

# **La mise en place de la livraison ou déploiement continu (CD)**

# Table des matières

<b>I. Livraison et déploiement continu, principes et différenciations</b>	<b>3</b>
<b>II. Exercice : Quiz</b>	<b>4</b>
<b>III. Étapes dans la mise en place d'un pipeline CD</b>	<b>5</b>
<b>IV. Exercice : Quiz</b>	<b>8</b>
<b>V. Essentiel</b>	<b>9</b>
<b>VI. Auto-évaluation</b>	<b>10</b>
A. Exercice .....	10
B. Test .....	13
<b>Solutions des exercices</b>	<b>13</b>

## I. Livraison et déploiement continu, principes et différenciations

**Durée : 1 h 30**

**Pré-requis :** savoir utiliser GIT, avoir fait le module sur l'intégration continue (CI), installer node.js et npm.

**Environnement de travail :** un ordinateur connecté à Internet.

### Contexte

La livraison continue (Continuous Delivery) ou déploiement continu (Continuous Deployment) est l'étape qui suit l'intégration continue (CI). C'est pour cette raison que l'on parle souvent de CI/CD ensemble, l'un va difficilement sans l'autre. Les tests une fois effectués et validés sur un environnement de développement permettent le déplacement du code en production. Le déploiement continu représente le processus qui consiste à automatiser les déploiements. Auparavant ces déploiements étaient réalisés manuellement

La livraison continue et le déploiement continu, bien qu'ils soient étroitement liés, définissent un degré d'automatisation différent.

### Définition Livraison continue (ou distribution continue)

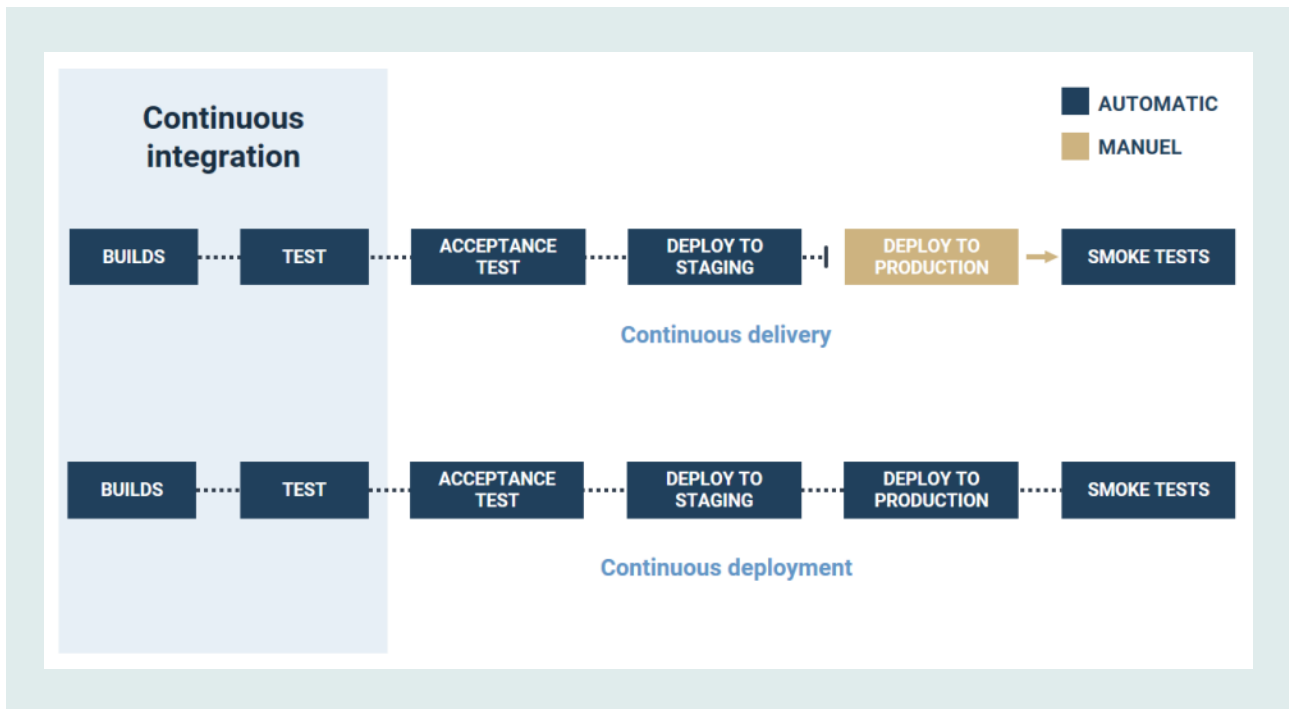
La livraison continue permet de toujours avoir à disposition une application prête à être déployée dans un environnement de production. Ce processus intervient après l'intégration continue. Pour mettre en place une livraison continue efficace, il est primordial d'avoir au préalable introduit l'intégration continue dans notre pipeline de développement.

Dans un pipeline de livraison continue, chacune des étapes est automatisée jusqu'à obtenir une version de notre application prête pour la production ce qui implique le déclenchement des processus de test et de publication dans des environnements définis. Une fois ce processus fini, l'équipe de développeurs est en mesure de déployer facilement et rapidement une application dans un environnement de production suite à une action manuelle.

### Définition Déploiement continu

Le déploiement continu représente le niveau maximum d'automatisation d'un pipeline CI/CD. Le déploiement continu automatise contrairement à la livraison continue le lancement d'une application dans un environnement de production.

Dans la théorie, une modification apportée par un développeur sur un projet doit pouvoir être publiée en quelques minutes suite à la validation du code de tous les tests automatisés. Ainsi il est très facile de suivre et de répondre aux demandes utilisateur en déployant une multitude de petites modifications tout en réduisant les risques de faire face à une erreur en production.



La mise en place d'un pipeline CD à la suite du pipeline CI repose sur de nouvelles étapes à respecter. C'est l'ajout de ses processus qui nous permet de tendre vers un pipeline CI/CD mature et efficace.

Ses étapes sont :

- Codifier son infrastructure en respectant les règles de l'Infrastructure-as-Code,
- Tester son application,
- Déployer son application,
- Monitorer l'application et la mise en place d'alerte.

#### Rappel

La différence entre la livraison continue et le déploiement continu concerne l'automatisation du déploiement en production. Dans le processus de livraison continue, la mise en production reste soumise à une validation par un être humain suivi d'un acte manuel. Quand cette étape est elle aussi automatisée, on parle de déploiement continu.

## Exercice : Quiz

[solution n°1 p.15]

### Question 1

La différence entre le déploiement continu et la livraison continue se trouve-t-elle au niveau de la phase de test ?

- ☐ Vrai
- ☐ Faux

### Question 2

La livraison continue livre automatiquement le code « *buildé et testé* » en production ?

- ☐ Vrai
- ☐ Faux

## Question 3

Que signifie IaC ?

- ☐ Infrastructure as Continuous
- ☐ Infrastructure as Code
- ☐ Infrastructure as Confirmation
- ☐ Infrastructure as Control

## Question 4

Que signifie l'abréviation CD ?

- ☐ Continuous Delivery
- ☐ Continuous Dependence
- ☐ Continuous Deployment
- ☐ Continuous Driven

### III. Étapes dans la mise en place d'un pipeline CD

#### La codification de l'infrastructure avec l'Infrastructure-as-Code (IaC)

L'IaC (Infrastructure-as-Code, ou Infrastructure en tant que code) consiste à gérer et approvisionner une infrastructure à l'aide de lignes de code plutôt que par des processus manuels.

L'IaC implique la création de fichiers de configuration qui contiennent les caractéristiques de l'infrastructure, ce qui facilite les modifications et la distribution des configurations. L'IaC permet également de s'assurer que le même environnement est fourni à chaque fois.

L'IaC est une partie importante de la mise en œuvre des pratiques DevOps et CI/CD. Elle soulage les développeurs de la plupart des tâches d'approvisionnement. Ils n'ont plus qu'à exécuter un script pour que leur infrastructure soit opérationnelle. Dans votre travail de tous les jours vous entendrez le plus souvent parler de conteneur ou d'image container.

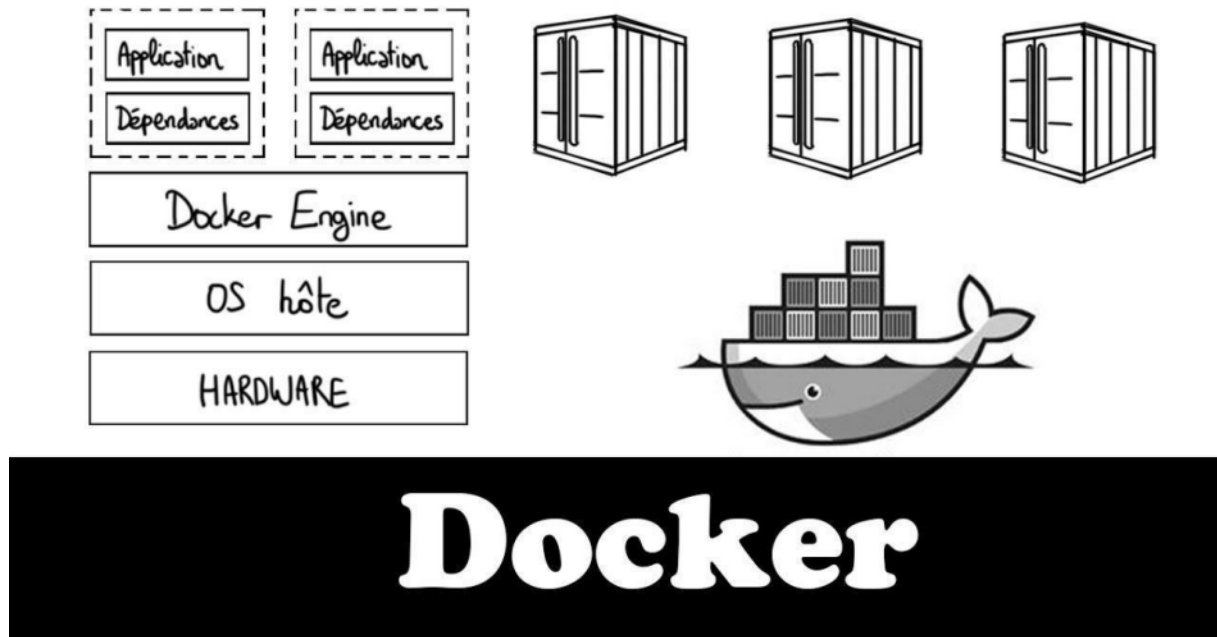
<b>Définition</b>	<b>Image container</b>
-------------------	------------------------

C'est un ensemble de processus logiciels léger et indépendant, regroupant tous les fichiers nécessaires à l'exécution des processus : code, runtime, outils système, bibliothèque et paramètres. Ils peuvent être utilisés pour exécuter des applications.

#### Avantages de mise en place IaC

- Créer des environnements à la demande,
- Créer des environnements complexes en quelques minutes,
- Intégrable et pilotable dans le pipeline CD,
- Faciliter la montée en version des environnements.

Outils disponibles : Docker, Chef, Puppet, Ansible et Terraform.



Source : Docker

### Tester son application en environnement de test

Tester son application sur un environnement de test presque identique à celui de la production nous permet d'obtenir des retours fidèles à ceux obtenables en production.

Les tests dans un pipeline CD sont :

- **Test d'acceptance**

Les tests d'acceptance (également appelés tests d'acceptation, UAT ou recette client) sont généralement exécutés par les personnes qui utilisent le produit dans leur pratique opérationnelle. Ils permettent de s'assurer que les changements mis en œuvre au travers de la solution informatique fonctionnent réellement avant que celle-ci ne soit en production.

Ces tests peuvent être automatisés ou manuels.

Outils disponibles: Confluence, FitNesse ou Ranorex.

- **Test de performance**

Les tests de performance (et de charge) ont pour principal objectif la validation d'une solution logicielle et de son architecture sous-jacente liées à une utilisation simultanée multi-utilisateur, permettant ainsi d'éviter certains problèmes en production. Ils permettent de garantir une qualité de service applicative dans des conditions réelles d'utilisation.

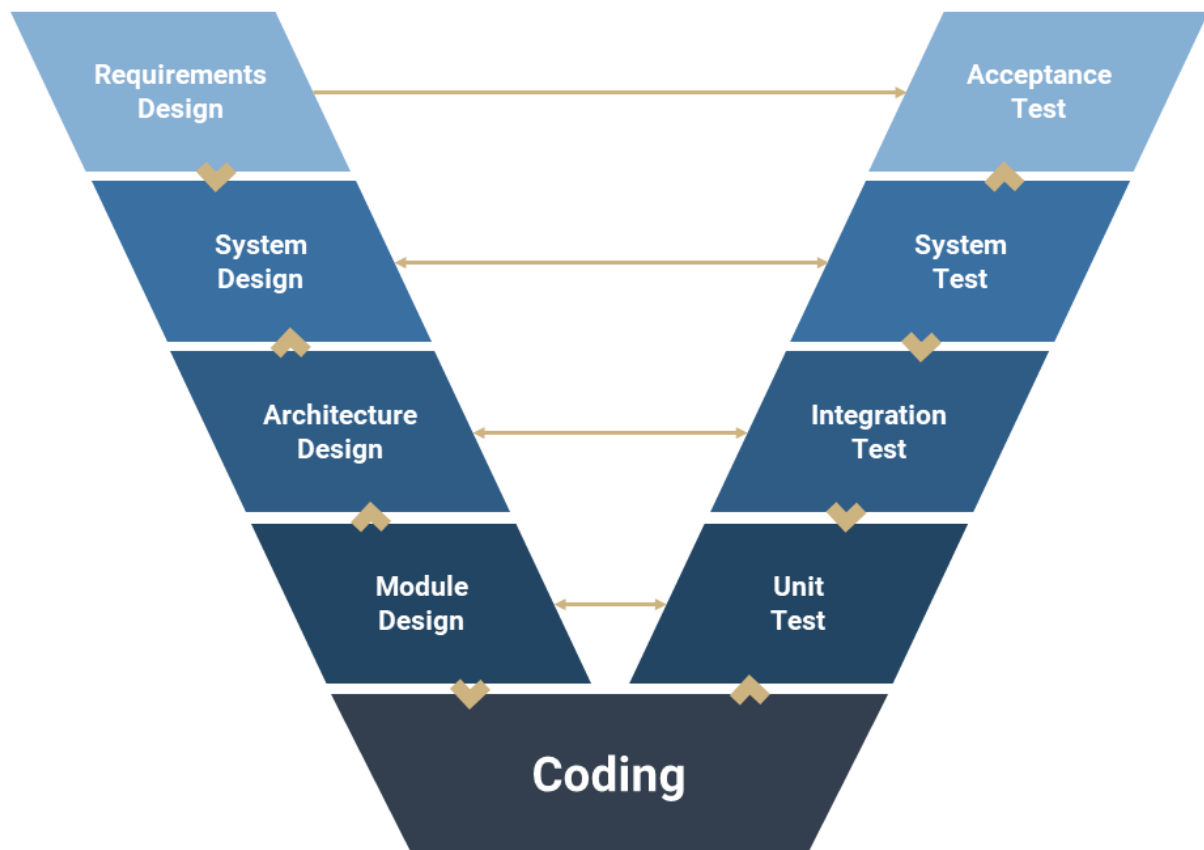
Outils disponibles : JMeter, Apache Bench ou Gatling.

- **Smoke test**

Ils vérifient les fonctionnalités de base de l'application. Rapides à exécuter, ils sont utiles juste après un nouveau build, afin de décider si l'on peut exécuter des tests plus coûteux, ou juste après un déploiement pour vérifier que l'application fonctionne comme prévu dans le nouvel environnement.

Exemple : vérifier que la base de données répond et est correctement configurée.

Outils disponibles : Selenium, SoapUI ou Cypress.



### Déployer son application

L'étape la plus importante de la livraison continue est le déploiement du package que nous avons précédemment créé lors de l'intégration continue.

La caractéristique distinctive entre le déploiement continu et la livraison continue est l'étape automatisée d'activation du nouveau code dans un environnement réel. Un pipeline de déploiement continu doit pouvoir annuler un déploiement en cas de déploiement d'erreurs ou de modifications importantes. Les outils de déploiement automatisés sont indispensables à un déploiement continu approprié.

Les avantages d'utiliser un outil pour automatiser le déploiement de l'application sont nombreux :

- Se concentrer sur le développement.
- Déployer les logiciels devient moins complexe, moins sujet aux erreurs et beaucoup plus reproductible.
- Déployer dans un nouvel environnement est facile.
- La fréquence des déploiements peut être plus élevée.

Outils disponibles: Spinnaker, XLDeploy ou UrbanCode.

### La supervision de l'application (monitoring) et la mise en place d'alerte

Le monitoring est une forme de surveillance qui teste et vérifie régulièrement le bon fonctionnement d'un site Web lors de la réponse aux entrées de l'utilisateur, comme la connexion ou l'achat.

Une application logicielle charge l'application Web dans un navigateur et, à l'aide d'un script d'automatisation, l'application exécute les interactions utilisateur et rapporte les performances de l'application Web à travers certaines métriques.

Le monitoring peut être mis en place sur n'importe quel environnement test, staging ou bien l'environnement de production.

Les métriques sont classées en 3 catégories. Chacune de ses catégories se traite différemment et ne sert pas le même objectif.

- **Les Métriques Systèmes**

Liées à l'infrastructure servant la partie applicative. Cette infrastructure est composée de différentes ressources qui peuvent être de bas niveau comme le matériel physique (CPU, RAM, Disques, Réseau) ou de plus haut niveau comme une base de données.

- **Les Métriques Applicatives**

C'est ce qui permet de mesurer l'expérience utilisateur.

Ces métriques permettent de répondre rapidement aux questions qui intéressent les utilisateurs finaux du service. Le service est-il disponible et remplit-il sa mission ? À quelle vitesse le fait-il ? Avec quels résultats ?

- **Les Événements**

En plus des métriques systèmes et applicatives, il y a certaines informations que l'on souhaite récupérer de façon plus sporadique. Certains systèmes permettent de superviser des événements. Les événements n'ont pas lieu de façon fréquente et il est souvent difficile, voire, impossible de les prévoir (exemple détection d'un virus).

Outils disponibles : Dynatrace, Sysdig ou New Relic.

Les métriques une fois en place il est primordial de disposer d'un retour rapide des utilisateurs de l'application. En effet, si un bug se retrouve en production il faut le détecter le plus rapidement possible, afin de le corriger.

Outils disponibles : Twitter, Slack ou Trello.

#### Rappel

Les étapes indispensables à la réalisation d'un pipeline CD mature et efficace sont :

1. La codification de l'infrastructure avec l'Infrastructure-as-Code,
2. Le test de votre application en environnement de test,
3. Le déploiement de votre application,
4. La supervision de l'application et la mise en place de notification d'alerte.

## Exercice : Quiz

[solution n°2 p.15]

### Question 1

Quel outil permet de créer des environnements à la demande ?

- ☐ Le Smoke test
- ☐ Les Containers
- ☐ Les métriques
- ☐ Le monitoring

### Question 2



Lequel de ses tests n'est pas réalisé dans la partie CD ?

- ☐ Test intégration
- ☐ Test système
- ☐ Test acceptance
- ☐ Test unitaire

Question 3

Les événements sont des métriques ?

- ☐ Vrai
- ☐ Faux

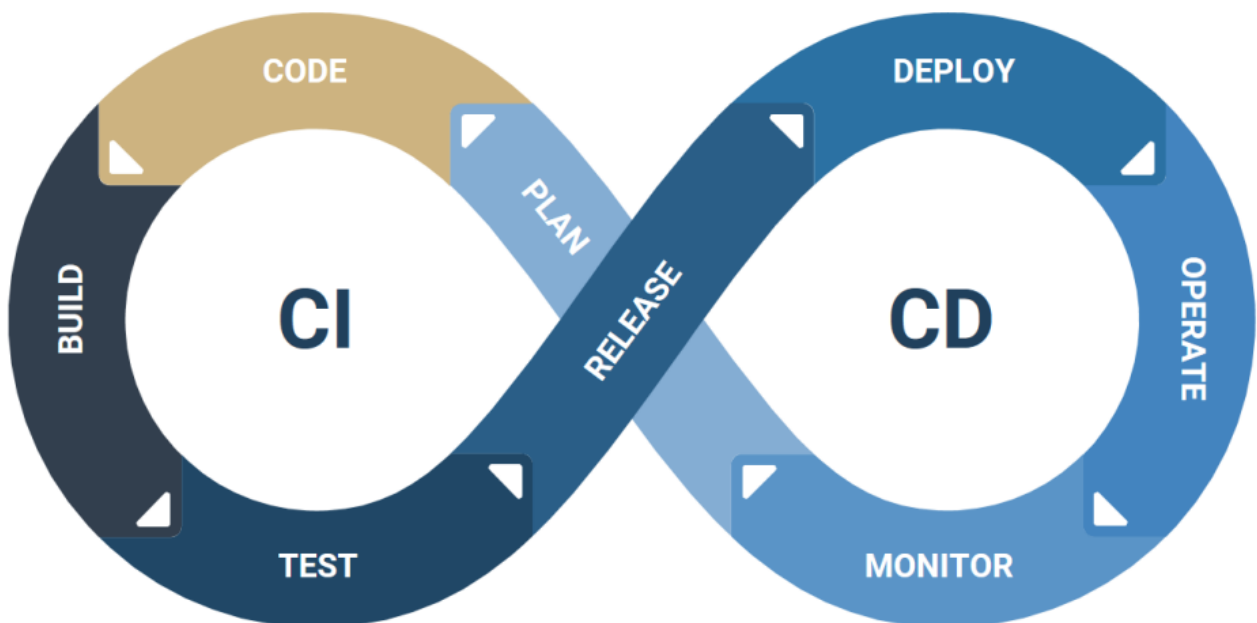
Question 4

Le terme staging désigne un environnement de production ?

- ☐ Vrai
- ☐ Faux

## V. Essentiel

La livraison continue ou déploiement continu est la suite logique du processus d'intégration continue dont l'objectif est la mise en place d'un pipeline CI/CD. Un pipeline CI/CD mature et efficace est une savante combinaison du respect des bonnes pratiques, des règles DevOps et des méthodes agiles. Mais l'itinéraire pour gravir cette montagne qui paraît si impressionnante peut être divisé en de nombreuses étapes. À chacune de ses étapes, une multitude d'outils permettent de faciliter notre ascension et on peut se retrouver au sommet bien plus rapidement que ce qu'on pouvait l'imaginer.



## VI. Auto-évaluation

### A. Exercice

L'objectif de ce cas pratique est de démystifier le déploiement continu, bien que la mise en place d'un pipeline respectant scrupuleusement toutes les étapes CI/CD restent un travail de grande envergure demandant une connaissance pointue dans de nombreux domaines, chacune de ses étapes peut être abordée indépendamment et incorporer dans un projet au besoin. De plus ce niveau d'automatisation peut être mis en place gratuitement il serait donc dommage de s'en priver. Le but de cet exercice est de déployer de manière continue une application React.js. Il n'est pas nécessaire d'avoir déjà travaillé sur React pour réussir ce cas pratique. Toutes les commandes nécessaires vous seront données.

Pourquoi utiliser React ? Car React est une framework qui nécessite de build (télécharger) son application avant de la déployer. La visualisation de ses processus vous permettra de comprendre les notions d'automatisations derrière un pipeline CI/CD.

Pré-requis :

- Créer un compte Netlify,
- Créer compte Gitlab,
- Installer node.js sur son ordinateur<sup>1</sup>,
- Installer npm sur son ordinateur<sup>2</sup>.

Mise en place :

Dans un premier temps nous allons construire une application de base React, la build, la test et la déployé manuellement, dans un second temps il vous faudra automatiser ses processus.

Une fois node.js et npm installées créons notre application ici nommée react-cicd-hero, dans votre terminal taper :

```
1 <code>
2 npx create-react-app react-cicd-hero //crée l'application react
3 </code>
```

Une fois votre application créée, votre terminal devrait afficher les commandes possibles pour utiliser votre application, on peut déjà noter que les commandes npm test et npm run build sont possibles, mais commençons par suivre les recommandations.

```
Created git commit.

Success! Created react-cicd-hero at D:\Local\www\react-cicd-hero
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd react-cicd-hero
  npm start

Happy hacking!
```

1 <https://nodejs.org/fr/>

2 <https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>

```
1 <code>
2 cd react-cicd-heo
3 npm start
4 </code>
```

Une fois npm start lancé, nous avons une application fonctionnelle qui tourne sur notre localhost, testons et déployons notre application manuellement.

On peut donc tester notre application :

```
1 <code>
2 npm test (exécute des tests internes à React, dans notre cas très rapide pour l'exemple)
3 </code>
```

Enfin, il faut « *build* » l'application.

```
1 <code>
2 npm run build
3 </code>
```

Le build va générer un dossier build à la racine de votre application c'est ce dossier qui contient tout le code de l'application prêt à être déployé sur un serveur.

Pour le déployer sur Netlify il suffit de se connecter et de se rendre sur l'onglet SITES et de glisser déposer le dossier build.

Votre site est en ligne après un CI/CD manuel.

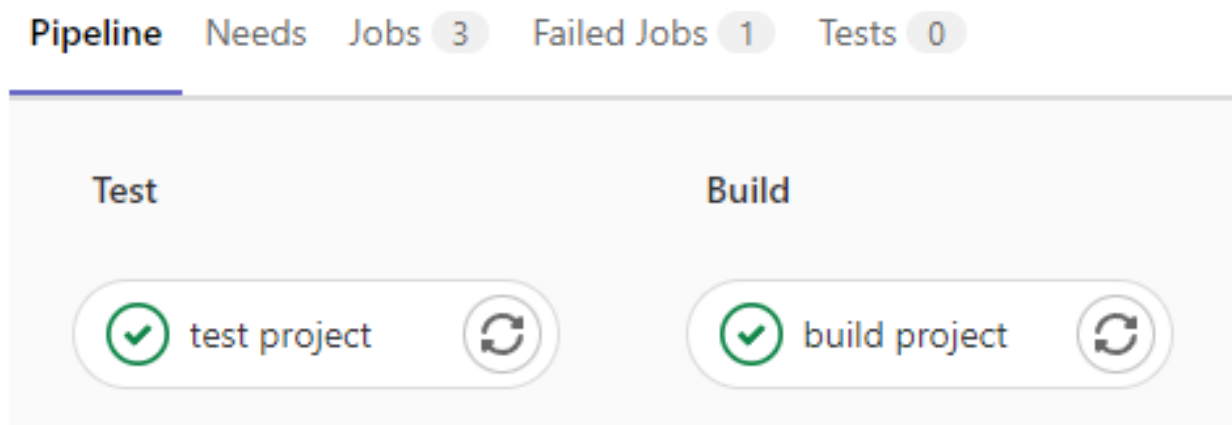
Automatiser la phase CI :

### Question 1

[solution n°3 p.16]

Lister les commandes qui vous permettent d'initier votre projet sur Gitlab et créer un fichier .gitlab-ci.yml composé de deux stages : test et build. Le job test project utilisera une image : node:15 et lancera les scripts npm install et npm test. Le job build project utilisera une image node:15 , lancera les scripts npm install et npm run build et un artifacts:paths: -build/ afin de ne pas perdre le dossier build générer et de pouvoir le déployer ultérieurement.

Si cette partie est correctement exécutée à chaque nouveau commit votre pipeline devrait ressembler à ça :



Vous pouvez cliquer sur chaque étape et voir que les scripts npm test et npm run build sont exécutés automatiquement par gitlab qui grâce à la ref de l'image et au script npm install à déployé un environnement similaire à celui de votre ordinateur.

Pour automatiser la phase CD, dans Gitlab créer les variables dans setting - CICD.

## Variables

Collapse

Variables store information, like passwords and secret keys, that you can use in job scripts. [Learn more.](#)

Variables can be:

- **Protected:** Only exposed to protected branches or tags.
- **Masked:** Hidden in job logs. Must match masking requirements. [Learn more.](#)

Environment variables are configured by your administrator to be **protected** by default.

Type	↑ Key	Value	Protected	Masked	Environments
Variable	NETLIFY_AUTH_TOKEN	*****	✓	✗	All (default)
Variable	NETLIFY_SITE_ID	*****	✓	✗	All (default)

Add variable

Reveal values

NETLIFY\_AUTH\_TOKEN se trouve sur netlify - user setting - applications - créer un nouveau acces toker et le reporter sur GITLAB.

NETLIFY\_SITE\_ID cliquer sur votre site créé juste avant dans netlify , puis dans site setting, SITE ID.

La création de variables permet la connexion Netlify - Gitlab, votre authentification et la connexion entre votre repository et un site.

## Question 2

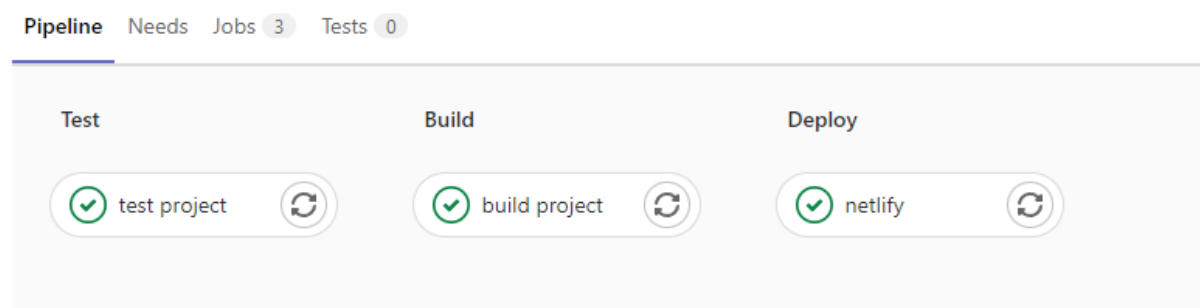
[solution n°4 p.17]

Modifier le fichier gitlab-ci.yml ci-dessus pour permettre le déploiement continu de votre projet. Les scripts qui vous seront nécessaire sont :

- npm install -g netlify-cli
- netlify deploy --dir=build --prod

Attention au moment du push il faut avoir Netlify d'ouvert dans une page de votre navigateur pour permettre l'authentification.

Si tout est réalisé correctement vous devriez apercevoir :



Pour aller plus loin ouvrez le fichier avec votre éditeur et modifiez le titre de votre page, puis push avec git le changement qui devrait automatiquement arrivé sur votre site Netlify.

Votre pipeline CI/CD est opérationnel.

**B. Test****Exercice 1 : Quiz**

[solution n°5 p.18]

## Question 1

Le déploiement continu reste soumis à une validation par un être humain ?

- ☐ Vrai
- ☐ Faux

## Question 2

Que représente le déploiement continu ?

- ☐ Le lancement d'une application dans un environnement de production
- ☐ Le niveau maximum d'automatisation d'un pipeline CI/CD

## Question 3

Les Métriques Applicatives sont composées de ressources de haut niveau telles que la base de données d'une application ?

- ☐ Vrai
- ☐ Faux

## Question 4

Utiliser un outil pour automatiser le déploiement d'une application permet d'augmenter la fréquence des déploiements ?

- ☐ Vrai
- ☐ Faux

## Question 5

Quelle est la dernière étape de la réalisation d'un pipeline de déploiement continu ?


- ☐ Le test de votre application en environnement de test
- ☐ La codification de l'infrastructure avec l'Infrastructure-as-Code (IaC)
- ☐ La supervision de l'application (monitoring) et la mise en place d'alerte
- ☐ Le déploiement de votre application

**Solutions des exercices**




**Exercice p. 4 Solution n°1****Question 1**

La différence entre le déploiement continu et la livraison continue se trouve-t-elle au niveau de la phase de test ?

- ☐ Vrai
- ☒ Faux
-  C'est au niveau de la méthode de déploiement.


**Question 2**

La livraison continue livre automatiquement le code « *buildé et testé* » en production ?

- ☐ Vrai
- ☒ Faux
-  C'est le déploiement continu qui est automatisé à 100 %.


**Question 3**

Que signifie IaC ?

- ☐ Infrastructure as Continuous
- ☒ Infrastructure as Code
- ☐ Infrastructure as Confirmation
- ☐ Infrastructure as Control
-  L'infrastructure as code (IaC) est une démarche qui vise à configurer une infrastructure informatique (virtuelle), d'une façon similaire à une programmation de logiciel, en utilisant des fichiers descripteurs et des codes.

**Question 4**

Que signifie l'abréviation CD ?


- ☒ Continuous Delivery
- ☐ Continuous Dependence
- ☒ Continuous Deployment
- ☐ Continuous Driven
-  La livraison continue (Continuous Delivery) et déploiement continu (Continuous Deployment) partagent leur acronyme CD.

**Exercice p. 8 Solution n°2**

### Question 1

Quel outil permet de créer des environnements à la demande ?


- ☐ Le Smoke test
- ☒ Les Containers
- ☐ Les métriques
- ☐ Le monitoring

 Tout comme dans le domaine des transports, les conteneurs informatiques stockent des objets pour les transporter. Ils permettent d'expédier des applications et leurs dépendances sur de multiples systèmes d'exploitation, quels qu'ils soient. Ils garantissent que leur contenu est identique au départ et à l'arrivée, et de manière sécurisée.

### Question 2

Lequel de ses tests n'est pas réalisé dans la partie CD ?


- ☐ Test intégration
- ☐ Test système
- ☐ Test acceptance
- ☒ Test unitaire

 Les tests unitaires appartiennent à la partie CI.

### Question 3

Les événements sont des métriques ?


- ☒ Vrai
- ☐ Faux

 Les évènements sont un des métriques prises en compte dans la supervision des applications.

### Question 4

Le terme staging désigne un environnement de production ?

- ☐ Vrai
- ☒ Faux

 C'est l'étape avant la mise en ligne publique Le staging est utilisé pour des démonstrations au client avant la mise en ligne finale, toutes les fonctionnalités sont testables comme les paiements sur un site e-commerce.



Liste des commandes pour initier le projet sur Gitlab :

```
1 cd fichier de l'app
2 git init --initial-branch=main
3 git remote add origin https://gitlab.com/User/nom de votre projet.git
4 git add .
5 git commit -m "Initial commit"
6 git push -u origin main
```

Composition du fichier .gitlab-ci.yml :

```
1 <code>
2 stages:
3   - test
4   - build
5
6 test project:
7   stage: test
8   image: node:15
9   script:
10    - npm install
11    - npm test
12
13 build project:
14   stage: build
15   image: node:15
16   script:
17    - npm run build
18   artifacts:
19     paths:
20     - build/
21
22 </code>
```

#### p. 12 Solution n°4

```
1 <code>
2 stages:
3   - test
4   - build
5   - deploy
6
7 test project:
8   stage: test
9   image: node:15
10  script:
11    - npm install
12    - npm test
13
14 build project:
15   stage: build
16   image: node:15
17   script:
18    - npm install
19    - npm run build
20  artifacts:
21    paths:
22    - build/
```

```


23
24 netlify:
25   stage: deploy
26   image: node:15
27   script:
28     - npm install -g netlify-cli
29     - netlify deploy --dir=build --prod
30
31 </code>

```

## Exercice p. 13 Solution n°5


### Question 1

Le déploiement continu reste soumis à une validation par un être humain ?

- ☐ Vrai
- ☒ Faux
-  Dans le processus de déploiement continu, même l'étape de déploiement en production est automatisée.


### Question 2

Que représente le déploiement continu ?

- ☐ Le lancement d'une application dans un environnement de production
- ☒ Le niveau maximum d'automatisation d'un pipeline CI/CD
-  Le déploiement continu représente le niveau maximum d'automatisation d'un pipeline CI/CD contrairement à la livraison continue qui correspond au lancement d'une application dans un environnement de production.


### Question 3

Les Métriques Applicatives sont composées de ressources de haut niveau telles que la base de données d'une application ?

- ☐ Vrai
- ☒ Faux
-  Les Métriques Applicatives permettent de mesurer l'expérience utilisateur.

### Question 4


Utiliser un outil pour automatiser le déploiement d'une application permet d'augmenter la fréquence des déploiements ?

- ☒ Vrai
- ☐ Faux
-  Déployer les logiciels devient moins complexe, moins sujet aux erreurs et donc la fréquence des déploiements peut être plus élevée.

**Question 5**

---

Quelle est la dernière étape de la réalisation d'un pipeline de déploiement continu ?

- ☐ Le test de votre application en environnement de test
- ☐ La codification de l'infrastructure avec l'Infrastructure-as-Code (IaC)
- ☒ La supervision de l'application (monitoring) et la mise en place d'alerte
- ☐ Le déploiement de votre application
-  Le Monitoring est une forme de surveillance qui teste et vérifie régulièrement le bon fonctionnement d'un site Web lors de la réponse aux entrées de l'utilisateur, comme la connexion ou l'achat. Elle est donc la dernière étape de la réalisation d'un pipeline CD.