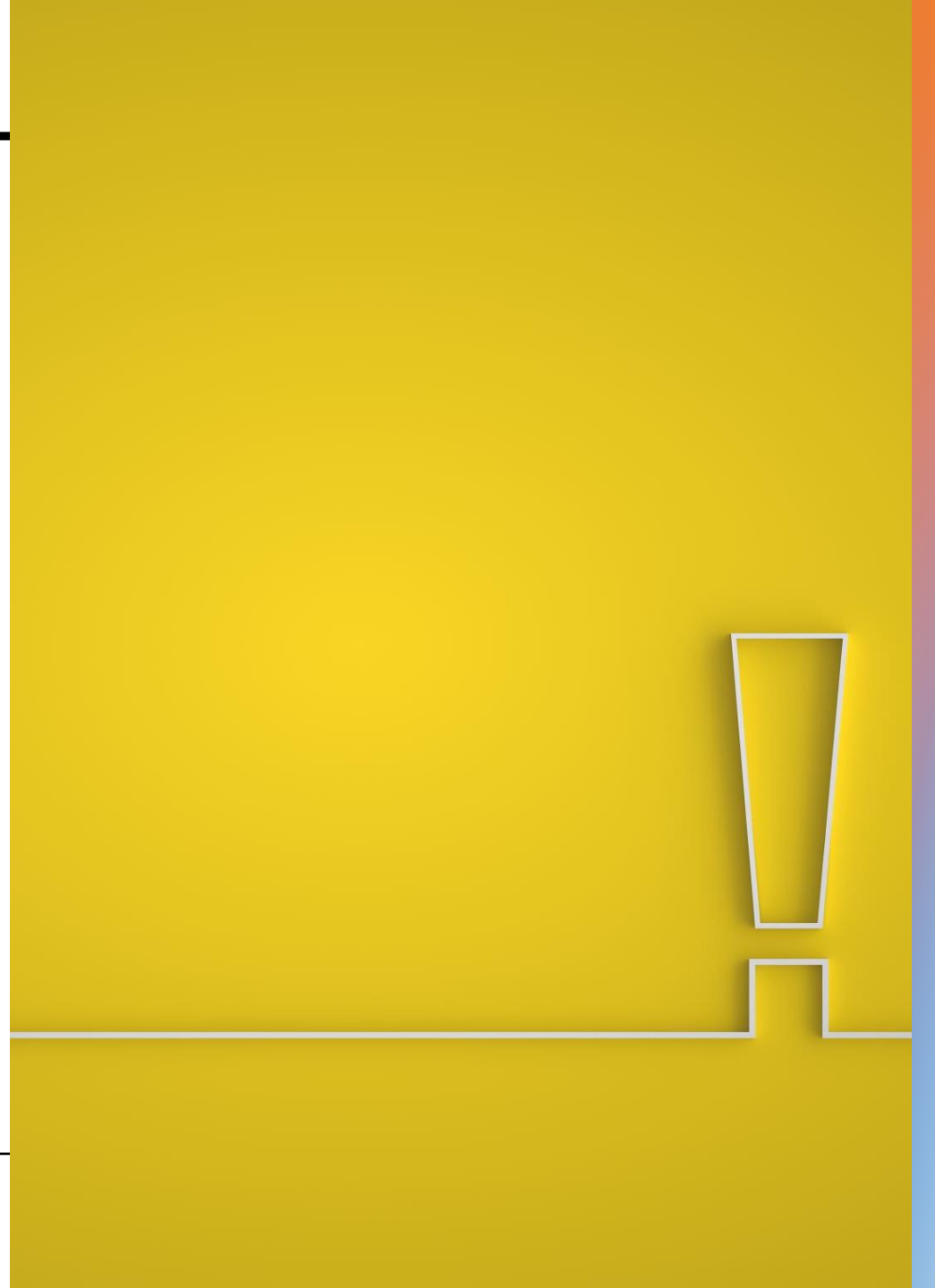
- Las pruebas unitarias se pueden ejecutar conforme a diferentes condiciones.
- JUnit ofrece anotaciones para habilitar o no dichos tests: @EnabledOnOS @EnabledOnJre @EnabledIfSystemProperty @EnabledIfEnvironmentVariable

```
@EnabledIf(“hasStock”)
@Test
void it_should_decrease_stock() {
}
boolean hasStock(){
}
```



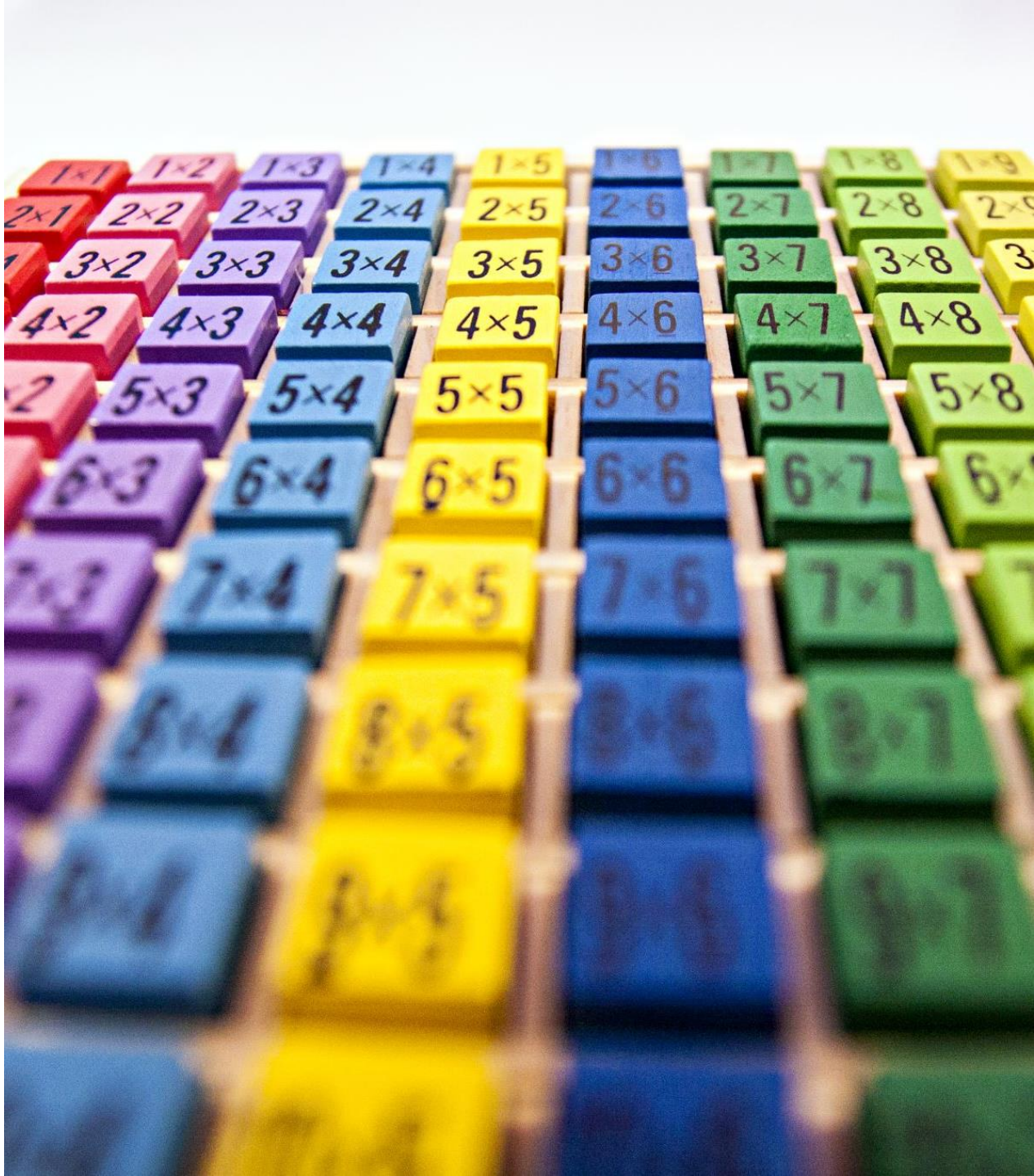
TESTS CONDICIONALES

- Permiten habilitar parte de un test en función de si se cumple una condición. assumeTrue(boolean condition); assumeFalse(boolean condition);
- Si la assumption no se cumple, el código a partir de ahí se deshabilita, evitando el fallo
- JUnit 5 permite ejecutar o no parte del método encapsulándolo en una expresión lambda. ○ assumingThat(boolean, () -> {});
- La expresión lambda sólo se va a ejecutar si la expresión evalúa a true.
- El código fuera de la expresión lambda, sí se ejecutará.



CLASES ANIDADAS

- Las clases anidadas permiten organizar los tests por diferentes criterios: funcionalidad, condicionalidad… Se anotan las clases con @Nested.
- Se puede incluir una descripción en dichas clases y los métodos que contiene con @DisplayName. Los tests aparecerán en el reporting agrupados por clases. Si falla un test de una clase @Nested, aparecerá como fallo el test y las clases contenedoras del mismo (hasta la clase raíz).

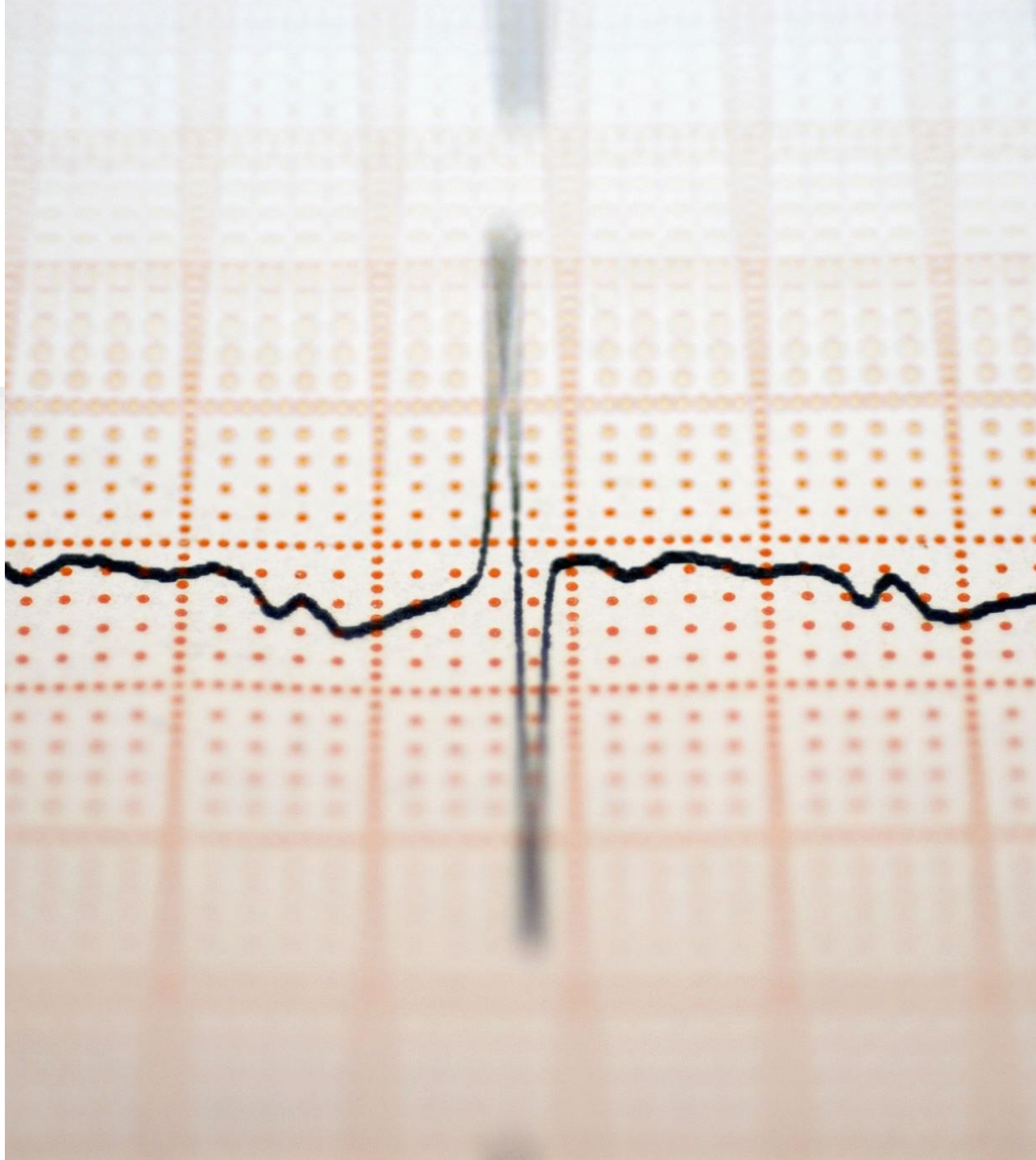


REPETIR TESTS

- JUnit 5 permite ejecutar varias veces el test. Útil en métodos que presentan cierta aleatoriedad. ○ Por ejemplo, crean valores random. Se anotan con @RepeatedTest(int repetitions)
- En el reporting, aparece la ejecución de las repeticiones. Es personalizable el nombre en cada ejecución del test. @RepeatedTest(int, message)
- Podemos usar variables para la creación de ese mensaje: {currentRepetition}{totalRepetitions}
- Podemos combinar el nombre con @DisplayName. @DisplayName: Será el título principal. @RepeatedTest: Nombre en cada repetición. @DisplayName puede ser inyectado en el mensaje de @RepeatedTest: ○ {displayName}

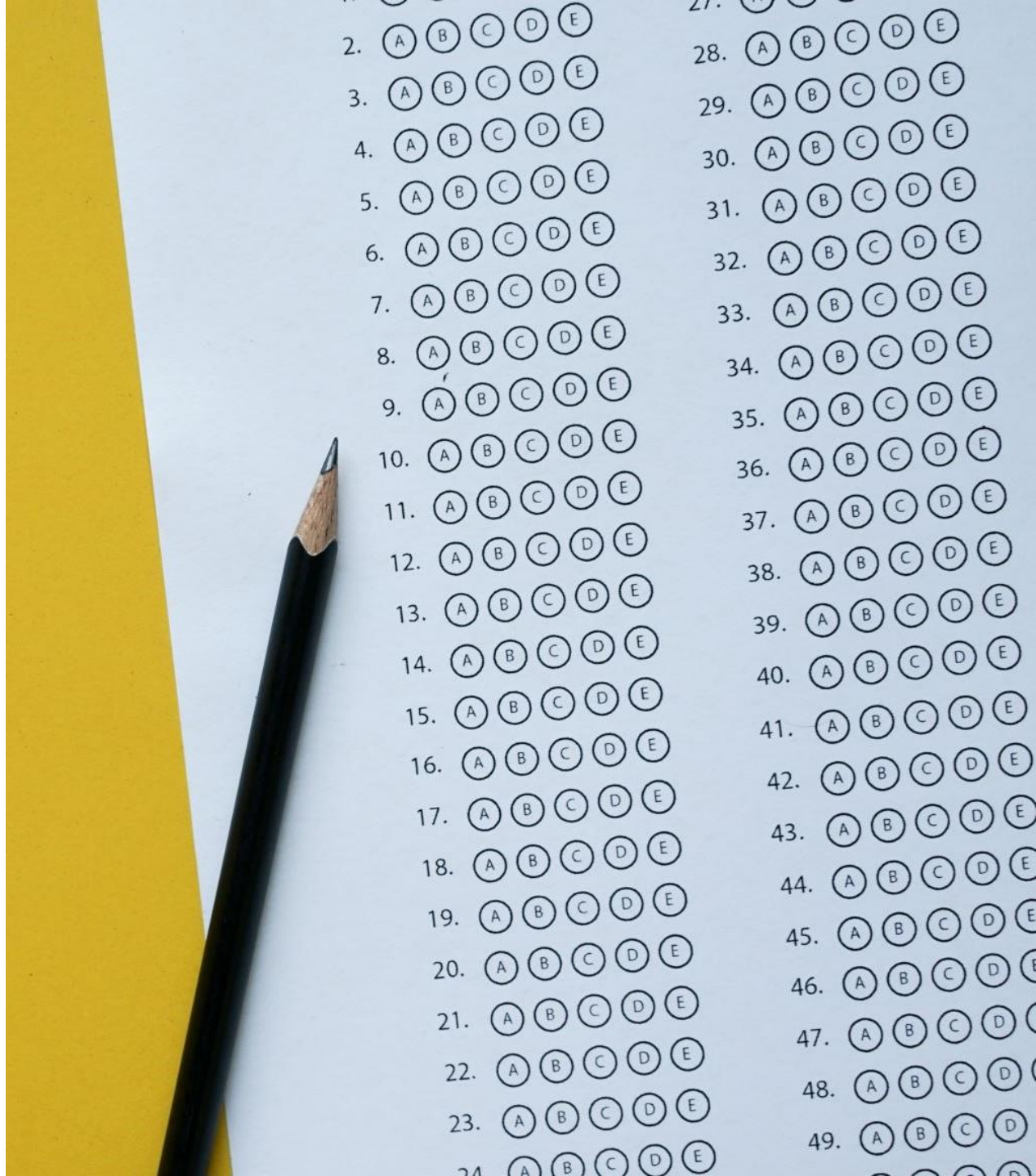
TESTS PARAMETRIZADOS

- Otra forma de repetir tests en JUnit 5.
- Permite que en cada repetición se ejecute con datos diferentes. Se inyectan mediante variables en los métodos.
- JUnit 5 permite proporcionar dichos datos mediante: @ValueSource(strings={}) Otros tipos: ints, doubles, booleans. @CsvSource({índice, valor}) @CsvFileSource(resources, delimeter, numLinesToSkip) @MethodSource(static methodName)



FILTRAR TESTS

- Permite ejecutar los tests de forma selectiva.
- Útil para identificar un test con el id de una tarea. Posibilidad de ejecutar grupos de tests.
- Se anota el test o la clase con @Tag. Se puede anotar con varios @Tag.
- Se especifican en el RunConfiguration / Maven.



EJERCICIO 2


packages
src
test
types
.babelrc.js
.editorconfig
.eslintignore
.eslintrc.js
.flowconfig
.gitignore
BACKERS.md
v2.6.0-beta.2
build: release 2.6.0-beta.2
build: build 2.6.0-beta.2
feat: dynamic directive arguments for v-on, v-bind and custom
origin/dynamic-directive-arguments
feat: dynamic args for custom
perf: improve scoped slots change detection accuracy (#9371)
test: test cases for v-on/v-bind dynamic arguments
refactor: v-bind dynamic arguments use bind helper
test: fix tests, resolve helper conflict
fix: fix middle modifier
feat: handle dynamic argument for v-bind.sync
origin/slot-optimization
perf: improve scoped slots
feat: dynamic directive arguments for v-bind and v-on
refactor: extend dom-props update skip to more
fix: fix checkbox event edge case in Firefox
test: fix tests in IE/Edge

mockito

¿POR QUÉ UTILIZARLO?

- Permite escribir y ejecutar tests unitarios de código integrado por varios componentes.
- Simula el comportamiento de componentes (mock, spy).
- Proporciona rapidez en las pruebas.
- Útil para seguir el paradigma TDD/BDD.
- Ejemplos de uso: Conexiones con BD. Servicios web. Clases de lenta ejecución. Clases con side-effects. Clases con un comportamiento indefinido.

MOCKITO BDD

Mockito permite basarnos en el comportamiento de otros componentes:

Dado que (Given): Preparación estado inicial.

Cuando (When): Se invoca al método.

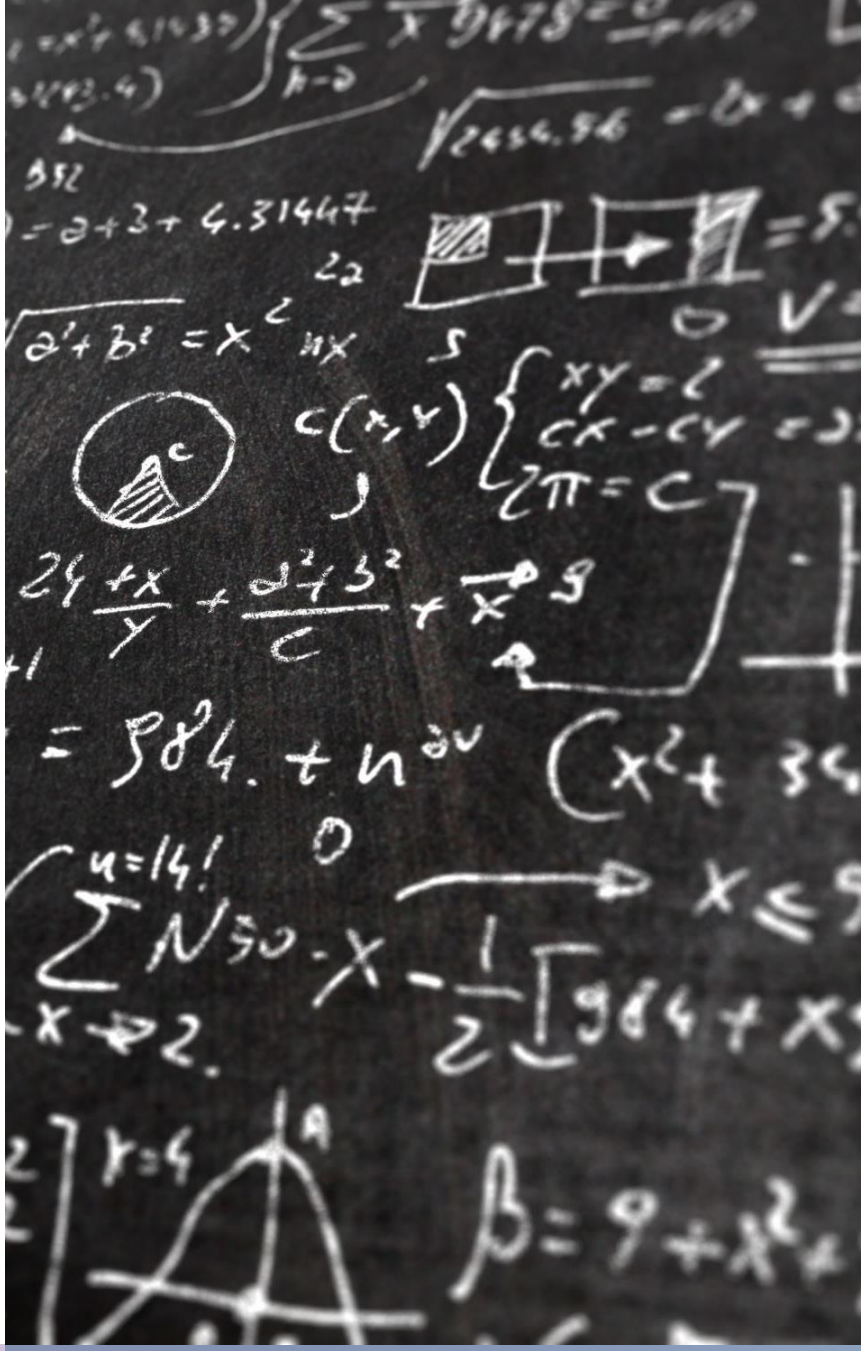
Entonces (Then): Validamos el comportamiento esperado

PRIMEROS TESTS

```java
@Test
Run Test | Debug Test | ✓
void our_first_test_with_mockito() {
    // Creamos el objeto ficticio.
    List<String> first_mock = mock(ArrayList.class);

    // Simulamos el comportamiento
    when(first_mock.get(0)).thenReturn("first_element");

    String first_element = first_mock.get(0);

    // Verificamos
    assertEquals("first_element", first_element);
    verify(first_mock).get(0);

}
```

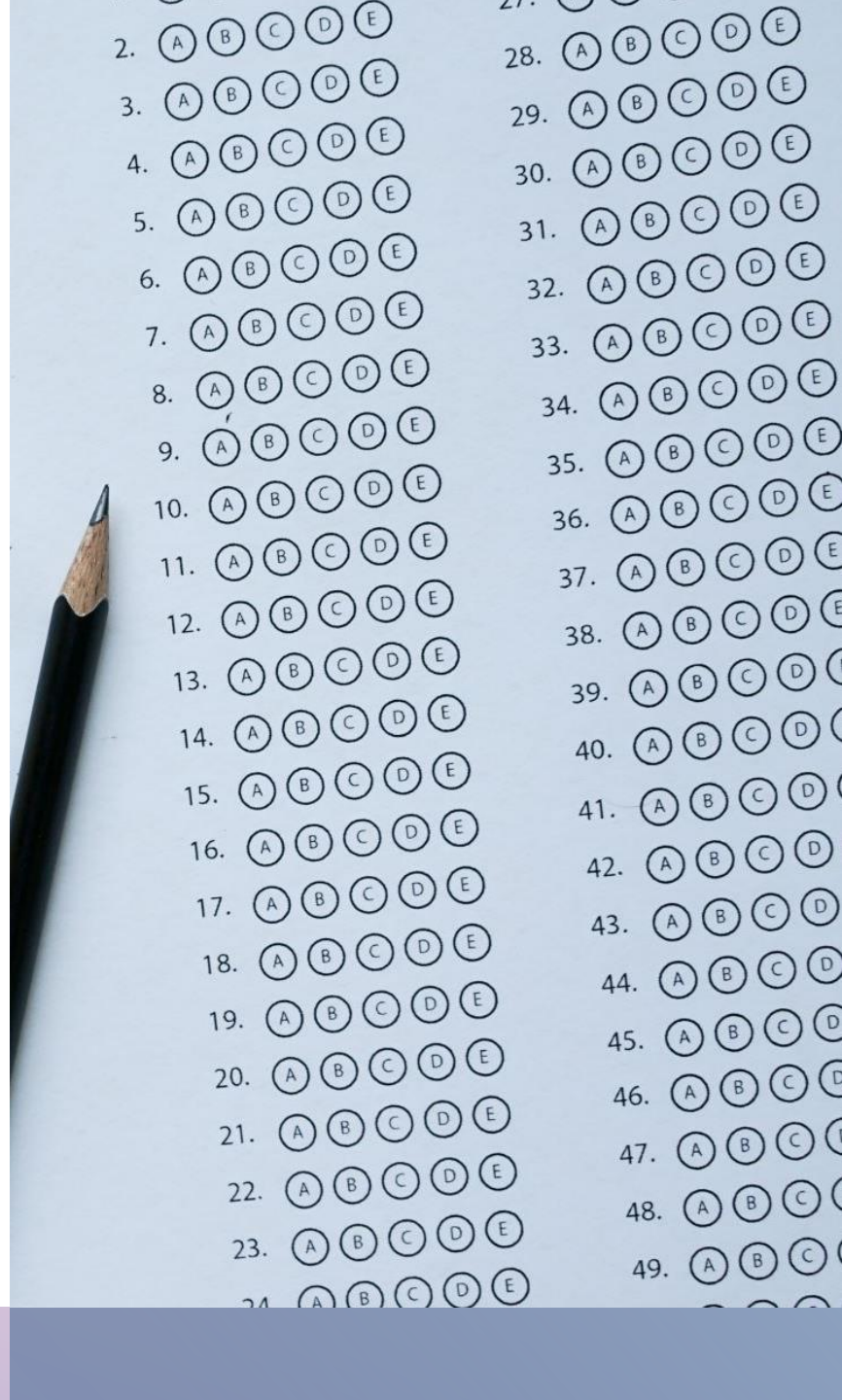
ARGUMENT MATCHERS

- Los ArgumentMatchers de Mockito permiten:
- Verificar argumentos con los que son llamados métodos de los mocks.
- Simular el comportamiento de un componente.
- eq() any(), anyInt(), anyString()... isNull(), isNotNull(), isA(Class c)



VERIFY

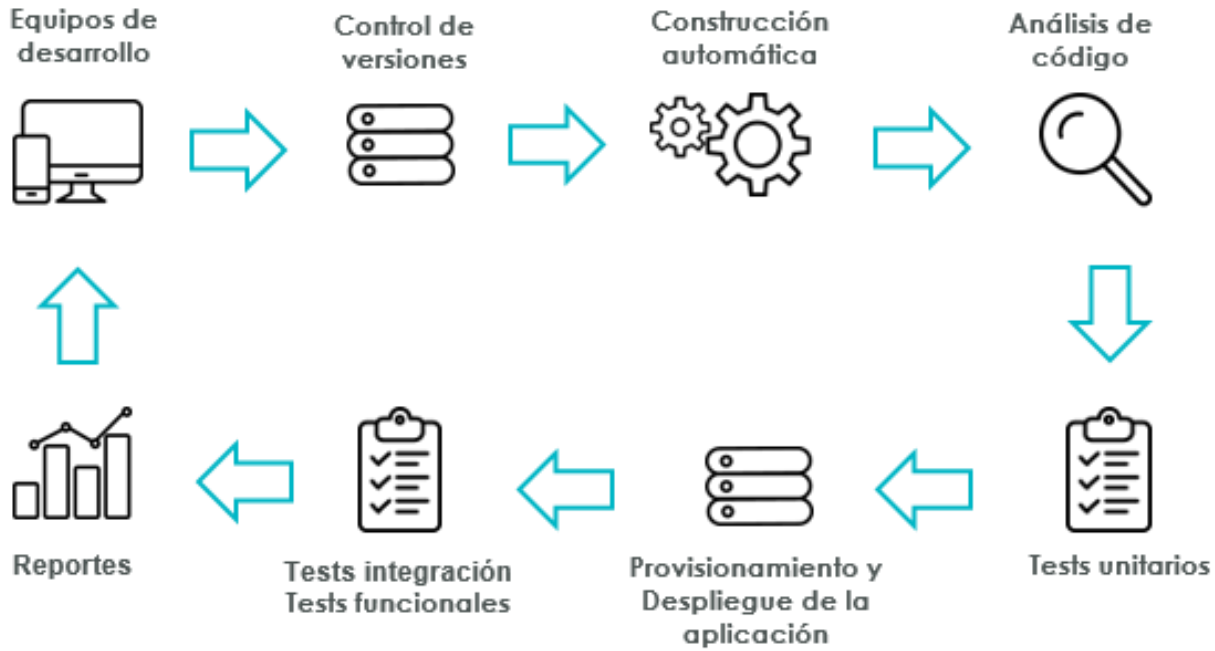
- Método estático.
- Permite testear la ejecución de un mock.
- verify(mock, [times(int number)]).método()
- verify(mock, never()).método()
- verifyNoInteractions()
- verify(mock, {atLeast(int), atMost(int), atLeastOnce(), atMostOnce()}

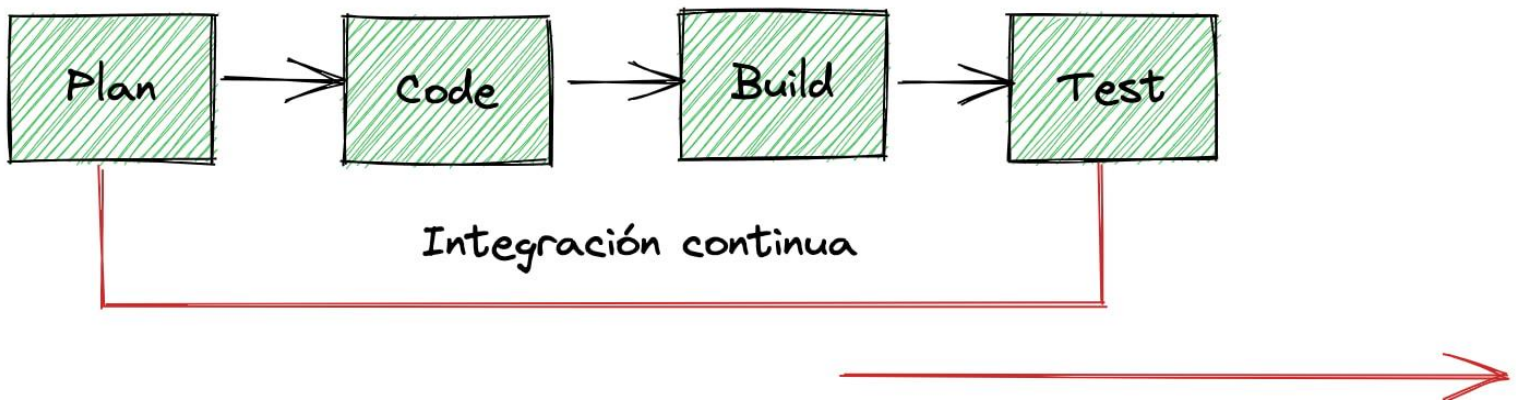


Modulo 3: **Jenkins**

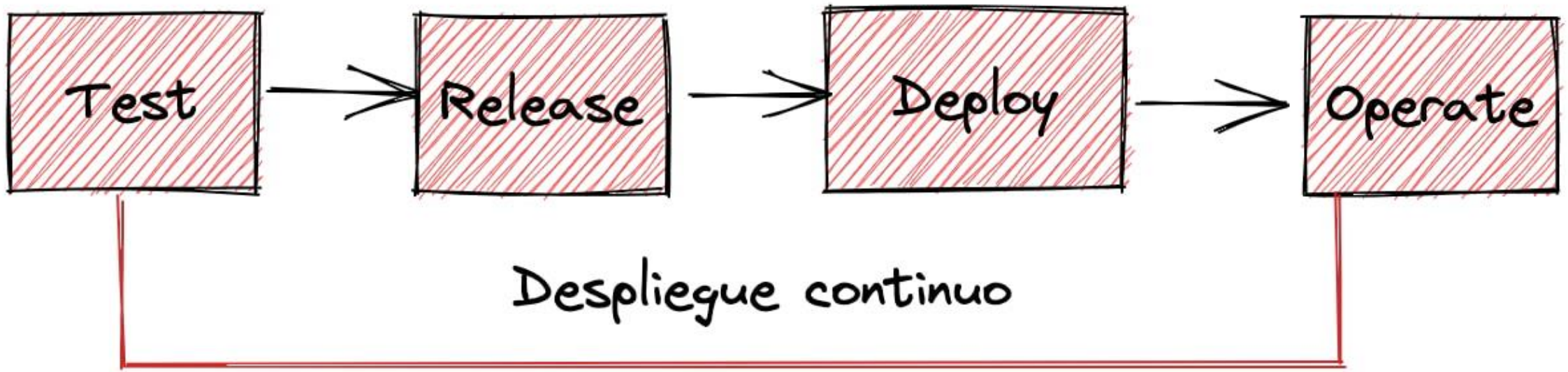
- Introducción
- Instalación y configuración
- Configuración de trabajos de ejecución
- Interfaz Git e integración con BitBucket
- Invocación de comandos y scripts externos
- Integración con Maven
- Notificaciones

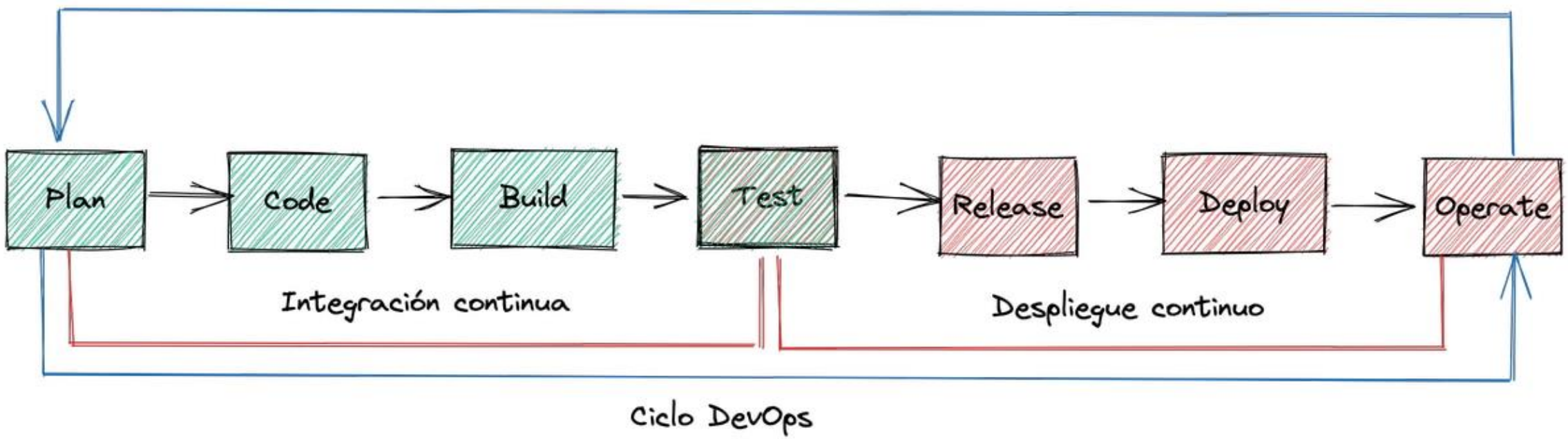

DevOps
INTEGRACIÓN CONTINUA
6 BENEFICIOS
1
Mejora la calidad del código
2
Detección de errores más rápida y fácil
3
Reduce tareas repetitivas y manuales
4
Puede crear versiones de pruebas en cualquier momento
5
Completa visibilidad del proyecto
6
Mayor confianza y seguridad del equipo de trabajo


Equipos de desarrollo
Control de versiones
Construcción automática
Análisis de código
Tests unitarios
Provisionamiento y Despliegue de la aplicación
Tests integración
Tests funcionales
Reportes

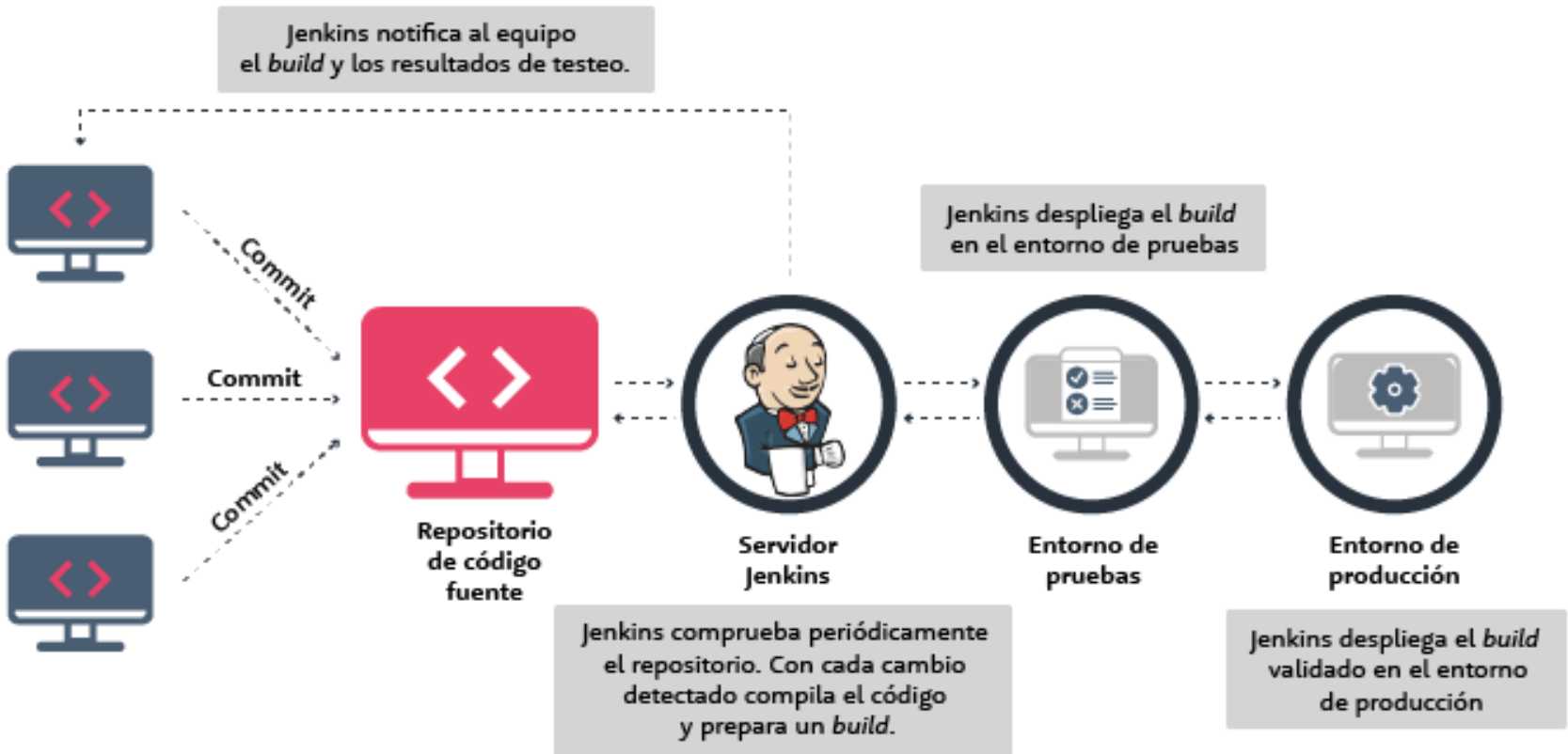


ETAPAS


Test
Release
Deploy
Operate
Despliegue continuo

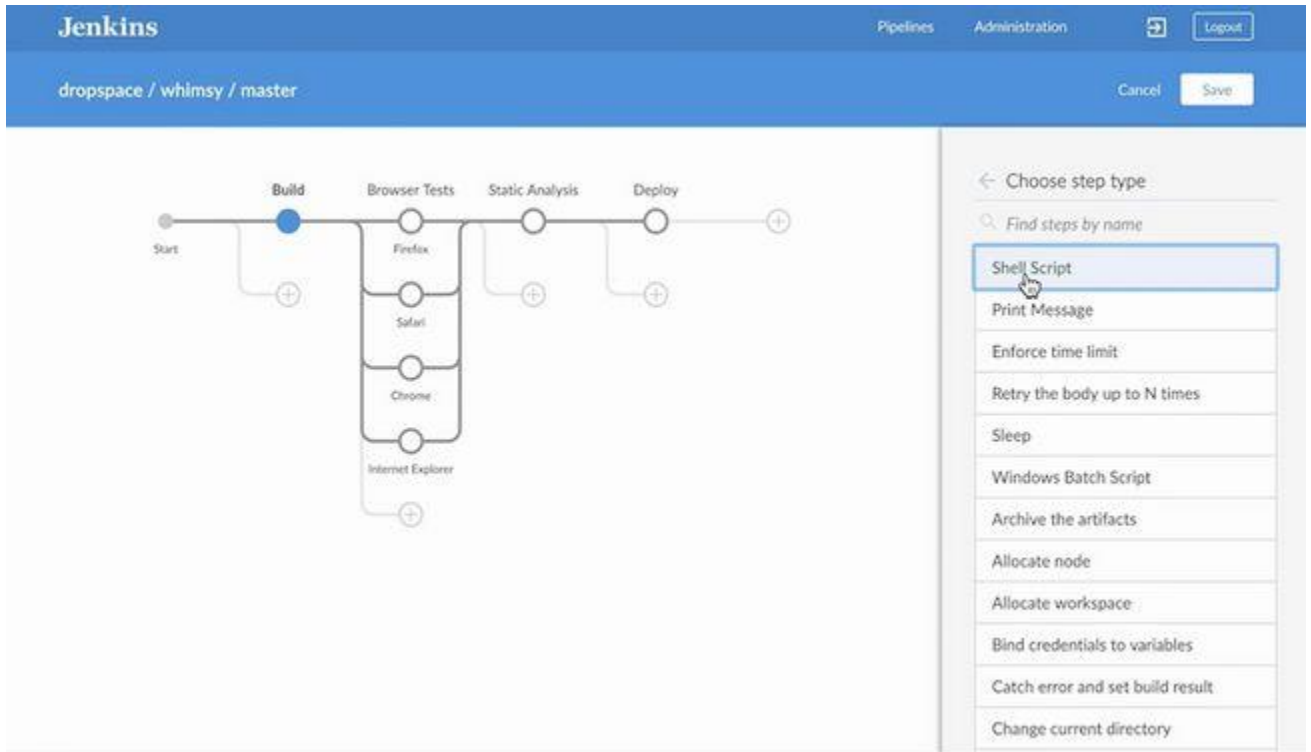

Plan
Code
Build
Test
Release
Deploy
Operate
Integración continua
Despliegue continuo
Ciclo DevOps

Jenkins


Jenkins notifica al equipo el *build* y los resultados de testeo.
Commit
Commit
Commit
Repositorio de código fuente
Jenkins despliega el *build* en el entorno de pruebas
Servidor Jenkins
Entorno de pruebas
Entorno de producción
Jenkins comprueba periódicamente el repositorio. Con cada cambio detectado compila el código y prepara un *build*.
Jenkins despliega el *build* validado en el entorno de producción

Instalación y configuración


Compilar
Test
Construcción
Despliegue
3s
6s
6s
5s
3s
5s
failed
3s
6s
5s
5s


Jenkins
Pipelines
Administration
Logout
dropspace / whimsy / master
Cancel
Save
Build
Browser Tests
Static Analysis
Deploy
Start
Firefox
Safari
Chrome
Internet Explorer
Choose step type
Find steps by name
Shell Script
Print Message
Enforce time limit
Retry the body up to N times
Sleep
Windows Batch Script
Archive the artifacts
Allocate node
Allocate workspace
Bind credentials to variables
Catch error and set build result
Change current directory