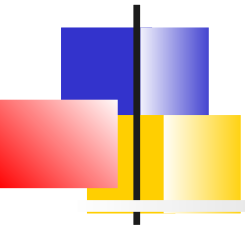
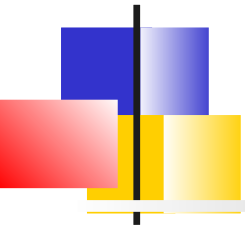


Recueil des pratiques de développement

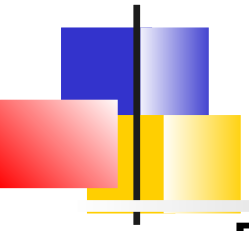
David THIBAU – 2024

david.thibau@gmail.com

Agenda

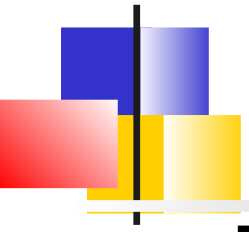


- **Parc applicatif**
- **Environnement de dév**
- **Gestion des sources**
- **Pilotage Projet**
- **CI/CD**
- **Qualité**
- **Releasing**
- **Tests d’intégration et d’acceptation**
- **Déploiement**
- **Observabilité**



Projets

- Parc applicatifs
  - Projet en maintenance
  - En cours de dév. ?
  - Fréquence de déploiement
  - Couplage entre applications
- Acteurs du projet
  - Combien de développeurs ? Mainteneur de projet
  - Métier
  - Infra



Stack Technologique

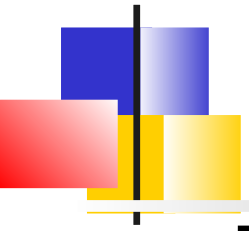
Persistance :

SQL, NOSQL, autres, LDAP, Keycloak, Broker ?

Langages :

Java, Kotlin (?), Javascript, TypeScript

Serveurs : Serveur JEE, Tomcat Embarqué, Netty embarqué, Node, Serverless



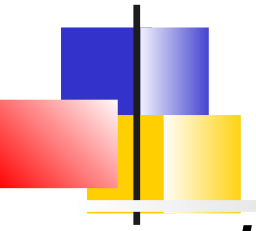
Frameworks

Framework maison : Retours ?

Spring Boot : Versions

Quarkus : ?

Ext.js ?

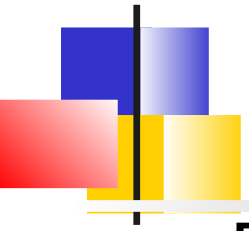


IDEs

Intellij IDEA (Communautaire / payant)
Dédié à Java et Kotlin

VSCode
Multi-langages

Eclipse : Le crépuscule ?
Lourdeur, Assistance moyenne



Copilot

Payant (licence corporate ou individuelle)

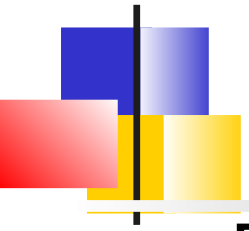
Pair-programming avec AI

Nécessite un compte GitHub

Compatible IntelliJ, VScode

Démo :

https://docs.github.com/fr/copilot/using-github-copilot/getting-started-with-github-copilot



SonarLint

Free

Détecte les mauvaises pratiques de codage en temps-réel

Peut être connecté à SonarQube et récupérer les personnalisations d’un projet

=> Permet d’apprendre les bonnes règles de codage tout langage
Mise à jour automatique des règles

IDEs pour SpringBoot

Plugins maintenu par l'équipe SB pour :

1) *VSCode*

2) Eclipse

3) Theia

Principaux Apports du plugin :

- **Ajout de starter** pour modifier son *pom.xml / build.gradle*
- Assistance pour l'**édition des propriétés de configuration** SB et applicative si starter *configuration-processor*
- ***Boot Dashboard*** (Démarrage / redémarrage des services), Intéressant si l'on développe plusieurs micro-services
- ***Run Configurations*** (Profil, Arguments de CLI)
- Redémarrage automatique si starter ***DevTools***

IntelliJ ?

Outils de build

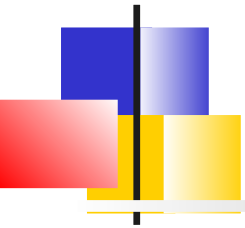
Maven, l'ancêtre :

- Très bien supporté dans les IDE
- Beaucoup de plugins disponibles
- Bien maîtrisé ?
- *pom.xml* verbeux et peu lisible
- Pas très performant et verbeux lors de l'exécution
- Limité à Java

Gradle, le challenger (choix par défaut *Spring Initializr*)

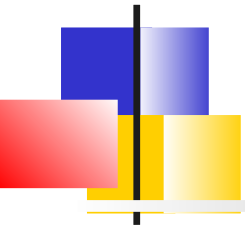
- Performant et rapide
- Beaucoup de plugins disponibles :
  - Plugin officiel => utilisable ad-hoc, exemple SB
  - Non officiel => Confiance relative, nécessite l'accès au soure
- Très customisable mais courbe d'apprentissage pas simple
- Moins bien supporté dans les IDE
- Bon support pour C++

TypeScript ? : ***ng*** ?



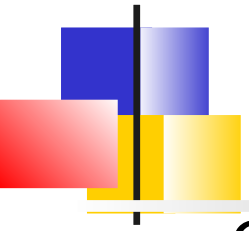
build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.2'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'org.formation'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'org.postgresql:postgresql'
    annotationProcessor 'org.springframework.boot:spring-boot-configuration-processor'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

tasks.named('test') {
    useJUnitPlatform()
}

```



build.gradle | DemoBuild-2/pom.xml ×

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XML
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/m
4      <modelVersion>4.0.0</modelVersion>
5      <parent>
6          <groupId>org.springframework.boot</groupId>
7          <artifactId>spring-boot-starter-parent</artifactId>
8          <version>3.2.2</version>
9          <relativePath/> <!-- lookup parent from repository -->
10     </parent>
11     <groupId>org.formation</groupId>
12     <artifactId>DemoBuild-2</artifactId>
13     <version>0.0.1-SNAPSHOT</version>
14     <name>DemoBuild-2</name>
15     <description>Demo project for Spring Boot</description>
16     <properties>
17         <java.version>17</java.version>
18     </properties>
19     <dependencies>
20         <dependency>
21             <groupId>org.springframework.boot</groupId>
22             <artifactId>spring-boot-starter-actuator</artifactId>
23         </dependency>
24         <dependency>
25             <groupId>org.springframework.boot</groupId>
26             <artifactId>spring-boot-starter-data-jpa</artifactId>
27         </dependency>
28         <dependency>
29             <groupId>org.springframework.boot</groupId>
30             <artifactId>spring-boot-starter-web</artifactId>
31         </dependency>
32
33         <dependency>
34             <groupId>org.springframework.boot</groupId>
35             <artifactId>spring-boot-devtools</artifactId>
36             <scope>runtime</scope>
37             <optional>true</optional>
38         </dependency>
39         <dependency>
40             <groupId>org.postgresql</groupId>
```



Tests Développeur

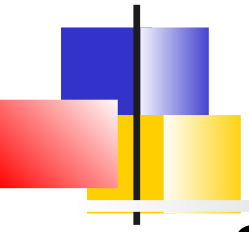
Opinion sur les tests unitaires ?

- Utilité (Dév, recette?)
- Granularité.
  TU dans un contexte de framework comme SB

Pratiques ? TDD

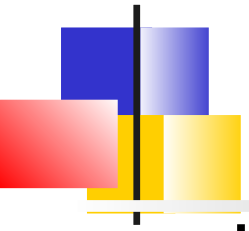
Calcul de couverture ?

Démo : Tests en continus Quarkus



Tests SpringBoot

- Support JUnit5, Hamcrest, AssertJ, Mockito, JsonPath
- Tests *système : @SpringBootTest*
- Tests d'intégration *: @DataJpaTest, @DatMongoTest, @JsonTest, @WebFluxTest*
- Sécurité *: @WithMockUser, ...*



Code, Deploy, Test

L’utopie : « Live Coding »

Les frameworks essaient d’y arriver :

- ng
- DevTools SpringBoot
- quarkus dev

La TDD peut raccourcir le cycle.
Plus rapide de lancer le test que de démarrer un serveur et effectuer des clicks

Services de support

Pour développer on besoin de services de support comme une BD par exemple:

- BD de test partagée.
  => Les développeurs se marchent dessus
- Installation locale.
  => Les développeurs ont ils vraiment la même installation
  => Que faire si l ‘on a besoin de 2 versions différentes pour 2 projets différents ?
- Base de données embarquée.
  => Très pratique car toujours dans un état connu
- docker-compose.
  => Très pratique car toujours dans un état connu
  => Plus proche des conditions réelles

Gestion des sources

Dépôt

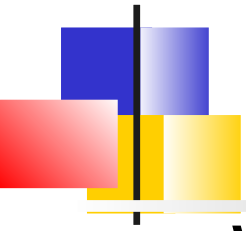
Git ou autre ?

Pratiques :

- IDE, Ligne de commande ?
- Utilisation de branches locales
- Fréquences des commits, des push
- Commandes avancées
  git rebase -i
  git cherry-pick
  ....

Référentiel projet

- Gitlab
- Github
- Custom ?

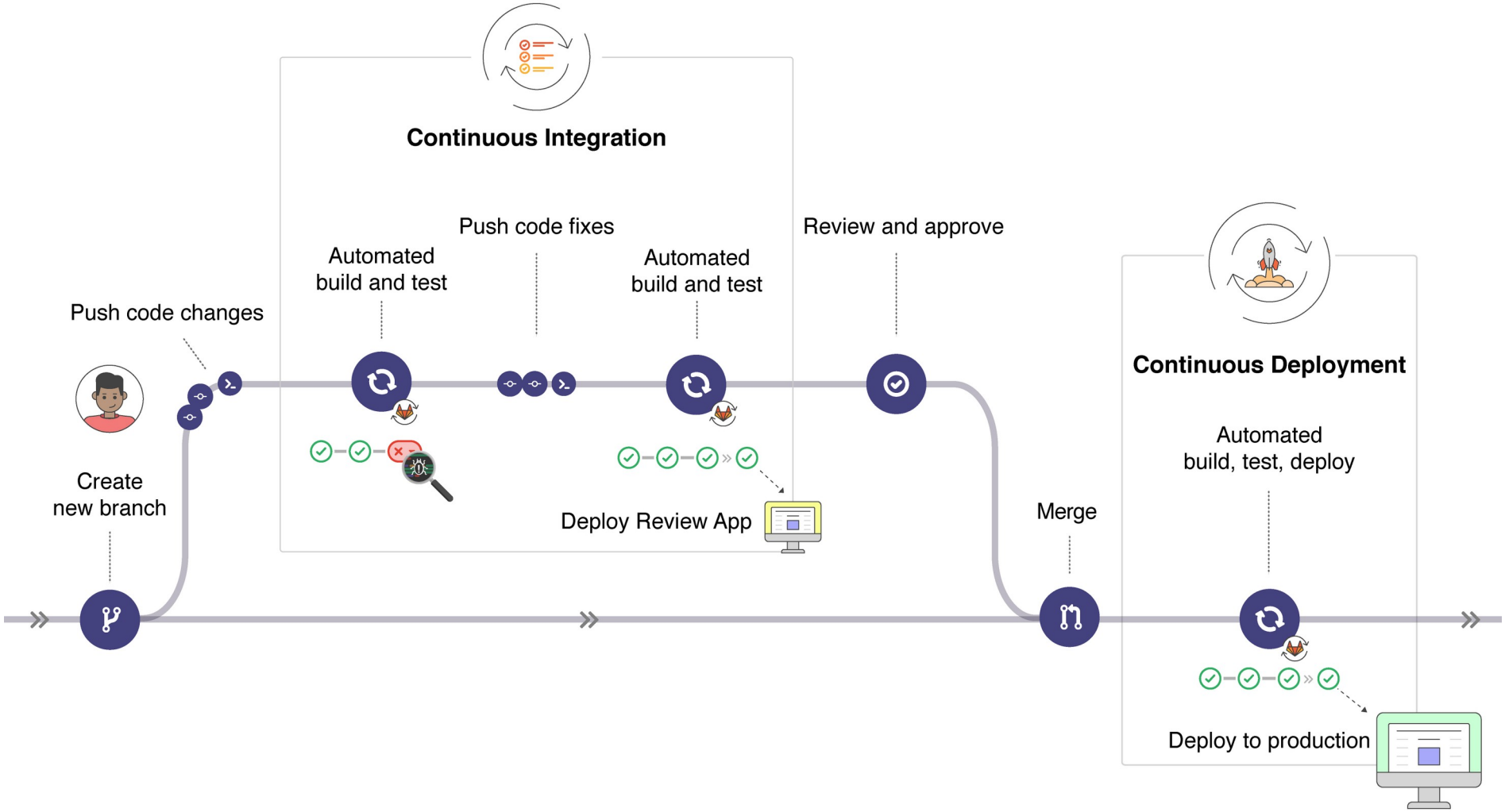


Workflow

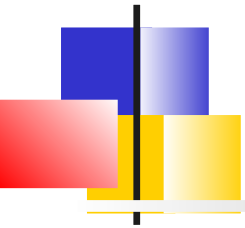
Worlflow de collaboration :

- Rôles dans un projet : Développeur, Mainteneur, PO ?
- PR, MR ?
- Branches stables : main, master, pré-prod, prod ?
- Revue de code avant intégration dans branches stables

Exemple Gitlab CI



Pilotage projet



Pratiques

Outils ?

Dashboard KanBan ?

Lien entre issue et commits

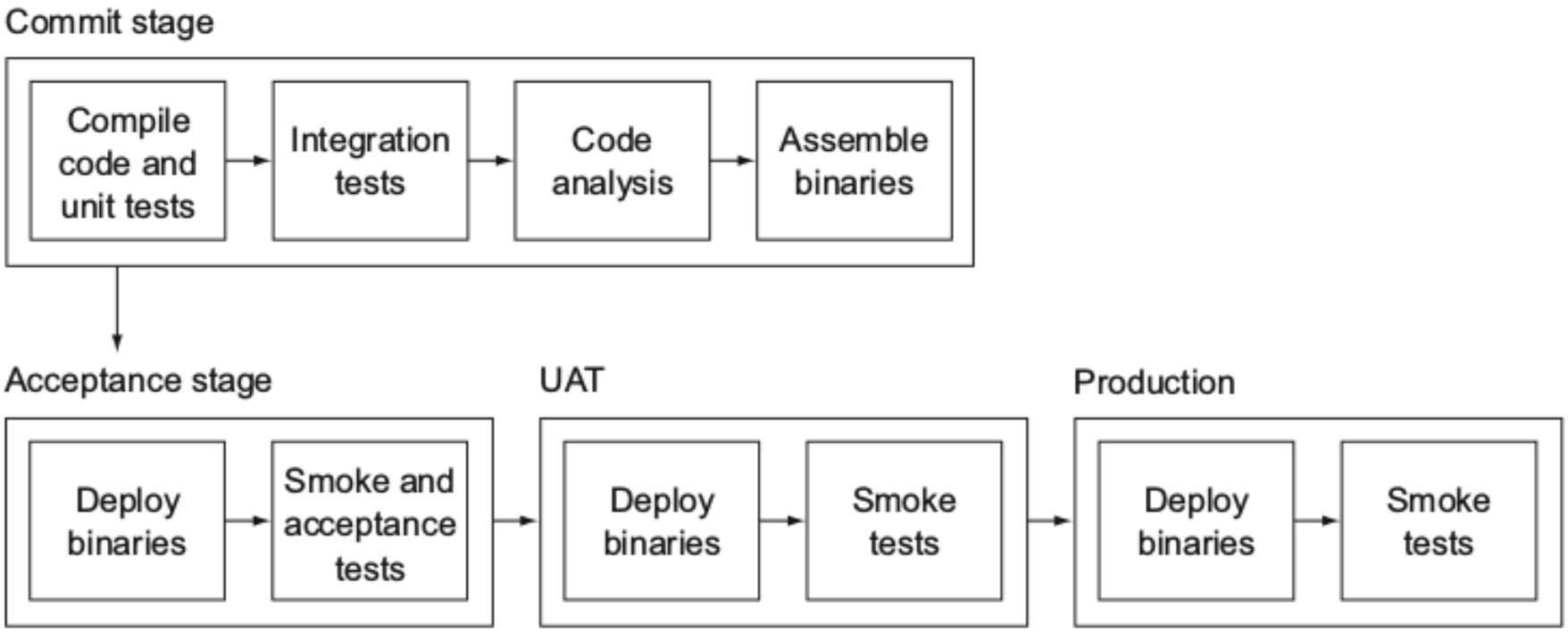
Implication du métier

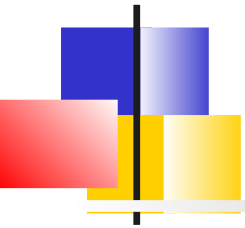
Démo : MR Gitlab

CI/CD

Distinction CI/CD

Exemple CD


Commit stage
Compile code and unit tests
Integration tests
Code analysis
Assemble binaries
Acceptance stage
Deploy binaries
Smoke and acceptance tests
UAT
Deploy binaries
Smoke tests
Production
Deploy binaries
Smoke tests



Pratiques

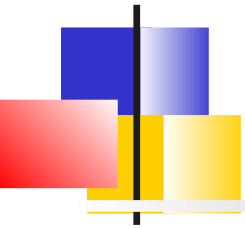
Outils :

- Jenkins pipeline en groovy
  => Souplesse
- Gitlab-ci déclaratif lié à Gitlab et aux containers

Publication de métriques :

- Rapport des tests unitaires, intégration performance
- Rapport des analyses statiques

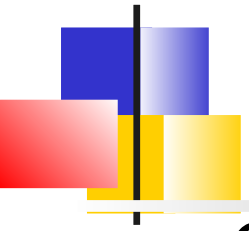
Qualité



Analyse statique

Outils :

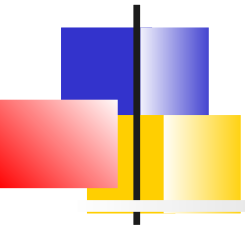
- Sonarqube : Détection de bugs, code smells, Vulnérabilité
- Gitlab / Code climate



Analyse statique de vulnérabilité (SAST)

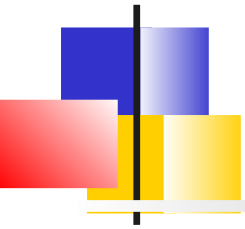
- Code source:
  - Free :
    - FindSecurityBug (Appli Java)
    - Gitlab CI
  - Commerciaux :
    - CheckMarx
    - Snyk
- Dépendances :
  - Dependency-check Maven plugin
  - Gitlab CI + Container Scanning

Analyse dynamique (DAST)



- Gitlab (version payante?)
  - DAST, API Fuzzing
- OWASP zap
- Burp (commercial)

Releasing



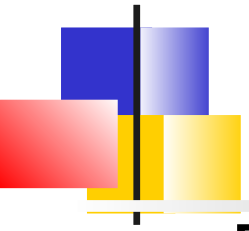
Dépôts d’artefacts

Outils :

- Nexus
- Artifactory
- Gitlab Package

Packaging :

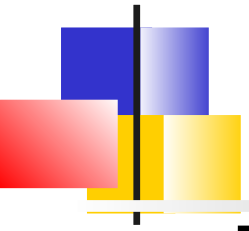
- War
- Jar
- Docker



Release

- Processus de release automatisé ?
  - Vérifier que tout est committé
  - Packaging et test
  - Mise à jour de la version, Commit, push et tag
  - Déploiement dépôt d’artefact
  - Incrément version, Commit et push
- Maven Release Plugin, Gradle Release Plugin

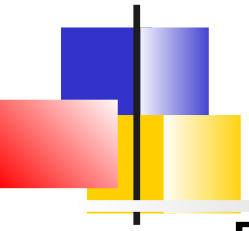
Tests d’intégration et d’acceptation



Pratiques

- Environnements Intégration, QA
- Quel type de test dans ces environnements
- Outils :
  - Selenium, QA

Déploiement



Pratiques

DevOps ou dépendant de l’équipe Infra ?

Fréquence des déploiements ?

Procédure de déploiement automatisé ?

A partir de dépôt ?

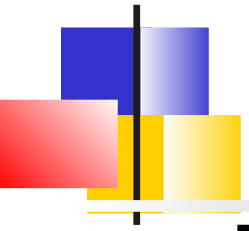
Déploiement immuable ?

Roll-back possible ?

Outil de migration de schéma

Ex : Déploiement immuable Kubernetes

Retours sur la production



Pratiques

- Endpoint de surveillance
  - Métriques disponibles
- Statistiques d’utilisation ?
- Mécanisme de feed-back utilisateurs finaux
- Gestion des issues en prod

Démo : Tableau de bord Grafana