Squash Test Factory

Usine de tests logiciels

Bertrand FRANCHET
Frédéric LAURENS

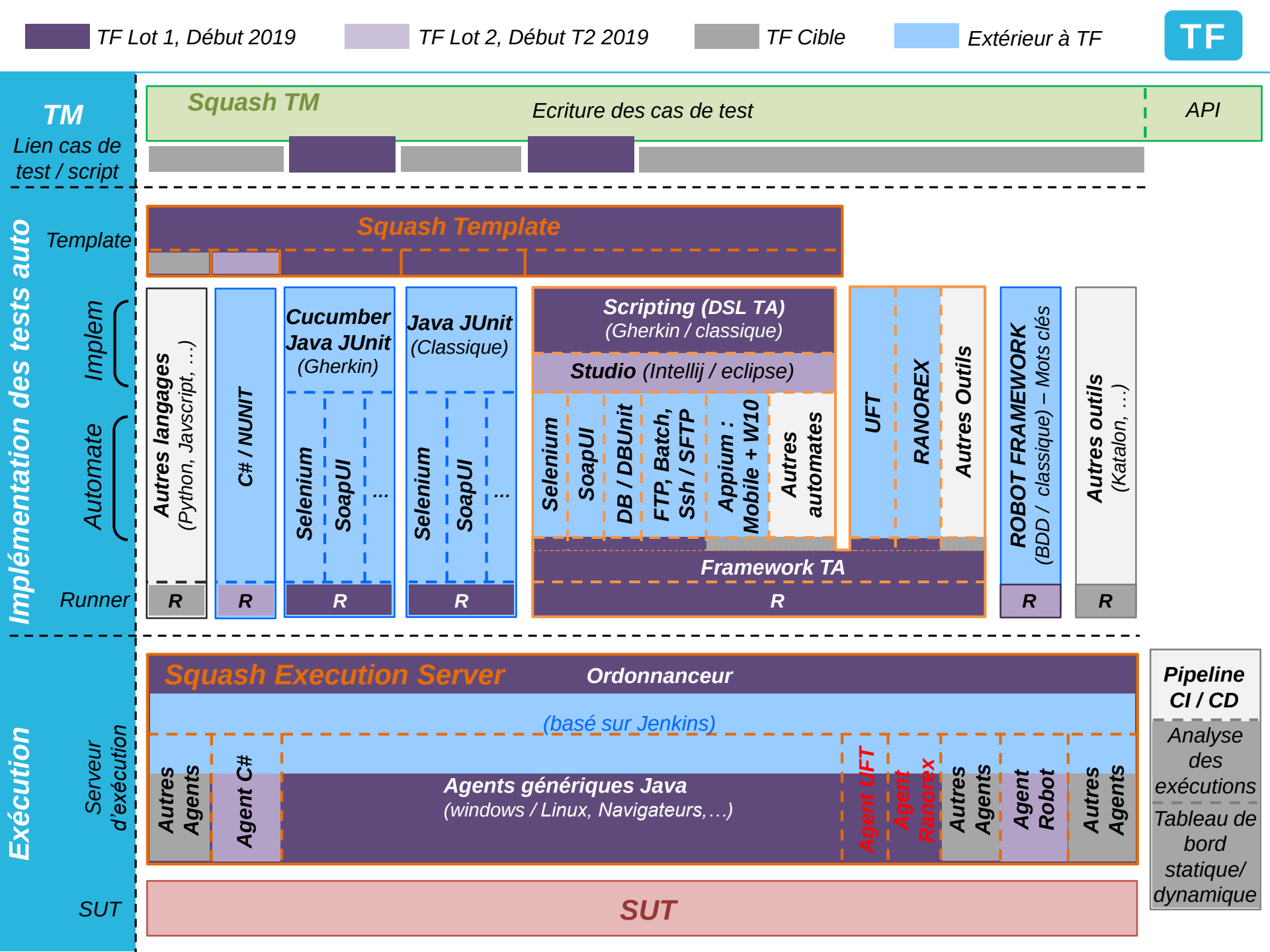
Squash TF Principe

Squash Test Factory : Principes/ Squash TA

TF

- **Constat**
  - Obligation d'utiliser un environnement de développement propriétaire Squash TA
  - Manque de packaging/installation difficile
  - Pas de mécanique/dynamique d'intégration des contributions externes
  - Pas de support du BDD

- **Principes**
  - Permettre d'automatiser avec d'autres frameworks que Squash TA
  - Fournir des environnements d'exécutions prépackagés + installeurs + fourniture de templates
  - Open source + Architecture modulaire + Amélioration Documentation + mécanisme d'intégration des contributions externes
  - Intégration Gherkin avec Squash TM


TF Lot 1, Début 2019
TF Lot 2, Début T2 2019
TF Cible
Extérieur à TF
TF
TM
Lien cas de test / script
Squash TM
Ecriture des cas de test
API
Implémentation des tests auto
Template
Implem
Automate
Runner
Squash Template
Autres langages (Python, Javscript, ...)
C# / NUNIT
Cucumber Java JUnit (Gherkin)
Selenium
SoapUI
...
Java JUnit (Classique)
Scripting (DSL TA) (Gherkin / classique)
Studio (Intellij / eclipse)
Selenium
SoapUI
DB / DBUnit
FTP, Batch, Ssh / SFTP
Appium : Mobile + W10
Autres automates
Framework TA
UFT
RANOREX
Autres Outils
ROBOT FRAMEWORK (BDD / classique) – Mots clés
Autres outils (Katalon, ...)
R
Exécution
Serveur d'exécution
SUT
Squash Execution Server
Ordonnanceur
(basé sur Jenkins)
Autres Agents
Agent C#
Agents génériques Java
(windows / Linux, Navigateurs,...)
Agent UFT
Agent Ranorex
Agent Robot
SUT
Pipeline CI / CD
Analyse des exécutions
Tableau de bord statique/ dynamique

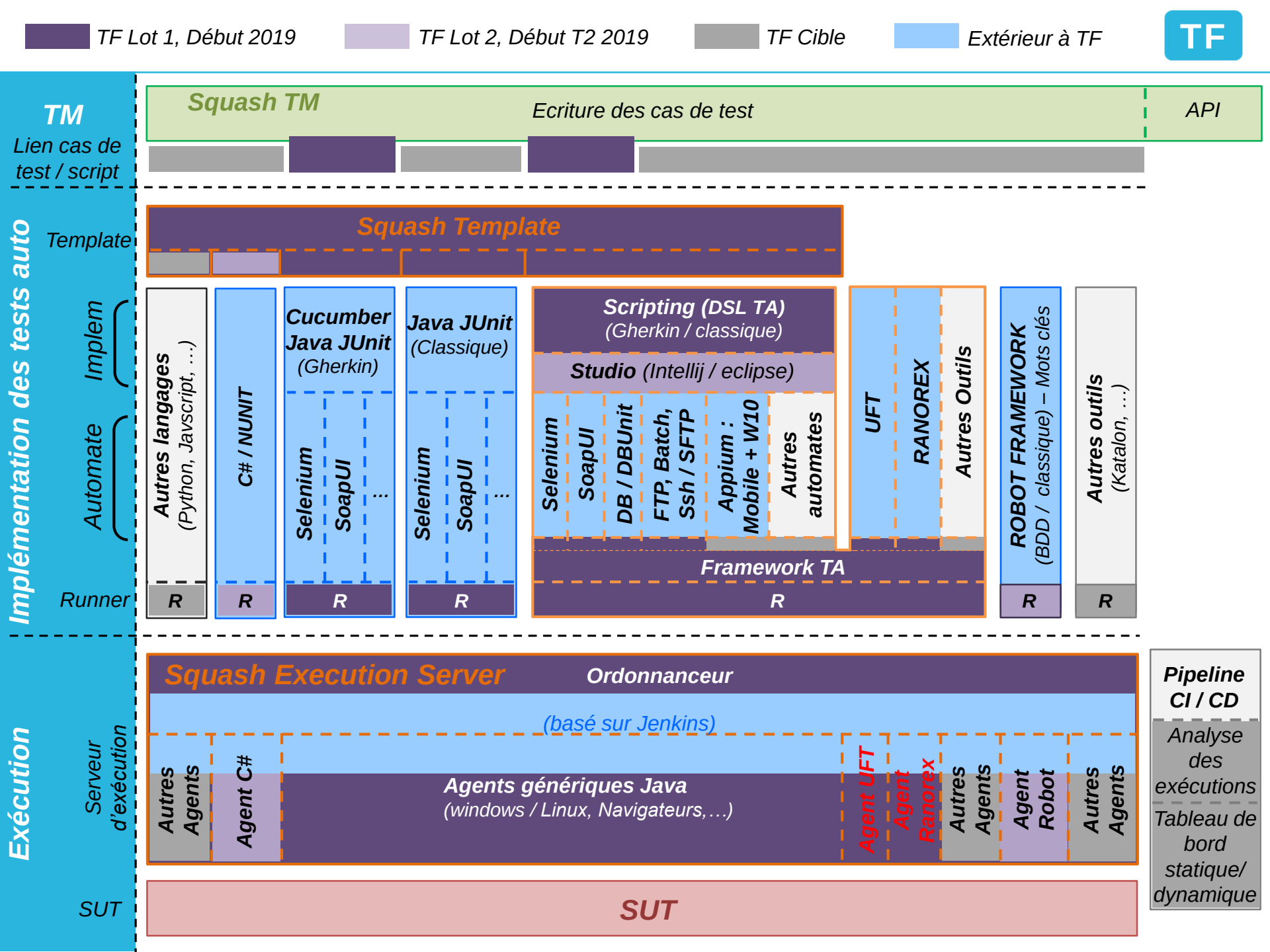
Squash Test Factory : **Les fonctionnalités**

TF

- **Ecriture** des cas de test (bdd / gherkin ou classique) + **workflow** testeurs - automaticiens (Squash TM) avec export Gherkin et génération de squelette Cucumber

- **Implémentation** des cas de test
  - Générateur basé sur des templates (Squash Template)
  - Différentes solutions d'implémentation des tests automatisés, avec leur studio. A choisir (non exclusif) selon le besoin / contexte parmi :
    - Le Scripting TA (orienté mots clés) incluant de nombreux connecteurs à des automates sachant gérer du web, des webservices, du client lourd windows 10 + un studio basé sur IntelliJ
    - Des studios propriétaires (s'appuyant ou non sur le framework Squash)
    - Des implémentations orientées pure code (Cucumber Java JUnit, Java JUnit)

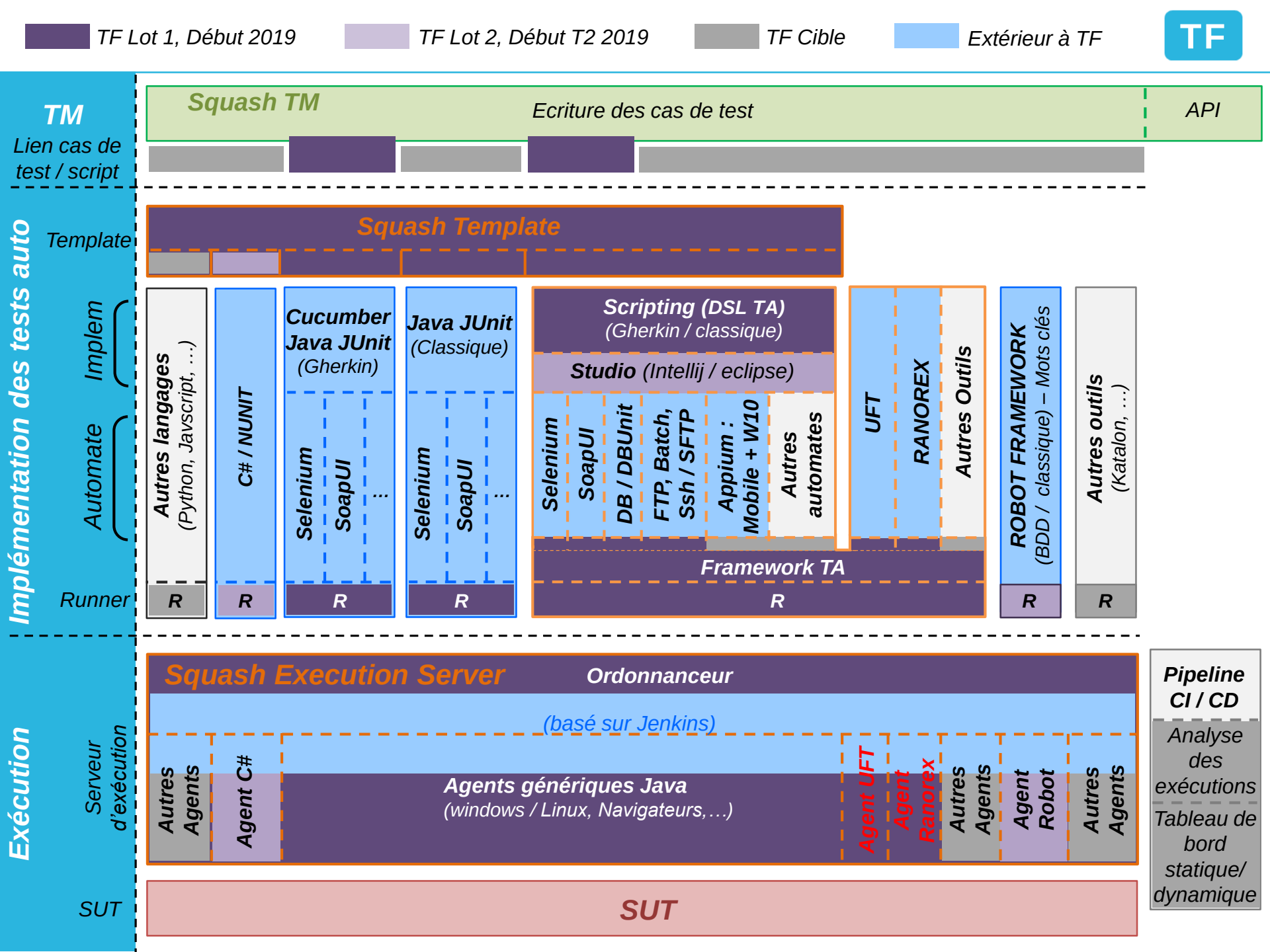
- **Exécution** des cas de tests sur différents types d'environnement (Squash Execution Server) avec **Intégration** avec les pipelines de CI / CD
  - Windows / Linux
  - Différents navigateurs / versions de navigateur
  - …


TF Lot 1, Début 2019
TF Lot 2, Début T2 2019
TF Cible
Extérieur à TF
TF
TM
Lien cas de test / script
Squash TM
Ecriture des cas de test
API
Implémentation des tests auto
Template
Implem
Automate
Runner
Squash Template
Autres langages (Python, Javscript, ...)
C# / NUNIT
Cucumber Java JUnit (Gherkin)
Selenium
SoapUI
...
Java JUnit (Classique)
Scripting (DSL TA) (Gherkin / classique)
Studio (Intellij / eclipse)
Selenium
SoapUI
DB / DBUnit
FTP, Batch, Ssh / SFTP
Appium : Mobile + W10
Autres automates
Framework TA
UFT
RANOREX
Autres Outils
ROBOT FRAMEWORK (BDD / classique) – Mots clés
Autres outils (Katalon, ...)
R
Exécution
Serveur d'exécution
SUT
Squash Execution Server
Ordonnanceur
(basé sur Jenkins)
Autres Agents
Agent C#
Agents génériques Java
(windows / Linux, Navigateurs,...)
Agent UFT
Agent Ranorex
Agent Robot
SUT
Pipeline CI / CD
Analyse des exécutions
Tableau de bord statique/ dynamique

Squash Test Factory : **Les composants**

TF

- **Squash Template** (Utilisation facultative en aide au démarrage scripting)
- Des « **Runner** » (Utilisation obligatoire selon le framework) :
  - Un Runner Java (pour les projets Cucumber Java JUnit / Java JUnit)
  - Un Runner pour les projets utilisant le scripting TA, UFT, Ranorex
- Le **scripting TA** et son studio (Utilisation facultative, alternative à un autre framework)
- **Squash Execution Server** (Utilisation obligatoire avec Installation possible à partir d'un jenkins existant (hors packaging)) comprenant :
  - un ordonnanceur (packaging docker / tar.gz) avec avec intégration pipeline CI/CD
  - des agents d'exécutions fonctions de l'implémentation

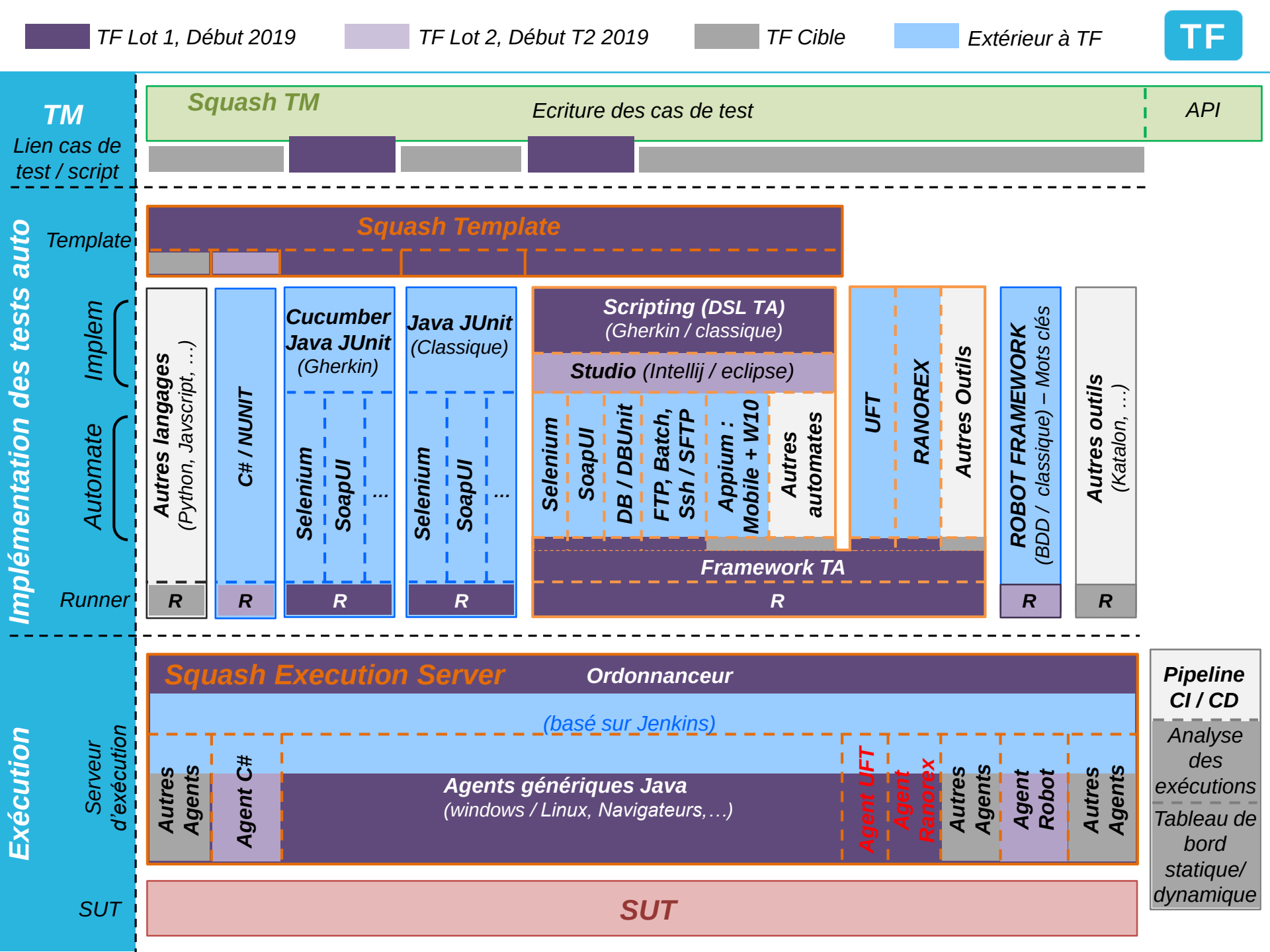

TF Lot 1, Début 2019
TF Lot 2, Début T2 2019
TF Cible
Extérieur à TF
TF
TM
Lien cas de test / script
Squash TM
Ecriture des cas de test
API
Implémentation des tests auto
Template
Implem
Automate
Runner
Squash Template
Autres langages (Python, Javscript, ...)
C# / NUNIT
Cucumber Java JUnit (Gherkin)
Selenium
SoapUI
...
Java JUnit (Classique)
Scripting (DSL TA) (Gherkin / classique)
Studio (Intellij / eclipse)
Selenium
SoapUI
DB / DBUnit
FTP, Batch, Ssh / SFTP
Appium : Mobile + W10
Autres automates
Framework TA
UFT
RANOREX
Autres Outils
ROBOT FRAMEWORK (BDD / classique) – Mots clés
Autres outils (Katalon, ...)
R
Exécution
Serveur d'exécution
SUT
Squash Execution Server
Ordonnanceur
(basé sur Jenkins)
Autres Agents
Agent C#
Agents génériques Java
(windows / Linux, Navigateurs,...)
Agent UFT
Agent Ranorex
Agent Robot
SUT
Pipeline CI / CD
Analyse des exécutions
Tableau de bord statique/ dynamique

Squash Template

- Des templates de projet **Maven Java JUnit** pour démarrer le projet d'automatisation (incluant le runner) :
  - Java JUnit
  - Java JUnit Selenium
  - Java JUnit SoapUI
  - Java JUnit Selenium SoapUI

- Des templates de projet **Maven Cucumber Java JUnit** pour démarrer le projet d'automatisation Gherkin (incluant le runner) :
  - Cucumber Java JUnit
  - Cucumber Java JUnit Selenium
  - Cucumber Java JUnit SoapUI
  - Cucumber Java JUnit Selenium SoapUI

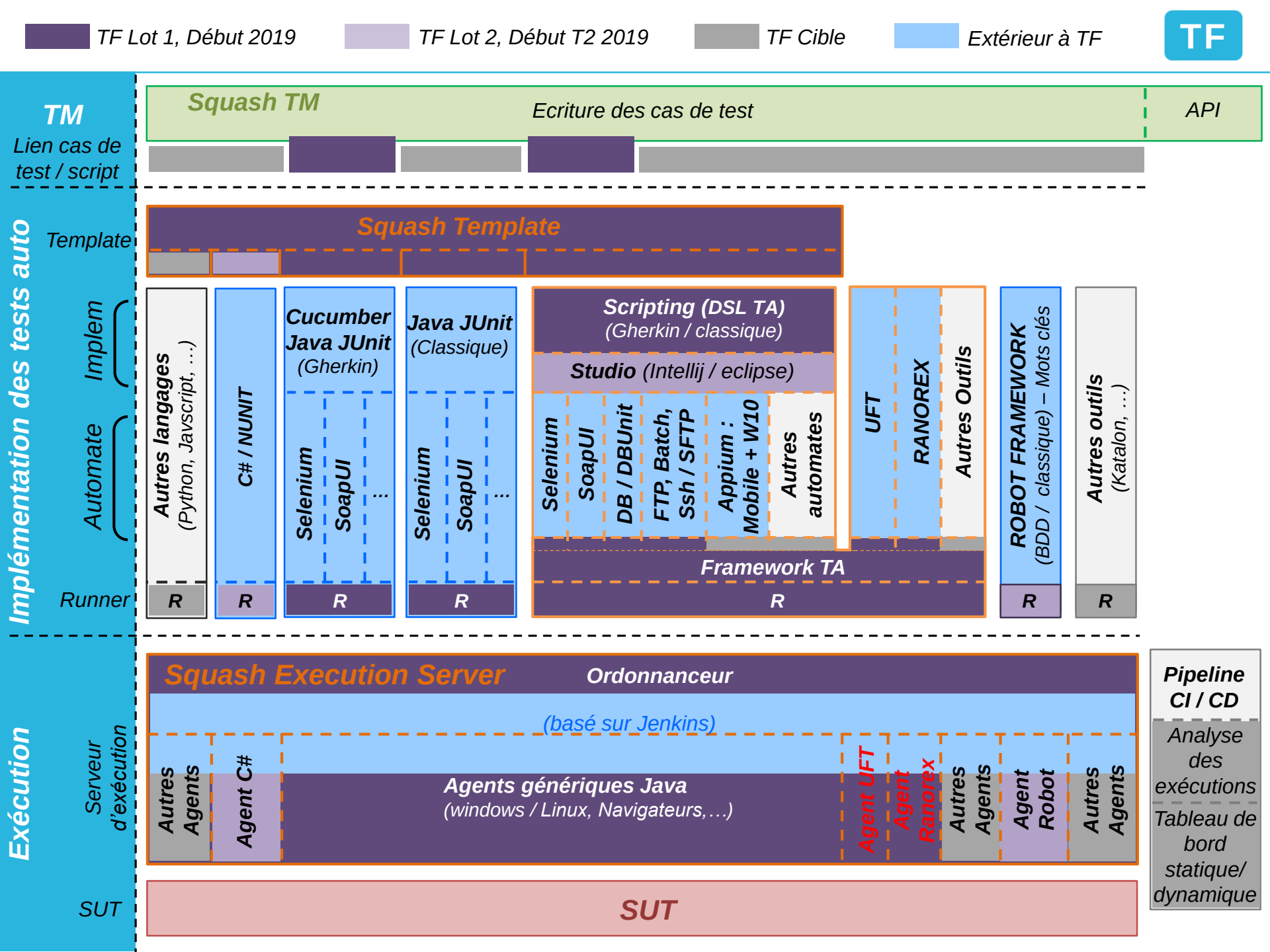
- Template de projet **Scripting TA**


TF Lot 1, Début 2019
TF Lot 2, Début T2 2019
TF Cible
Extérieur à TF
TF
TM
Lien cas de test / script
Squash TM
Ecriture des cas de test
API
Implémentation des tests auto
Template
Implem
Automate
Runner
Squash Template
Autres langages (Python, Javscript, ...)
C# / NUNIT
Cucumber Java JUnit (Gherkin)
Selenium
SoapUI
...
Java JUnit (Classique)
Scripting (DSL TA) (Gherkin / classique)
Studio (Intellij / eclipse)
Selenium
SoapUI
DB / DBUnit
FTP, Batch, Ssh / SFTP
Appium : Mobile + W10
Autres automates
Framework TA
UFT
RANOREX
Autres Outils
ROBOT FRAMEWORK (BDD / classique) – Mots clés
Autres outils (Katalon, ...)
R
Exécution
Serveur d'exécution
SUT
Squash Execution Server
Ordonnanceur
(basé sur Jenkins)
Autres Agents
Agent C#
Agents génériques Java
(windows / Linux, Navigateurs,...)
Agent UFT
Agent Ranorex
Agent Robot
SUT
Pipeline CI / CD
Analyse des exécutions
Tableau de bord statique/ dynamique

Runner

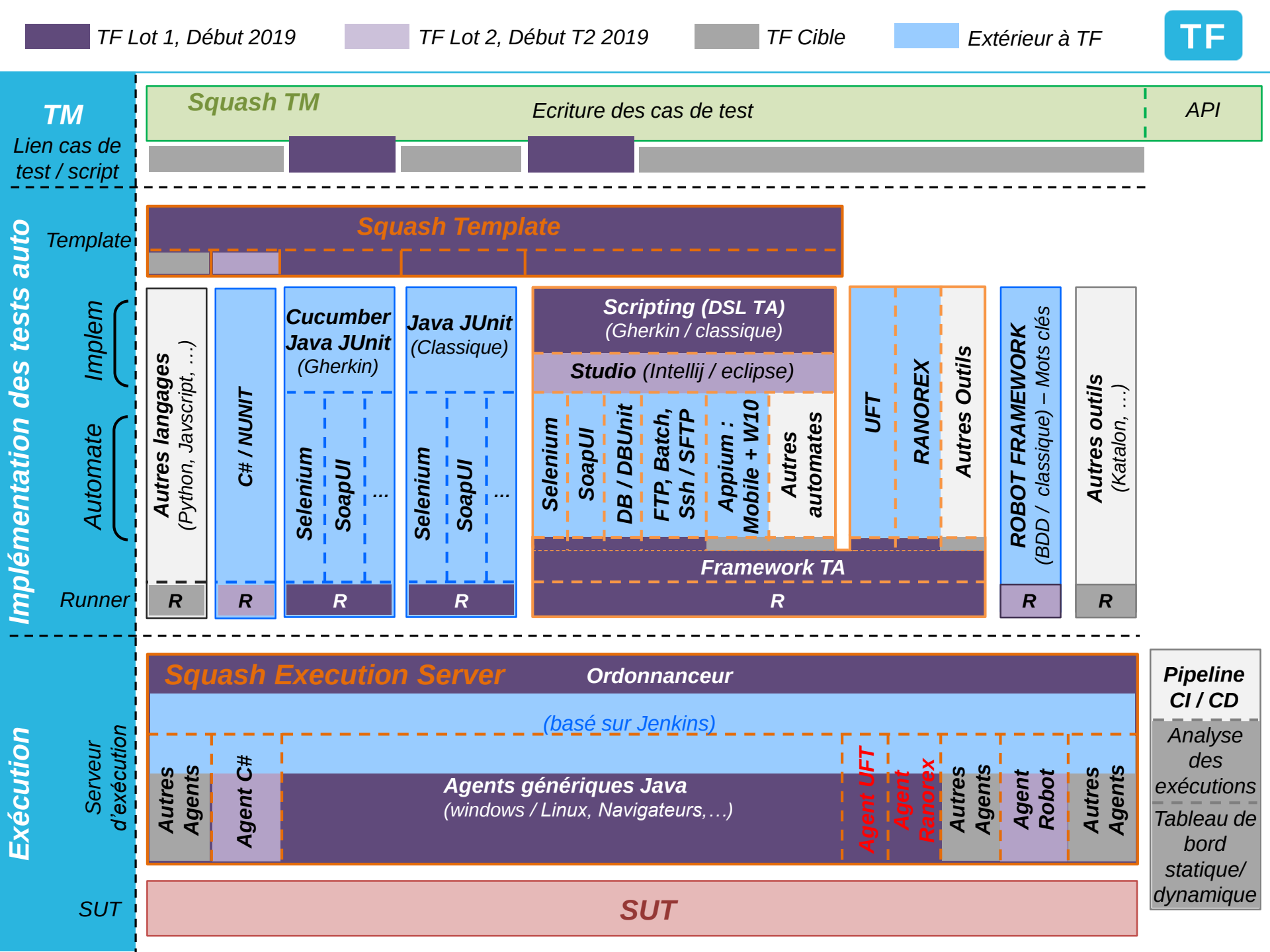
TF

- Les « runner » assurent la **connectivité** entre Squash TM et l'exécution des tests automatisés. Ils doivent :
  - Gérer la suite de test à exécuter (filtre / json du paramètre « ta.test.suite »)
  - Lancer l'exécution
  - Gérer la génération de rapport
  - Remonter les statuts d'exécutions dans Squash TM
  - Remonter les rapports d'exécutions dans Squash TM
  - Gérer les liens automatiques entre les cas de test de Squash TM (Gherkin / classique) et les tests automatisés
- Les « runner » peuvent se présenter sous la forme :
  - D'un runner complet
  - D'un add-on à un runner existant
- Dans le cadre des implémentations « Dev » le runner est le plus **transparent** possible.


TF Lot 1, Début 2019
TF Lot 2, Début T2 2019
TF Cible
Extérieur à TF
TF
TM
Lien cas de test / script
Squash TM
Ecriture des cas de test
API
Implémentation des tests auto
Template
Implem
Automate
Runner
Squash Template
Autres langages (Python, Javscript, ...)
C# / NUNIT
Cucumber Java JUnit (Gherkin)
Selenium
SoapUI
...
Java JUnit (Classique)
Scripting (DSL TA) (Gherkin / classique)
Studio (Intellij / eclipse)
Selenium
SoapUI
DB / DBUnit
FTP, Batch, Ssh / SFTP
Appium : Mobile + W10
Autres automates
Framework TA
UFT
RANOREX
Autres Outils
ROBOT FRAMEWORK (BDD / classique) – Mots clés
Autres outils (Katalon, ...)
R
Exécution
Serveur d'exécution
SUT
Squash Execution Server
Ordonnanceur
(basé sur Jenkins)
Autres Agents
Agent C#
Agents génériques Java
(windows / Linux, Navigateurs,...)
Agent UFT
Agent Ranorex
Agent Robot
SUT
Pipeline CI / CD
Analyse des exécutions
Tableau de bord statique/ dynamique

Scripting TA

- Un **DSL** (Domain Specific Language) orienté mots clés qui comprend :
  - Un **catalogue de mot clés** (macro paramétrable), **extensible** si besoin, mettant à disposition les fonctionnalités de base du framework tout en masquant la tuyauterie bas niveau
  - L'implémentation des tests automatisés par **composition** des mots clés
- Un **framework** qui fait le lien entre le DSL et les automates
- Un **studio** basé sur IntelliJ / Eclipse permettant de faire :
  - de l'auto-complétion et
  - de la coloration syntaxique


TF Lot 1, Début 2019
TF Lot 2, Début T2 2019
TF Cible
Extérieur à TF
TF
TM
Lien cas de test / script
Squash TM
Ecriture des cas de test
API
Implémentation des tests auto
Template
Implem
Automate
Runner
Squash Template
Autres langages (Python, Javscript, ...)
C# / NUNIT
Cucumber Java JUnit (Gherkin)
Selenium
SoapUI
...
Java JUnit (Classique)
Scripting (DSL TA) (Gherkin / classique)
Studio (Intellij / eclipse)
Selenium
SoapUI
DB / DBUnit
FTP, Batch, Ssh / SFTP
Appium : Mobile + W10
Autres automates
Framework TA
UFT
RANOREX
Autres Outils
ROBOT FRAMEWORK (BDD / classique) – Mots clés
Autres outils (Katalon, ...)
R
Exécution
Serveur d'exécution
SUT
Squash Execution Server
Ordonnanceur
(basé sur Jenkins)
Autres Agents
Agent C#
Agents génériques Java
(windows / Linux, Navigateurs,...)
Agent UFT
Agent Ranorex
Agent Robot
SUT
Pipeline CI / CD
Analyse des exécutions
Tableau de bord statique/ dynamique

Squash Execution Server

- Basé sur **Jenkins** :
  - Connecteurs scm
  - Architecture Maitre / Esclave
  - Pipeline
  - Publication de rapports
  - API Rest
- Déployable via image **docker** :
  - Facilite les déploiements
- Des agents d'exécution par :
  - **Environnement** d'exécution (intégration, recette, …)
  - **Technologie** (Linux / Windows, Chrome / Firefox , …)

PROJET

Squash Test Factory : La communauté

- Selon le même modèle que Squash TM, Squash TF existe en version **Community** (gratuite et open source) et **Commerciale** (même code source avec les agents d'exécution spécifiques UFT et Ranorex).

- Le projet est **piloté par Henix**, qui a sous sa responsabilité les dépôts de source, les releases et la définition de la roadmap.

- Les **contributions** et extensions sont les bienvenues :
  - Contributions : enrichissement/amélioration sur le périmètre supporté par Henix
    - Validation technique et fonctionnelle par le Projet
  - Extensions : maintenance supportée par le propriétaire
    - Référencement par le projet (marketplace)

- Ou puis-je trouver du support ?
  - via la communauté sur le forum
  - via Henix sur la version commerciale (contour de l'offre en cours de définition)

Squash Test Factory : Roadmap

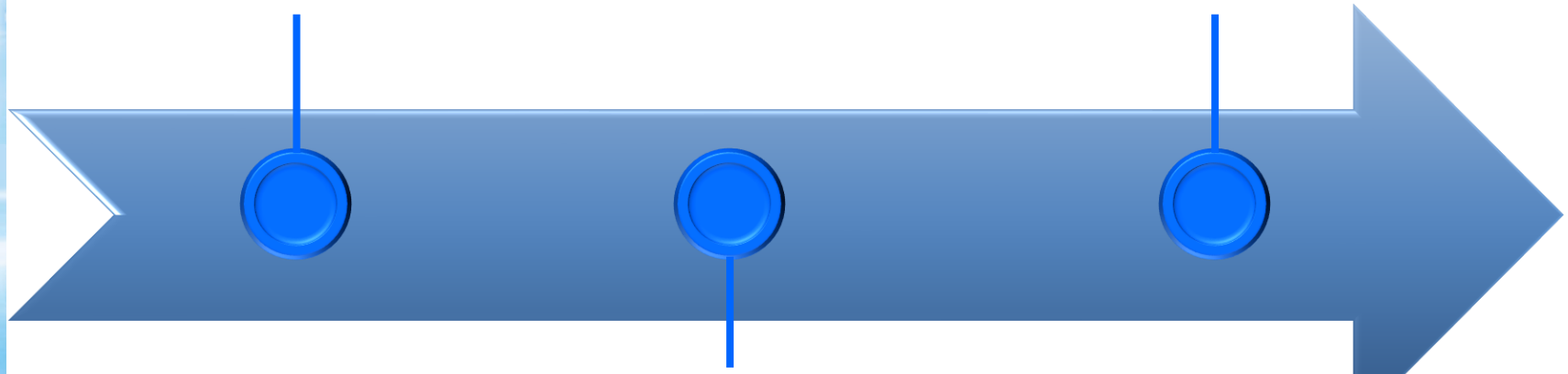
TF

Début 2019

- *Le runner Java*
- *Les templates de projet Java*
- *Squash Exécution Server (docker / tar.gz)*
- *Réorientation mots clés de la documentation du scripting TA*
- *Workflow testeurs – automaticiens (avec TM)*

Ultérieurement

- *Intégration Robot framework*
- *Support d'autres langages (Runner, Templates)*
- *Intégration plus fine de selenium dans le framework TA*
- *Offrir une partie des fonctionnalité TA sous forme de librairie aux projets java*
- *Liaison automatique CT TM – Test auto*

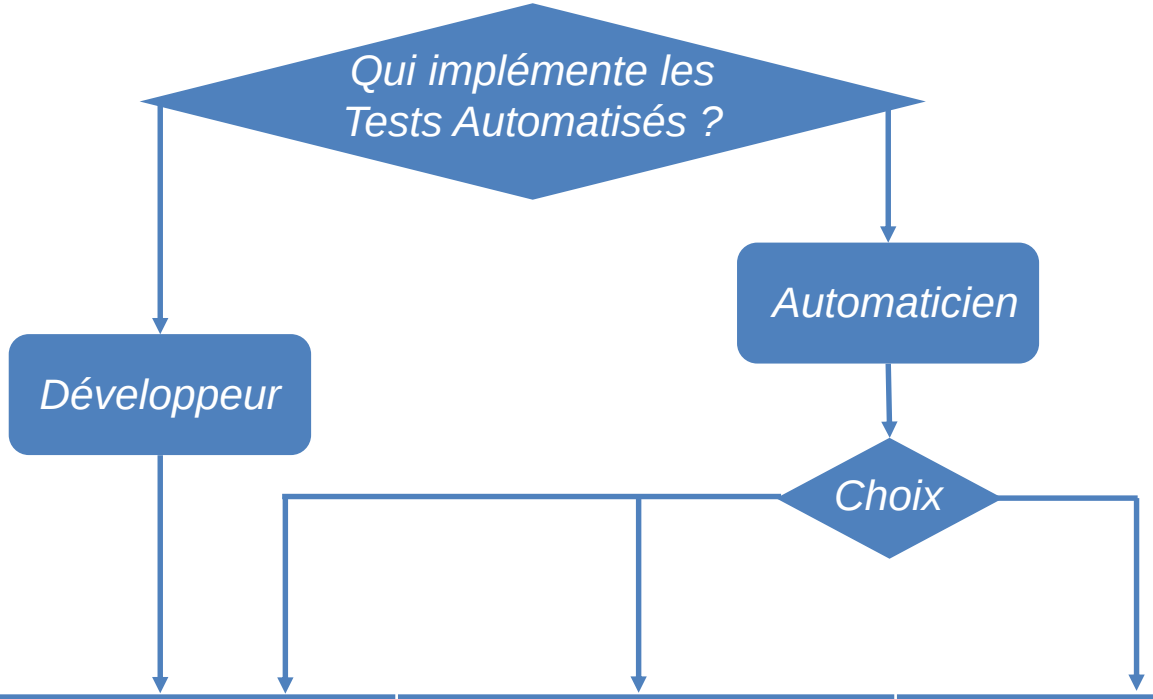


Début T2 2019

- *Runner & template pour un nouveau langage (C# Nunit ?)*
- *Studio IntelliJ pour le scripting TA*
- *Framework TA*
  - *Intégration d'appium*
  - *Support JUnit5*
  - *Réorientation mots clés, étape 2*
    - *Evolution du DSL (périmètre à définir)*

EXEMPLES D'UTILISATION

Impact de l'organisation sur le choix du framework



	Environnement Dev	Studio TA	Autre framework
Gherkin	Cucumber Java JUnit	DSL TA	Uft, Ranorex, Robot framework, …
Classique	Java JUnit	DSL TA	

Squash Test Factory : Exemple d'utilisation n°1

TF

- **Description**
  - Une équipe agile qui utilise gherkin pour écrire ses cas de test
  - Les développeurs du SUT (Environnement Java) implémentent les tests automatisés
- **Avantages**
  - Utilisation du langage naturel pour l'écriture des cas de test (Gherkin)
  - Workflow d'automatisation testeurs – automaticiens (Squash TM)
  - Environnement intégré pour l'exécution (Squash Execution Server)
  - Grande liberté d'écriture des tests automatisés sans surcouche (Projet type « Dev »)
  - Capitalise sur le langage habituel du développeur (Projet type « Dev »)
- **Points de vigilance**
  - Tout est fait dans le projet Java
    - Maintenabilité et robustesse des tests automatisés à gérer
    - Fonctionnalités de gestion des données à développer (Productivité ?)

CT Gherkin : workflow testeurs - automaticiens

TF

Espace Cas de Test (TM)

Ecrit les CT en Gherkin :

- CT #51
- CT #52
- CT #53

Testeur

Espace Automatisation (TM)

Suit l'avancement

Espace Exécution (TM)

GIT

Poste de Travail

Automaticien

Espace Automatisation (TM)

Orchestrateur (TF)

Agent (TF)

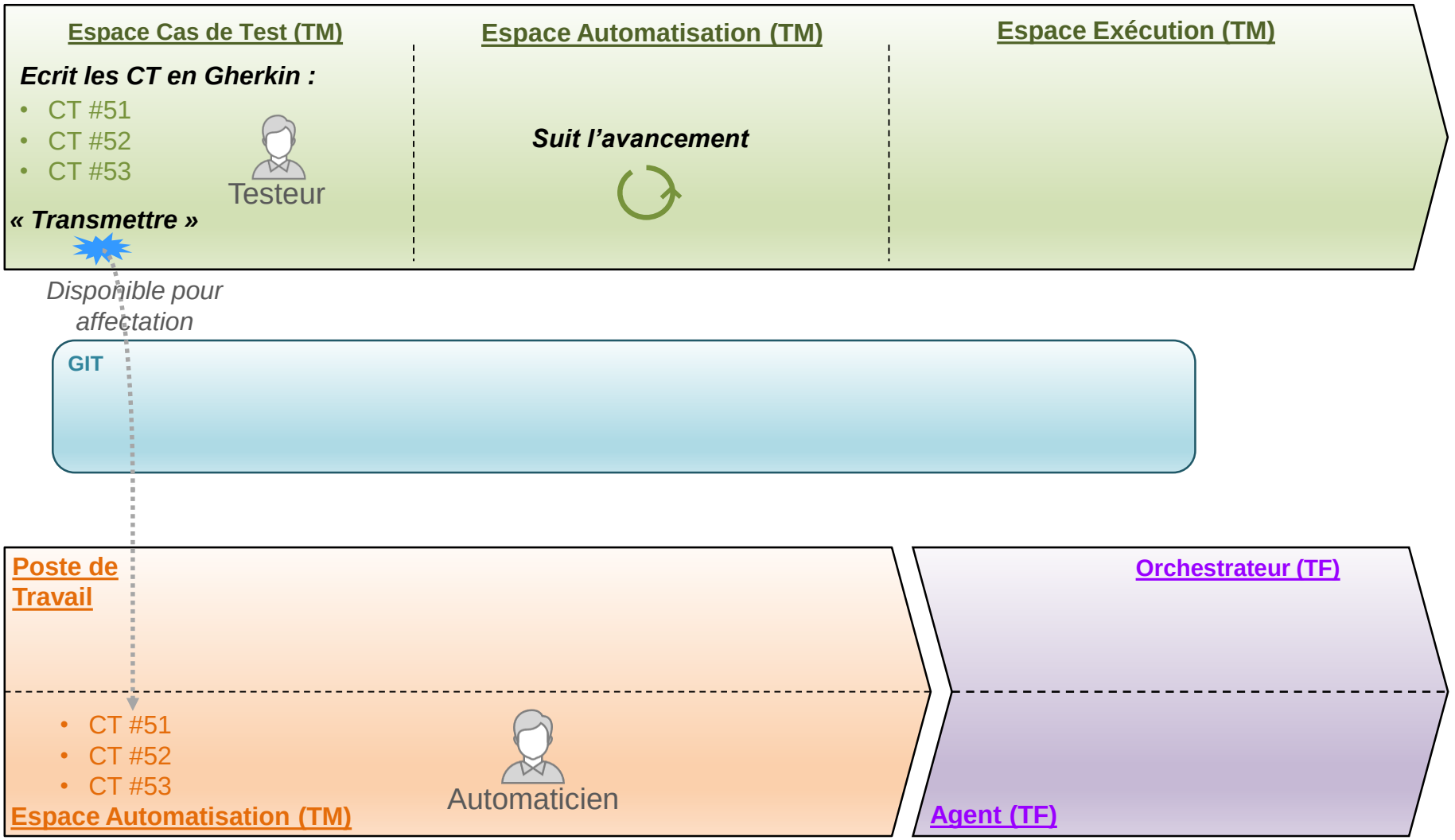
.feature (gherkin)

.java

SUT

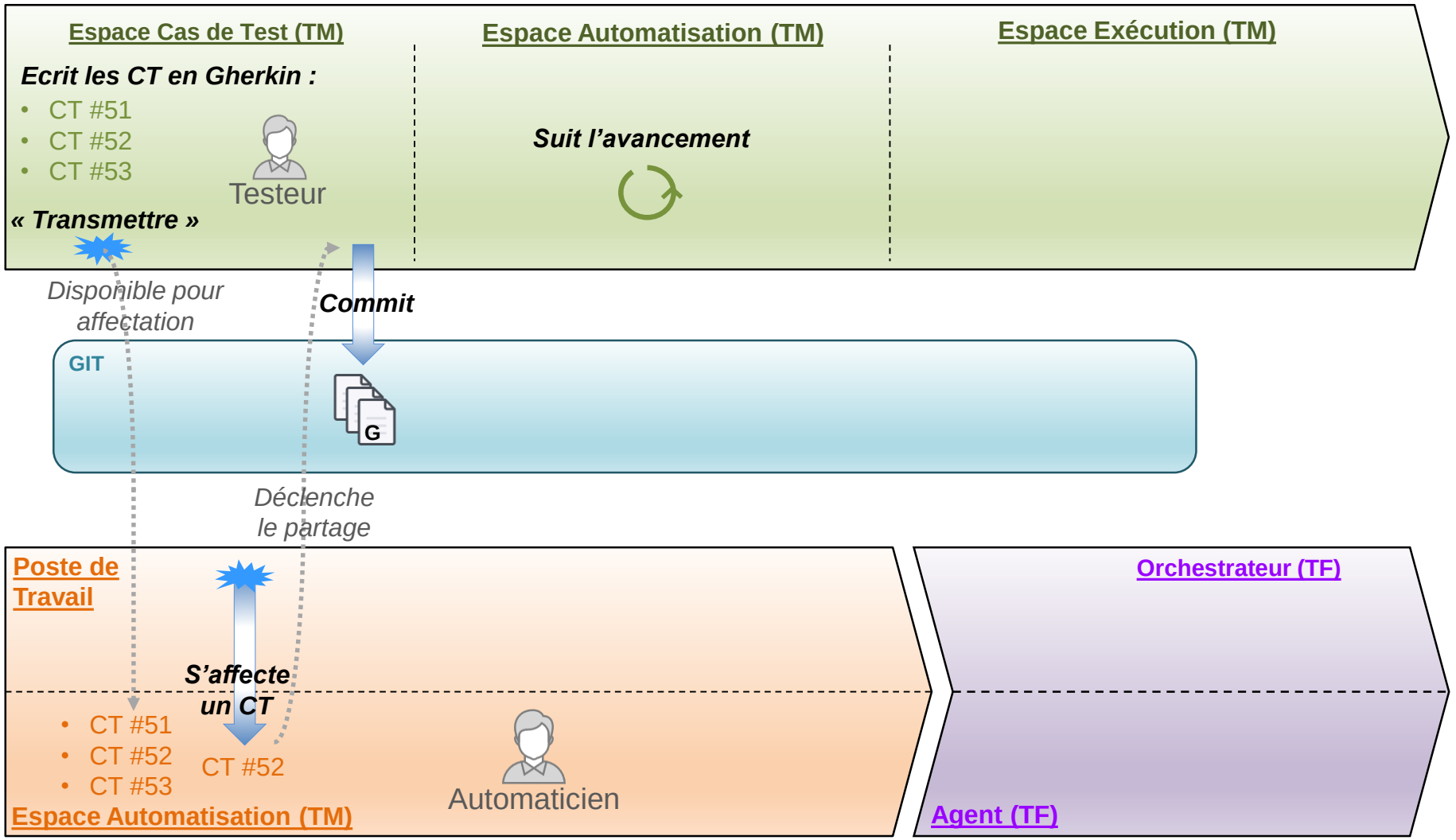
CT Gherkin : workflow testeurs - automaticiens

TF



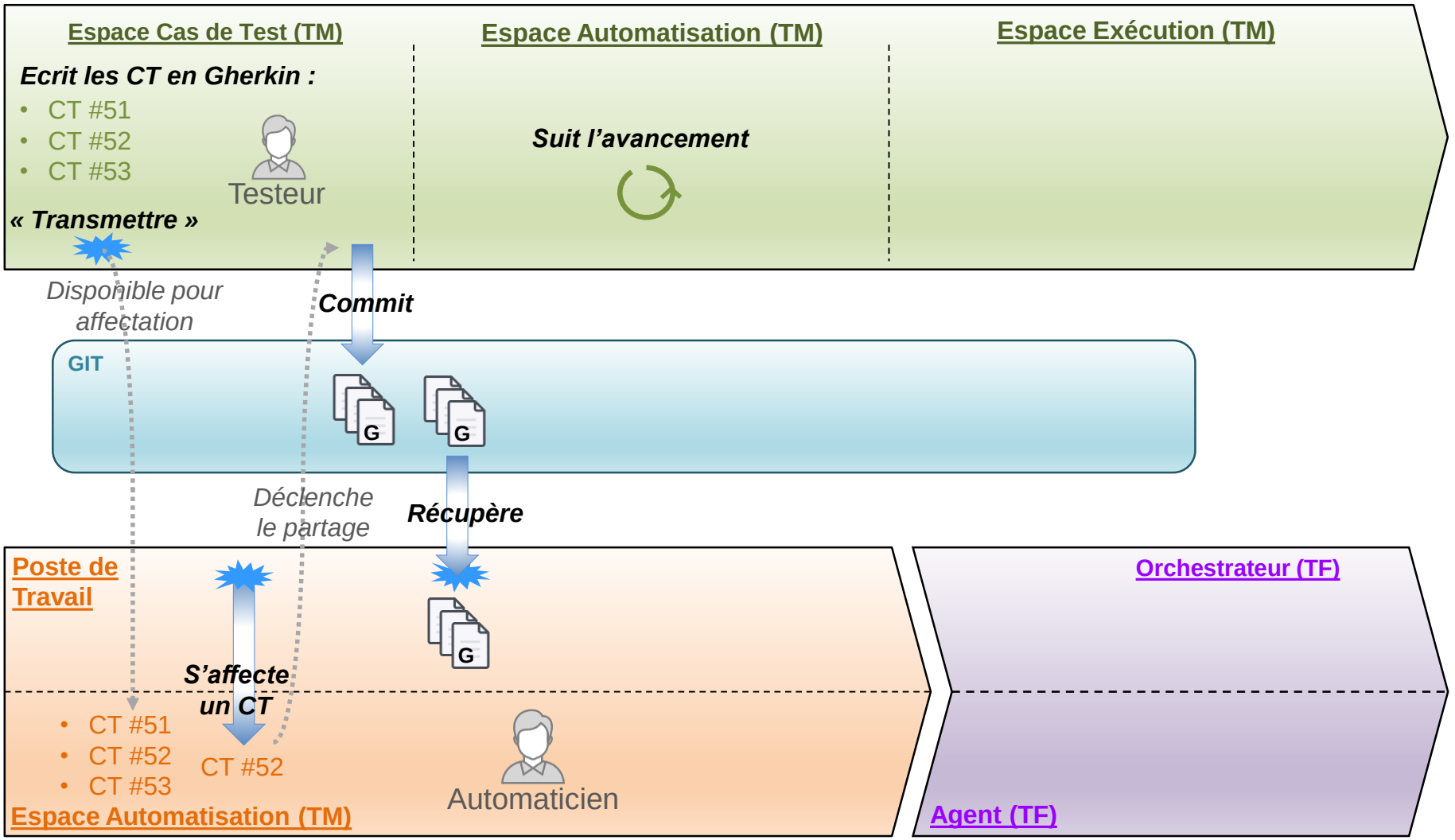
CT Gherkin : workflow testeurs - automaticiens

TF



CT Gherkin : workflow testeurs - automaticiens

TF



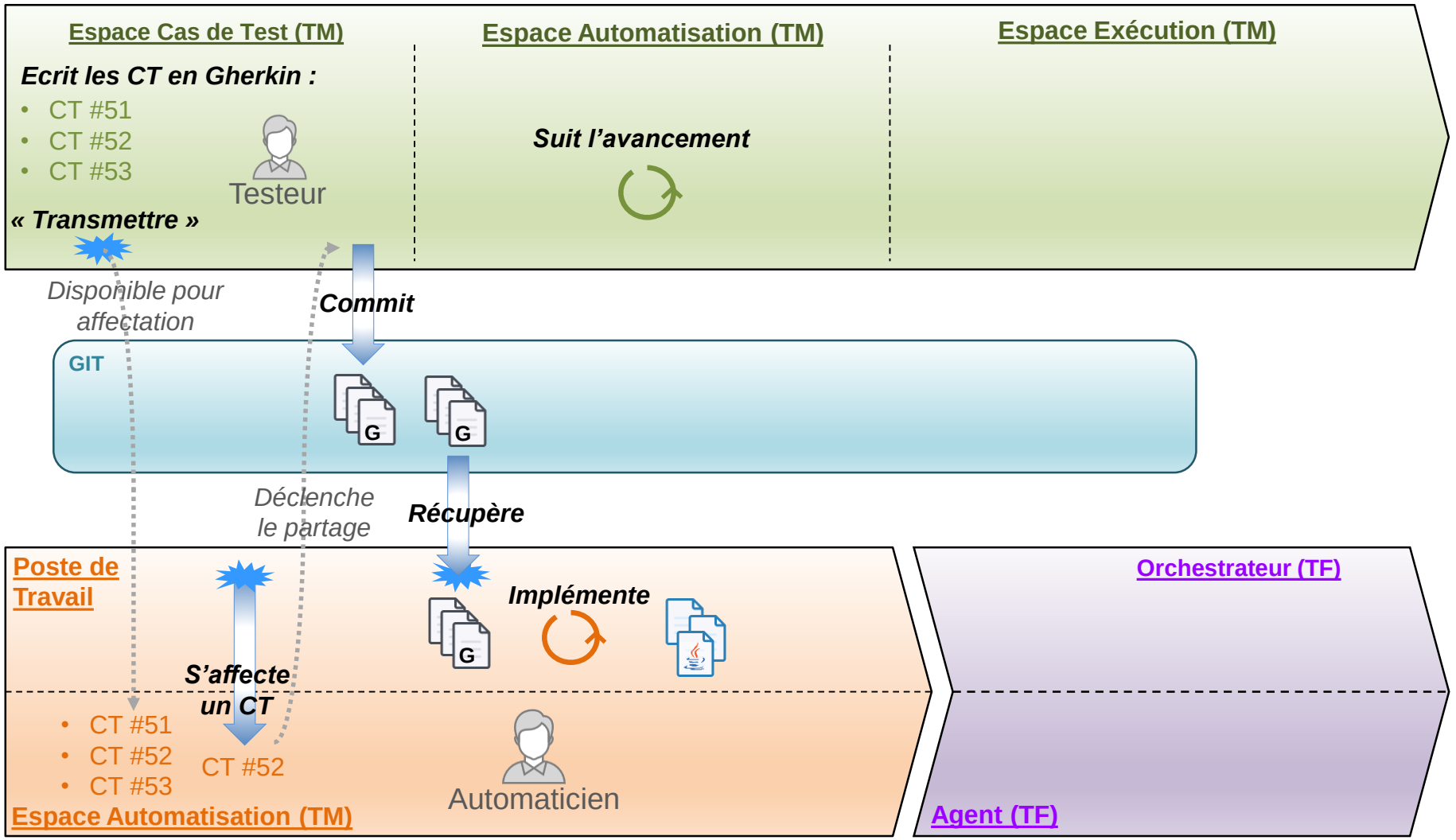
G .feature (gherkin)

.java

SUT

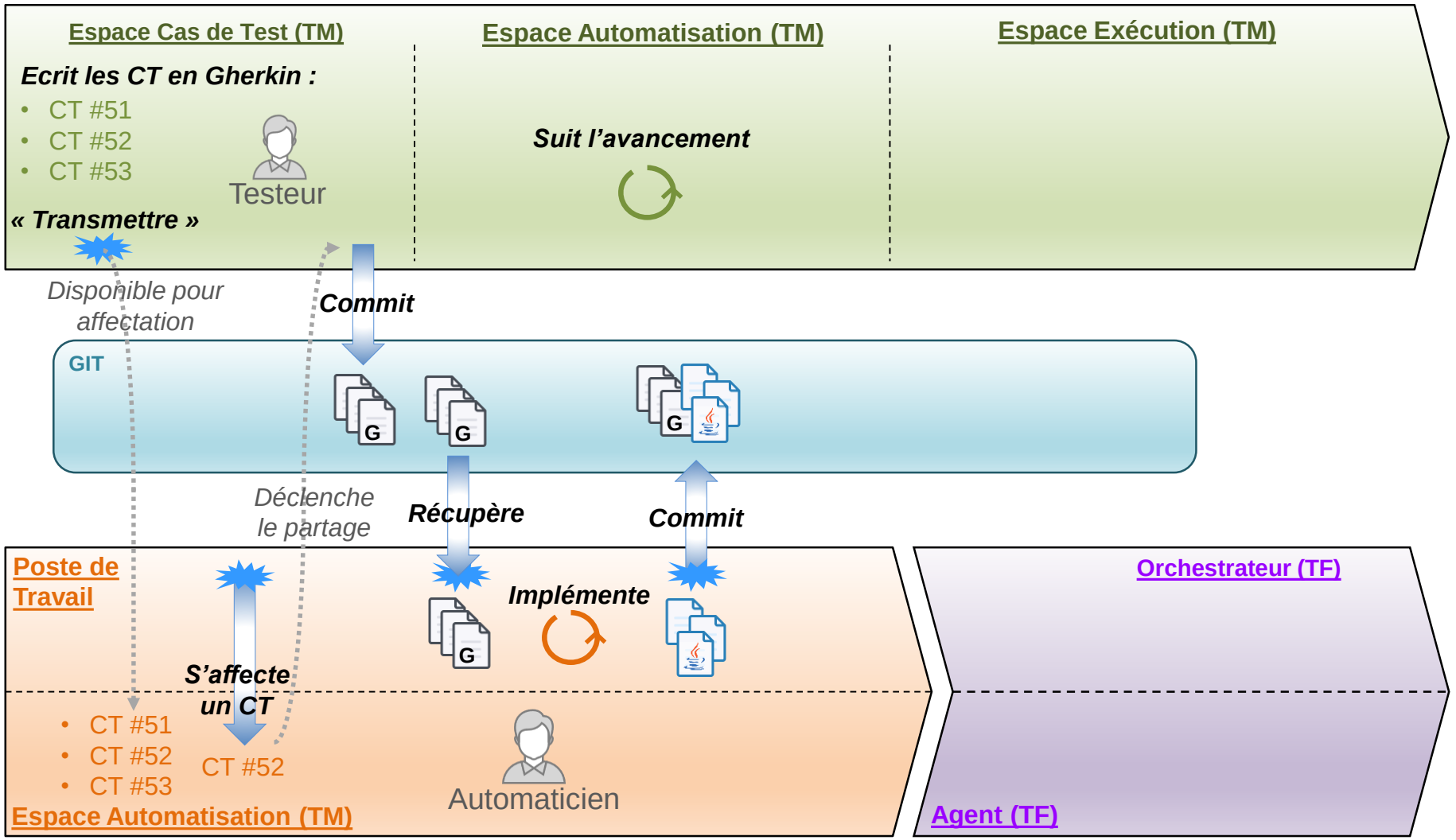
CT Gherkin : workflow testeurs - automaticiens

TF

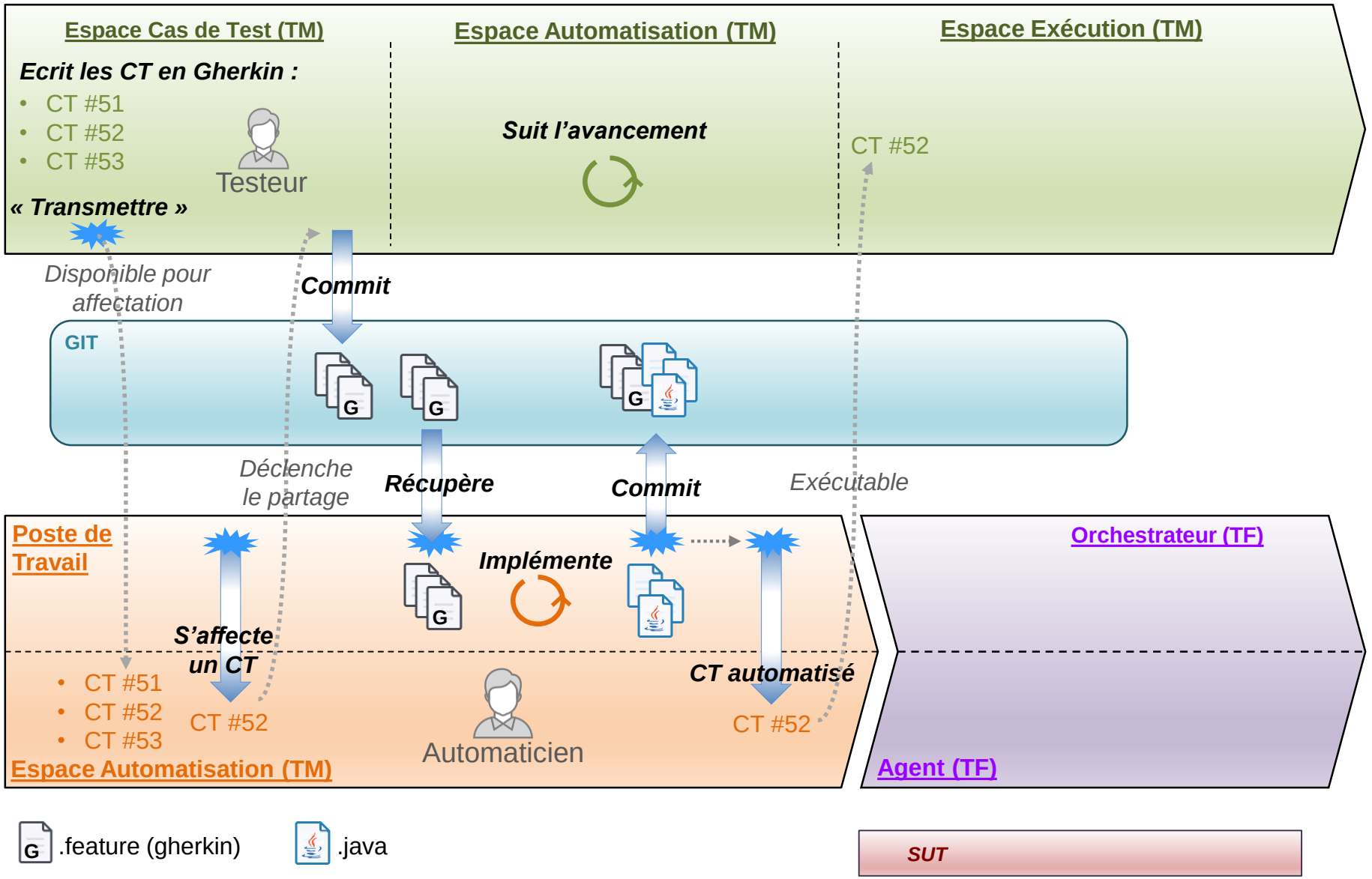


CT Gherkin : workflow testeurs - automaticiens

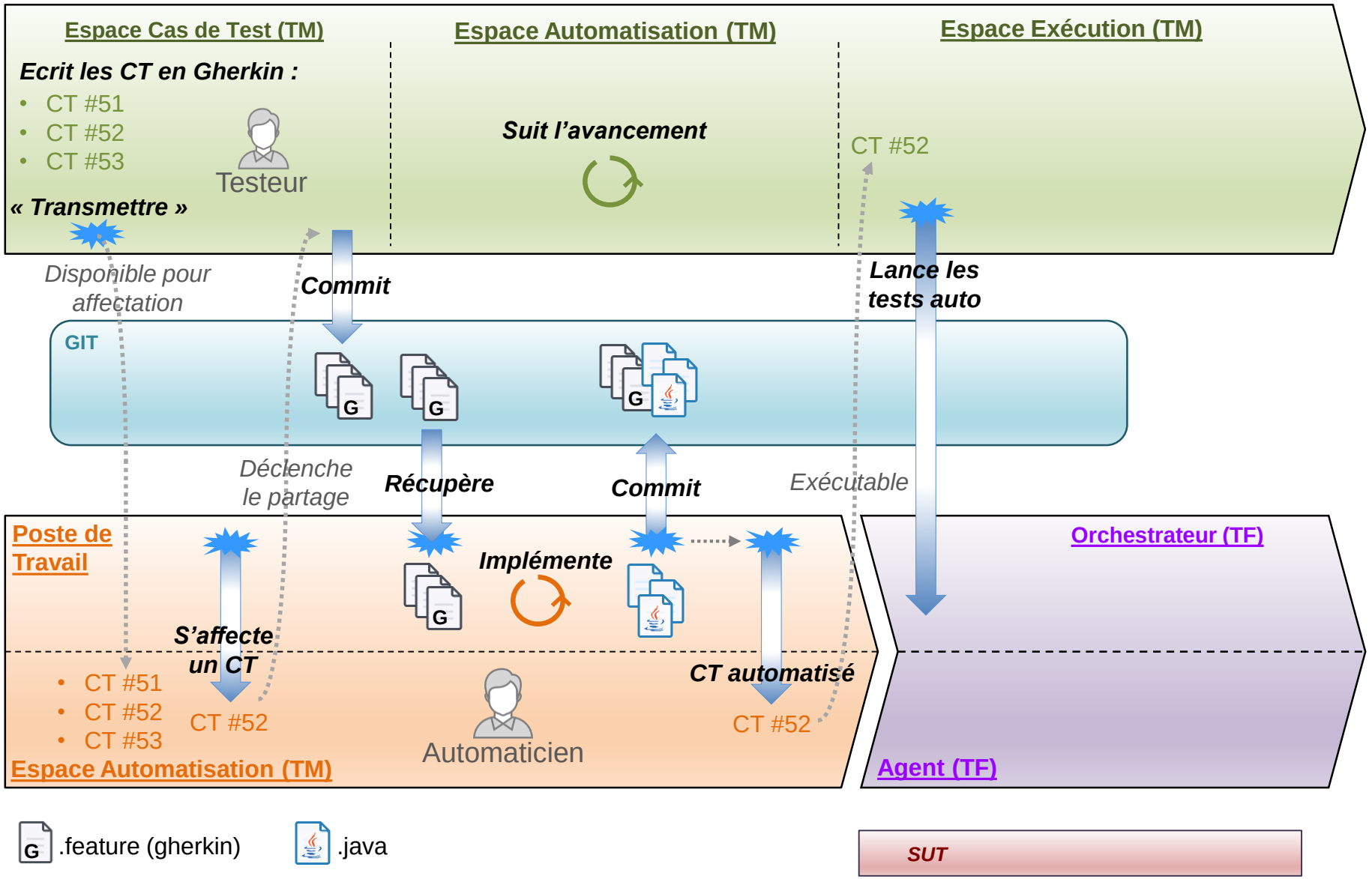
TF



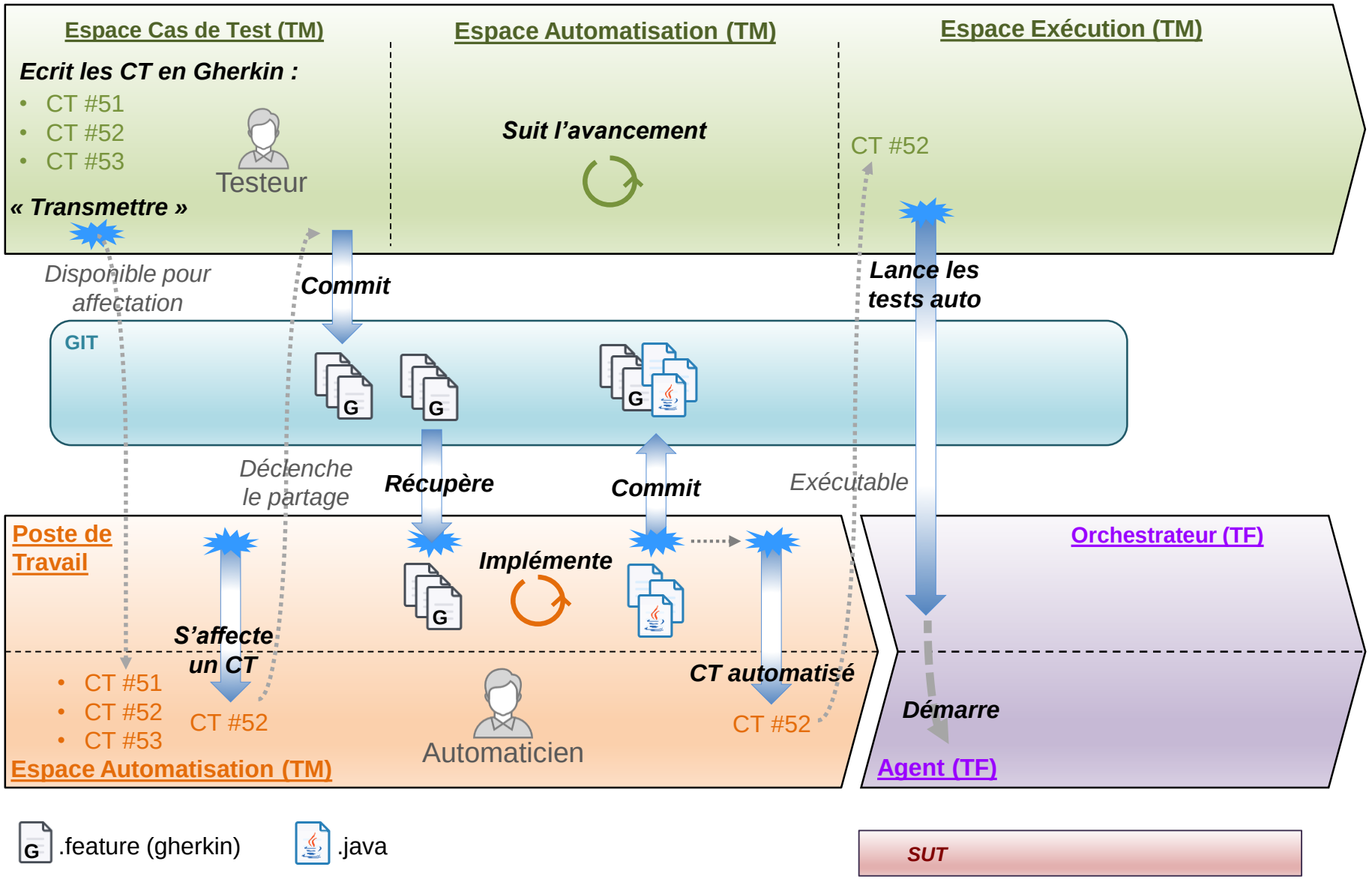
CT Gherkin : workflow testeurs - automaticiens



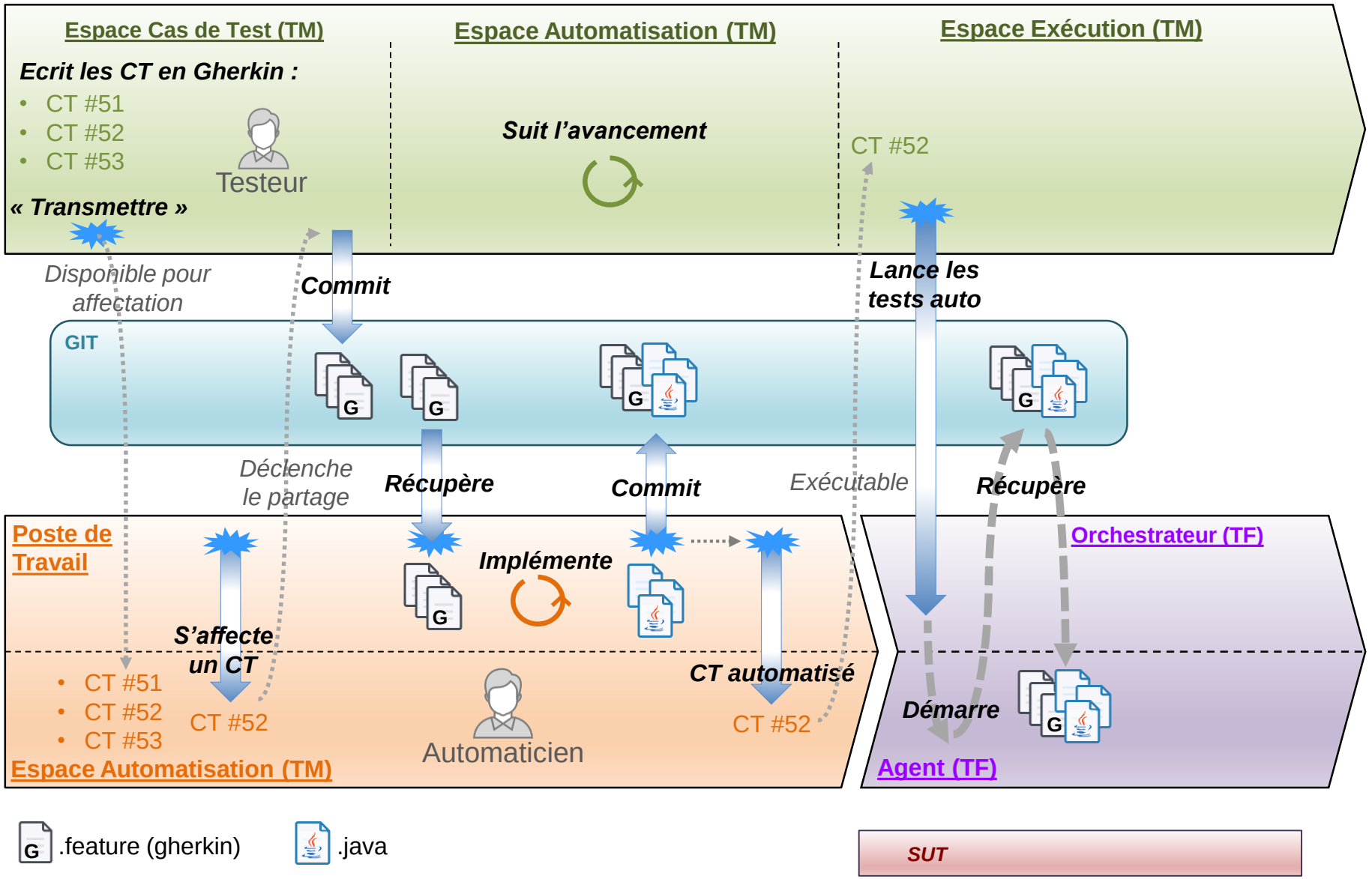
CT Gherkin : workflow testeurs - automaticiens



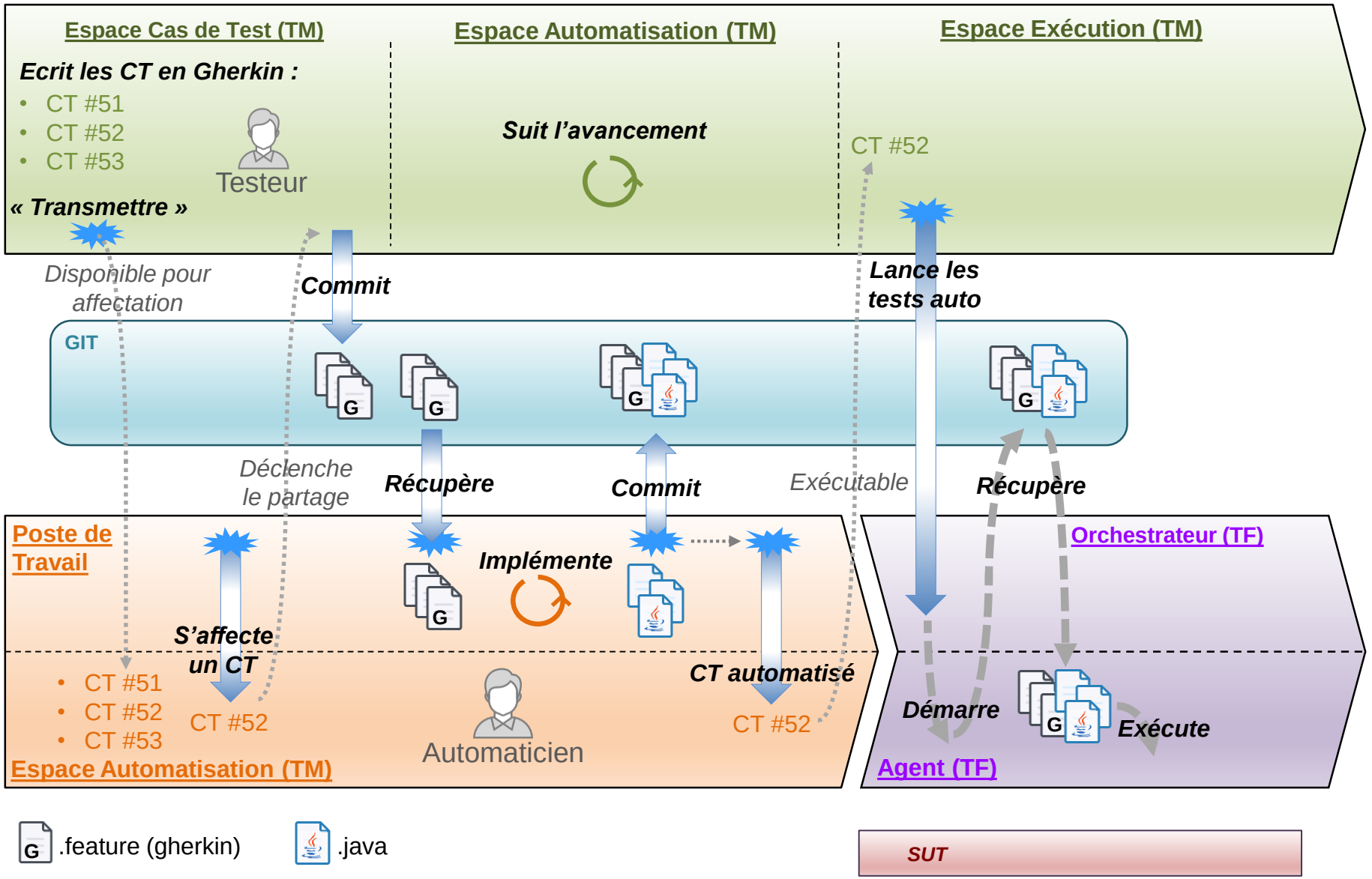
CT Gherkin : workflow testeurs - automaticiens



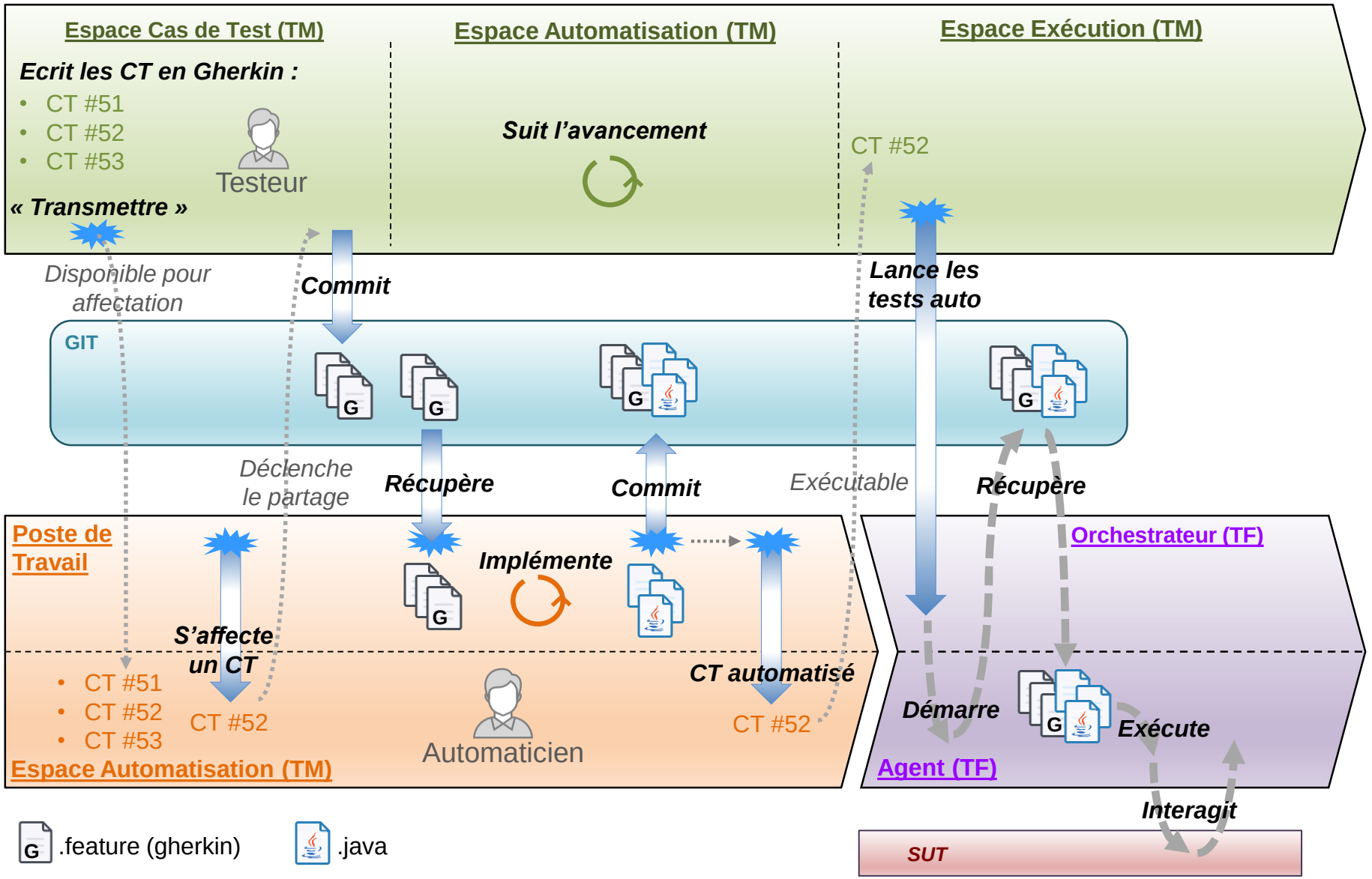
CT Gherkin : workflow testeurs - automaticiens



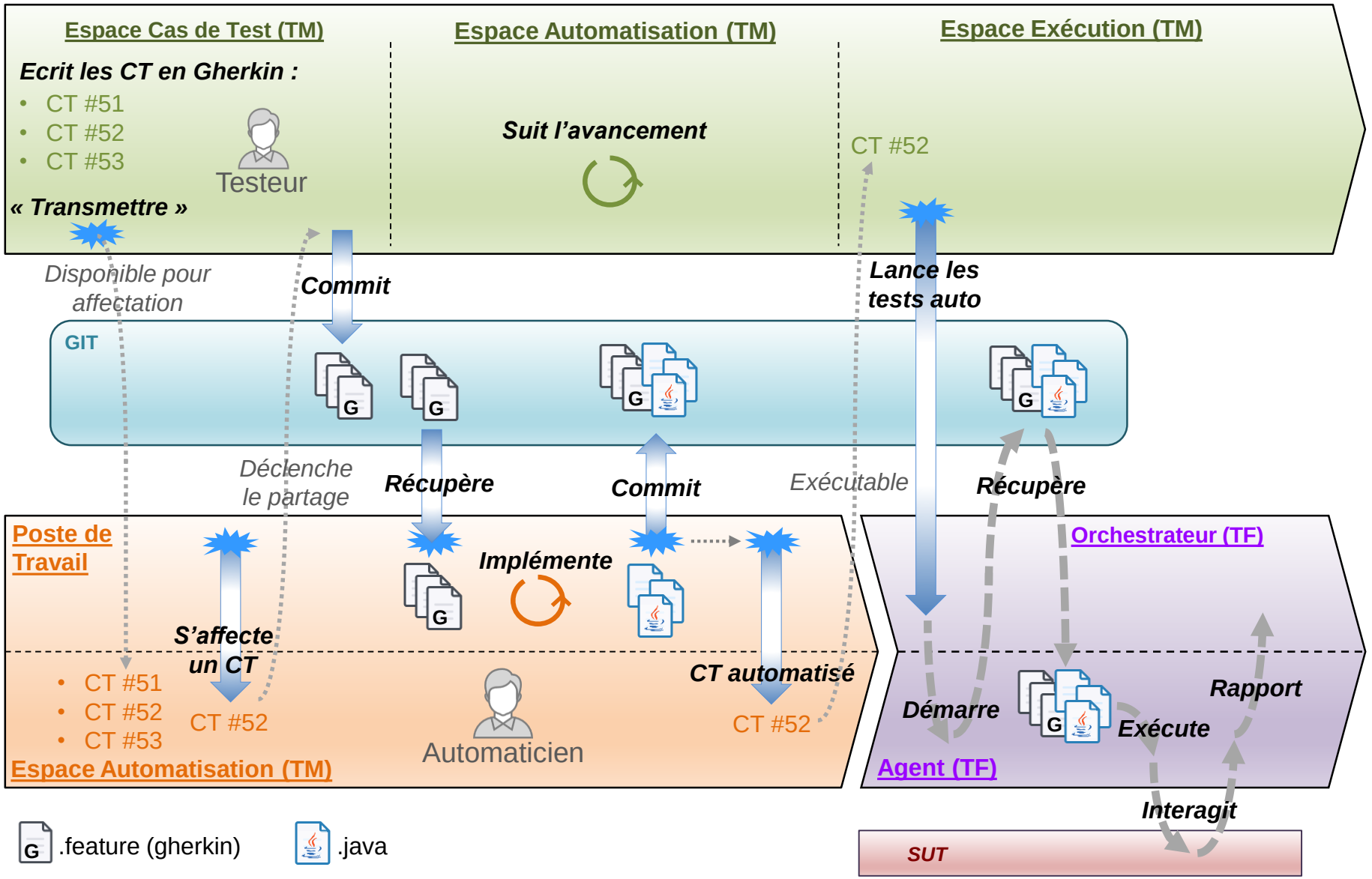
CT Gherkin : workflow testeurs - automaticiens



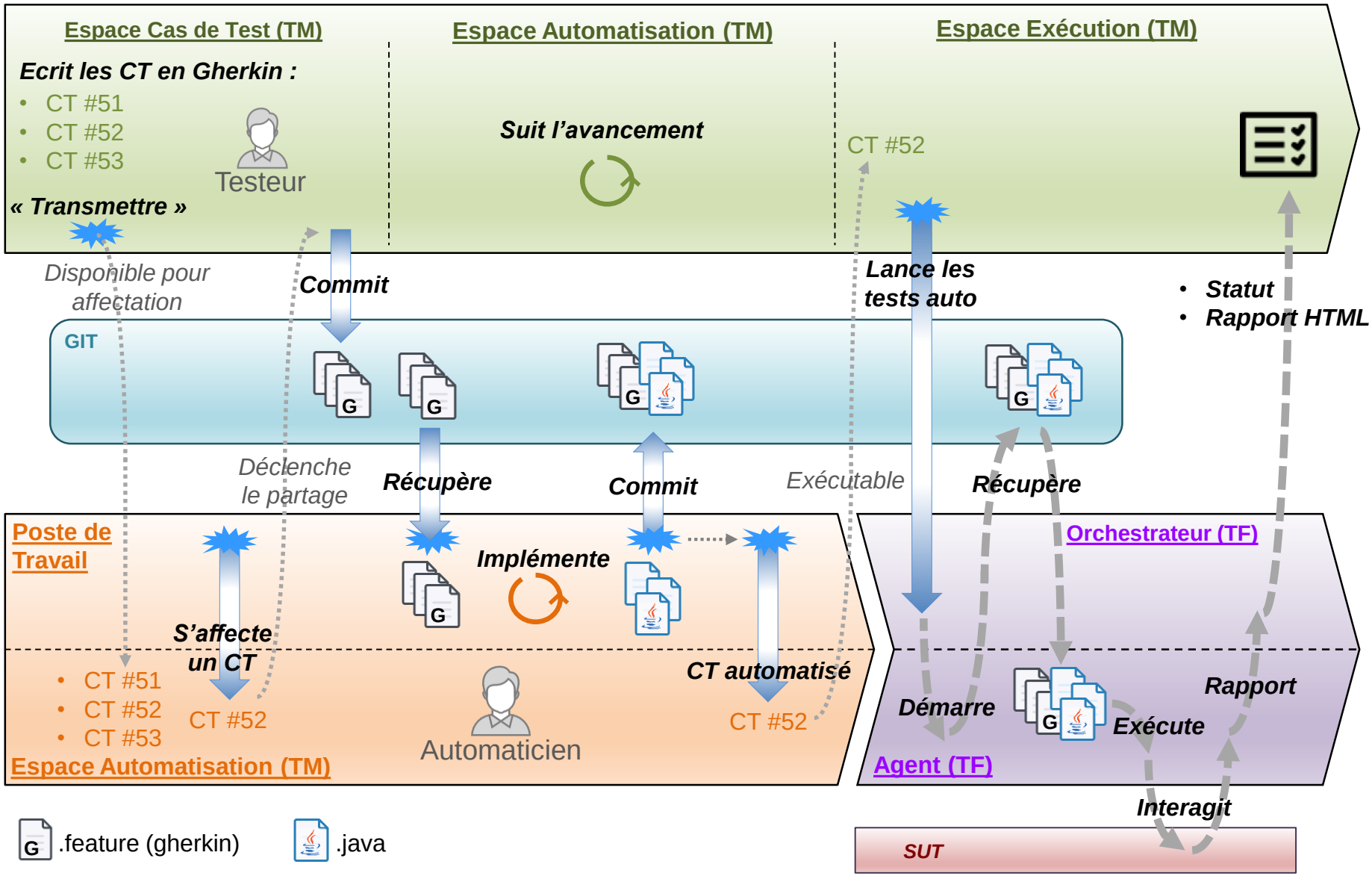
CT Gherkin : workflow testeurs - automaticiens



CT Gherkin : workflow testeurs - automaticiens



CT Gherkin : workflow testeurs - automaticiens



Squash Test Factory : Exemple d'utilisation n°2

TF

- **Description**
  - Une cellule de test TNR qui écrit des cas de test de manière classique
  - Les tests automatisés sont implémentés par des automaticiens avec le scripting TA
  - Eventuellement des scripts existants (UFT par exemple)
- **Avantages**
  - Worflow d'automatisation testeurs – automaticiens (Squash TM)
  - Environnement intégré pour l'exécution (Squash Execution Server)
  - Meilleur robustesse et maintenabilité/productivité des tests automatisés grâce aux capacités du framework (Structuration des tests, fonctionnalités pré câblées, Scripting orienté mot clés, …) (Scripting + framework TA)
  - Permet d'écrire des tests multi automates (Scripting + framework TA) en réutilisant les scripts UFT pour éventuellement les décommissionner au fur et à mesure
- **Pt de vigilance**
  - Contraint par le process et les fonctionnalités du framework TA

Squash Test Factory : Exemple d'utilisation n°3

TF

- **Description**
  - Existence d'un patrimoine de TNR utilisant un outil maison propriétaire
  - Désire avoir la remontée de statut dans Squash TM et utiliser l'environnement d'execution prépackagé

- **Avantages :**
  - Permet de récupérer un patrimoine existant
  - Environnements d'exécution intégrés permettant la remontée d'information dans TM (Squash Execution Server)
  - Worflow testeurs – automaticiens
  - Permet une transition en douceur (si volonté de migration)
  - Exécution de tests hétérogénes (si souhait)

- **Inconvénients :**
  - Intégration du framework à TF (à développer si elle n'existe pas encore)
  - Ajout d'une passerelle intermédiaire supplémentaire