

---

# Java Avancé

## Cours 1 : Outillage

---

Arsène Lapostolet

19 Janvier 2024

# Les outils du module

- Plateforme de développement : Machine Virtuelle Java (JVM)
- Langage de programmation : Java 21
- Environnement de développement intégré : IntelliJ IDEA Community
- Système de Gestion de Version : Git et Github
- Système de build : Gradle

# Git



# Qu'est ce que Git ?

Système de Gestion de Version :

- Versionner le code : suivre précisément les changements
- Naviguer entre les versions
- Ne pas perdre de changements
- Revenir en arrière
- Coopération à plusieurs sur le même code

# Concepts importants

- **Commit** : version du code sauvegardée à un instant T.
- **Branche** : historiques de commits qui évoluent en parallèle

Créer un dépôt local :

```
1 mkdir dossierPourMonDepot  
2 cd dossierPourMonDepot  
3 git init
```

Bash

# Base d'un workflow Git

- Créer un commits
  1. `git add .` : ajouter tous les changements au suivi git
  2. `git commit -m "message de commit"` : crée un commit avec les changements trackés
- Créer une branche :
  1. `git branche nomDeLaBranche` : créer la branche
  2. `git checkout nomDeLaBranche` : se positionner sur la branche

# Fusion

Pour appliquer les changement présents sur une branche à une autre :

```
1 git checkout brancheCible  
2 git merge brancheAFusionner
```

**Bash**

On appliquer les changement des la branche `brancheAFusionner` sur la branche `brancheCible`.

Attention aux conflits !

# Dépôt distant

```
1 git remote add nomLocalDuDepotDistant urlDuDepotDistant Bash
```

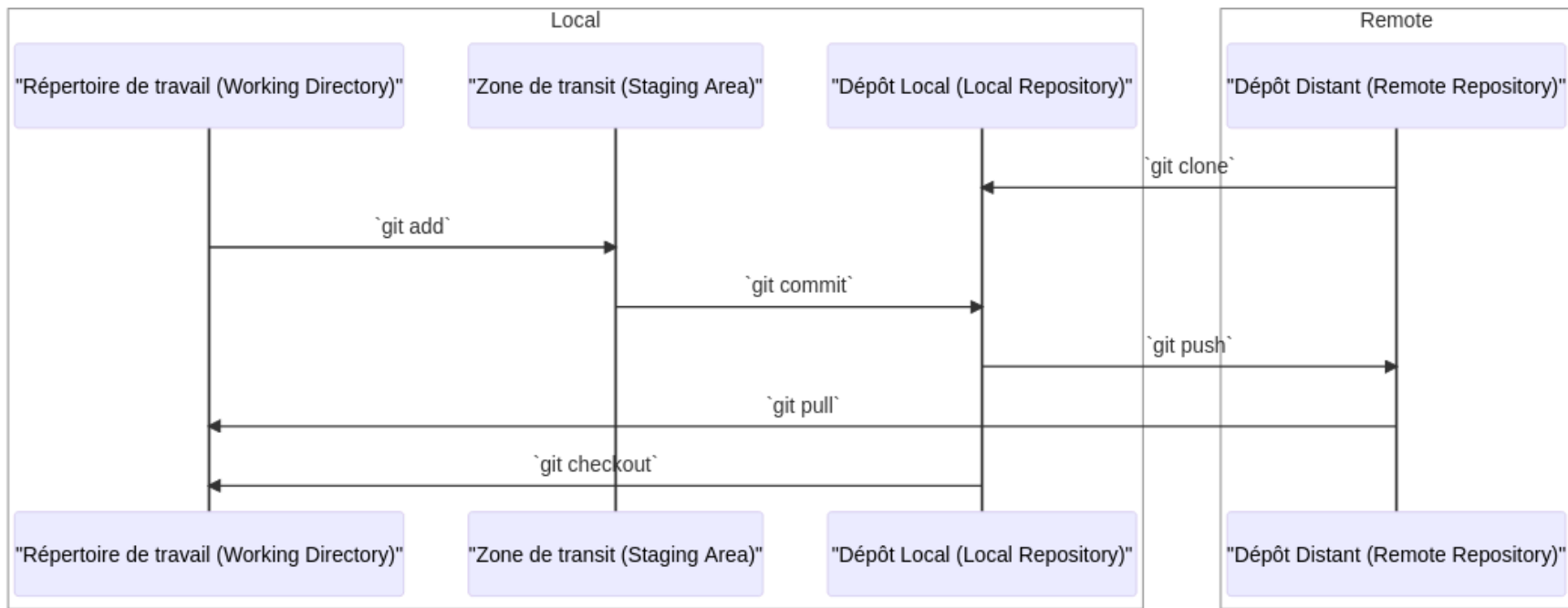
1. Sauvegarder son travail en lieu sur
2. Coopérer avec d'autres personnes

Interaction :

- `git push nomDeLaBranche` : pousser une branche locale vers le dépôt
- `git pull nomDeLaBranche` : récupérer une branche distante dans son dépôt local



# Récap des espaces



# Récap du workflow

1. Se positionner sur la branche principale : `git checkout master`
2. Mettre à jour la branche principale : `git pull origin master`
3. Créer une nouvelle branche à partir de la branche principale : `git branch maFeature`
4. Se positionner sur la nouvelle branche : `git checkout maFeature`
5. Faire des changements dans le code

6. Ajouter les changements à git : `git add .`
7. Créer un nouveau commit avec les changements : `git commit -m "message de commit"`
8. Pousser les changements : `git push origin maFeature`
9. Retourner sur master : `git checkout master`
10. Fusionner la branche de feature : `git merge maFeature`

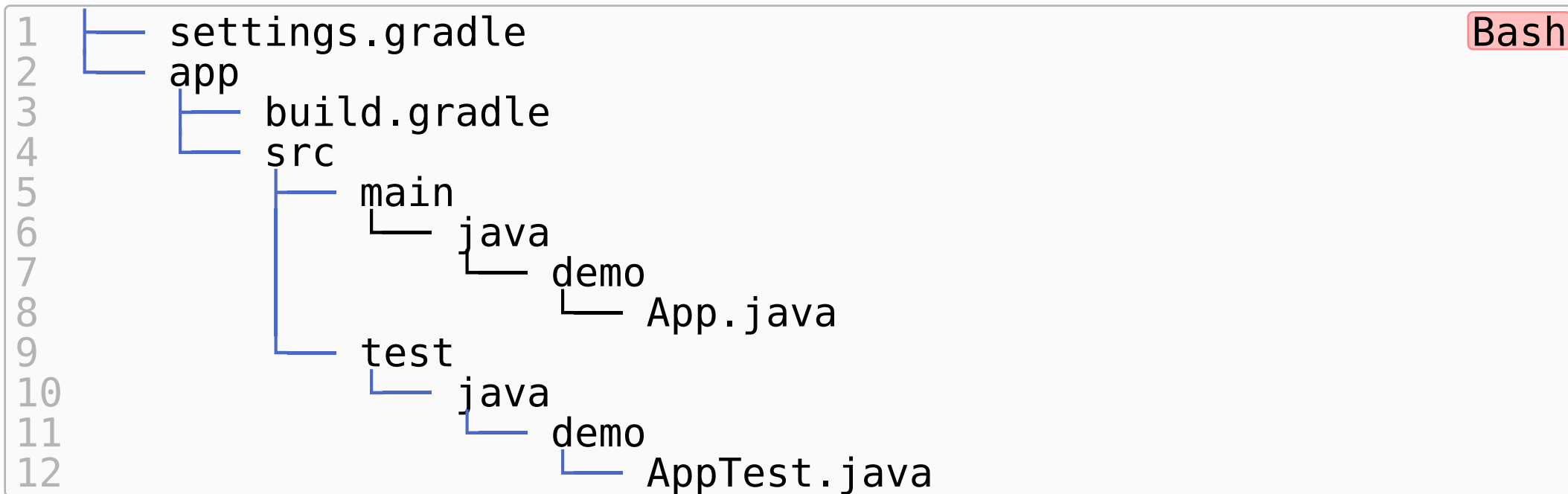
# Gradle

---

# Qu'est ce que Gradle ?

- Outil de build
  - Structurer le projet
  - Décrire comment le compiler, packager, exécuter, tester

# Structure d'un projet Gradle



## settings.gradle

```
1 rootProject.name = 'mon-projet'  
2 include('app')
```

Groovy

## app/build.gradle

```
1  plugins {  
2      id 'application'  
3  }  
4  
5  repositories {  
6      mavenCentral()  
7  }  
8  
9  application {  
10     mainClass = 'fr.arsenelapostolet.App'  
11 }  
12
```

Groovy



```
13
14 dependencies {
15     testImplementation 'org.junit.jupiter:junit-jupiter:5.9.2'
16     testRuntimeOnly 'org.junit.platform:junit-platform-
    launcher'
17     implementation 'com.google.guava:guava:31.1-jre'
18 }
19
20 java {
21     toolchain {
22         languageVersion = JavaLanguageVersion.of(21)
23     }
24 }
25
```

```
26  
27 tasks.named('test') {  
28     useJUnitPlatform()  
29 }
```

# Commandes de base

- Pour exécuter l'application : `./gradlew run`
- Pour exécuter les tests : `./gradlew test`

Sur windows : `./gradlew` est remplacé par `./gradlew.bat`