UNIVERSITÉ CÔTE D'AZUR

Département INFORMATIQUE
IUT Nice Côte d'Azur

Introduction à l'Intégration Continue

Johann Mortara

IUT Nice Côte d'Azur
Département Informatique

13 Octobre 2020

Rappels sur les tests et la qualité du code

Règle n°1 : Les tests, c'est TRÈS important !

Règle n°2 : Écrire des tests, c'est bien. Les exécuter, c'est mieux.

Ce qu'on vous a appris :

1. Coder
2. Écrire les tests
3. S'assurer que les tests passent
4. Pousser le code sur Git

Attends c'est pas ça qu'il faut faire ?

M'aurait-on donc menti ?

Je vois pas où est le problème...

Je savais bien que ça servait à rien tout ça !

C’est une bonne pratique !

Tester, à quoi ça sert ?

Tester, à quoi ça sert ?

1. Vérifier que le code qui vient d'être écrit a le comportement attendu…

Tester, à quoi ça sert ?

1. Vérifier que le code qui vient d'être écrit a le comportement attendu…

2. … et qu'il n'a pas altéré le reste du projet !

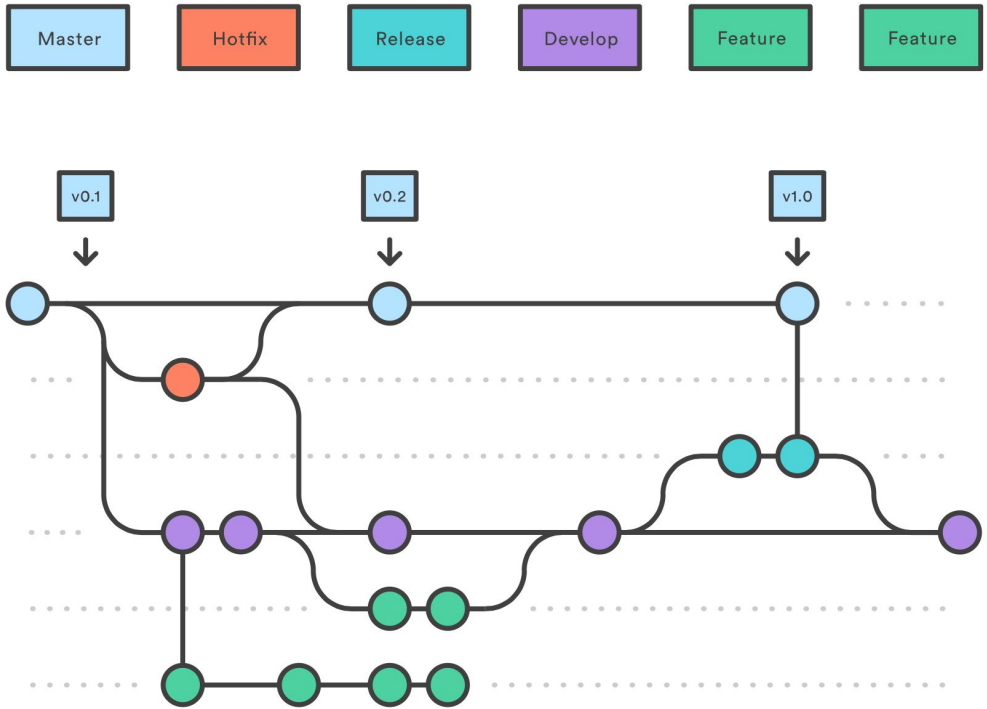
Tester, à quoi ça sert ?

1. Vérifier que le code qui vient d'être écrit a le comportement attendu…

2. … et qu'il n'a pas altéré le reste du projet !

⇒ maîtriser le code **et son évolution**

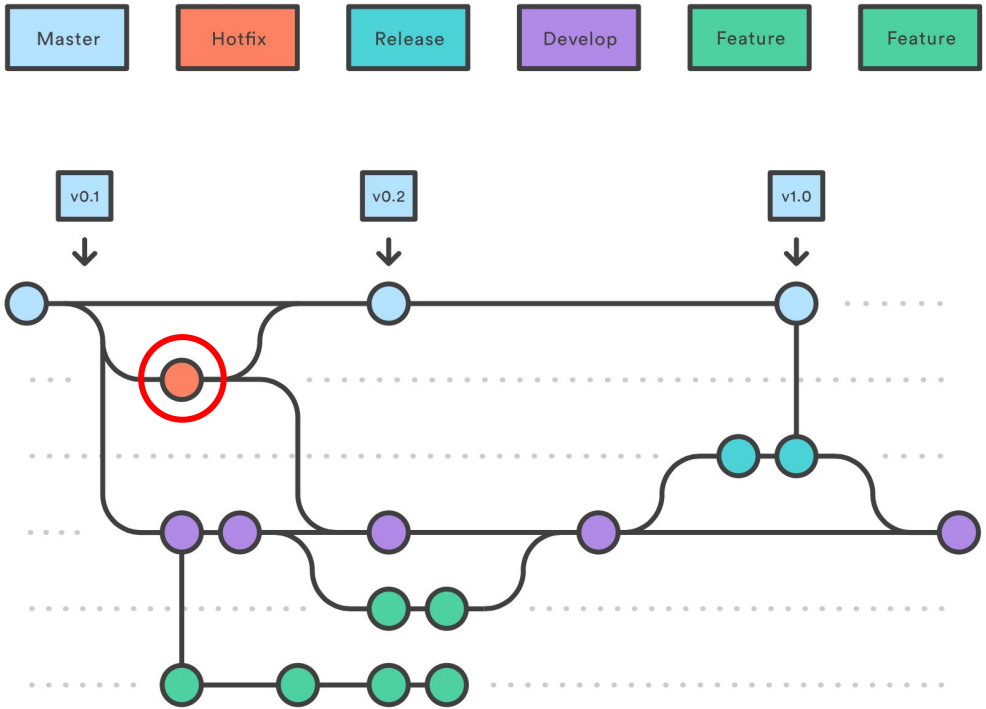
Gérer son développement avec des branches



Source : https://blog.engineering.publicissapient.fr/2018/03/28/gitflow-est-il-le-workflow-dont-jai-besoin/

Gérer son développement avec des branches

Hotfix : nouveaux tests pour **reproduire le bug**

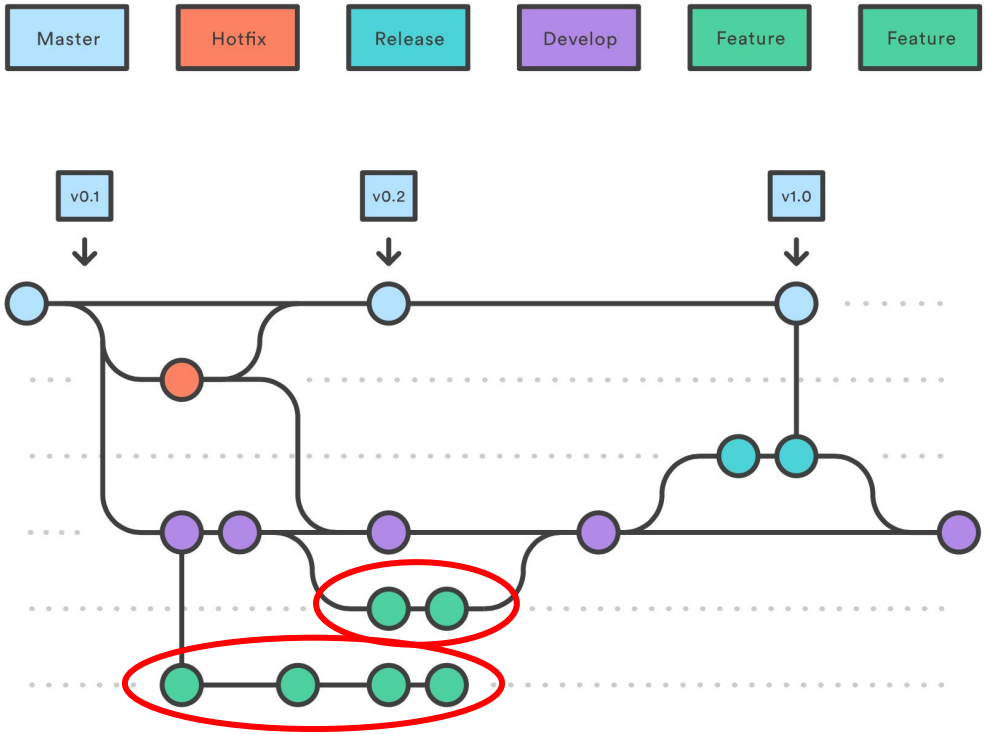


Source : https://blog.engineering.publicissapient.fr/2018/03/28/gitflow-est-il-le-workflow-dont-jai-besoin/

Gérer son développement avec des branches

Hotfix : nouveaux tests pour **reproduire le bug**

Feature : nouveaux tests pour les **nouvelles fonctionnalités**



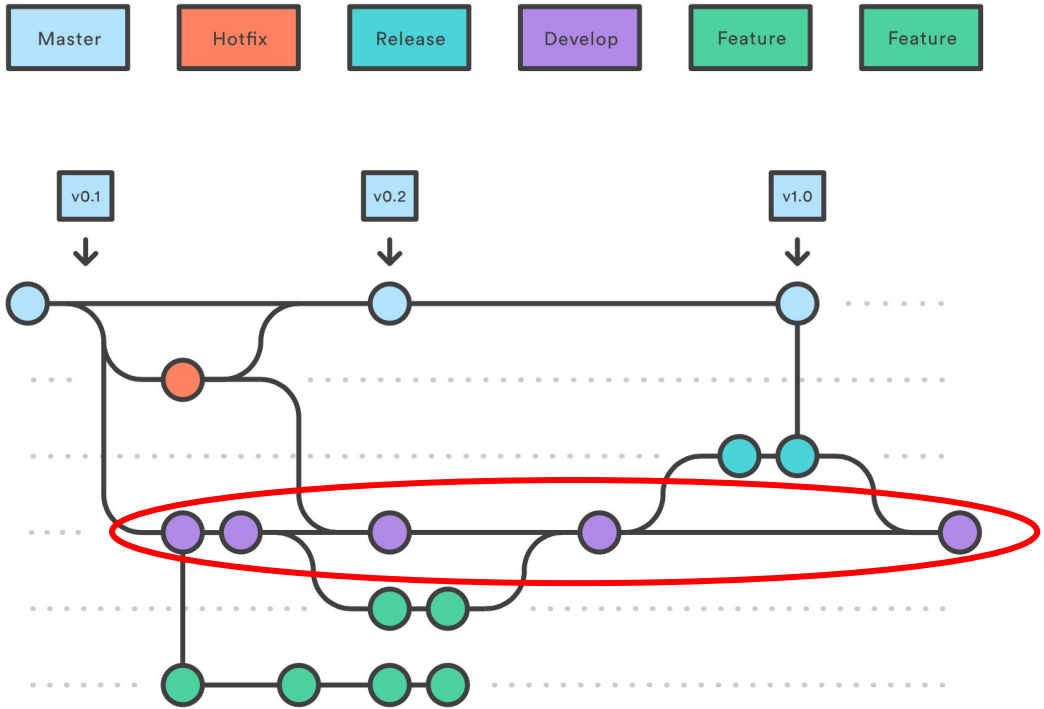
Source : https://blog.engineering.publicissapient.fr/2018/03/28/gitflow-est-il-le-workflow-dont-jai-besoin/

Gérer son développement avec des branches

Hotfix : nouveaux tests pour **reproduire le bug**

Feature : nouveaux tests pour les **nouvelles fonctionnalités**

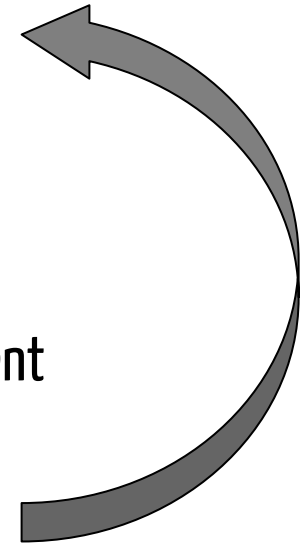
Develop : **intégration des features** et exécution de tous les tests



Source : https://blog.engineering.publicissapient.fr/2018/03/28/gitflow-est-il-le-workflow-dont-jai-besoin/

Ce qu'on vous a appris :

1. Coder
2. Écrire les tests
3. S'assurer que les tests passent
4. Pousser le code sur Git



Ce qu'on vous a appris (v2) :

1. Coder
2. Écrire les tests
3. S'assurer que les tests passent
4. Pousser le code sur Git

Maîtrise de l'évolution

5. S'assurer que **tous** les tests passent toujours

Maîtrise du code

Ce qu'on vous a appris (v2) :

1. Coder
2. Écrire les tests
3. S'assurer que les tests passent
4. Pousser le code sur Git

Maîtrise du code

Maîtrise de l'évolution

5. S'assurer que **tous** les tests passent toujours

⇒ **Non-régression**

Question :

Combien de temps faut-il pour exécuter tous les tests d'un projet ?

A. < 5 minutes

B. < 30 minutes

C. Plusieurs heures

D. Ça dépend

Mozilla Firefox

Settings | Report Duplicate

Very High Activity

13,463

I Use This!

Analyzed about 4 hours ago. based on code collected about 5 hours ago.

Languages

Total Lines :	31,189,242	Code Lines :	22,559,384	Percent Code Lines :	72.3%
Number of Languages :	48	Total Comment Lines :	4,799,854	Percent Comment Lines :	15.4%
		Total Blank Lines :	3,830,004	Percent Blank Lines :	12.3%

Source : https://www.openhub.net/p/firefox/analyses/latest/languages_summary

Mozilla Firefox

Settings | Report Duplicate

Very High Activity

13,463

I Use This!

Languages

Analyzed about 4 hours ago. based on code collected about 5 hours ago.

Total Lines :	31,189,242	Code Lines :	22,559,384	Percent Code Lines :	72.3%
Number of Languages :	48	Total Comment Lines :	4,799,854	Percent Comment Lines :	15.4%
		Total Blank Lines :	3,830,004	Percent Blank Lines :	12.3%

Source : https://www.openhub.net/p/firefox/analyses/latest/languages_summary

Mozilla Firefox

Settings | Report Duplicate

Very High Activity

13,463

I Use This!

Languages

Analyzed about 4 hours ago. based on code collected about 5 hours ago.

Total Lines :	31,189,242	Code Lines :	22,559,384	Percent Code Lines :	72.3%
Number of Languages :	48	Total Comment Lines :	4,799,854	Percent Comment Lines :	15.4%
		Total Blank Lines :	3,830,004	Percent Blank Lines :	12.3%

Tout doit être testé !

Source : https://www.openhub.net/p/firefox/analyses/latest/languages_summary

15/09/2019

Total Lines :	28,730,555	Code Lines :	20,548,088	Percent Code Lines :	71.5%
Number of Languages :	48	Total Comment Lines :	4,575,107	Percent Comment Lines :	15.9%
		Total Blank Lines :	3,607,360	Percent Blank Lines :	12.6%

+ 2M LoC

09/10/2020

Total Lines :	31,189,242	Code Lines :	22,559,384	Percent Code Lines :	72.3%
Number of Languages :	48	Total Comment Lines :	4,799,854	Percent Comment Lines :	15.4%
		Total Blank Lines :	3,830,004	Percent Blank Lines :	12.3%

Source : https://www.openhub.net/p/firefox/analyses/latest/languages_summary

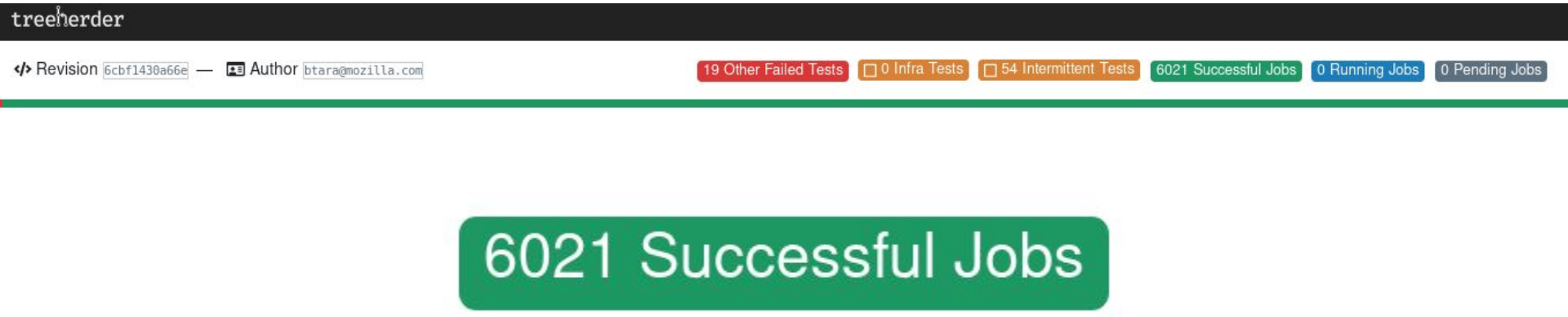
Comment s’assurer que tout ça fonctionne ?

Comment s’assurer que tout ça fonctionne ?

Avec des tests !

Beaucoup de tests...

15/09/2019



Source : https://treeherder.mozilla.org/testview.html?repo=mozilla-central&revision=6cbf1430a66ef5aa5f066ed77677f47ea44fc0e1

Temps total d'exécution des tests de Firefox :

950 h ≃ 40 jours

Réponse :

Combien de temps faut-il pour exécuter tous les tests d'un projet ?

~~A. < 5 minutes~~

~~B. < 30 minutes~~

~~C. Plusieurs heures~~

D. Ça dépend

... et ça peut être énorme !

Question :

Une nouvelle version de Firefox est déployée...

A. toutes les 12 heures

B. toutes les 3 semaines

C. tous les mois

D. tous les 2 mois

Release timeline

Firefox is released at intervals of six to eight weeks (not counting urgent patch updates), meaning that every six to eight weeks there will be a new version of Firefox Release.

From mozilla-central to mozilla-release

- Firefox Nightly is released every 12 hours with all the changes landed on mozilla-central.
- Every 6 to 8 weeks, we merge the code from mozilla-central to our mozilla-beta branch. The mozilla-beta branch should now only get patches aimed at stabilizing the release. Any patch on mozilla-central that we want backported to our mozilla-beta branch should follow the approval rules for uplifts.
- Beta 1 and Beta 2 are built from this beta branch and used to build and ship Firefox Developer Edition as a stabilization step before shipping Firefox Beta to our much wider Beta audience.
- Starting with Beta 3, Firefox Beta is released twice a week for Desktop, leaving us with 12 to 16 betas every cycle unless we have chemspills leading to additional betas. Firefox Beta 3 is shipped to a subset of our Beta population. The full Beta population gets updated starting with beta 4 only.
- At the end of the Beta cycle, a final build is validated by our QA and tagged for release into the mozilla-release branch.

6 to 8 weeks

every 12 hours

Source : https://wiki.mozilla.org/Release_Management/Release_Process

Réponse :

Une nouvelle version de Firefox est déployée…

A. toutes les 12 heures

~~B. toutes les 3 semaines~~

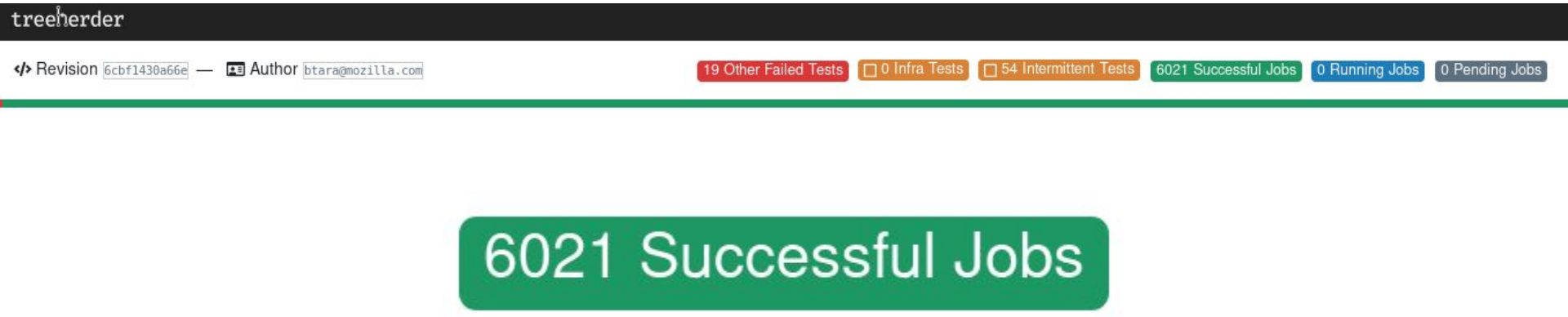
~~C. tous les mois~~

D. tous les 2 mois

… et c'est testé !

15/09/2019

Comment est-ce possible ?



Source : https://treeherder.mozilla.org/testview.html?repo=mozilla-central&revision=6cbf1430a66ef5aa5f066ed77677f47ea44fc0e1

L'intégration continue

C'est pas trop tôt !

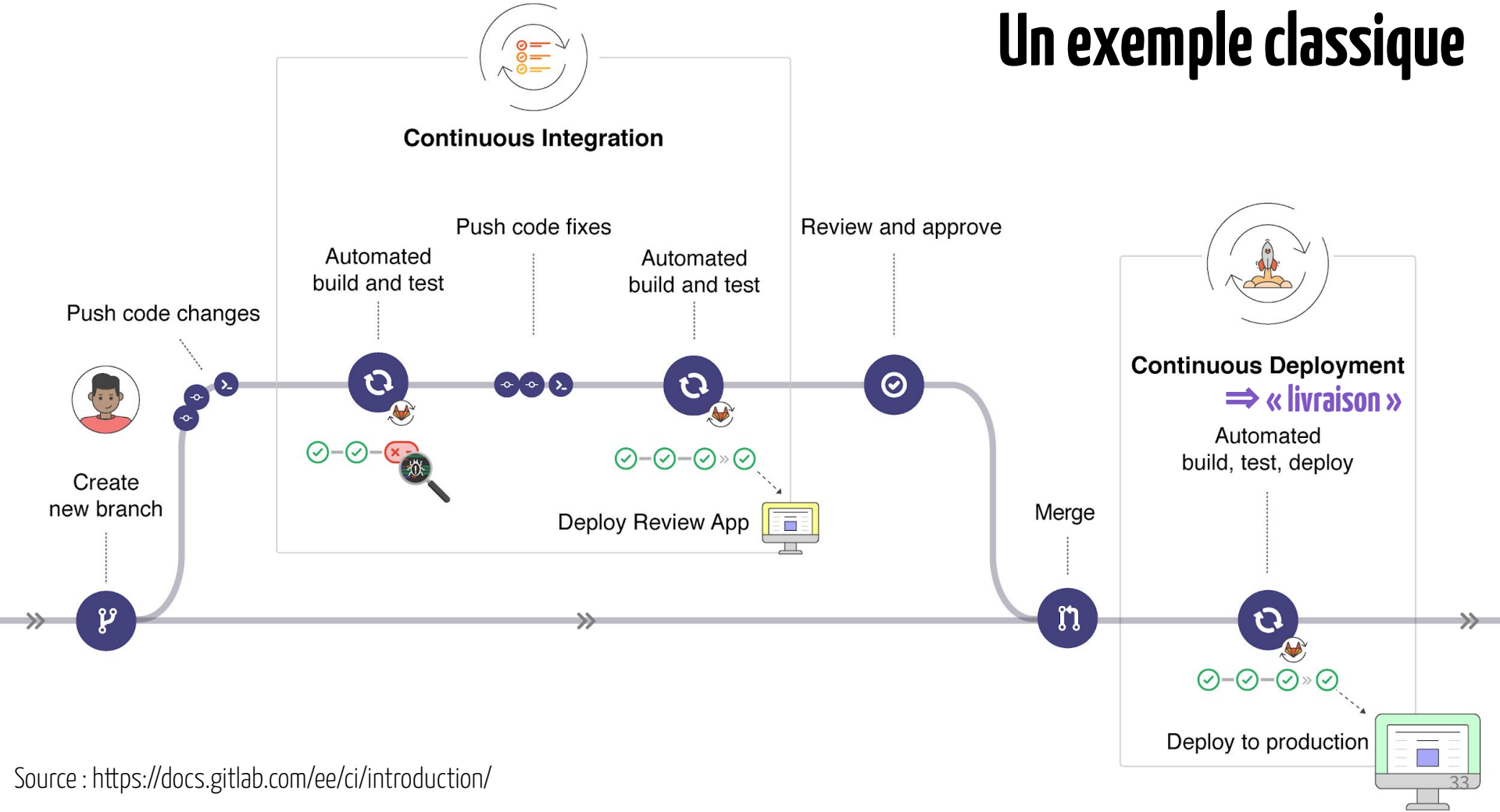
Principe de base

Automatiser la construction

du logiciel pour pouvoir assurer la livraison

d'un système qui fonctionne

Un exemple classique



Source : https://docs.gitlab.com/ee/ci/introduction/

Firefox, c'est...

- De l'affichage
- Des fichiers de configuration (`about:config`)
- Une base de données embarquée
- Un support pour des plugins personnalisés
- La gestion de plusieurs protocoles
- Support de fichiers PDF, MP3, etc...

Et bien d'autres choses encore !

Firefox, c'est…

- De l'affichage
- Des fichiers de configuration ← **TUs**
- Une base de données embarquée ← **TUs**
- Un support pour des plugins personnalisés
- La gestion de plusieurs protocoles
- Support de fichiers PDF, MP3, etc… ← **TUs**

Et bien d'autres choses encore !

Firefox, c'est...

- De l'affichage
- Des fichiers de configuration ← TUs
- Une base de données embarquée ← TUs
- Un support pour des plugins personnalisés
- La gestion de plusieurs protocoles
- Support de fichiers PDF, MP3, etc... ← TUs

Et bien d'autres choses encore !

Comment s'assurer que l'ouverture d'un fichier PDF reste fonctionnel quelle que soit ma configuration ?

De nouveaux types de test

Besoin de tester les briques séparément ET ensemble

⇒ tests d'intégration

Besoin d'essayer le déploiement sur plusieurs systèmes

⇒ tests opérationnels

Besoin de s'assurer qu'une fonctionnalité respecte les spécifications

⇒ tests fonctionnels

4 Testing levels
4.1 Unit testing
4.2 Integration testing
4.3 System testing
4.4 Operational acceptance testing
5 Testing types, techniques and tactics
5.1 Installation testing
5.2 Compatibility testing
5.3 Smoke and sanity testing
5.4 Regression testing
5.5 Acceptance testing
5.6 Alpha testing
5.7 Beta testing
5.8 Functional vs non-functional testing
5.9 Continuous testing
5.10 Destructive testing
5.11 Software performance testing
5.12 Usability testing
5.13 Accessibility testing
5.14 Security testing
5.15 Internationalization and localization
5.16 Development testing
5.17 A/B testing
5.18 Concurrent testing
5.19 Conformance testing or type testing
5.20 Output comparison testing

Source : https://en.wikipedia.org/wiki/Software_testing

De multiples variantes...

De multiples variantes de tests pour de nombreux supports

	Windows	OSX	Debian	Arch
stable (ESR)	✓	✓	✓	✓
latest	✓	✓	✓	✓
canary	✓	✗	✓	

Variantes de Firefox
(extrait)

https://treeherder.mozilla.org/#/jobs?repo=mozilla-central&tier=1%2C2%2C3&revision=600f47bbfeb2b8dd8feb52dc9b0df0c72e01da9e

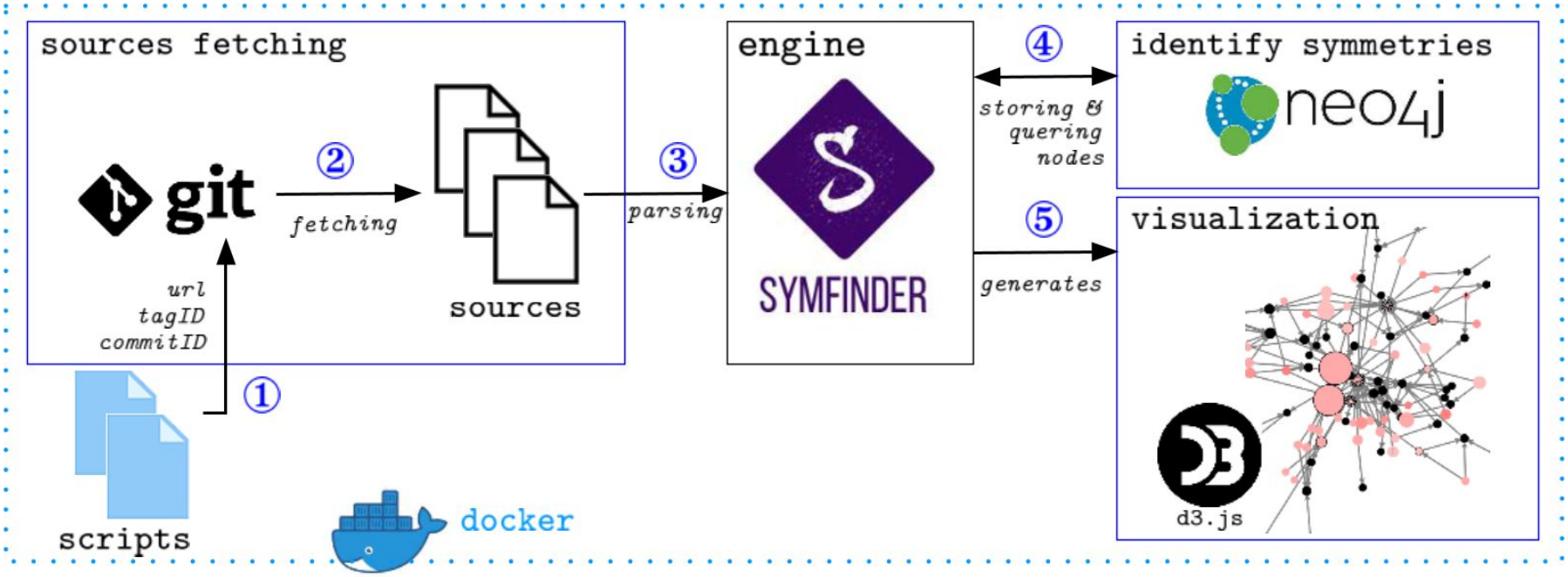
- Complete -

macosx64 opt	TM(gd-s)
win32 opt	TW32(gd-s)
win64 opt	TW64(gd-s)
Linux opt	BR [tier 3](+2) TL32(gd-s)
Linux debug	B BR Sym SM(arm) [tier 3](+2)
Linux shippable opt	Bs Rpk UgsB p(N ach af an ar ast az be bg bn bo br brx bs ca ca-valencia cak ckb cs cy da de dsb el en-CA en-GB eo es-AR es-CL es-ES es-MX et eu fa ff fi fr fy-NL ga-IE gd gl gn gu-IN he hi-IN hr hsb hu hy-AM hye ia id is it ja ka kab kk km kn ko lij lo lt ltg lv meh mk mr ms my nb-NO ne-NP nl nn-NO oc pa-IN pl pt-BR pt-PT rm ro ru scn si sk sl son sq sr sv-SE szl ta te th tl tr trs uk ur uz vi wo xh zh-CN zh-TW) ms(+106) ps(+106) BMR(+106) Bpgo(+5) L10n(B19* B20* +42) c-Up(+106) BMcs(+106) L10n-Rpk(+105)
Linux 18.04 shippable opt	W[tier 2](+10)
Linux x64 opt	B BR Bb Bbc Bocf V [tier 2](AB Bp) SM(+6) TL(gd-s) [tier 3](Bg +2) SM[tier 3](smoosh) l10n-bump(+11) condprof[tier 2](firefox) Static-Analysis[tier 2](coverity-full-analysis +2)
Linux x64 asan	Bd Bo Bocf Bof
Linux x64 tsan	Bo Bof
Linux x64 debug	B BR Bb Bbc Bf H Sa Sa Sym [tier 2](Bp) SM(+9) [tier 3](+2) SM[tier 2](+3) SM[tier 3](smoosh) Searchfox[tier 2](idx)
Linux 18.04 x64 tsan opt	M(20 +20) X(+8)
Linux x64 asan reporter opt	BoR BoRs Rpk UgsBoR ms(N) BMR(BoR) BMcs(BoR) c-Up(BoR)
Linux x64 WebRender Shippable opt	T(+16) Rap(amazon amazon-c imgur imgur-c mm-a tumblr tumblr-c twitch twitch-c twitter twitter-c wa) Rap[tier 2](apple apple-c ares6 bing bing-c dom ebay ebay-c fandom fandom-c fb fb-c fb-r fb-r-c gdoc gdoc-c gmail gmail-c godot godot-b godot-c godot-i google google-c gsheets gsheets-c gslides gslides-c imdb imdb-c instagram instagram-c linkedin linkedin-c microsoft microsoft-c mm-h netflix netflix-c office office-c outlook* outlook-c paypal paypal-c pinterest pinterest-c reddit reddit-c sb sp ss wikipedia wikipedia-c wm wm-b wm-c wm-i yahoo-mail yahoo-mail-c yahoo-news yahoo-news-c yandex yandex-c youtube youtube-c ytp ytp-h264 ytp-h264-sfr ytp-widevine-h264-sfr ytp-widevine-hfr) T-swr(+4) Rap[tier 3](+2) T-fis[tier 2](damp* +14) T-swr[tier 3](bcv) Btime[tier 3](+3) Rap-fis[tier 2](outlook* +80) Rap-live[tier 2](+2) Btime-Prof[tier 3](+2)
Linux x64 WebRender opt	WR(+2)
Linux x64 WebRender debug	WR(wrench)
Linux x64 shippable opt	Bs Rpk UgsB T(+17) p(bs +106) ms(+106) ps(+106) Rap(+12) T[tier 3](f) BMR(+106) Bpgo(+4) Rap[tier 2](outlook* +70) L10n(+42) c-Up(+106) BMcs(+106) Rap[tier 3](+2) Btime[tier 3](+64) T-Prof[tier 2](+16) Rap-Cr[tier 2](fandom-c* imgur* +69) L10n-Rpk(gu-IN* ltg +105) Rap-ChR[tier 2](gsheets-c* youtube-c* +75) Rap-Cr[tier 3](+3) Rap-Prof[tier 2](+16) Rap-live[tier 2](+2) Rap-ChR[tier 3](+2) Rap-Prof[tier 3](+2) Btime-fis[tier 3](outlook-c-vismet +62) js-bench-sm[tier 2](+5) Btime-Prof[tier 3](+2) js-bench-v8[tier 2](+4)

Langues supportées

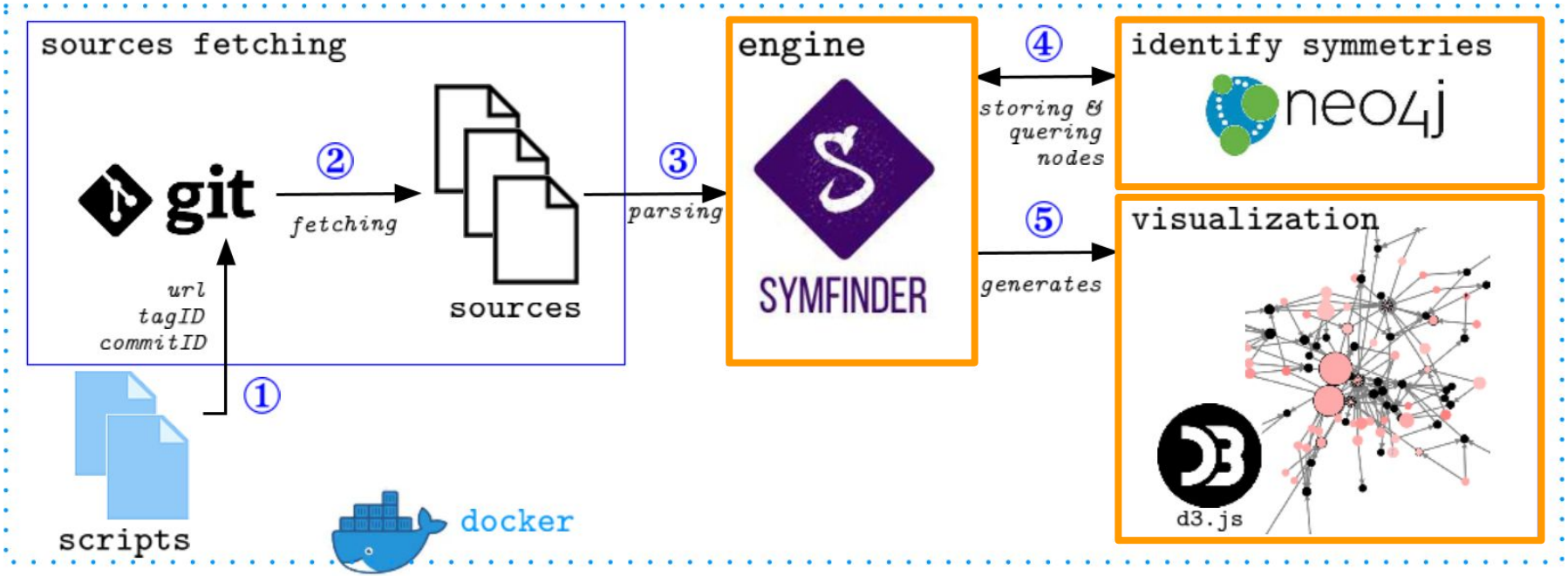
Tests de rendu visuel sur des sites populaires

Un cas concret : symfinder



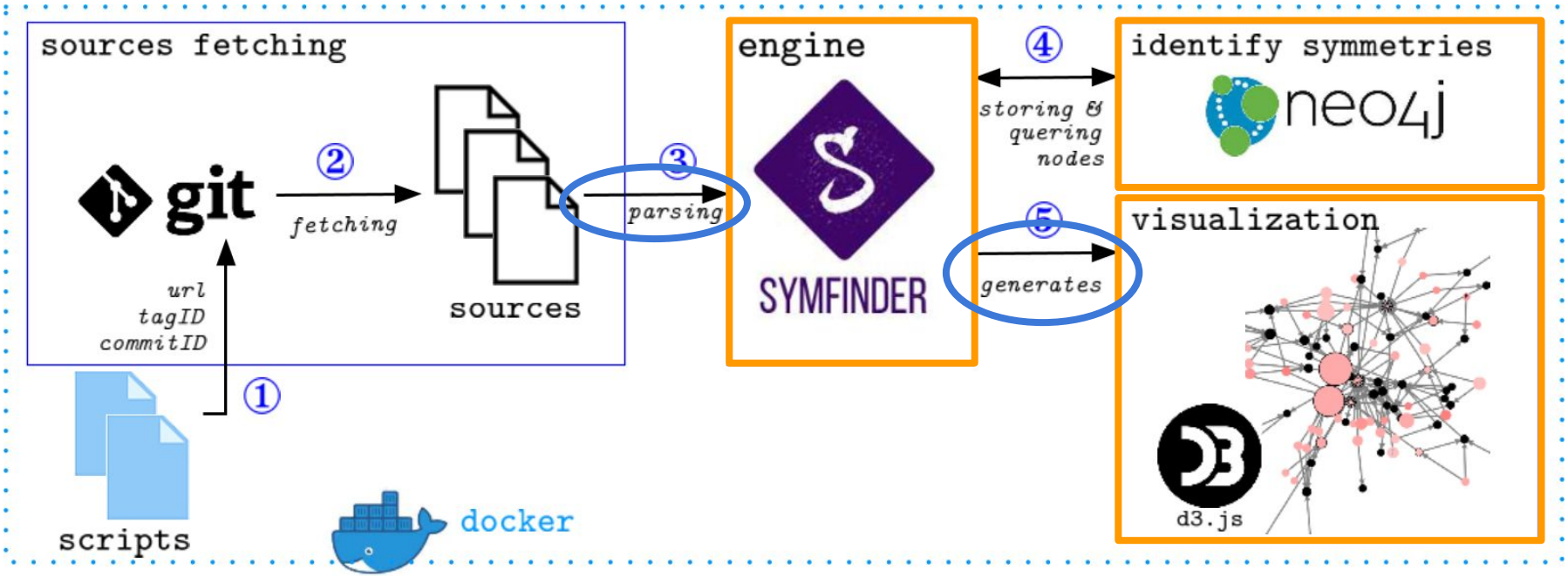
https://deathstar3.github.io/symfinder-demo/

Tests unitaires



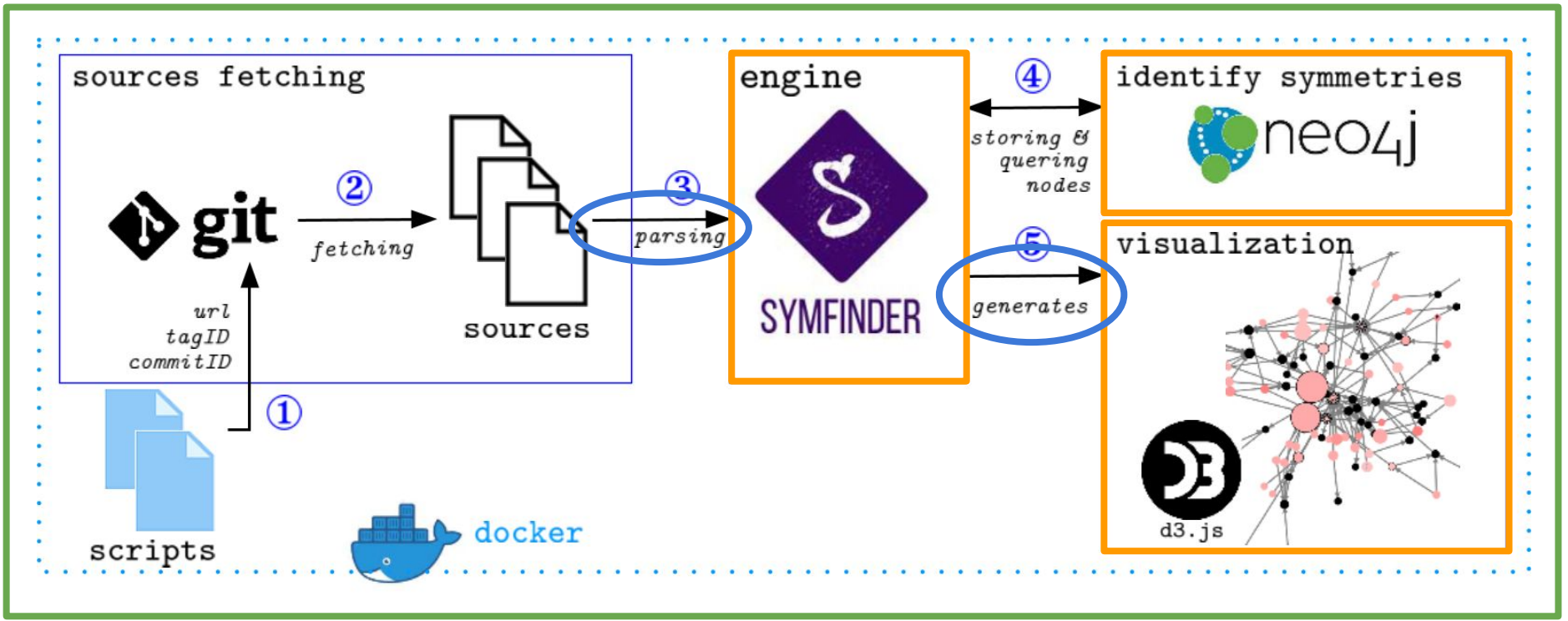
https://deathstar3.github.io/symfinder-demo/

Tests unitaires + tests d'intégration



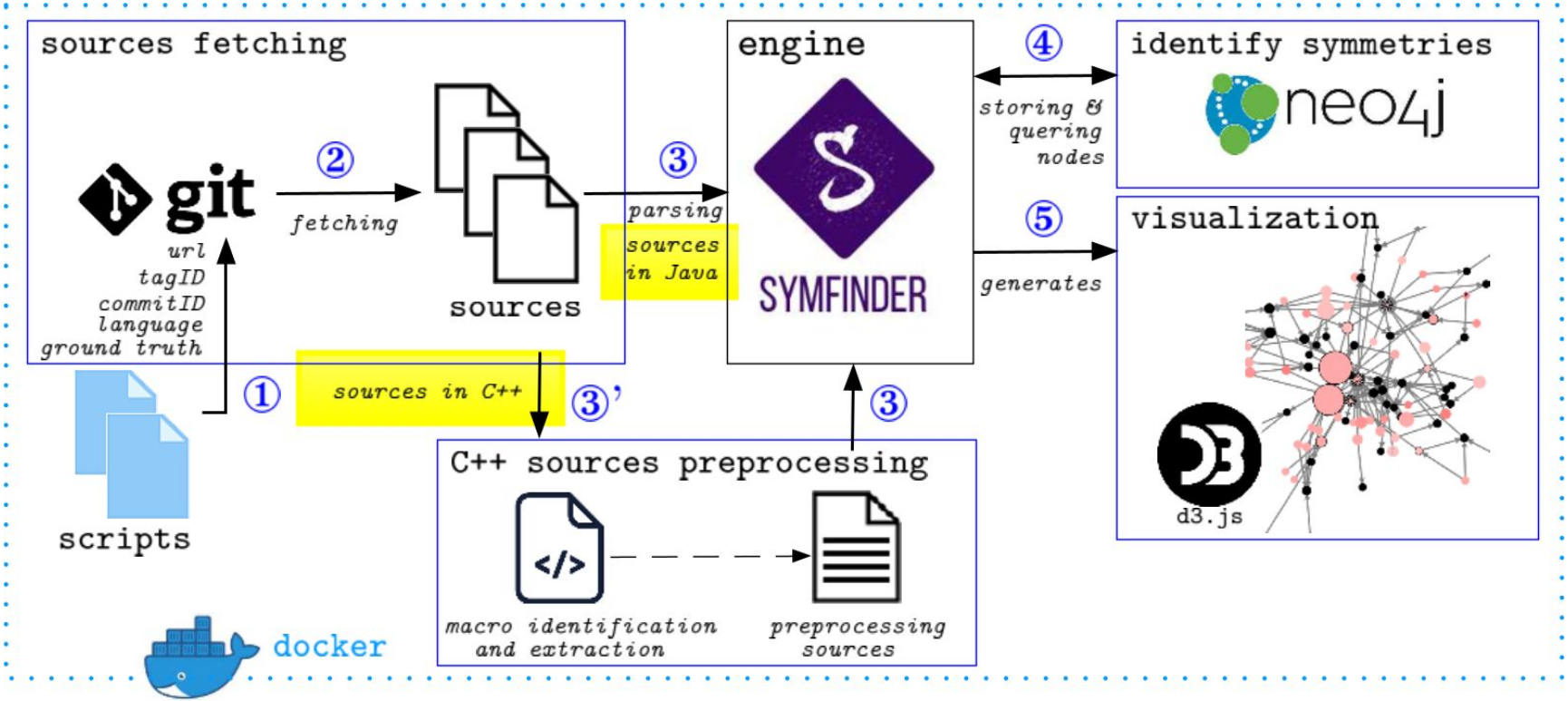
https://deathstar3.github.io/symfinder-demo/

Tests unitaires + tests d'intégration + tests d'acceptation



https://deathstar3.github.io/symfinder-demo/

symfinder-cpp : une variante de symfinder pour C++



https://deathstar3.github.io/symfinder-demo/

Configuration de la chaîne d'intégration

Travis CI

```yaml
language: minimal

jobs:
  include:
    - stage: "Unit tests"
      name: "Building symfinder engine"
      script: docker build -f docker/symfinder/Dockerfile -t deathstar3/symfinder-engine:local .
    - name: "Building sources fetcher"
      script: docker build -f docker/sources_fetcher/Dockerfile -t deathstar3/symfinder-fetcher:local .
    - name: "Building symfinder runner"
      script: docker build -f docker/runner/Dockerfile -t deathstar3/symfinder-runner:local .
    - name: "Building Neo4j procedures"
      script: docker build -f docker/neo4j/Dockerfile -t deathstar3/symfinder-neo4j:local .
    - name: "Visualization unit tests"
      script: ./run_visualization_tests.sh
    - stage: "Integration tests"
      name: "Tests on sample projects"
      script: ./run_integration_tests.sh
    - name: "Acceptance tests on a Java project"
      script: ./run_acceptance_tests.sh junit
    - name: "Acceptance tests on a C++ project"
      script: ./run_acceptance_tests.sh decaf
```

Unit tests — 4 min 8 sec

✓ # 13.1	AMD64	Xenial	Building symfinder engine	4 min 7 sec
✓ # 13.2	AMD64	Xenial	Building sources fetcher	26 sec
✓ # 13.3	AMD64	Xenial	Building symfinder runner	2 min 25 sec
✓ # 13.4	AMD64	Xenial	Building Neo4j procedures	1 min 46 sec
✓ # 13.5	AMD64	Xenial	Visualization unit tests	47 sec

Integration tests — 23 min 11 sec

✓ # 13.6	AMD64	Xenial	Tests on sample projects	23 min 9 sec
✓ # 13.7	AMD64	Xenial	Acceptance tests on a Java project	7 min 26 sec
✓ # 13.8	AMD64	Xenial	Acceptance tests on a C++ project	6 min 48 sec

Des outils pour réaliser de l'intégration continue

Petit exemple d'application

On conçoit un client de messagerie instantanée (ex. Discord).

Quels types de tests pourrait-on implémenter ?

Quelques exemples

Tests de performance (temps au lancement, réactivité de l'application)

Tests de sécurité (chiffrement des échanges)

Tests UI / UX (déclenchement des notifications, affichage du statut des contacts)

Tests d'accessibilité (passage en contraste inversé, tailles des polices)

Tests d'internationalisation (vérification de l'affichage des traductions)

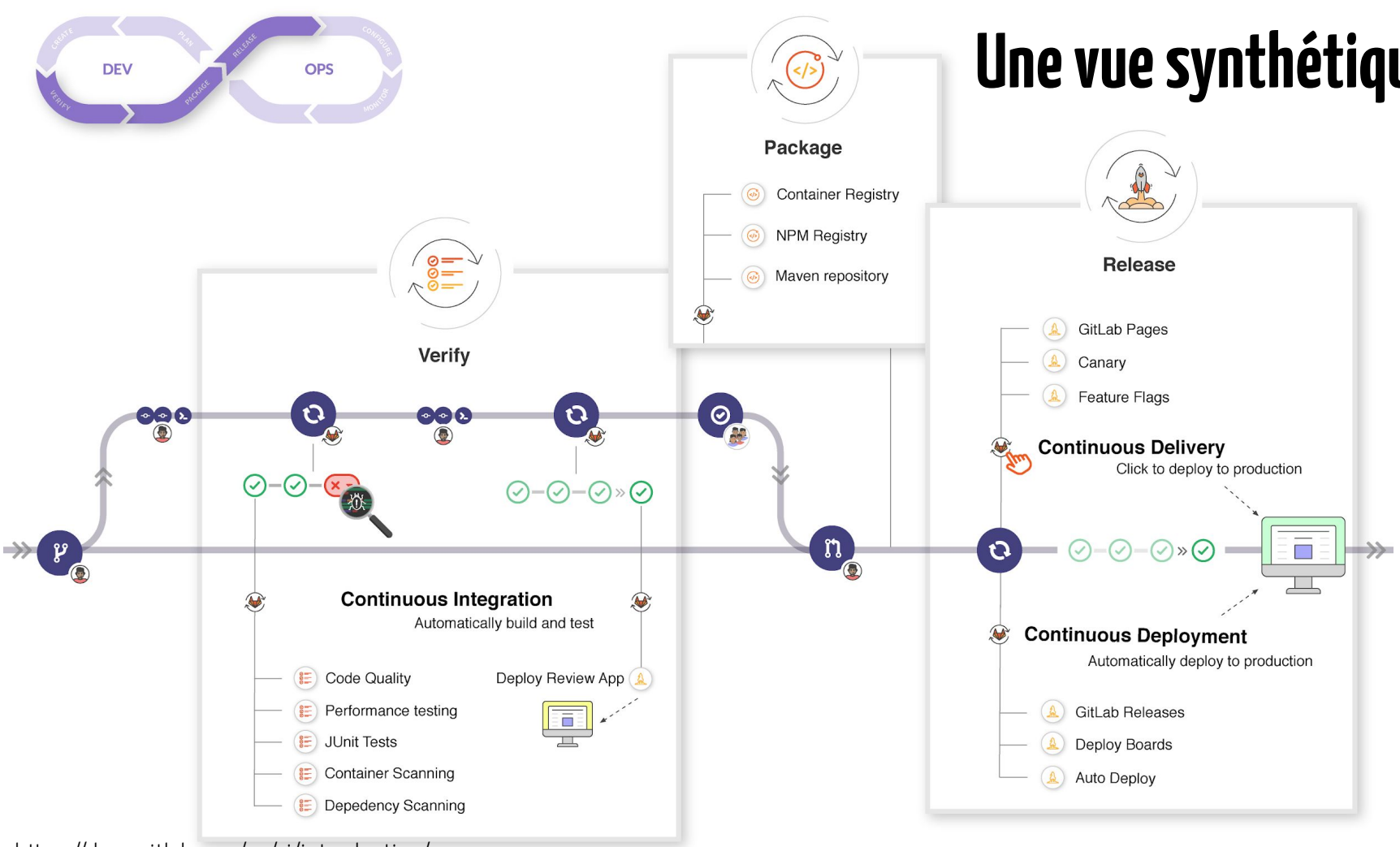

Parés au
déploiement !

Exemples de phases de déploiement / livraison

- Création des différents exécutables pour chaque OS (Windows, OSX, Linux...)
- Déploiement des nouvelles versions sur les serveurs
- Mise à jour du site web
- Déclenchement des mises à jour automatiques

  ou notification qu'une mise à jour est disponible

Une vue synthétique



Source : https://docs.gitlab.com/ee/ci/introduction/

Merci !