Cours 3
Mise en place d’une architecture Web

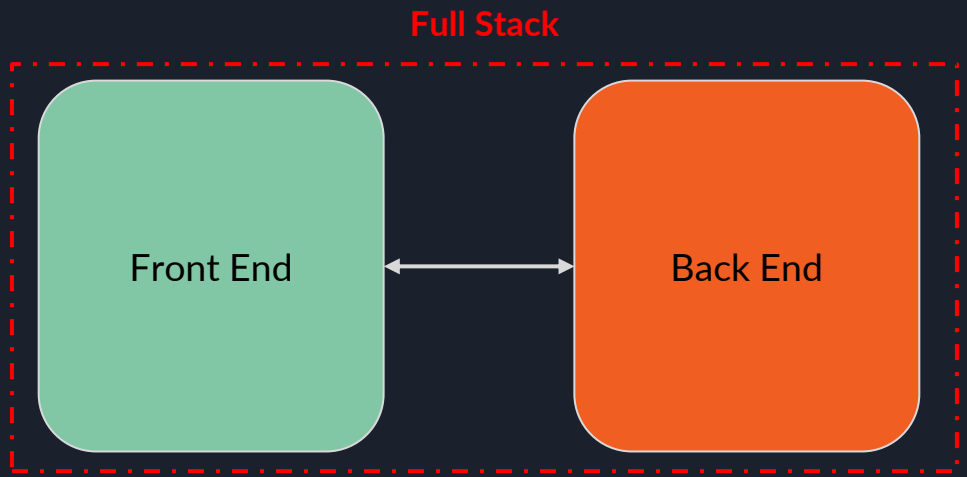
Un cours de Yann Fornier

Session 3 : Mise en œuvre d’une architecture web

- Les différentes technologies utilisées pour mettre en œuvre une architecture web (Java, .NET, Node.js, etc.)
- La création d’un environnement de développement et de déploiement pour les services web (outils de build, de test, de déploiement, etc.)
- La gestion des erreurs et des exceptions dans les services web
- **Exercice** : Mise en place d’un workflow GitOps pour le déploiement d’une application web
- **QCM** : connaissances sur la mise en œuvre d’une architecture web

Les différents éléments d’une application web

Le front end et le backend sont deux parties d’une application web qui travaillent ensemble pour fournir une expérience utilisateur cohérente.



Développeur.se Full stack

Le couteau-suisse de la Tech

Profil

Projets web (site ou app)
Front-end / Back-end
Généraliste
Vision produit globale

Compétences

- ●●●●● **Curieux.se**
- ●●●●○ Communiquant(e)
- ●●●●○ Rigoureux.se
- ●●●●● **Agile**

Formation

Ecole en ingénierie **informatique**

Parcours

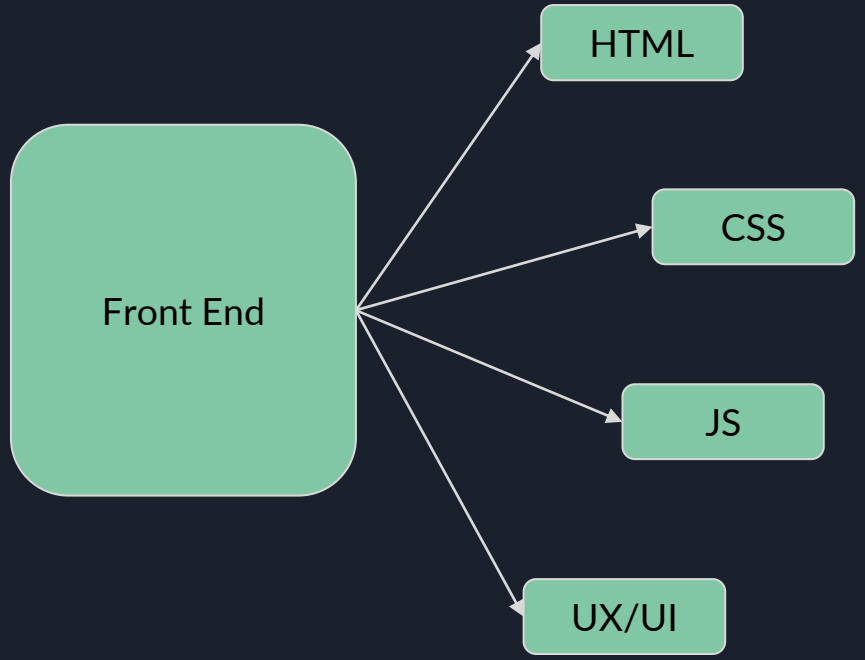
Autodidacte
Généralement expert(e)
dans un domaine
Appétences métier

Carrière

Management de projet et d'équipes
Architecte / **Expert** front, back ou Ops

Le Front End

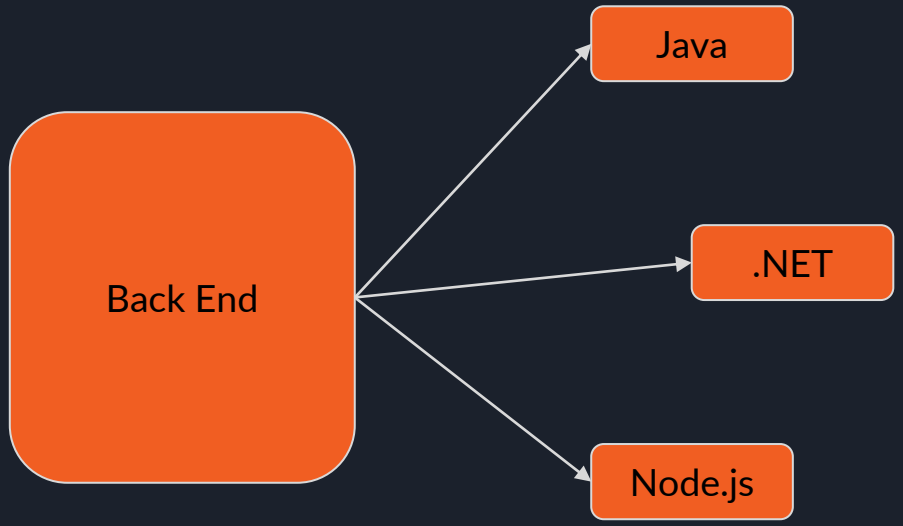
Le front end, également appelé “client”, se réfère aux éléments d’une application web qui sont visibles et utilisables par l’utilisateur.



Le Back End

Le backend, également appelé “serveur”, se réfère aux éléments d’une application web qui sont exécutés côté serveur et qui sont généralement invisibles pour l’utilisateur.

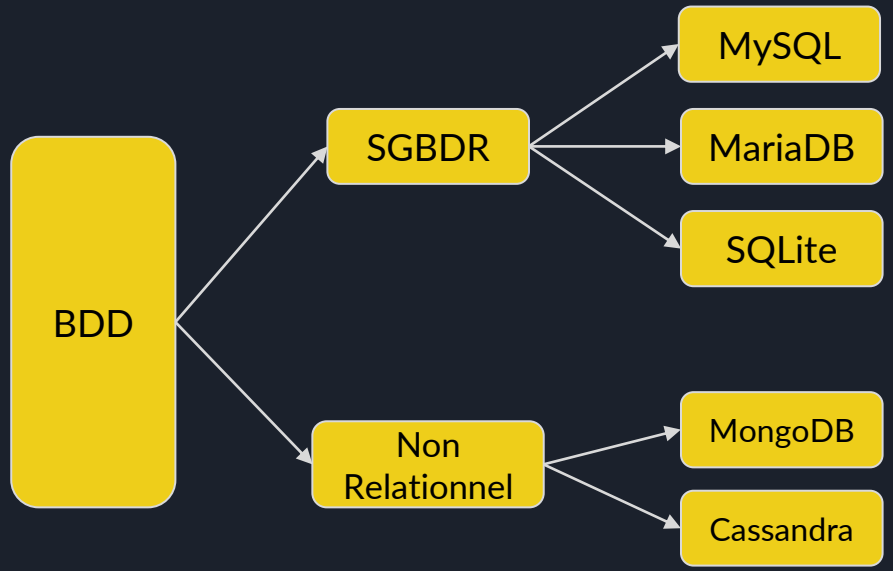
Il comprend généralement du code qui s’exécute sur un serveur web et qui est responsable de la gestion des données, de la logique métier et des communications avec les bases de données et d’autres services.



Les bases de données

Les bases de données sont des systèmes de gestion de données qui permettent de stocker, de gérer et de récupérer des données de manière structurée et efficace.

Elles sont souvent utilisées en conjonction avec des applications web pour stocker et gérer les données de l'application, telles que les utilisateurs, les produits et les transactions.



Les différentes technologies utilisées pour mettre en oeuvre une architecture web

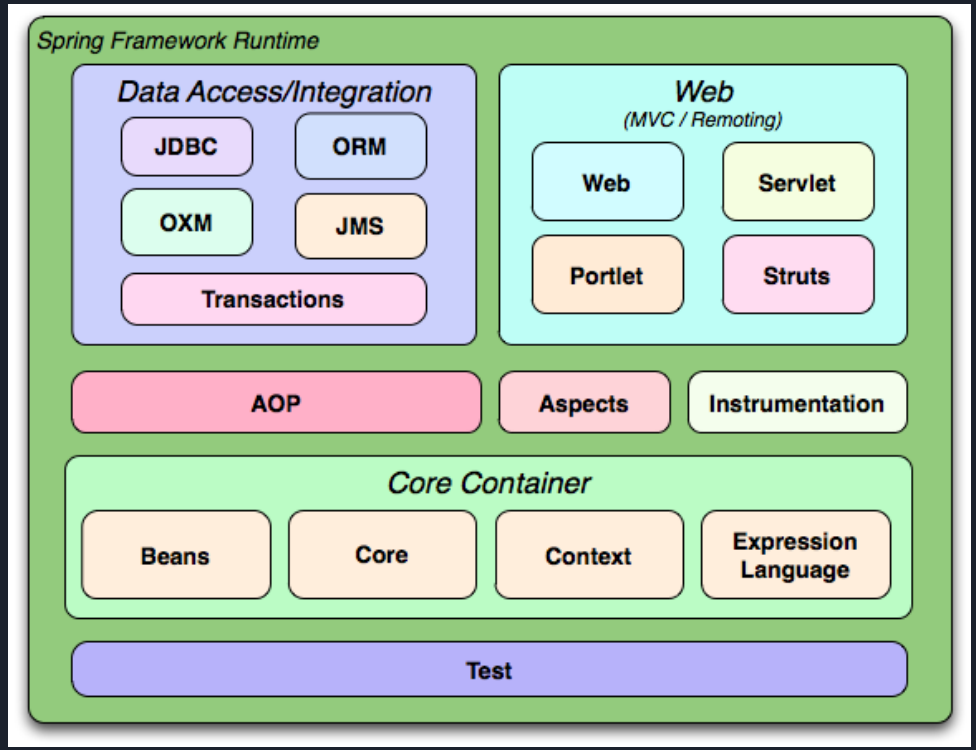
Il existe de nombreuses technologies qui peuvent être utilisées pour mettre en œuvre une architecture web. Chacune de ces technologies a ses propres caractéristiques et spécificités, qui en font un choix plus ou moins adapté selon les besoins et les contextes de l'application.

- Java
- .NET
- Node.js
- Python Django

Java

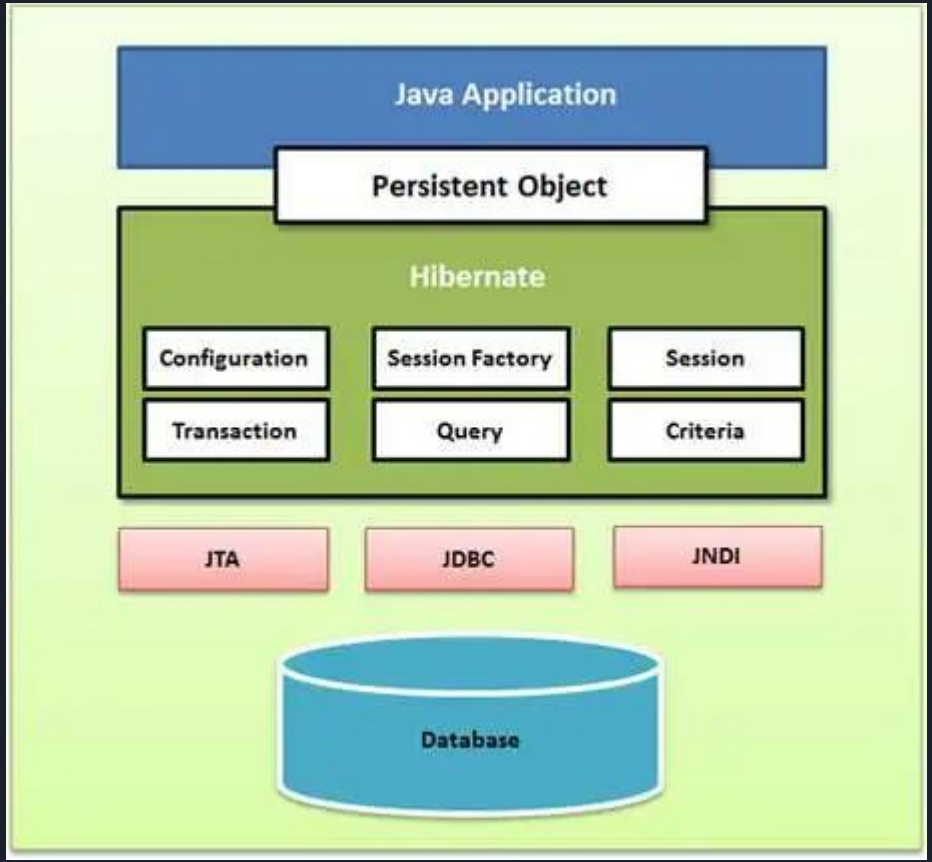
Java est un langage de programmation populaire et polyvalent, qui est utilisé pour développer des applications web côté serveur. Il est souvent utilisé avec des frameworks tels que Spring ou Hibernate, qui facilitent la mise en œuvre de l'architecture backend de l'application. Java est particulièrement adapté aux applications à grande échelle et aux environnements distribués, grâce à sa robustesse, sa portabilité et sa sécurité.

Framework Java Spring



Framework Java Hibernate

Hibernate est un framework open source gérant la persistance des objets en base de données relationnelle



.NET

.NET est un framework de développement Microsoft qui fournit un ensemble d'outils et de bibliothèques pour développer des applications web côté serveur (backend).

Le framework .NET est particulièrement adapté aux applications d'entreprise et aux environnements Microsoft, grâce à sa puissance, sa flexibilité et sa compatibilité avec les autres technologies Microsoft.

C#

VB.NET

F#

Node.js

Node.js est un environnement d'exécution JavaScript côté serveur (backend), qui permet de développer des applications web en utilisant le même langage côté serveur (backend) et côté client.

Il est basé sur le moteur JavaScript V8 de Google Chrome et utilise un modèle d'exécution asynchrone pour gérer les requêtes HTTP et les événements de manière performante.

Node.js est particulièrement adapté aux applications en temps réel et aux environnements de microservices, grâce à sa légèreté, sa rapidité et sa scalabilité.

Python Django

Django est un framework de développement web open source basé sur le langage de programmation Python. Il fournit un ensemble de bibliothèques et d'outils pour développer rapidement et facilement des applications web côté serveur (backend).

Django est particulièrement adapté aux applications web de petite et moyenne envergure, grâce à sa simplicité, sa flexibilité et sa puissance. Il est souvent utilisé pour développer des applications web à fort contenu dynamique, comme des blogs, des sites de commerce électronique ou des réseaux sociaux.

La création d’un environnement de développement et de déploiement pour les services web

Le cycle de vie d’une application

Lorsqu’on développe et déploie des services web, il est important de mettre en place un environnement de développement et de déploiement adapté et optimisé. Cet environnement doit permettre de gérer les différentes étapes du cycle de vie de l’application, depuis la création du code jusqu’à la mise en production, en passant par le test et la validation.

Cas d’étude en groupe

Par groupe, vous allez étudier et présenter une des méthodologies citées ci-dessous :

Groupe 1 : Méthodologie AGILE SCRUM
Groupe 2 : Méthodologie Cycle en V
Groupe 3 : Méthodologie DevOps
Groupe 4 : Méthodologie Cascade

Les différents environnements :

Il existe plusieurs types d'environnements informatiques, qui peuvent être utilisés à différentes fins et dans différentes situations.

- Environnement de développement
- Environnement de test (staging)
- Environnement de production

L’environnement de développement

Environnement de développement (dev): Cet environnement est utilisé par les développeurs.

L’environnement de test (staging)

Environnement de test (staging): Cet environnement est utilisé pour préparer et tester le code et les applications avant qu’ils ne soient déployés en production. Il peut être configuré de manière similaire à l’environnement de production, afin de permettre aux équipes de vérifier comment le code et les applications se comporteront une fois en production.

L’environnement de production

Cet environnement est utilisé pour exécuter le code et les applications de manière permanente et pour les rendre disponibles aux utilisateurs finaux.

Conclusion

Il est important de maintenir des environnements séparés pour chaque étape du développement et du déploiement, afin de s’assurer que le code et les applications sont testés et déboguées correctement avant d’être mis en production.

Cela peut aider à éviter les erreurs et les problèmes de performance en production. Il est également à noter que le nombre d'environnement varie selon les entreprises.

Git Workflow

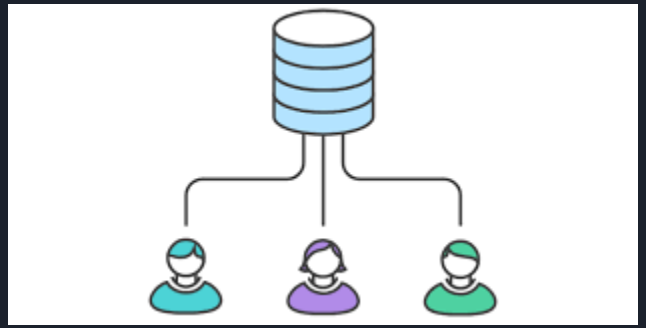
Un workflow Git est un ensemble de règles et de pratiques qui définissent la manière dont une équipe de développement utilise et gère un système de contrôle de version (SCV) tel que Git. Un workflow Git peut inclure des règles sur la manière de créer et de fusionner des branches, de rédiger des messages de commit, de gérer les conflits, de publier du code, etc.

Le Workflow Github Flow

Le workflow Github Flow est un workflow Git qui a été conçu par l'équipe de développement de Github. Il se base sur la création de branches pour chaque fonctionnalité.

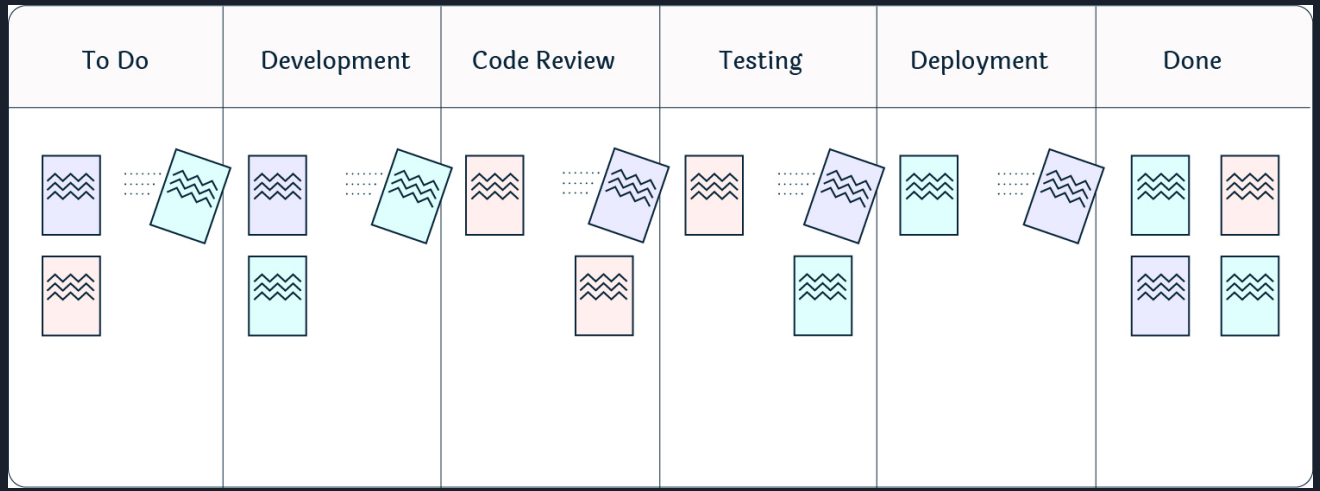
Le Workflow Centralisé

Le workflow centralisé est le plus simple et le plus classique des workflow Git. Il se base sur un dépôt centralisé (sur un serveur ou sur un service en ligne) qui sert de référence commune pour tous les membres de l'équipe. Chaque membre peut créer des branches à partir de ce dépôt central et y apporter des modifications, qui doivent être validées et fusionnées par un responsable avant d'être publiées.

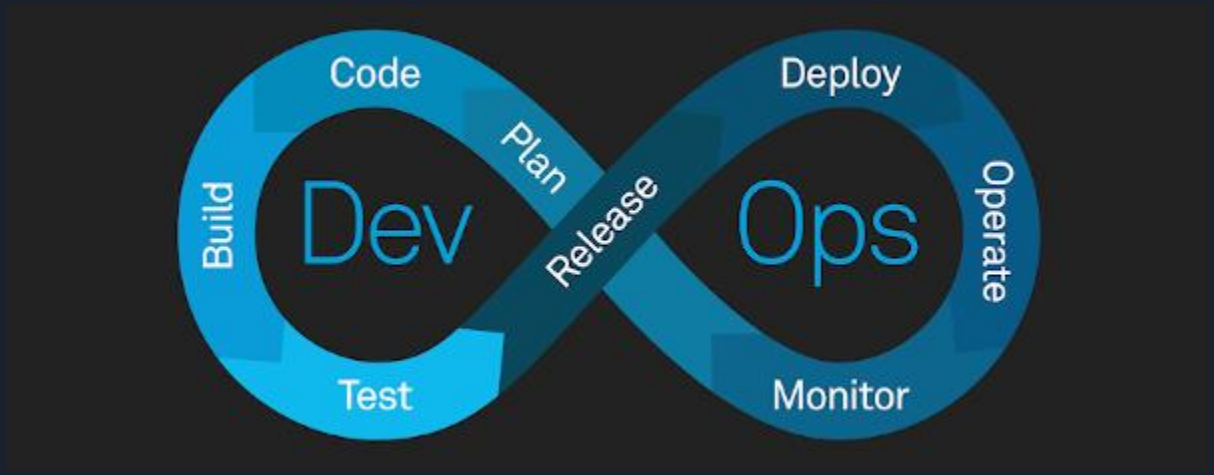


Le Workflow Kanban Flow

Workflow Kanban Flow : Le workflow Kanban Flow est un workflow Git qui s'inspire du modèle Kanban de gestion de projet. Il se base sur la création de tâches et de cartes dans un tableau Kanban, qui représentent les différentes étapes du développement (à faire, en cours, terminé, etc.). Chaque carte peut être associée à une branche Git, qui contient le code correspondant à cette tâche. Le workflow Kanban Flow permet de visualiser et de suivre l'avancement du projet en temps réel, et de s'adapter aux changements de priorité et de contexte.



Le CI/CD


Code
Plan
Build
Dev
Test
Release
Deploy
Operate
Ops
Monitor

L’intégration Continue (CI)

L’intégration continue (CI) est une pratique de développement de logiciel qui consiste à intégrer les modifications et les ajouts de code de manière régulière et automatisée dans le code source principal du projet. L’objectif de l’intégration continue est de s’assurer que le code source est stable et fonctionnel à tout moment, en détectant et en corrigeant les erreurs et les bugs de manière précoce.

L’intégration Continue (CI)

Pour mettre en place une intégration continue, il est nécessaire de configurer un environnement de développement et de déploiement automatisé, qui inclut des outils de build, de test et de déploiement. Ces outils permettent de construire, de tester et de déployer le code source de manière automatisée, en s’appuyant sur un système de contrôle de version (Git, SVN, etc.) et sur un serveur d’intégration continue (Jenkins, Travis CI, etc.).

L’intégration Continue (CI)

L’intégration continue s’appuie sur une stratégie de développement en continu, qui consiste à développer et à intégrer le code de manière continue et itérative, plutôt que de le faire de manière séquentielle et ponctuelle. Cette stratégie permet de réduire les temps de développement et de déploiement, de s’assurer de la qualité et de la stabilité du code, et de faciliter la collaboration et la communication au sein de l’équipe de développement.

Outils de build

Les outils de build sont des logiciels qui permettent de compiler, d'assembler et de packager le code source de l'application en vue de sa mise en production. Ils peuvent automatiser et optimiser différentes tâches de build, comme la génération de fichiers binaires, la création de fichiers de configuration, l'inclusion de dépendances, la minification de code, etc. Exemples d'outils de build : Maven, Gradle, Ant, Webpack.

Remarque :

Il existe deux grandes catégories de langages de programmation : les langages compilés et les langages interprétés.

Langages compilés vs Langages interprétés

Langages compilés

- Doivent être compilés avant exécution
- Exécution plus rapide
- Exécution plus efficace
- Compilation longue et obligatoire
- Sécurité du code
- Environnement de dev lourd

Langages interprétés

- Plus simple d'utilisation
- Pas de compilation
- Exécution plus lente (interpréteur)
- Modification flexible
- Code lisible par tous
- Environnement de dev léger

Outils de test

Les outils de test sont des logiciels qui permettent de vérifier et de valider le bon fonctionnement et la qualité de l'application. Ils peuvent exécuter différents types de tests, comme les tests unitaires, les tests d'intégration, les tests de performance, les tests de sécurité, etc. Les résultats des tests sont généralement reportés dans des rapports, qui permettent de suivre l'état de l'application et de détecter les éventuels problèmes.

JUnit TestNG

Tests de performance

Les tests de performance sont des tests qui mesurent les indicateurs de performance d'un système, tels que sa vitesse, sa capacité de traitement, sa fiabilité, etc. Ils permettent de s'assurer que le système est capable de gérer les charges de travail prévues et de répondre aux exigences de performance définies.

Les tests de performance sont souvent exécutés par les testeurs ou les administrateurs système au cours de la phase de déploiement, pour valider les performances du système en conditions réelles.

Tests de sécurité

Les tests de sécurité sont des tests qui vérifient la sécurité d'un système, en détectant et en corrigeant les failles et les vulnérabilités qui pourraient être exploitées par des attaquants. Ils permettent de s'assurer que le système est protégé contre les menaces externes et internes, et qu'il respecte les règles de sécurité en vigueur. Les tests de sécurité sont souvent exécutés par des experts en sécurité ou des sociétés de testing spécialisées, pour valider la sécurité du système avant son déploiement.

Tests d’acceptation

Les tests d’acceptation sont des tests qui vérifient que le système répond aux besoins et aux attentes des utilisateurs ou des clients. Ils permettent de s’assurer que le système est facile à utiliser, ergonomique et fonctionnel, et qu’il respecte les standards et les normes en vigueur. Les tests d’acceptation sont souvent exécutés par les utilisateurs ou les clients finaux au cours de la phase de déploiement, pour valider l’adéquation du système aux besoins et aux attentes de l’organisation.

Déploiement Continu (CD)

Le déploiement continu (Continuous Deployment en anglais) est une pratique de développement logiciel qui consiste à automatiser et à intégrer de manière continue les étapes de build, de test et de déploiement d'une application. Le but du déploiement continu est de permettre une livraison rapide et fréquente de nouvelles fonctionnalités et de corrections de bugs, tout en maintenant un niveau élevé de qualité et de stabilité du code.

Déploiement Continu (CD)

Pour mettre en œuvre le déploiement continu, il est nécessaire de mettre en place une chaîne d'intégration et de déploiement (CI/CD en anglais) qui permet de gérer de manière automatisée et transparente les différentes étapes du processus de déploiement. La CI/CD peut être basée sur des outils de build, de test et de déploiement tels que Jenkins, Travis CI, CircleCI, etc.

CD sur un serveur de test

CD dans le cloud

Déploiement continu sur un serveur de staging

Dans ce scénario, chaque fois qu'un développeur valide un commit sur la branche de développement, le processus de CI/CD déclenche automatiquement la compilation et les tests de l'application, puis déploie le code sur un serveur de staging (environnement de préproduction). Les tests de non-régression et les validations manuelles peuvent être effectuées sur ce serveur avant de déclencher le déploiement sur le serveur de production.

Déploiement sur un serveur de production

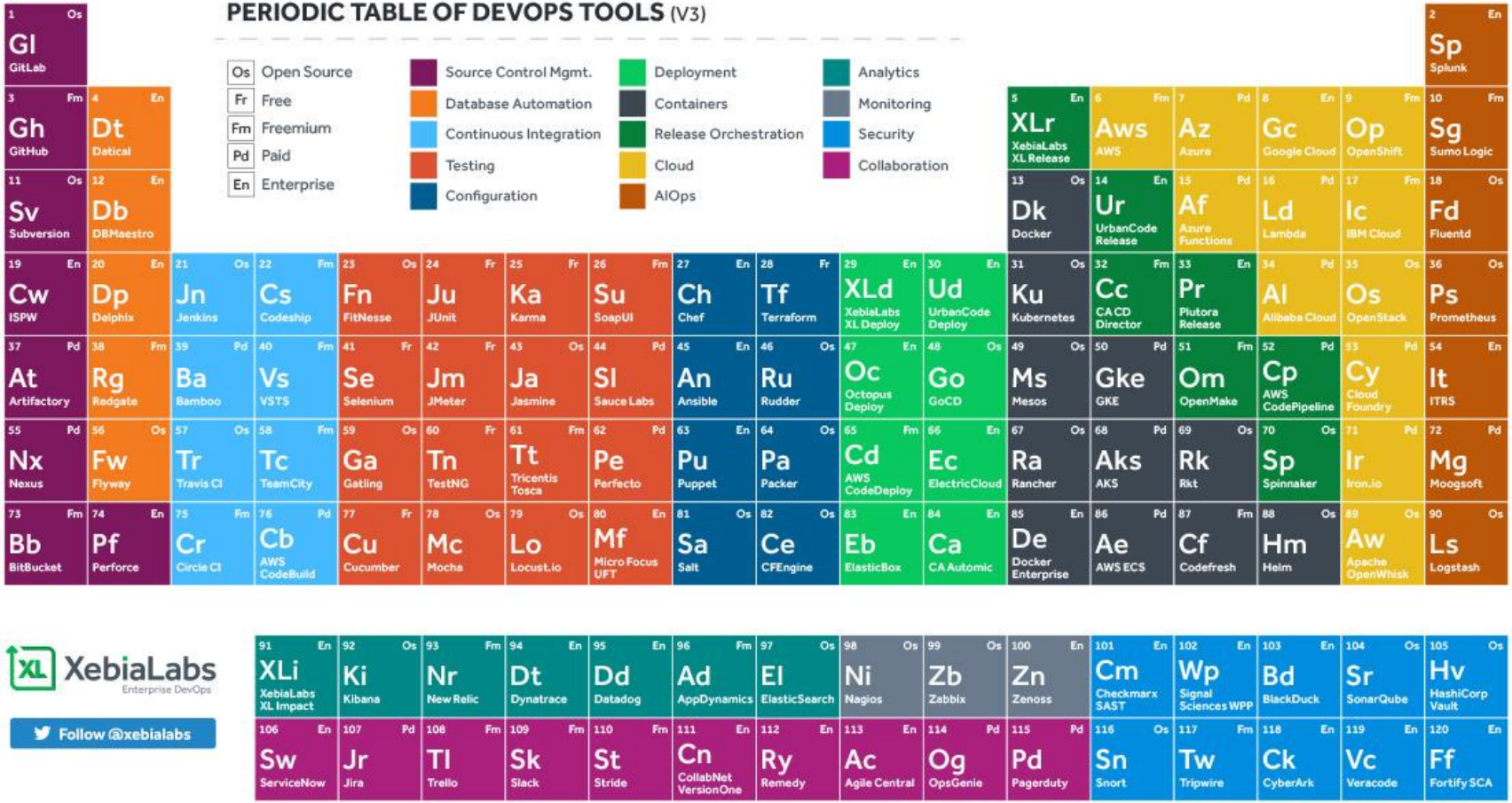
Dans ce scénario, le processus de CI/CD déploie automatiquement le code sur le serveur de production dès qu'il est validé sur la branche de développement. Cette approche est plus risquée, car elle nécessite de mettre en place des tests de qualité et de stabilité très rigoureux pour éviter les erreurs de déploiement.

Déploiement continu sur le cloud

Dans ce scénario, le processus de CI/CD déploie automatiquement l'application sur un environnement de cloud computing, comme AWS ou Google Cloud. Cette approche permet de bénéficier d'une scalabilité et d'une flexibilité importantes, ainsi que d'une facilité de gestion et de maintenance.

Outils de déploiement

Les outils de déploiement sont des logiciels qui permettent de déployer l'application sur un ou plusieurs serveurs de production. Ils peuvent automatiser et optimiser différentes tâches de déploiement, comme le transfert de fichiers, la configuration de l'application, le redémarrage de services, la mise à jour de la base de données, etc. Ils peuvent également gérer les différentes versions de l'application et les rollbacks en cas de problème. Exemples d'outils de déploiement : Jenkins, Github Action, Gitlab CI, ArgoCD ...


PERIODIC TABLE OF DEVOPS TOOLS (V3)
Os Open Source
Fr Free
Fm Freemium
Pd Paid
En Enterprise
Source Control Mgmt.
Database Automation
Continuous Integration
Testing
Configuration
Deployment
Containers
Release Orchestration
Cloud
AIOps
Analytics
Monitoring
Security
Collaboration
1 Os Gl GitLab
2 En Sp Splunk
3 Fm Gh GitHub
4 En Dt Datical
5 En XLr XebiaLabs XL Release
6 Fm Aws AWS
7 Pd Az Azure
8 En Gc Google Cloud
9 Fm Op OpenShift
10 Fm Sg Sumo Logic
11 Os Sv Subversion
12 En Db DBMaestro
13 Os Dk Docker
14 En Ur UrbanCode Release
15 Pd Af Azure Functions
16 Pd Ld Lambda
17 Fm Ic IBM Cloud
18 Os Fd Fluentd
19 En Cw ISPW
20 En Dp Delphix
21 Os Jn Jenkins
22 Fm Cs Codeship
23 Os Fn FitNesse
24 Fr Ju JUnit
25 Fr Ka Karma
26 Fm Su SoapUI
27 En Ch Chef
28 Fr Tf Terraform
29 En XLd XebiaLabs XL Deploy
30 En Ud UrbanCode Deploy
31 Os Ku Kubernetes
32 Fm Cc CA CD Director
33 En Pr Plutora Release
34 Pd Al Alibaba Cloud
35 Os Os OpenStack
36 Os Ps Prometheus
37 Pd At Artifactory
38 Fm Rg Redgate
39 Pd Ba Bamboo
40 Fm Vs VSTS
41 Fr Se Selenium
42 Fr Jm JMeter
43 Os Ja Jasmine
44 Pd Sl Sauce Labs
45 En An Ansible
46 Os Ru Rudder
47 En Oc Octopus Deploy
48 Os Go GoCD
49 Os Ms Mesos
50 Pd Gke GKE
51 Fm Om OpenMake
52 Pd Cp AWS CodePipeline
53 Pd Cy Cloud Foundry
54 En It ITRS
55 Pd Nx Nexus
56 Os Fw Flyway
57 Os Tr Travis CI
58 Fm Tc TeamCity
59 Os Ga Gatling
60 Fr Tn TestNG
61 Fm Tt Tricentis Tosca
62 Pd Pe Perfecto
63 En Pu Puppet
64 Os Pa Packer
65 Fm Cd AWS CodeDeploy
66 En Ec ElectricCloud
67 Os Ra Rancher
68 Pd Aks AKS
69 Os Rk Rkt
70 Os Sp Spinnaker
71 Pd Ir Iron.io
72 Pd Mg Moogsoft
73 Fm Bb BitBucket
74 En Pf Perforce
75 Fm Cr Circle CI
76 Pd Cb AWS CodeBuild
77 Fr Cu Cucumber
78 Os Mc Mocha
79 Os Lo Locust.io
80 En Mf Micro Focus UFT
81 Os Sa Salt
82 Os Ce CFEngine
83 En Eb ElasticBox
84 En Ca CA Automic
85 En De Docker Enterprise
86 Pd Ae AWS ECS
87 Fm Cf Codefresh
88 Os Hm Helm
89 Os Aw Apache OpenWhisk
90 Os Ls Logstash
XebiaLabs
Enterprise DevOps
Follow @xebialabs
91 En XLi XebiaLabs XL Impact
92 Os Ki Kibana
93 Fm Nr New Relic
94 En Dt Dynatrace
95 En Dd Datadog
96 Fm Ad AppDynamics
97 Os El ElasticSearch
98 Os Ni Nagios
99 Os Zb Zabbix
100 En Zn Zenoss
101 En Cm Checkmarx SAST
102 En Wp Signal Sciences WPP
103 En Bd BlackDuck
104 Os Sr SonarQube
105 Os Hv HashiCorp Vault
106 En Sw ServiceNow
107 Pd Jr Jira
108 Fm Tl Trello
109 Fm Sk Slack
110 Fm St Stride
111 En Cn CollabNet VersionOne
112 En Ry Remedy
113 En Ac Agile Central
114 Pd Og OpsGenie
115 Pd Pd Pagerduty
116 Os Sn Snort
117 Fm Tw Tripwire
118 En Ck CyberArk
119 En Vc Veracode
120 En Ff Fortify SCA

Exercice : Mise en place d’un workflow GitOps pour le déploiement d’une application web

En groupe de 4 étudiants, vous allez devoir imaginer et mettre en place un workflow GitOps pour le déploiement d’une application web (celle de votre choix). Vous devrez prendre en compte les éléments suivants dans votre workflow :

Exercice : Mise en place d’un workflow GitOps pour le déploiement d’une application web

La gestion des sources de code : comment gérer les versions et les branches de votre application ? Qui est responsable de la review du code et comment se fait-elle ?

La gestion des builds et des tests : comment automatiser la compilation et les tests de votre application ? Qui est responsable de la vérification de la qualité du code et comment se fait-elle ?

La gestion des déploiements : comment déployer votre application sur différents environnements (développement, staging, production) ? Qui est responsable de la validation des déploiements et comment se fait-elle ?

La gestion des rollbacks : comment gérer les cas où un déploiement ne se passe pas comme prévu et il faut revenir à une version précédente de l’application ?

La documentation : comment documenter votre workflow de manière à ce qu’il soit facilement compréhensible et reproductible par d’autres membres de l’équipe ?

Exercice : Mise en place d’un workflow GitOps pour le déploiement d’une application web

Pour réaliser cet exercice, vous devrez imaginer un workflow GitOps adapté aux besoins de votre application et le mettre en place sur un repository Git. Vous devrez également documenter votre workflow de manière à ce qu’il soit facilement compréhensible par d’autres membres de l’équipe.

Note : vous pouvez utiliser des outils tels que GitLab, Jenkins ou GitHub Actions pour mettre en place votre workflow.