Annexe 7.6 - Plateforme d'intégration continu de l'ANTAI (PIC ANTAI)

1 Description et utilisation de la PIC ANTAI

La plateforme d'intégration et de déploiement continu de l'ANTAI (PIC ANTAI) est une solution d'intégration et de déploiement continu. L'infrastructure de la PIC ANTAI est hébergée dans le centre de données du site CNT à Rennes. Le titulaire du présent marché de TME est chargé de sa maintenance et de son évolution.

La PIC ANTAI intègre actuellement les composants techniques suivants qui sont susceptibles d'évolution au cours du présent marché :

- Gitlab Community pour la gestion des référentiels de code source ;
- Gitlab CI Community pour l'orchestration des « pipelines » ; Pour l'orchestration, Jenkins Community est aussi toujours présent sur la PIC ANTAI afin d'assurer la compatibilité avec les anciens applicatifs métier de l'ANTAI.
- SonarQube Community pour la génération des rapports relatifs à la qualité du code et aux vulnérabilités SSI ;
- Nexus Community pour le stockage des « builds » applicatifs, des dépendances, des outillages, etc.
- Ansible Automation Platform pour la réalisation des déploiements applicatifs et la création des infrastructures techniques.

Sur le plan fonctionnel, la PIC ANTAI offre les services suivants :

- Un référentiel des sources complet, incluant un système de gestion des versions, qui permet de stocker l'ensemble des codes sources des composants dont l'ANTAI est propriétaire ;
- Un outillage d'intégration continue qui permet de compiler automatiquement les sources, d'exécuter les tests unitaires, de contrôler la qualité et la sécurité du code et de générer les livrables d'installation des plates-formes ANTAI ;
- Un outillage de déploiement automatisé ;
- Un service de stockage ;
- Un outillage d'exécution de tests automatisés (Cucumber, Selenium, Bruno, etc.) pour certains applicatifs.

Les titulaires des marchés de TMA doivent synchroniser leur propre référentiel des sources avec le dépôt Gitlab intégré à la PIC ANTAI. Cette synchronisation doit permettre de déposer les codes sources ajoutés et modifiés au sein de la PIC ANTAI *a minima* à la fin de chaque itération d'un cycle de développement agile. Pour cela, ils doivent s'assurer en phase de reprise que la version de GitLab qu'ils installent au sein de leurs propres environnements de développement est bien compatible avec celle de la PIC ANTAI (la version définitive sera indiquée par l'ANTAI lors du lancement des différents marchés). Les titulaires des marchés de TMA réalisent les développements/paramétrages des « pipelines » d'intégration sous Gitlab CI en coordination avec l'ANTAI et le titulaire du présent marché de TME. Le titulaire du présent marché veille en permanence au bon fonctionnement global de la PIC.

Les titulaires des marchés de TMA et celui du présent marché de TME constituent un catalogue de composants Gitlab standards qui sera mis à disposition et utilisé tout au long de l'exécution du présent marché par leurs équipes de développement, pour créer et mettre à jour leur

« pipelines » d'intégration continue et de déploiement continu sans recréer des composants spécifiques d'intégration.

En cas de mise à jour de composants dans ce catalogue, les équipes de développement utiliseront ces nouvelles versions, dès leur cycle suivant de développement. Ceci afin de limiter le nombre de versions de composants utilisées. Seules deux versions d'un même composant pourront être disponibles au catalogue.

De manière similaire au fonctionnement des titulaires de marché TMA, le titulaire du marché de TME doit livrer les sources de ses développements dans la PIC ANTAI. Il pourra toutefois les livrer directement dans la PIC ANTAI sans besoin d'un dépôt de source dans ses propres environnements de développement qui serait à synchroniser avec celui de la PIC ANTAI.

2 Structuration des dépôts des codes sources sous Gitlab

La gestion des dépôts GitLab est cruciale pour assurer une organisation efficace et évolutive du code source.
Le système d'information de l'ANTAI est structuré en systèmes applicatifs (SA) eux même composés de sous-systèmes applicatifs (SSA).

2.1 Organisation hiérarchique des dépôts

Au sein de GitLab, l'utilisation de **groupes et de sous-groupes** permet de refléter la structure du système d'information en SA et SSA :

- Groupe principal pour l'identification du SI concerné (ex. : BPO, Cœur…)
    - Premier niveau de sous-groupe pour l'identification des SA (ex. : Collecte)
        - Second niveau de sous-groupe pour l'identification des SSA (ex. : IRIS)
            - Dernier niveau hiérarchique composé des dépôts (ex. : Micro-service A, Front-end C…)

2.2 Granularité des dépôts Gitlab

Une application logicielle peut être constituée de différents modules et/ou de micro-services. Les modules applicatifs (exemple : IHM, Front End, Back-End, …) doivent être stockés au sein de dépôts distincts.
De même, concernant les micro-services, les codes sources doivent être stockés au sein de dépôts distincts (micro-service A, micro-service B) avec des exigences spécifiques sur la mise en place d'un fichier « README » qui décrit la configuration du micro-service et toutes ses dépendances.

2.3 Mutualisation et gestion des dépendances

Les composants développés pouvant être utilisés par d'autres projets dans une logique de mutualisation et de réutilisation des développements. Ces développements doivent être regroupés au sein d'un dépôt « Common » qui peut être utilisé par les autres dépôts relatifs au périmètre du présent marché.

2.4 Gestion des branches

La gestion des branches peut être héritée du contexte applicatif historique.

Concernant les nouveaux développements, le titulaire doit respecter la gestion des branches suivantes acceptant le « merge request » :

- Une branche « develop » qui constitue la version de travail des développements ;
- Une branche « release » qui constitue la branche de stockage des versions candidates à la production. Cette version sera préalablement testée dans le cadre de la VABF.
- Une branche « master » qui porte le code en cours d'exécution en production. Elle est donc mise à jour lors de la mise en production d'une version candidate.

En dehors de ces 3 branches, il peut exister d'autres branches permettant de répondre à des besoins spécifiques (ex. : « feature », « bugfix », « hotfix » etc.)

2.5 Versionnement des applications

L'ANTAI s'inspire du manifeste « Semantic Versioning » (https://semver.org/lang/fr/) pour les règles de versionnement des logiciels. Le titulaire doit respecter la méthode de versionnement suivante pour la maintenance applicative des composants qu'il gère au titre du présent marché.
En phase de reprise, le titulaire doit proposer pour validation à l'ANTAI une méthode précise de versionnement, de marquage et de constitution des notes de version pour ce qui concerne la maintenance applicative du présent marché. Une fois validée, le titulaire devra former en continu ses collaborateurs sur ces pratiques et vérifier régulièrement leur application au sein des dépôts.

2.6 Numérotation

La numérotation des versions se fait selon un modèle à trois niveaux :

- Génération (G) : évolution majeure entraînant des changements non rétro-compatibles ;
- Release (R) : nouvelles fonctionnalités ou améliorations significatives ;
- Correction (C) : Corrections de bugs ou petites améliorations.

Exemples de numérotations :

- 1.2.0 : deuxième release de la première génération ;
- 1.2.3 : troisième correction après la release 2 de la génération 1.

2.7 Versionnement des micro-services

Chaque micro-service doit posséder sa propre version indépendante pour permettre :

- Des mises à jour spécifiques sans impacter les autres services ;
- Une traçabilité précise des évolutions et des corrections ;
- Une flexibilité maximale dans le déploiement.

2.8 Versionnement global d'une application

Une application métier combinant plusieurs micro-services et/ou modules doit être elle-même versionnée.
La numérotation suit les mêmes règles que défini précédemment.

La livraison de cette version applicative doit être accompagnée d'une matrice de configuration, indiquant la combinaison des différentes versions de services et micro-services qui la compose.
L'application doit avoir été testée dans ces mêmes conditions de configuration.

2.9 Note de version

Toute version d'un micro-service ou d'une application ou d'un module doit être accompagnée d'une note de version décrivant le contenu de celle-ci (ex. : numéro du ticket Jira, titre et description des fonctionnalités…). Le titulaire doit faire valider le modèle de note de version à l'ANTAI au cours de la phase de reprise du présent marché.