# « Quand Squash (DevOps) teste Squash : Un exemple de Test Factory Open Source* »

Club qualité logicielle 19/11/2020

*Auteur :* *Florian Gautier <fgautier@henix.fr>*

* Majoritairement

# Quand Squash (DevOps) teste Squash : Un exemple de Test Factory Open Source*

1. Le Test en Continu
2. Les « Test Factory »
3. Squash (DevOps) teste Squash : (Notre) implémentation de Test Factory
4. Aperçu de la roadmap de « Squash DevOps »

* Majoritairement



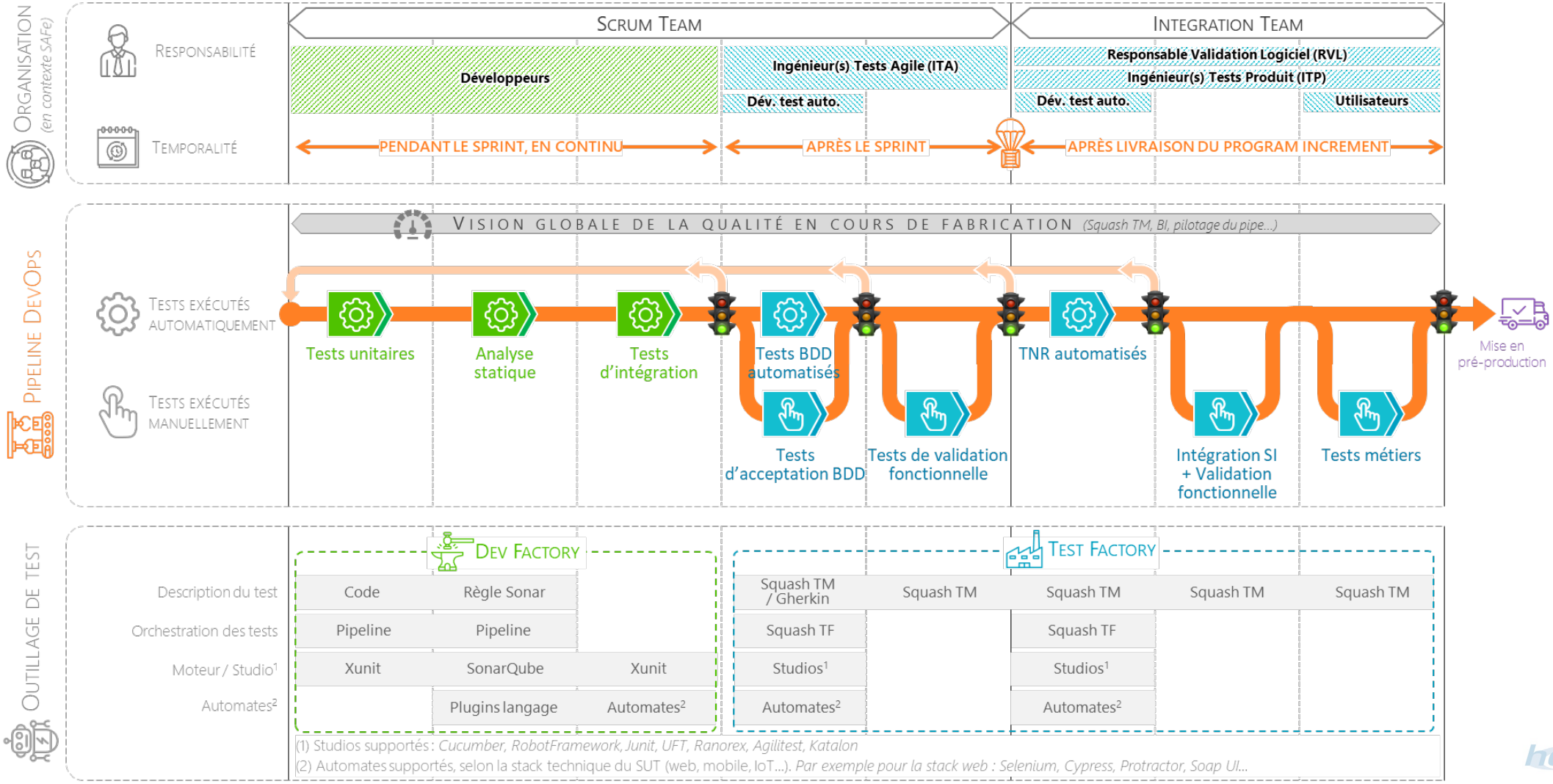
henix

# 1 Le Test en Continu

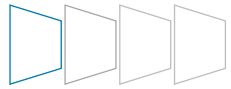
- 1.1 – Objectif
- 1.2 – Le Test en Continu et le développement de l'outillage des pipelines CI/CD
- 1.3 – Le déploiement en continu : Des environnements de test à disposition

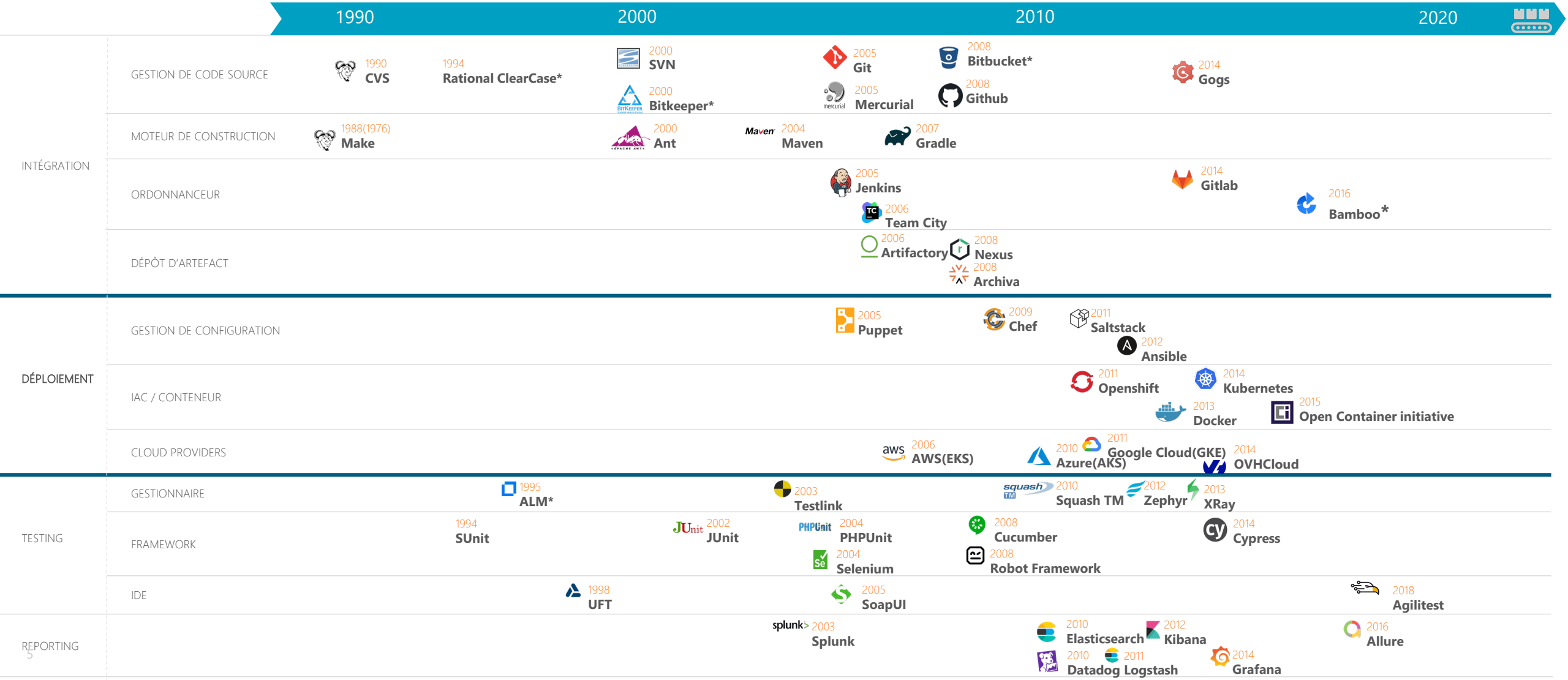
# LE TEST EN CONTINU : (NOTRE) OBJECTIF

**Objectif** : L'intégration fluide des séquences de tests fonctionnels automatisés (ou manuels) via les pipelines d'intégration continue (CI) et de déploiement continu (CD)



# 1.2 Le test en continu et le développement de l'outillage des pipelines CI/CD





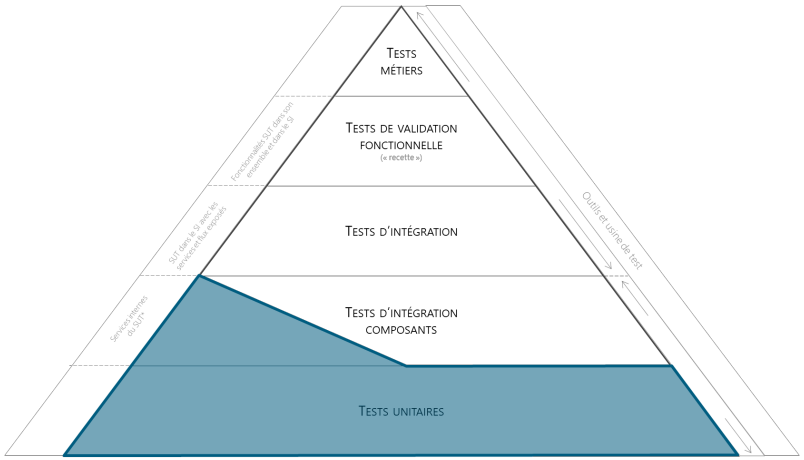
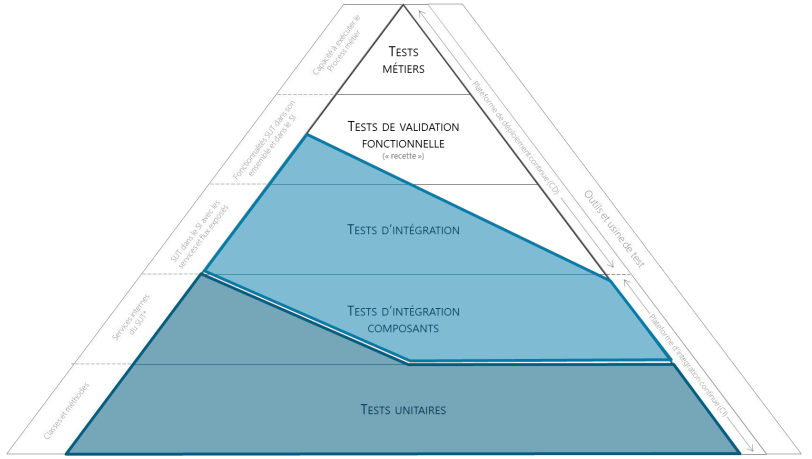

* Liste non exhaustive

** Des erreurs/typos au niveau des dates ont pu se glisser


5

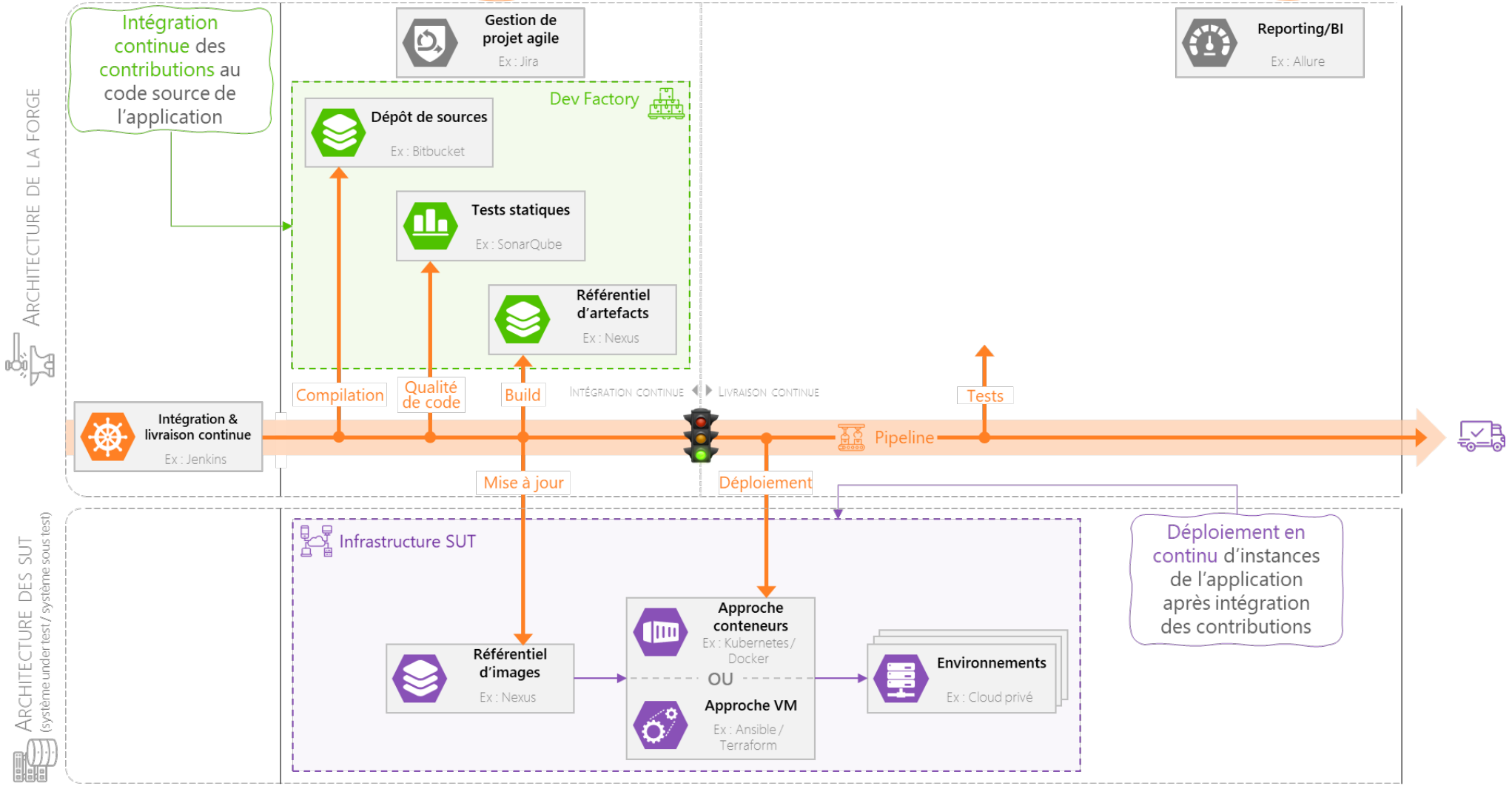
# Le déploiement en continu : Des environnements de test à disposition

Le pipeline de CD met à disposition des environnements disponibles aux testeurs manuels et aux automates de tests (Robot Framework, Sélénium, Soap UI,…)

| | Dans la CI | Après le CD |
|---|---|---|
| Système testé | Le Code Source de l'application sous test | L'application sous test déployée |
| Analyse | Statique* | Dynamique |
| (potentielle) Cible d'automatisation des tests |  |  |

# Le déploiement en continu : Des environnements de test à disposition

Au-delà des CI, Avec l'émergence des CD, le pipeline met à disposition des environnements disponibles aux testeurs manuels et aux automates de tests (Robot Framework, Sélénium, Soap UI,...)

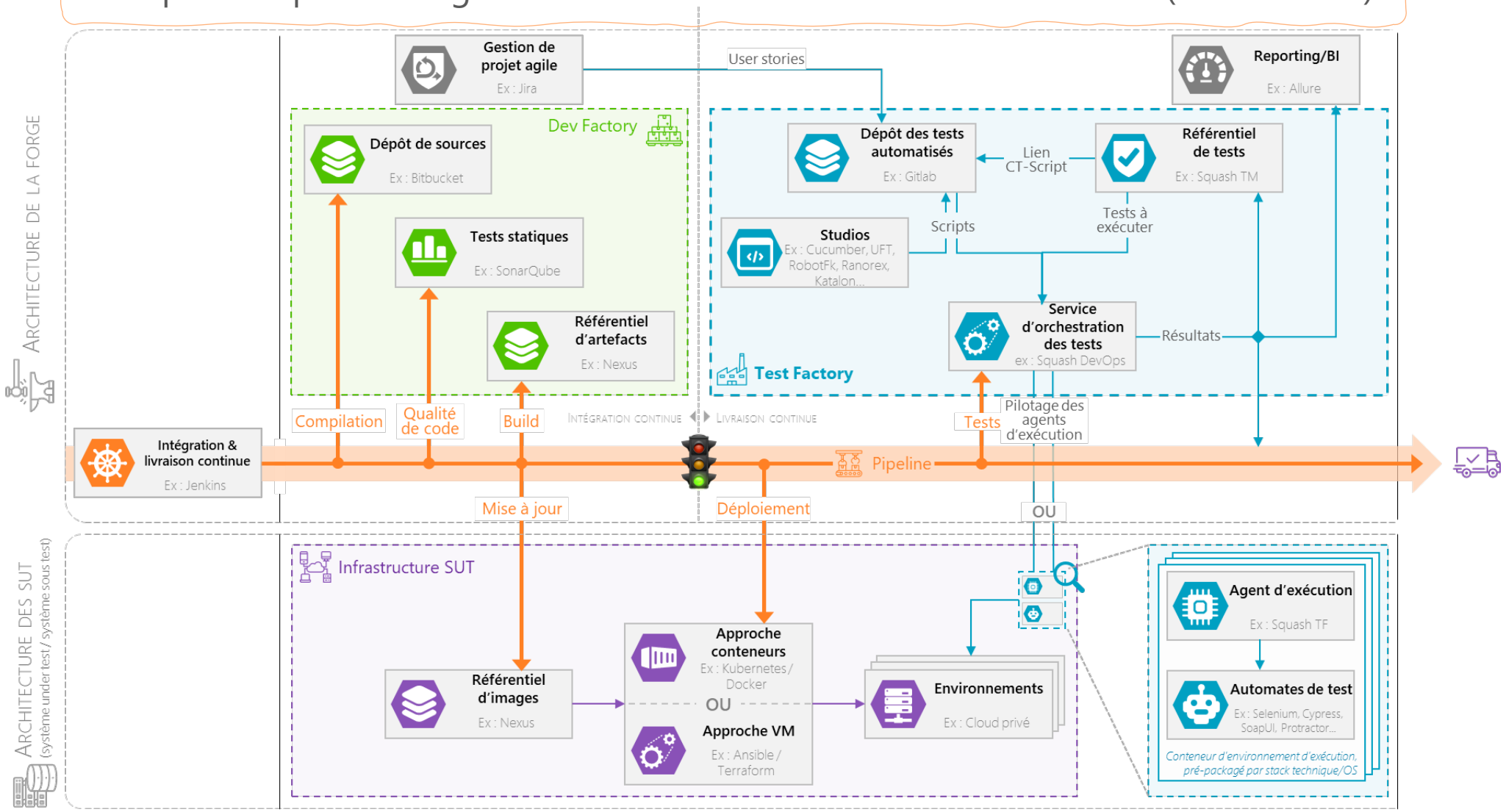


# 2 LES «TEST FACTORY »

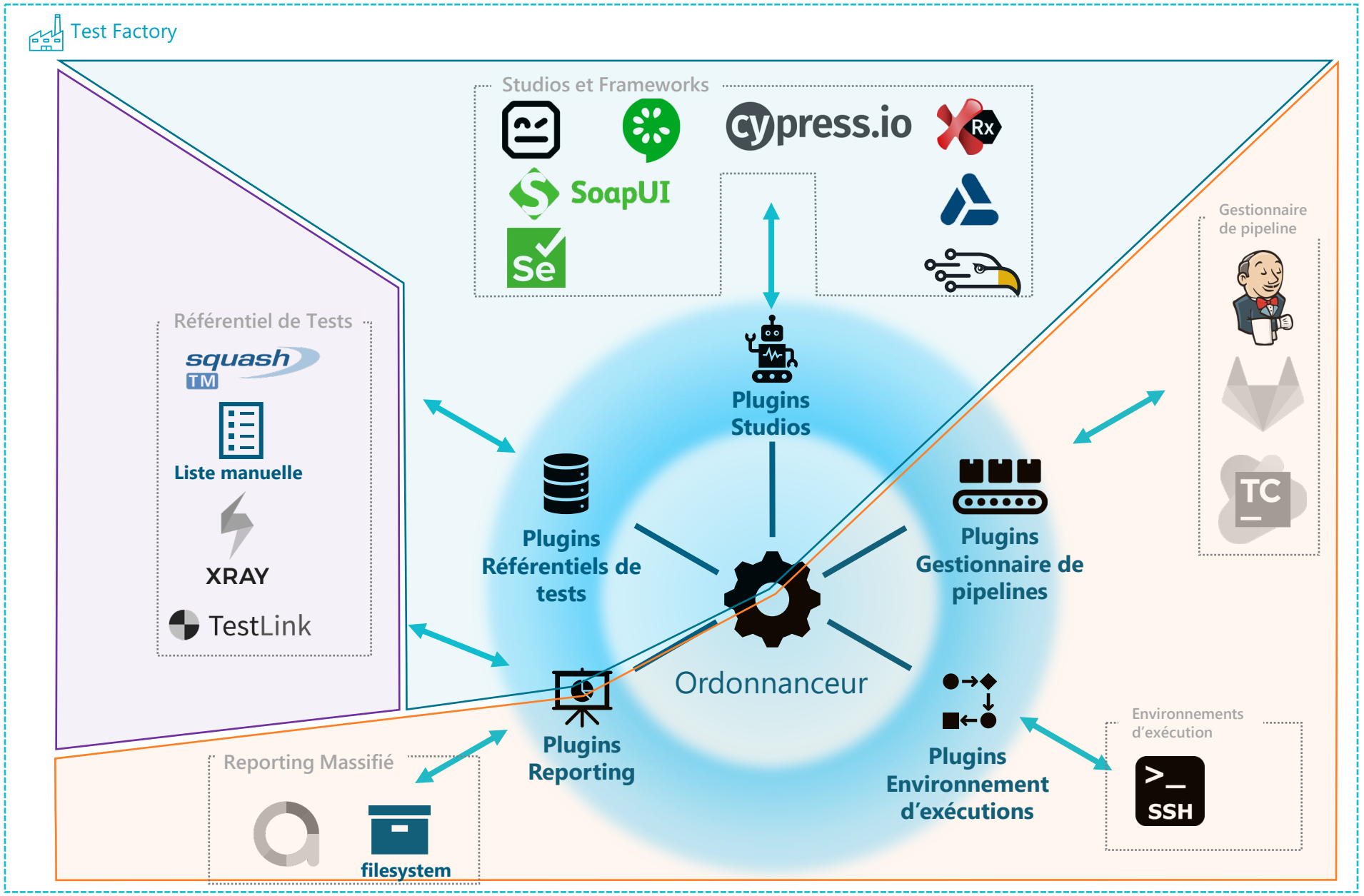
- 2.1 – « Test Factory » : (Notre) définition
- 2.2 – L'architecture en (micro)-services de l'ordonnanceur de de la « Test Factory »

# « Test Factory »: (Notre) Définition (1/2)

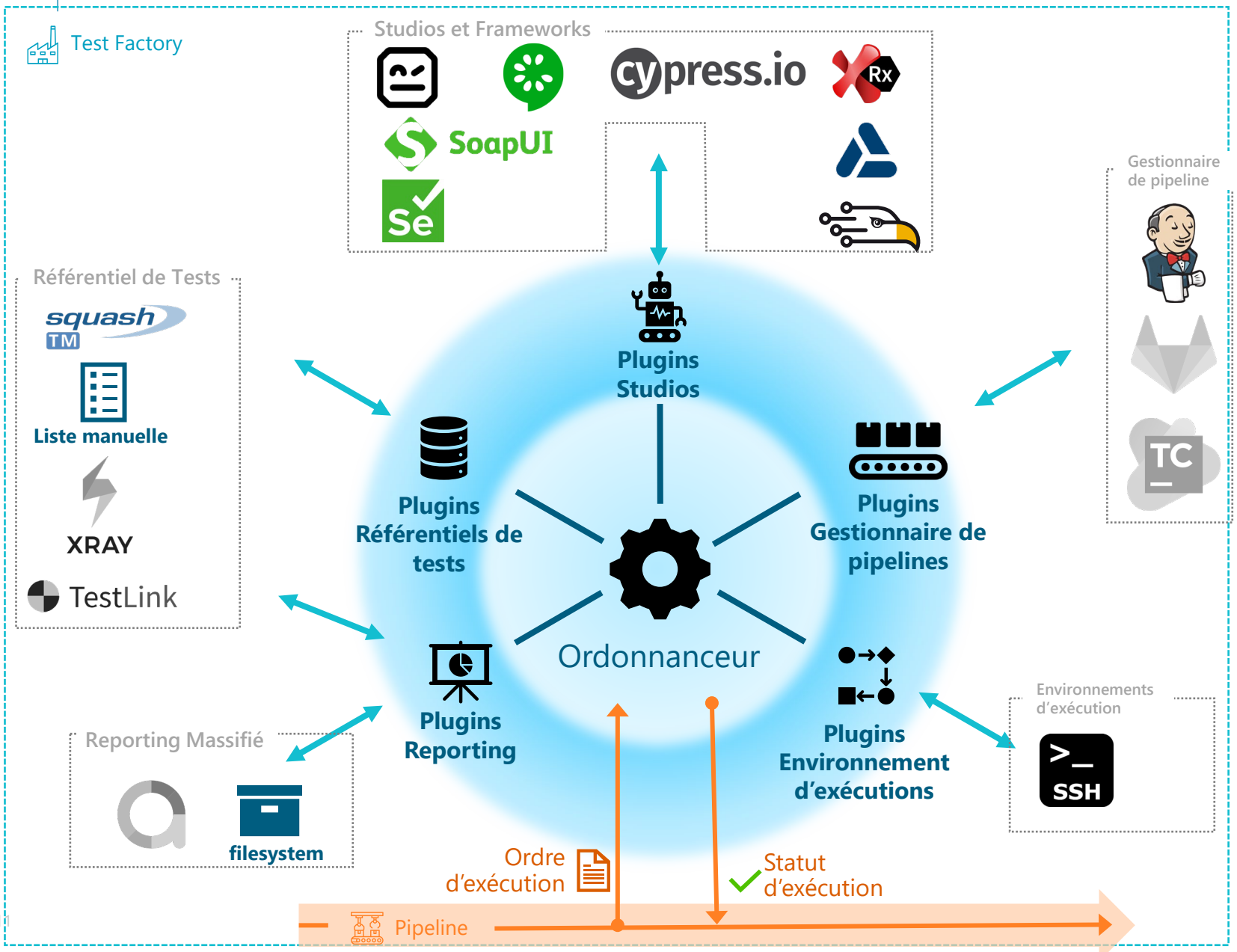
**Définition** : L'instanciation et la mise à disposition du pipeline de CI/CD des composants pour l'intégration et le lancement des tests fonctionnels (automatisés)



# « Test Factory »: (Notre) Définition (2/2)



# L'ARCHITECTURE EN (MICRO-)SERVICES DE L'ORDONNANCEUR DE LA « TEST FACTORY »




- Des micro-services
  - Un service a une responsabilité unique et précise (CQRS)
    - Transmission des résultats vers un outil de reporting
    - Lien avec un environnement d'exécution
    - Traduction pour l' automate choisi de l'ordre d'exécution
    - Etc …
  - Un service central (ordonnanceur - bus) pour la coordination des services
- Interface d'entrée : Un descripteur « As Code » du workflow de test
  - Structuré en Yaml ou Json
  - Définit le workflow de test à exécuter
  - Qui peut-être enrichi par les différents plugins
  - Permet d'être colocalisé avec les sources du projets
- Interface de sortie : Un statut d'exécution
  - Technique pour valider la bonne continuation du pipeline
  - Une Quality Gate

# 3 SQUASH (DEVOPS) TESTE SQUASH : (NOTRE) IMPLÉMENTATION DE TEST FACTORY

- 3.1 – Infrastructure pour la Factory et les SUT
- 3.2 – Intégration Continue
- 3.3 – Déploiement Continu
- 3.4 – Test en Continu : Utilisation des composants Squash pour la Test Factory
- 3.5 –Sécurisation de la plateforme
- 3.6 – Cas d'usage : Le pipeline complet de « Squash DevOps »

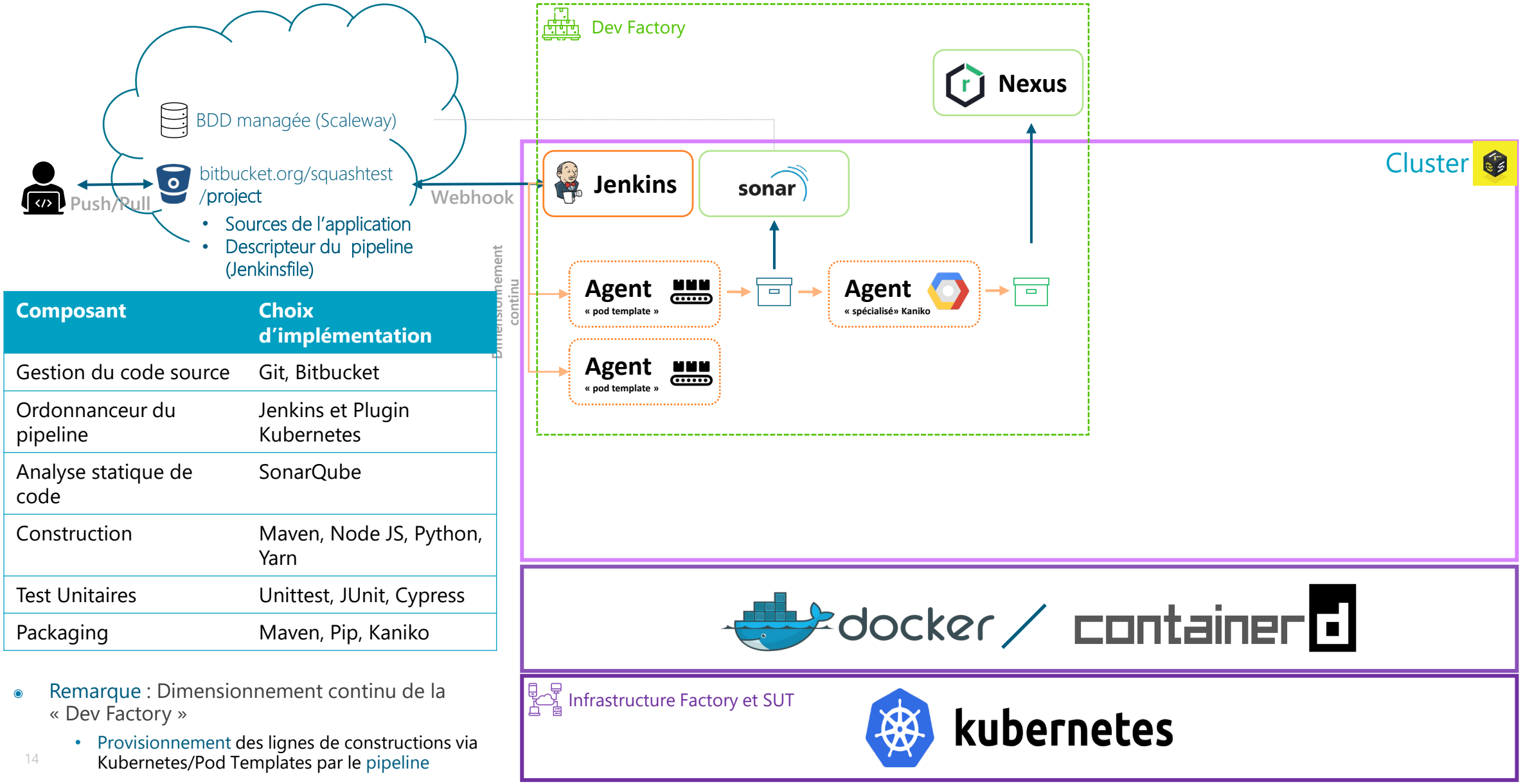
# 3.1 L'INFRASTRUCTURE POUR LA FACTORY ET DES SUT

| Composant | Choix d'implémentation |
|---|---|
| Gestionnaire de containeurs | Kubernertes / K3S |
| Containeurs | Docker / Containerd |

- Remarques :
  - Une Factory est soumise à une charge variable
    - Utilisée fortement pendant les constructions et les tests
    - Surtout en journée : le développeur est une espèce diurne
    - « En attente » la majorité du temps
  - Utilisation de Kubernetes/K3S pour sa capacité de mise à l'échelle horizontale
  - **Attention :** L'adaptation à la charge n'est pas un pré-requis pour le déploiement d'une « Test Factory » dans l'absolu.

# Intégration Continue

## CI « classique » Bitbucket/Jenkins/Sonar/Nexus

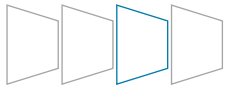


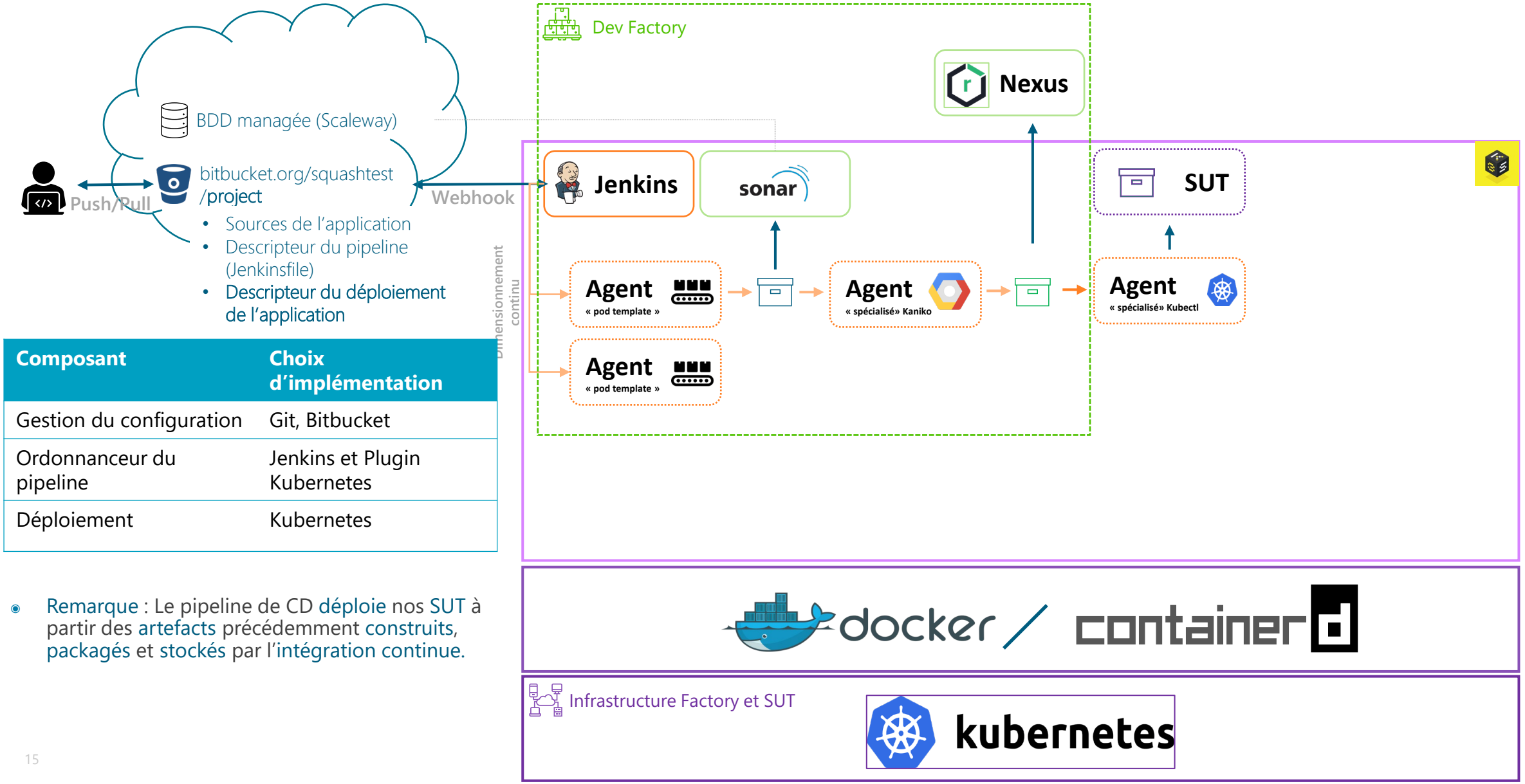

| Composant | Choix d'implémentation |
|---|---|
| Gestion du code source | Git, Bitbucket |
| Ordonnanceur du pipeline | Jenkins et Plugin Kubernetes |
| Analyse statique de code | SonarQube |
| Construction | Maven, Node JS, Python, Yarn |
| Test Unitaires | Unittest, JUnit, Cypress |
| Packaging | Maven, Pip, Kaniko |

- Remarque : Dimensionnement continu de la « Dev Factory »
  - Provisionnement des lignes de constructions via Kubernetes/Pod Templates par le pipeline

# Déploiement Continu

## Utilisation de Kubernetes pour déployer nos SUT

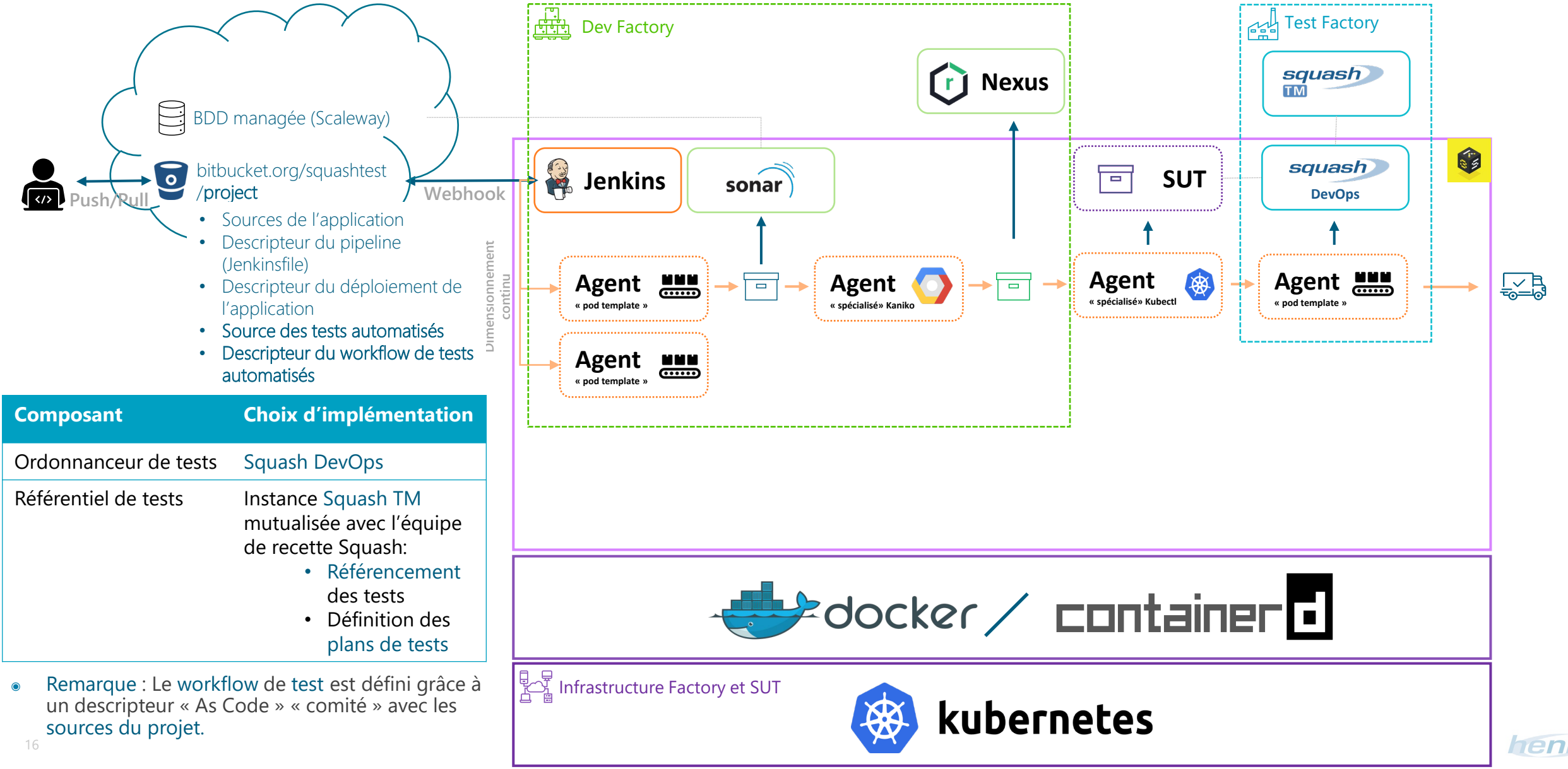





| Composant | Choix d'implémentation |
|---|---|
| Gestion du configuration | Git, Bitbucket |
| Ordonnanceur du pipeline | Jenkins et Plugin Kubernetes |
| Déploiement | Kubernetes |

- Remarque : Le pipeline de CD déploie nos SUT à partir des artefacts précédemment construits, packagés et stockés par l'intégration continue.

# TEST EN CONTINU (1/2):
## LES COMPOSANTS SQUASH POUR LA « TEST FACTORY »




| Composant | Choix d'implémentation |
|---|---|
| Ordonnanceur de tests | Squash DevOps |
| Référentiel de tests | Instance Squash TM mutualisée avec l'équipe de recette Squash:<br>• Référencement des tests<br>• Définition des plans de tests |

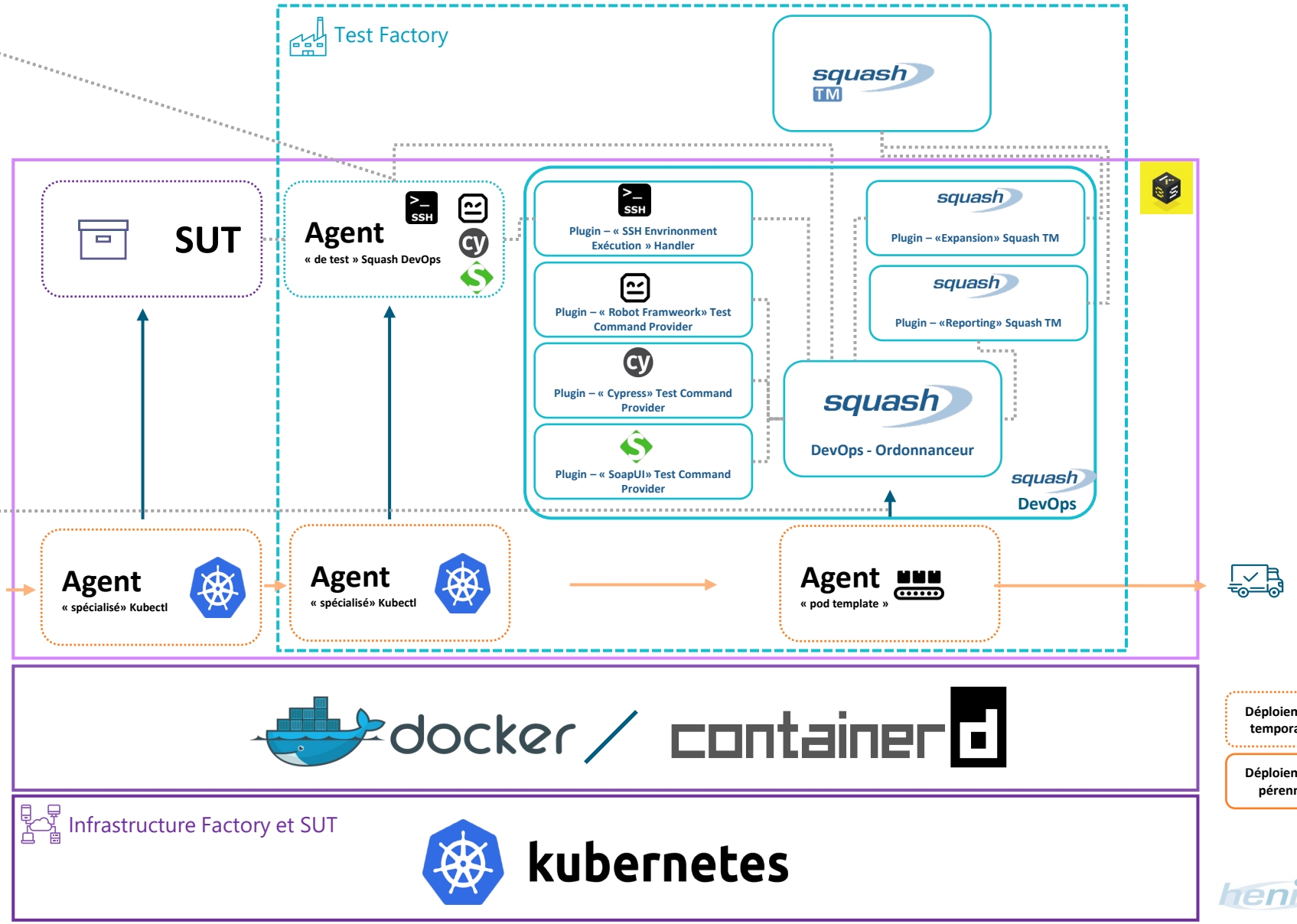
- Remarque : Le workflow de test est défini grâce à un descripteur « As Code » « comité » avec les sources du projet.

3.4

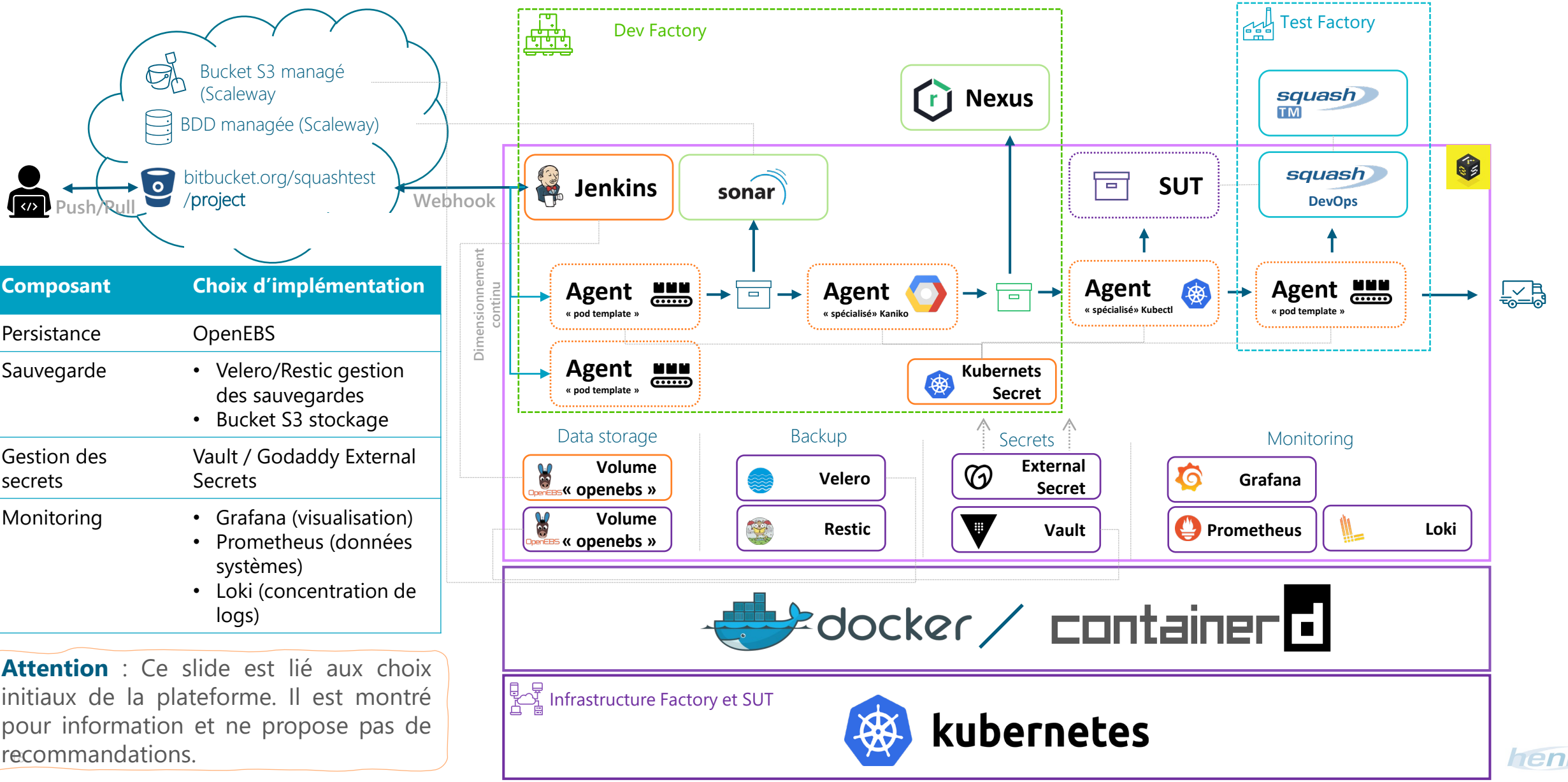
# Test en Continu (2/2):
## Zoom sur la « Test Factory »

- Sources des tests automatisés
- Descripteur « As Code » du workflow de tests

| Composant | Choix d'implémentation |
|---|---|
| Ordonnanceur de tests | Ordonnanceur de Squash DevOps |
| Référentiel de tests | Instance Squash TM mutualisée avec l'équipe de recette Squash:<br>• Référencement des tests<br>• Définition des plans de tests |
| Automates et studios | • Robot Framework<br>• Cypress<br>• Soap UI |
| Environnements d'exécution | « SSH » déployé par le pipeline CI/CD |
| Reporting | Reporting des exécutions dans Squash TM |



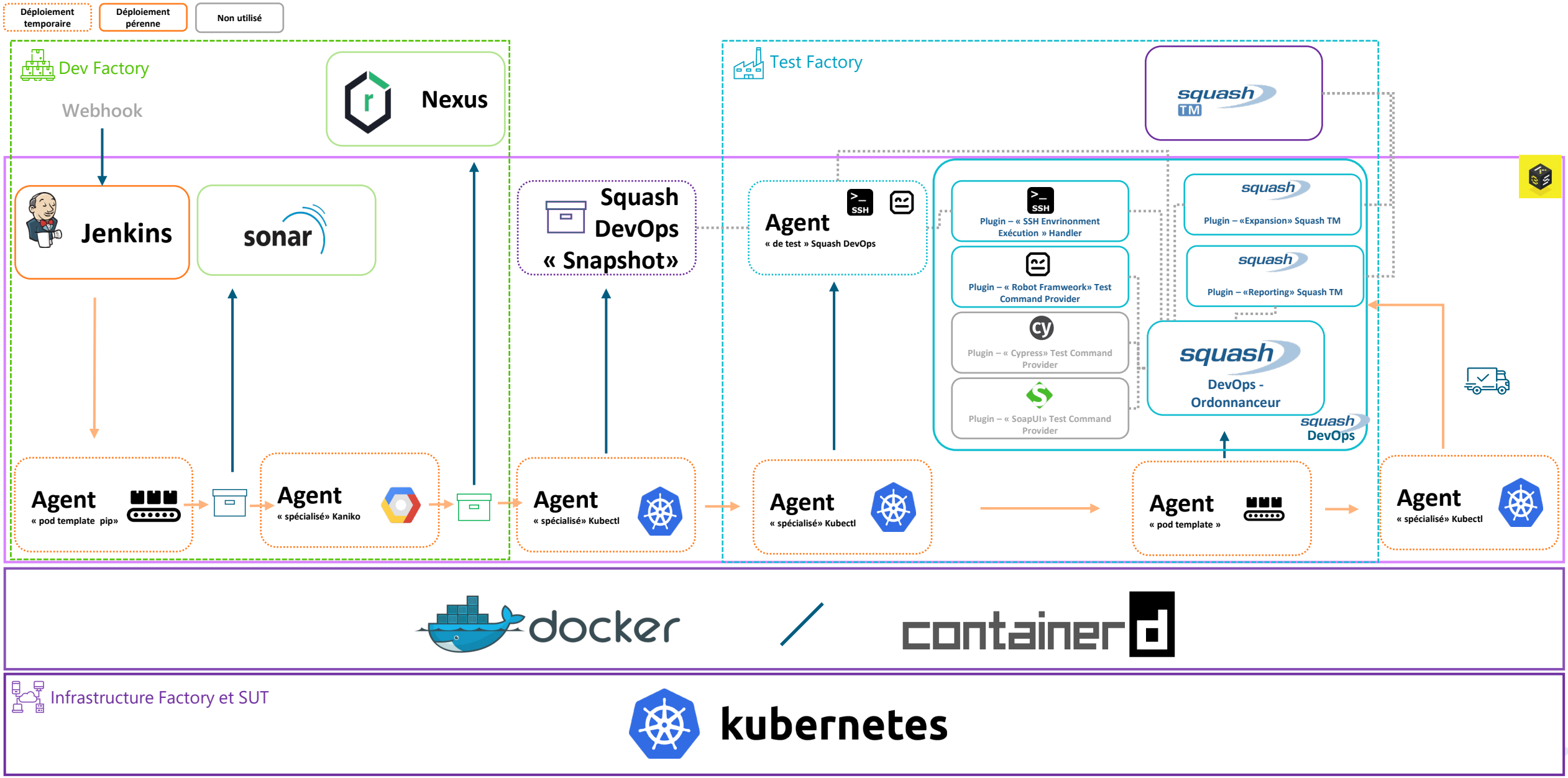
# 3.5 SÉCURISATION DE LA PLATEFORME




| Composant | Choix d'implémentation |
|---|---|
| Persistance | OpenEBS |
| Sauvegarde | • Velero/Restic gestion des sauvegardes<br>• Bucket S3 stockage |
| Gestion des secrets | Vault / Godaddy External Secrets |
| Monitoring | • Grafana (visualisation)<br>• Prometheus (données systèmes)<br>• Loki (concentration de logs) |

**Attention** : Ce slide est lié aux choix initiaux de la plateforme. Il est montré pour information et ne propose pas de recommandations.

# 3.6 Cas d'usage : Le pipeline complet de « Squash DevOps »



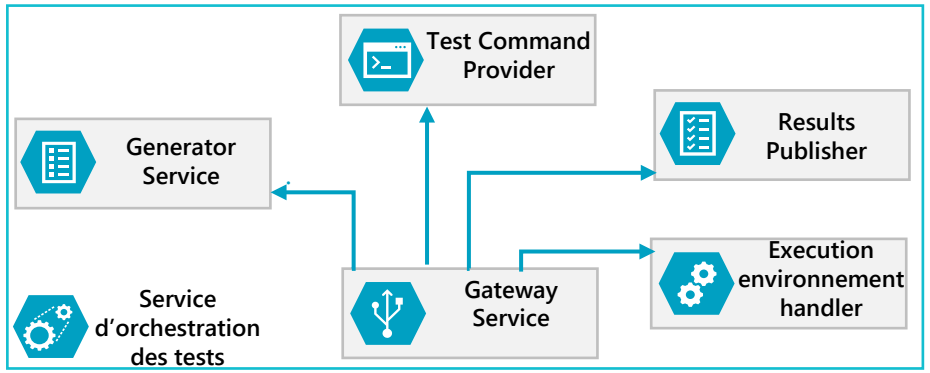
# 4 Aperçu de La roadmap de « Squash DevOps »

# La Roadmap « Squash DevOps »

Plus d'informations dans la présentation de l'équipe Squash cet après-midi !

- Les dates


- Deux packagings disponibles
  - Une image « all in one » incluant l'ensemble des services
  - Chaque service autonome




- Contenu


* Inclus dans la version béta

** Inclus dans la v1