



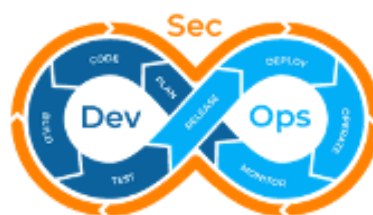
MINISTÈRE
DE L'INTÉRIEUR
ET DES OUTRE-MER

*Liberté
Égalité
Fraternité*

Cadre de Cohérence Technique (CCT)

Volet : Cloud (IT) Native

à portée interministérielle



Version : Béta

Date : 26/05/2024

Auteur : Direction de la transformation numérique du Ministère de l'Intérieur et des Outre-Mer.

A propos de l'appel à commentaire de ce document.

Ce même document est disponible sur le repository GitHub de la direction du numérique du MI :

<https://github.com/dnum-mi/CCT-Cloud-Native/>

Vous pouvez remonter vos commentaires et suggestions sur ce document de plusieurs manières:

- 1/ utiliser le fichier de relecture proposé [Fichier pour commentaires:](#)
 - <https://github.com/dnum-mi/CCT-Cloud-Native/blob/main/gabarit-pour-commentaires.ods>
 - et l'envoyer à :
dnum-architecture-entreprise@interieur.gouv.fr
- 2/ déclencher des 'Pulls request' sur le repository Github
- 3/ enregistrer des issues dans github

TABLE DES MATIÈRES

1 - Introduction	5
2 - Le contexte, les enjeux, la vision	6
3 - Principes généraux cadre Cloud Native	9
Périmètres du CCT et configurations prises en compte	10
Gestion des non-conformités, dérogations et contribution	11
Le modèle organisationnel, de responsabilité et de collaboration Cloud Native	12
Préconisations générales d'architecture et technique	17
Des spécificités à prendre en compte sur la création des conteneurs	18
Des spécificités à prendre en compte sur la topologie réseau et les ouvertures de flux	19
Des spécificités à prendre en compte autour de la qualité et de la sécurité des applications	19
Modèle d'intégration d'une application dans le cadre Cloud Native	20
4 - Présentation de l'offre interMinistérielle Cloud Pi Native et de ses évolutions pressenties	21
Les magasins de composants kubernetes et d'image de base	22
6 - Introduction à l'offre de service du ministère	23
5 - Référentiel d'exigences et modalités d'usage	24
6 - Annexes	26
Les normes industrielles, institutionnelles applicables	26
Liens vers autres contenus utiles(informatif)	27
Glossaire	28
7 - Référentiel d'exigences applicables au CCT Cloud Pi Native	37

version préliminaire pour appel à commentaire

--- page vide ---

1 - Introduction

Ce présent volet, du cadre de cohérence technique Cloud Pi Native, s'adresse aux développeurs, architectes et en général aux acteurs se projetant dans la planification, l'élaboration et la maintenance de produits numériques basés sur la conteneurisation et Kubernetes. Il présente le cadre et les exigences qui permettent à une direction d'application de faciliter la construction d'application de qualité et l'accès à l'offre de service proposée.

Ce cadre est soutenu par la plateforme de service Cloud Pi Native¹, proposant un parcours simple et prédictif, limitant les frictions organisationnelles et mettant à disposition des configurations préconfigurées pour réduire les charges de travail et cognitives des équipes.

La construction de produits numériques robustes basés sur les principes agiles et DevSecOps permet une plus grande réactivité des équipes, permet grâce à la mise en place de cycle plus court à quelques semaines permet d'obtenir un meilleur feedback des usagers pour produire des produits numériques de qualité et sécurisé qui répondent au besoin.

Le cloud dit *native*, basé sur la technologie de conteneurisation et d'orchestration, tel que Kubernetes rend plus robuste et fluide la construction, le déploiement et l'exécution d'applications. Ces technologies sont adoptées massivement par l'ensemble des d'organisation et des développeurs.

L'ensemble de l'offre Cloud Pi Native intègre les composantes suivantes :

- L'utilisation de clusters kubernetes sur cloud public pour des cas d'usage standard ou un hébergement ministériel on-premises durcis, soit en tant que namespace as a service (ou clusters dédiés) jusqu'au niveau DR pour les cas d'usage sensibles ;
- une console adaptées autonomisant les projet, intégrant un service de construction DevSecOps d'image applicative conteneurisée, de gestion du code d'infrastructure, et un système gitops assurant le déploiement automatisé, sécurisé depuis l'internet ;
- un modèle de sécurité (dast et sast) ainsi qu'un dashboard dédié à l'amélioration en continue de la sécurité pour les projet, responsable sécurité des métiers et les acteurs centraux cyber;
- un corpus documentaire et d'exemples permettant l'autoformation ;
- un modèle de maturité pour permettre la progression et le coaching des équipes.

Ce document et les ressources associées permettent à également pour objectif de :

- guider les concepteurs d'applications afin d'optimiser les architecture produites selon les normes industrielles rigoureuses, les meilleures pratiques DevSecOps et du Cloud Native tout en maintenant une capacité d'innovation ;
- mettre à disposition un référentiel d'exigences favorisant les bonnes pratiques et la conformité ;
- diminuer ressources consommées (financière, RH, énergétique) par la réduction de la quantité de code à produire et la modularité et l'efficience des architecture applicatives ;
- maintenir un niveau de compatibilité minimal entre les offres cloud ;
- fluidifier les déploiement, l'homologation en continu, le maintien en qualité ;
- mettre en place un modèle de responsabilité et de collaboration adapté ;
- disposer d'une trajectoire soutenable pour ceux en charge de maintenir l'offre.

¹ [lien vers fondation](<https://github.com/cncf/toc/blob/main/DEFINITION.md#fran%C3%A7ais>)

Le lecteur est invité à vérifier qu'il dispose de la dernière version de ce document de présent ainsi que de la liste d'exigences.

2 - Le contexte, les enjeux, la vision

Audience : ce paragraphe s'adresse à tout acteur considérant l'usage de l'offre Cloud Pi Native du ministère de l'intérieur, il présente les principes fondateurs. Le cloud : des nouvelles possibilités techniques, une collaboration étendue des acteurs pour répondre aux enjeux d'un contexte exigeant, incertain et accéléré.

À travers sa doctrine « Cloud au centre », l'État encourage l'ensemble des acteurs publics à se saisir de son potentiel afin de développer une nouvelle génération de services numériques de qualité et qui répond au besoin dans un cadre de normes juridiques adapté à l'usage. Cela répond notamment au besoin accru d'agilité et d'efficacité de l'administration.

Le Cloud est une approche d'accès à l'infrastructure d'hébergement à travers une interface standardisée et rendant abstraite et normalisée les technologies sous-jacentes. Cette abstraction permet une automatisation poussée, supprimer les frictions organisationnelles, les non-qualités et lenteurs induites par des opérations manuelles. L'infrastructure est pilotable par du logiciel et donc automatisable avec des processus reproductibles.

La technologie Cloud Native fait référence à l'usage de la conteneurisation et Kubernetes. Kubernetes² est une technologie d'orchestration de conteneur issue des travaux, il y a plus de 15 ans de Google, mise à disposition en open source et gérée par la fondation linux, permettant de simplifier le flux de production logiciel et rendre encore plus efficace et sécurisée l'usage des infrastructures techniques, la résilience des hébergements et apporter une souplesse organisationnelle accrue. Les grands services de l'internet s'appuient sur cette technologie, elle permettent une résilience extrême et permet d'absorber un trafic extrêmement important.

Les architectures des applications se simplifient avec une abstraction de plus en plus grande de l'infrastructure avec notamment la montée en puissance de services managés, fonctions-as-a-service dans l'objectif recherché de diminuer la quantité de code produite (et donc de réduire les bugs et la complexité de la maintenance) ainsi que grâce à une standardisation industrielle poussée et la conteneurisation, permet de réduire la charge des équipes, déchargée de nombreuse problématique de gestion de l'infrastructure de leur application et du déploiement.

L'ensemble des organisations ayant mis en œuvre cette technologie telles que EDF, Orange, des services de vente en ligne, des Banques, Airbus, Urssaf, etc... ont vu également leurs efficacités de l'usage du numérique augmenter, il y a un *avant et un après* Kubernetes.

Le ministère de l'intérieur, l'un des premiers acteurs étatiques à avoir proposé une offre Cloud il y a plus de 5 ans, étend son offre de service, en proposant l'offre Cloud Pi Native combinant une offre d'hébergement kubernetes sécurisée jusqu'au niveau DR. Cette offre

² <https://fr.wikipedia.org/wiki/Kubernetes> , mot grec signifiant pilote ou timonier.

est accompagnée d'un modèle DevSecOps permettant une fluidité organisationnelle accrue et un renforcement de la qualité des solutions numériques.

Les approches cloud, devops et l'agilité ont progressivement permis de concilier des postures antagonistes : les développeurs ayant besoin de pouvoir déployer fréquemment, et l'exploitation ayant au contraire besoin de stabilité et de diminuer les risques liés au changement. La clé réside dans une collaboration étendue de tous les acteurs en prenant compte de la sécurité à toutes les étapes en continue : le DevSecOps.

Une évolution des pratiques pour un numérique efficient et éco-responsable et réactif

Les contraintes s'accroissent sur la production de services numériques, le standard de qualité général a augmenté massivement avec les acteurs du net et industrielle qui produisent des solutions ergonomiques, sécurisées qui montent à l'échelle facilement. Un fossé important s'est creusé entre l'efficacité du numérique 'legacy' et ce monde moderne.

L'environnement change rapidement, il devient incertain, il faut réagir de plus en plus rapidement, la pression augmente sur la production de solutions numériques ergonomique, nécessitant des cycles raccourcis, la prise en compte de l'éco responsabilité, de l'accessibilité et le maintien d'un haut standard de qualité et de sécurisation.

Seul un changement majeur de pratiques s'appuyant sur l'opportunité du Cloud Native et du DevSecOps permettent de satisfaire ce nouveau standard d'exigences.

Le mode produit et l'agilité sont indispensables en complément de l'utilisation du cloud, extrait de la doctrine cloud au centre:

Citation issue de la doctrine cloud au centre de l'État : “ *L'adoption du cloud doit s'accompagner de celle des pratiques associées à l'excellence dans la production de services numériques (proximité entre métiers et équipes informatiques, scalabilité, agilité, « devops », « continuous delivery » qui sont les garants de l'adaptation des produits à leurs utilisateurs) ; “*

Les principales caractéristiques du modèle opérationnel Cloud Native:

Le fonctionnement évolue vers un partage des responsabilités dans la chaîne de production orienté vers la livraison du service vers les usagers.

L'équipe projet intégrée (ou équipe produit intégrée) voit ses prérogatives étendues:

- Elle est organisée autour du produit numérique livré. Elle fonctionne en modalité intégrée et de manière autonome en lien avec la vision et les contraintes fixées.
- Elle est composée de développeurs, architectes, ergonomes (ux-design), juriste, gestionnaire du changement, etc... orchestrée par le product owner et généralement facilitée par un coach / scrum master agile.
- Elle est focalisée sur l'ergonomie, la qualité et la performance de la solution mise à disposition des usagers. C'est le modèle «You build it, you run it, you support it ». (vous l'avez construit vous l'opérez)

Temporairement ou de façon permanente selon la taille de produit, une équipe d'aide, appelée "service team", peut être chargée, au sein de l'équipe produit, de mettre en place l'automatisation et les environnements de travail et de production³.

Cette dernière doit être aguerrie à ces technologies et l'offre cloud pi native.

L'hébergeur assure quant à lui, la mise à disposition d'une offre de service hautement disponible et sécurisée. L'usage de l'offre est réalisé via une console, une interface technique normée (API), une documentation et des exemples accélèrent la prise en main.

Il assure également la fourniture de services d'abstraction tels que des services managés ou fonctions as a service, un pipeline devsecops etc...

L'équipe intégrée est autonome et travaille sans échange avec l'hébergeur. La console mise à disposition assure ce lien avec une documentation associée. L'ensemble des opérations réalisées manuellement auparavant lors des étapes d'élaboration sont complètement automatisées. Le code logiciel est testé en permanence et automatiquement par un outillage : l'orchestrateur DevSecOps.

Pour assurer la qualité du code, plusieurs principes sont mis en œuvre, une couverture de test (100% sur le back-end), l'analyse statique du code, l'analyse réursive de vulnérabilité des composants importés (bibliothèque, images), des tests de sécurité et de performances, tests spécifiques liés à des besoins de vérifications particulières.

L'équipe de développement est prévenue au plus tôt par l'orchestrateur DevSecOps en cas de non-qualité. L'orchestrateur offre des options techniques pour intégrer les retours dans le flux de travail du développeur afin de procéder à la correction au plus tôt des anomalies, cette modalité s'appelle "shift-left". *In fine*, seul un logiciel ayant atteint le niveau de qualité fixé peut être déployé.

Les développeurs assurent continuellement la qualité et la sécurité du code produit, la non-régression et l'absence de vulnérabilité soutenus par l'outillage mis en place, la chaîne devsecops, les environnement de codage (IDE) et gestionnaire de code deviennent de plus en plus performant et identifient dès l'écriture ou l'intégration du code les erreurs d'algorithmes ou si un secret (mots de passe, clé API) est laissé dans le code ou aide le développeur à mieux produire du code optimisé via l'utilisation de l'intelligence artificielle.

Les architectures sont modulaires et les composants internes ainsi que les interfaces entre applications sont découplées, c'est à dire rendus indépendants techniquement et organisationnellement via des interfaces normées (API). Cela permet la maintenance et les évolutions indépendantes entre composants. Ce découplage contribue fortement à réduire le coût des changements et faciliter les transitions technologiques en cas d'obsolescence.

³ Le lecteur est invité pour plus d'explication à consulter l'ouvrage "Team topologies" de Manuel Pais et Matthew Skelton. <https://teamtopologies.com/>

La conception de l'architecture, le choix des langages sont faits pour une efficience de l'usage des ressources d'infrastructures et également sur l'impact sur le poste de travail.

La conteneurisation, l'élasticité dynamique de la solution d'orchestration de conteneurs kubernetes, la mutualisation des infrastructures physiques soutiennent fortement cette efficience.

Le non-respect de ce principe de modularité a été identifié par la direction interministérielle du numérique comme l'une des causes d'échec des grands programmes de l'État. La modularité, l'indépendance technique et organisationnelle des modules lors du déploiement, permettent de réduire la taille des déploiements ce qui contribue à réduire les risques et fluidifier les mises en production. Les déploiements peuvent ainsi être rendus transparents pour des usagers et les retours arrière éventuels sont facilités. Il est ainsi possible de déployer en confiance, plusieurs évolutions par jour si nécessaire.

L'automatisation permet de mieux contrôler et rendre les actions prédictibles, faciliter la reprise en cas d'incident et minimiser les coûts de maintien en conditions opérationnelles et de sécurité des applications. Plus aucune intervention manuelle n'est effectuée sur les environnements ce qui permet de réduire la variance, les non-qualités, et aussi de (re)construire très rapidement des plateformes.

In fine, la conception doit s'inscrire dans une démarche d'éco-conception et de sobriété numérique des conceptions (green IT) permettant un usage plus efficient des ressources qu'elles soient RH, financières ou énergétiques. L'État devant être exemplaire⁴.

3 - Principes généraux cadre Cloud Native

Audience : ce paragraphe s'adresse à la communauté des concepteurs et architectes solutions, le lecteur est réputé compétent et formé sur les sujets abordés .

Le cadre de cohérence technique régule et normalise les différents domaines associés à l'élaboration et au maintien des ressources partagées nécessaires à la mise à disposition de solutions numériques de qualité répondant au besoin. Il s'assure que l'ensemble peut-être mis en œuvre de manière cohérente avec une consommation minimisée des ressources : financière, RH et écologique. Il recommande ou fixe les mesures permettant d'atteindre l'objectif, tout en favorisant l'innovation, la prise en compte de l'obsolescence régulière des technologies et la manœuvre RH nécessaire (formation continue, recrutement ...)

Le volet du CCT Cloud Native du ministère de l'intérieur, hérite de normes industrielles, interministérielles et européennes. La portée de ce document est interministérielle, il a fait l'objet d'échanges avec la direction interministérielle du numérique et des ministères primo-accédants.

Pour le ministère de l'Intérieur, il encadre la conception et l'hébergement d'applications qu'elles soient hébergées dans les datacenters du ministère ou bien à l'externe sur des

⁴ cf guide d'éco-conception de la direction interministérielle cité en référence.

clouds publics. Ce volet s'applique notamment lors de la conception d'une nouvelle application ou une évolution significative d'une application existante. (cf Doctrine Cloud au centre)

Pour les autres entités étatiques, ce volet sert de présentation de bonnes pratiques et présente les conditions générales d'utilisation de l'offre Cloud Pi Native.

Ce document ainsi que le référentiel d'exigences sont annexés aux dossiers de consultation des entreprises. La portée administrative étant précisée par le règlement de consultation (RC) du marché lui-même, typiquement le RC et le Cadre des clauses administratives particulières (CCAP) peuvent exclure administrativement un candidat en cas de non-respect des exigences P primordiales. (hors dérogation soumise, instruite et accordée).

D'autres référentiels d'exigences ou des guides peuvent être applicables ou conseillés. voir plus loin le chapitre sur les cadres de normes supérieures.

Offre de service Cloud (π) Native :

Concerne la description de l'offre de service managé d'infrastructure Cloud π et d'une chaîne DevSecOps assurant l'homologation en continu et le déploiement en production. Cf. présentation de l'offre plus loin dans ce document. Cette offre évoluera selon les demandes et financement disponible vers services d'abstraction de l'infrastructure, tel que service managés, fonction, etc...

Poste de travail agent :

Dans le cadre d'une application rendue accessible sur le poste de travail de l'agent, le lecteur est invité à se conformer également au volet *Environnement Numérique de Travail*, notamment sur les aspects d'intégration au SSO et la politique des navigateurs.

Ouverture des données :

Sur la thématique de l'ouverture et de la circulation de la donnée, le projet est invité à se mettre en conformité avec le volet idoine. Cela concerne notamment le référencement des objets métiers dans le référentiel de cartographie des données et la mise à disposition d'une facilité technique d'accès à la donnée basée sur un standard d'échange de type API.

L'ensemble des acteurs de l'État est invité à faire circuler la donnée au profit d'une simplification du fonctionnement des administrations et d'un service public ergonomique et proactif . (cf rapport Bothorel, lois CRPA et 3DS, ...)

Périmètres du CCT et configurations prises en compte

Ce CCT concerne l'usage de l'offre Cloud Pi Native dans les configurations précisées ci-dessous.

Le ministère de l'intérieur dispose de plusieurs capacités d'hébergement d'application. Ces offres peuvent être historiques et liées à une entité (ex: Sgami, ANTS) ou centrales.

Les offres d'hébergement centrales sont découpées en plusieurs catégories :

- **Physique ou virtualisées** généralement de type VMWare tel qu'Isocèle (DTNUM), STIG (ANFSI). Cette offre est accessible qu'au MIOM et seul l'exploitant gère la plateforme et les actes d'intervention techniques via Ticketing ITMS
- **Offre Cloud Pi⁵** : offre IaaS basée sur OpenStack. La gestion interne des 'tenants' est à la main du développeur, le reste via ticketing. Note : Le développeur gère son outillage projet en autonomie
- **Offre Cloud Pi Native** (objet de ce volet de CCT) : offre s'appuyant sur Cloud Pi, avec un espace d'exécution sur kubernetes, des services d'accélération de construction applicatif en sus.

Les configurations suivantes sont prises en compte par ce volet Cloud (Pi) Native du CCT.

- Hébergement sur les clusters kubernetes managés par le ministère de l'Intérieur, jusqu'au niveau « donnée restreinte » ;
- Hébergement sur un cluster kubernetes externe au ministère, compatible avec la sensibilité des données manipulées ;
- Hébergement sur un cluster kubernetes dédié et géré par l'application;
- Une approche hybride multi-clusters (plusieurs environnements de production on prem ou cloud public).

Pour l'ensemble de ces configurations l'usage de la chaîne DevSecOps managée par le Ministère de l'Intérieur est impératif. (hors cadre dérogatoire accordée)

Gestion des non-conformités, dérogations et contribution

L'évolution rapide des technologies cloud peut conduire à ce que le cadre CCT restreigne l'innovation. Il est également souhaité, pour éprouver le modèle, de notifier le département architecture d'entreprise du Ministère de l'intérieur au plus tôt des éventuelles impossibilités ou limitations remarquées. Les directions d'applications ou les organisations utilisatrices peuvent contribuer, via un échange préalable, à enrichir les fonctionnalités de l'offre ou du cadre lui-même. Sur l'offre la contribution est effectuée directement sur le repository open source de la solution via un pull request.

En cas de non-conformité au CCT ou absence de contribution à l'offre, une demande de dérogation dûment motivée sera formulée à l'avance par la direction d'application. Seule la notification de la décision permet d'amender le besoin de conformité au cadre, temporairement ou de manière pérenne dans le cadre d'une dérogation. Dans le cadre d'une dérogation, la direction d'application prend à sa charge le surcoût complet de possession. (formation, homologation, personnel assurant la tme, etc...)

Lors de l'utilisation du cadre et de l'offre Cloud PI Native, toute organisation souhaitant décliner ce cadre dans un document de norme inférieur pour un besoin propre est invitée à référencer la dernière version de ce document en l'état. Dans la hiérarchie des normes, une instruction de niveau inférieur ne peut entrer en conflit ou contredire ce présent document.

⁵ PI : produit de l'Intérieur

Le modèle organisationnel, de responsabilité et de collaboration Cloud Native

L'architecture, le modèle de responsabilité et d'organisation à mettre en place est orienté pour maximiser la qualité, la sécurité, la fluidité opérationnelle et l'évolutivité du produit en tirant parti au maximum des possibilités offertes par la technologie kubernetes, un flux de production DevSecOps et une collaboration étendue entre les acteurs.

L'élargissement de la responsabilité du développeur et de l'équipe produit

La responsabilité de l'équipe produit est élargie dans le cadre Cloud Native. Elle élabore et exploite une solution qui répond au besoin métier généralement une automatisation d'un ou plusieurs processus métiers . L'équipe s'assure de la qualité et de la disponibilité du service rendu à l'utilisateur selon le précepte : « You build it, you run it ». L'équipe s'organise de façon intégrée, si nécessaire avec de l'externalisation, pour couvrir l'ensemble des aspects nécessaires de la conception à l'exploitation des produits.

Le développeur, en particulier, met à disposition d'un point de vérité du code sous la forme d'un ou plusieurs dépôts de code logiciel fonctionnel et d'infrastructure. Il met en place un flux intégré et continu de production en s'appuyant sur un orchestrateur primaire DevSecOps qu'il construit et opère.

Le développeur initialise et supervise le pipeline de l'orchestrateur secondaire opéré par le ministère de l'Intérieur (cf plus loin sur la configuration de l'offre Cloud Pi Native). Il intègre les étapes de vérification de sécurité génériques imposées par le ministère et spécifiques issus de la démarche d'homologation.

L'ensemble combiné des orchestrateurs primaire et secondaire assure la fonction d'homologation et de déploiement en continu du produit numérique.

Dans le cas de la détection d'une non-qualité critique, telle une vulnérabilité critique, la progression du déploiement est bloquée, le développeur est prévenu en temps réel et doit corriger au plus tôt les défauts remontés. Cette approche permet de garantir un niveau de qualité, évite des régressions et maintient la dette technique au niveau le plus bas.

Sur le plan organisationnel le développeur met généralement en place :

- un contrôle de qualité au plus tôt, par exemple par un assistant et la revue de code ;
- l'agilité avec des itérations courtes de constructions et de vérification des usagers ;
- le découpage des livraisons en lot de taille de réduite ;
- la mise en place d'une culture de collaboration étendue et des pratiques intégrant la sécurité à toutes les étapes.

La répartition des responsabilités s'établit de la manière suivante :

L'équipe produit intégrée :

- est responsable de l'application, de la qualité du code et du bon fonctionnement de l'application pendant l'ensemble du cycle de vie de l'application.
- est responsable de définir et d'ajuster l'infrastructure en s'appuyant sur l'élasticité du cloud.(sur la base de l'offre Cloud adaptée selon la sensibilité des données)

- s'appuie sur les patterns applicatives et services managés mis à disposition, les magasins de charts helms et des *operators* disponibles. Il est déconseillé, par exemple, de repackager une base de données alors qu'un *operator*, un chart ou un service managé est disponible. (simplification et systématisation du mcs)
- fournit un code de qualité exempt de défaut d'algorithmes, de qualité et de vulnérabilité ;
- met en œuvre le flux de production, logiciel local permettant d'assurer la production et la démonstration d'un code de qualité exempt d'anomalie fonctionnelle, de non-qualité et de vulnérabilité, notamment dans les librairies importées ;
- pose les normes de développement des langages utilisés ;
- met en place des pratiques DevSecOps visant un maintien de la qualité dans le temps avec les composantes suivantes (cf outillage DevSecOps) :
 - test driven development ;
 - couverture de test unitaire à 100% du back-end de l'application ;
 - couverture significative des tests du front de l'application ;
 - analyse statique de qualité du code ;
 - analyse réursive des vulnérabilités des librairies importées ;
 - utilisation exclusivement d'images sources maintenues en condition de sécurité et certifiées (distribution LTS) ;
 - conception des tests d'intégration en sandbox ;
 - fourniture des outils nécessaires à la remontée de l'état de santé des briques applicatives destinées à fonctionner en production (healthcheck) ;
 - fourniture des indicateurs nécessaires au suivi en temps réel de la qualité en condition opérationnelle de sa solution (exports prometheus) ;
 - exploitation des logs remontés.
- met en place un hébergement sur une plateforme kubernetes afin d'assurer la démonstration du bon fonctionnement de l'application avec la solution qu'il préfère soit internalisée (avec un moyen de mener des démonstrations) ou sur cloud public.
- met en œuvre l'intégration technique et organisationnelle avec la chaîne DevSecOps de l'offre Cloud Pi Native et initialise le flux logiciel global (cf plus bas).
- maintient un point de vérité du code logiciel ainsi que celui du code d'infrastructure. Celui-ci est accédé par la chaîne DevSecOps étatique, la sécurisation d'accès issus par token.
- est responsable de la surveillance de l'ensemble des pipelines, y compris pour celui géré côté ministère.
- met en place une intégration du flux de retour d'anomalie "shift-left" des orchestrateurs afin de permettre une correction au plus tôt des anomalies.
- effectue l'apprentissage comportemental du firewall applicatif Web (WAF) vis-à-vis de l'application dans le cadre fixé par le ministère. (anticipation avant la mise en production)
- est invité à mettre en œuvre ce pipeline au plus tôt dans le processus de réalisation.

Notes : (cf exigences CCT)

L'équipe de développement respecte les règles suivantes permettant une qualité de code en progression et un maintien de la sécurité :

- minimise la portion spécifique de code développés en s'appuyant sur le catalogue des services proposés. (revoir régulièrement)
- met en place une couverture de test unitaire complète du back-end (et fourni les moyens de vérification automatisé à la chaîne secondaire)

- mener une analyse de code systématique le plus tôt possible (les langage et IDE modernes fournissent des fonctions de ce type)
- mener une analyse de CVE des dépendants importées et apporter des corrections.

L'équipe intégrée met en œuvre une activité continue de refactoring du code produit. Ainsi, la qualité du code ne peut pas être décroissante.

Elle fournit les preuves que des tests de sécurité, de qualité, de robustesse des algorithmes ont été mis en œuvre, et qu'ils n'ont pas remontés de vulnérabilités ou d'erreurs majeures. En s'appuyant notamment sur les logs des analyses des outils de la chaîne primaire. Elle fournit la preuve (ex: le document) des normes de développement et pratiques permettant de maîtriser la qualité du code produit. (refactoring, peer review, etc..)

Note : l'équipe s'assure qu'après le dernier déploiement stable de l'application, toutes nouvelles vulnérabilités critiques et importantes seront détectées et corrigées. En cas de non correction des anomalies dans un délai de plusieurs mois et surtout si l'application est exposée sur internet, l'hébergement de la solution pourra être suspendu pour maintenir en intégrité les données.

L'exploitant Ministériel de la console et de l'orchestrateurs DevSecOps :

- Il assure de la disponibilité et de la qualité de fonctionnement de la chaîne DevSecOps et maintient à jour la documentation sur le fonctionnement des interfaces et assure les évolutions fonctionnelles ;
- Il assure les retours d'anomalies "shift-left" lors des opérations de déploiement continu. L'intégration et leurs traitements sont à la charge du Concepteur / développeur ;
- Il contribue à la mise en place et l'évolution du catalogue d'operator Kubernetes, de charts helm et du référentiel de pattern de référence;
- Il est également en lien avec les autorités d'homologation afin de s'assurer que l'ensemble est en condition d'assurer la protection d'ensemble;
- Il intègre les propositions d'évolution "pull request" proposée en fonction de son plan de charge et d'une négociation préalable.
- Il fournit la documentation dont les éléments d'orientation ou d'aide à la migration permettant au développeur de s'orienter vers l'opérateur cloud qui hébergera son produit. (cela dépend également des décisions projets ou d'homologation)

L'opérateur Cloud ministériel ou Public :

Il assure le maintien en condition de disponibilité et de sécurité de l'offre d'hébergement, l'interface API de management, la console, la gestion capacité et les offres de services managées.

Dans le cas de l'offre ministérielle du ministère de l'Intérieur, plusieurs services additionnels peuvent être consommés, tel que la supervision applicative ou de sécurité, la fourniture de poste d'administration sécurisés.

Le Cloud PI est constitué de plusieurs régions (elle même constituées de plusieurs centres de calculs) en zones de sensibilité usuel et DR.

Chacune des régions peuvent faire l'objet d'une affinité ou restriction de services, sur la nature des données, des typologies de projets, ministériels et interministériel, chaîne de SecOps secondaire; etc...

La gouvernance du produit 'Cloud Pi Native' à portée interministérielle et ministérielle définit la politique d'ensemble, cette dernière traite d'aspects tel que : restriction ou affinité d'accès aux régions, projets, chaîne secondaire ;

- politique de données définie dans l'extension de la définition de "données restreintes" ;
- la capacité en matière de ressources nécessaires à héberger de nouveaux projets ;
- la localisation géographique et le nombre de centres de calculs constituant sa région ;
- les services mis à disposition ;
- les possibilités d'interconnexion réseau avec d'autres systèmes ;

La version courante de l'offre et la politique d'usage est mise à disposition sur portail et notamment vers l'utilisateur lors de la connexion à la console. La version courante est présentée dans les grandes lignes au paragraphe 4.

Des pratiques complémentaires sont introduites dans la configuration Cloud Native :

Le "GitOps", contraction de git et opération, est indispensable à la gestion des applications Cloud Native avec Kubernetes. Ce mode d'organisation du code d'infrastructure permet de maîtriser la description de l'infrastructure de production avec les mêmes pratiques de revue collaborative que celle du logiciel. Il est par exemple strictement interdit de faire des modifications «à la main » sur l'environnement de production, toute variation est supprimée, l'infrastructure réelle est strictement celle décrite par les fichiers d'infrastructure.

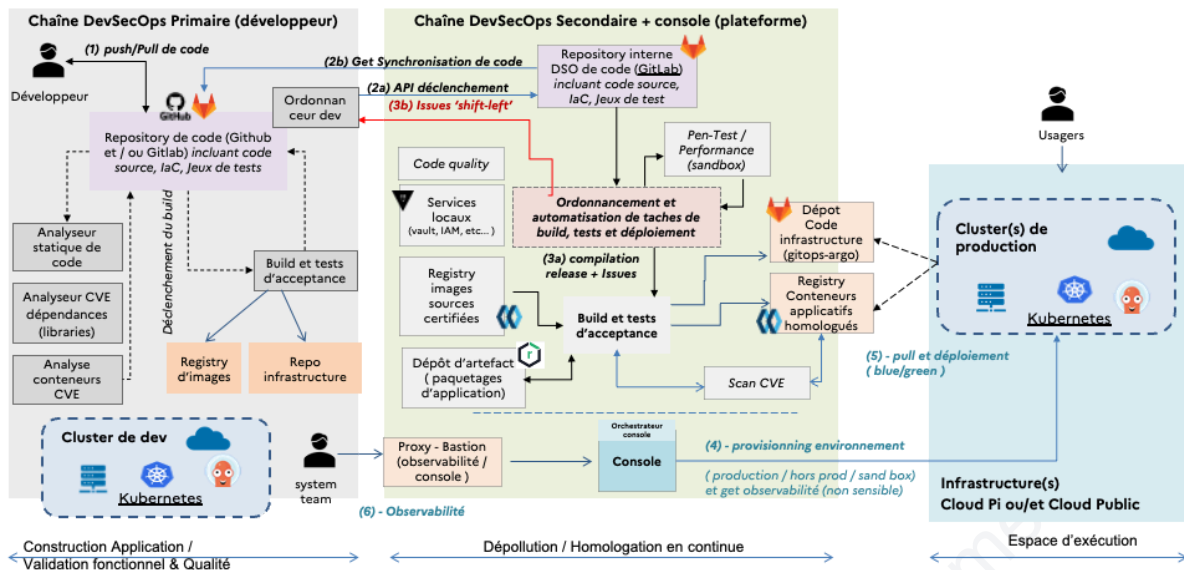
Le **"shift-left"** (vers la gauche du processus de développement) fait référence à la remontée le plus tôt possible vers le développeur des anomalies identifiées par la chaîne de déploiement et de vérification DevSecOps. Ce flux est notamment mis en œuvre depuis la chaîne secondaire.

Présentation du cycle d'usage de l'offre pour les directions d'applications:

Phase d'initialisation du projet

- Le développeur initialise l'environnement de développement, il est autonome pour les choix techniques, il respecte les exigences organisationnels et de processus automatisé permettant de maintenir une qualité constante ;
- Le développeur décide de l'infrastructure d'hébergement en fonction des contraintes sur les données et la liste des options autorisées et maintenue par la Dinum et l'ANSSI en lien avec la doctrine Cloud au centre, notamment : Cloud Pi, cloud externe de confiance ou plateforme dédiée (si besoin spécifique)
- Le développeur commande, (signature de convention), initialise l'espace projet au ministère et configure selon son choix d'infrastructure les environnements désirés. Il récupère les clés techniques nécessaires à l'intégration des pipelines.
- Le développeur effectue l'intégration des pipelines, cf labels (2) , et (4) si l'infrastructure est externe.
- Il vérifie que l'ensemble du pipeline est opérationnel à partir d'un code d'exemple fourni de type "hello word".

Principe de fonctionnement du pipeline d'ensemble (chaînes primaire et secondaire)



- **[1]** Le code logiciel ainsi que celui de description des infrastructures sont produits au sein de l'espace du développeur/concepteur, généralement en externe au Ministère de l'Intérieur.
- **[2a] [2b]** Une interface bi-directionnelle entre l'espace du concepteur / développeur et celui de DSO permet en push-pull à la chaîne secondaire de récupérer automatiquement l'ensemble du code et des dépendances nécessaires.
- Le développeur déclenche par appel API les services de synchro / build / deploy. (il n'y accède pas directement sauf via un Bastion)
- **[3a][3b]** La chaîne d'orchestration DevSecOps effectue la récupération du code, des tests de qualité du code, scan de vulnérabilité des dépendances, la reconstruction, les tests de nos régressions, des tests d'homologation, vérification des manifests / charts etc... au regard des politiques de sécurité et dépose les images certifiées sur la registry de la chaîne ainsi que le code d'infrastructure.
- Le concepteur/développeur accède à un retour d'information détaillée sur le succès ou sur les éventuels défauts, lors du build, deploy de l'application par DSO. (via message ou webhook)
- **[4]** : La console provisionne si nécessaires les environnements en 'poussant' les ressources nécessaires une fois (secrets, application.yaml (argo), certificats, etc...)
- **[5]** : L'infrastructure vérifie régulièrement les changements sur le dépôt d'infrastructure (ou déclenchement forcé par API) et synchronise l'environnement à la cible visé et opère une bascule blue-green transparente, cf ArgoCD (si échec l'environnement de prod reste inchangé)
- **[6]** : Le développeur accède à un proxy d'observation du fonctionnement de l'application
- Note : Les développeurs n'accèdent pas directement à la production. Seuls les administrateurs habilités peuvent y avoir accès via bastion.

Préconisations générales d'architecture et technique

Ce chapitre précise les aspects importants liés à l'usage de kubernetes dans le cadre du ministère de l'intérieur. Il est attendu que les acteurs soient correctement formés à la solution kubernetes et se maintiennent à jour. La technologie évoluant rapidement.

Le fondement des normes techniques est issu du cadre "Cloud Native", largement accepté et appliqué au sein de l'État et le secteur privé, tel que les "15 factors".

C'est le respect de ces normes qui permet à la fois d'adresser les enjeux de performance en termes de vitesse de livraison et de qualité de service, mais aussi de normaliser les applicatifs pour une meilleure évolutivité et maîtrise de la dette technique. Enfin, elles assurent une intégration fluide au sein des systèmes d'information Ministériels.

Un des principes cœurs est de laisser un certain degré de liberté au concepteur/développeur sur le fonctionnement interne de son application. Au contraire, les interactions avec les autres applications et services seront particulièrement contraintes.

Il est à noter qu'uniquement la plateforme d'orchestration de conteneurs Kubernetes est considérée dans le cadre de ce cadre de cohérence, celle-ci étant considérée comme l'état de l'art, et open-source de surcroît.

À propos de la solution mutualisée d'hébergement Cloud π Native et de l'ergonomie pour le développeur

La philosophie générale est de balancer la réduction de la surface d'attaque cyber et l'ergonomie indispensable pour le développeur. La réponse définitive ne peut être que celle éprouvée en run, nous livrons ici une intention qui doit être rodée opérationnellement.

- Sur les **environnements côté "primaires"**, le développeur est (doit être) maître 'root,' il les maîtrise et les utilise pour son besoin de déboguer tester des choses, etc... l'administration n'intervient pas ici.
- Sur les éventuels **d'environnement hors production côté secondaire au MI**, l'environnement est dédié au développeur il dispose d'un accès root à sa plateforme selon les règles de la PSSI applicables. (voir plus loin)
- Sur **l'environnement de production côté MI**, les développeurs n'accèdent pas directement, cet accès s'effectue via une console dédiée et un flux "gitops". (via l'usage d'ArgoCD). L'accès à la production nécessite une accréditation renforcée et adaptée et un poste 'rouge' (niveau habilitation SD).
- Sur **l'environnement de production chez un hyperscaler public**, le développeur gère sa plateforme. Généralement, il met en place un modèle d'opération réduisant les accès à la plateforme de production aux seuls administrateurs habilités.

Notes :

A propos de l'accès à des ressources d'environnements d'exécution situé côté administration, la règle fixée par la PSSI (politique SSI de l'état) s'applique, notamment :

L'accès hors production nécessite une accréditation / enquête et un poste de travail adapté et maîtrisé par l'administration que l'on puisse raccorder au réseau du MI.

Pour l'accès distant, un poste "bastion" adapté, que l'on appellera ici "poste jaune", sera nécessaire. (cela s'appelle SPAN aujourd'hui).

Sur l'environnement de production l'approche est différente et plus contrainte, l'accès à la production est restreint au strict minimum et nécessitera une accréditation renforcée et adaptée et un poste 'rouge' (niveau habilitation SD).

Il est recommandé d'anticiper au plus les contraintes en activant rapidement l'environnement de production et de tester le plus tôt le déploiement en production pour que cette partie sujets soit vite maîtrisée par les équipes.

Nous réfléchissons à la mise en place d'un "proxy" en lecture seule (au sens une représentation distante) des éléments de vie du système tel que l'état des pipelines et l'observabilité au développeur situé côté primaire ainsi un flux d'échange et de collaboration automatisé, dans la limite où cela n'augmente pas la surface d'attaque.

Des tests de compatibilité avec d'autres solutions d'hébergement d'acteurs du cloud public sont menés, les premiers retours sont concluants.

Spécificités à prendre en compte sur la création des conteneurs

Kubernetes impose une rigueur un peu plus élevée à l'initialisation que d'autres solutions.

Les images de conteneurs sont conçues pour s'exécuter "rootless" sur des ports réseaux internes sont > 1024 et ne nécessite pas d'élévation de privilèges.

L'image est minimaliste, concentrée sur une fonction primaire ou atomique du système.

Elles sont conçues pour démarrer très rapidement avec le moins de dépendance possible (dans l'intérêt de la résilience et du bon fonctionnement de l'orchestrateur)

L'image n'embarque pas de services exposant des services réseau non essentiels à son fonctionnement tel que SSHD.

L'image ne se modifie pas elle-même au lancement. (ce qui souvent nécessite des élévations de privilèges)

L'écriture dans les pods n'est possible qu'avec le montage d'un volume.

(note : L'exécution du pod bloquée si ces règles ne sont pas respectées).

cf. <https://docs.openshift.com/>, et en particulier

https://docs.openshift.com/container-platform/4.15/openshift_images/create-images.html (toujours vérifier la dernière version)

Important : le lancement de pod en 'root' est interdit et bloqué au lancement. Ce point n'est pas modifiable. Ceci est un point d'attention majeur, la quasi-totalité des conteneurs à disposition sur les plateformes de partage de conteneurs ne sont pas rootless.

Les pods sont responsables de vérifier au lancement, si l'application est dans la condition initiale de 1er lancement, ou bien s'il faut initialiser ou modifier d'autres ressources telles qu'une base de données.

L'architecture de l'application, hors persistance de données, est conçue pour être complètement stateless, c'est-à-dire, sans aucune persistance de sessions, états et liens, les pods peuvent être basculés à la volée d'un nœud à l'autre sans préavis.

Spécificités à prendre en compte sur la topologie réseau et les ouvertures de flux

L'organisation de réseau est segmenté par type de service porté par le flux. (flux usagers, interdatacenters, interapplicatifs). Les flux réseau sont ouverts en lecture des manifests kubernetes / helms notamment Ingress et egress.

Spécificités à prendre en compte autour de la qualité et de la sécurité des applications

L'objectif d'ensemble est de s'assurer que le code produit est de qualité constante ou accrue, exempt de vulnérabilités algorithmiques ou importées néfastes.

Pour atteindre ces objectifs plusieurs mécanismes doivent être mis en place par l'équipe de développement intégrée :

- minimiser la portion spécifique de code développés en s'appuyant sur le catalogue de service proposé.
- mettre en place une couverture de test unitaire complète du back-end (et fournir les moyens de vérification automatisé à la chaîne secondaire)
- mener une analyse de code systématique le plus tôt possible (les langage et IDE modernes fournissent des fonctions de ce type)
- mener une analyse de CVE des dépendants importées

La chaîne secondaire reconstruit les images à partir des codes sources et procède aux mêmes tests avec des outils complémentaires. L'orchestration du pipeline secondaire est gérée par l'équipe et intègre les tests de vérification issue de la démarche d'homologation de l'application qui fixe les seuils de blocage de déploiement.

Les tests typiques consistent à vérifier la qualité du code (et la bonne couverture des tests), et le bon fonctionnement de l'application (non régression) et le scan de vulnérabilité.

L'équipe de développement reçoit via l'interface "shift left" une notification des rapports qui doit être intégrée au flux de travail pour correction.

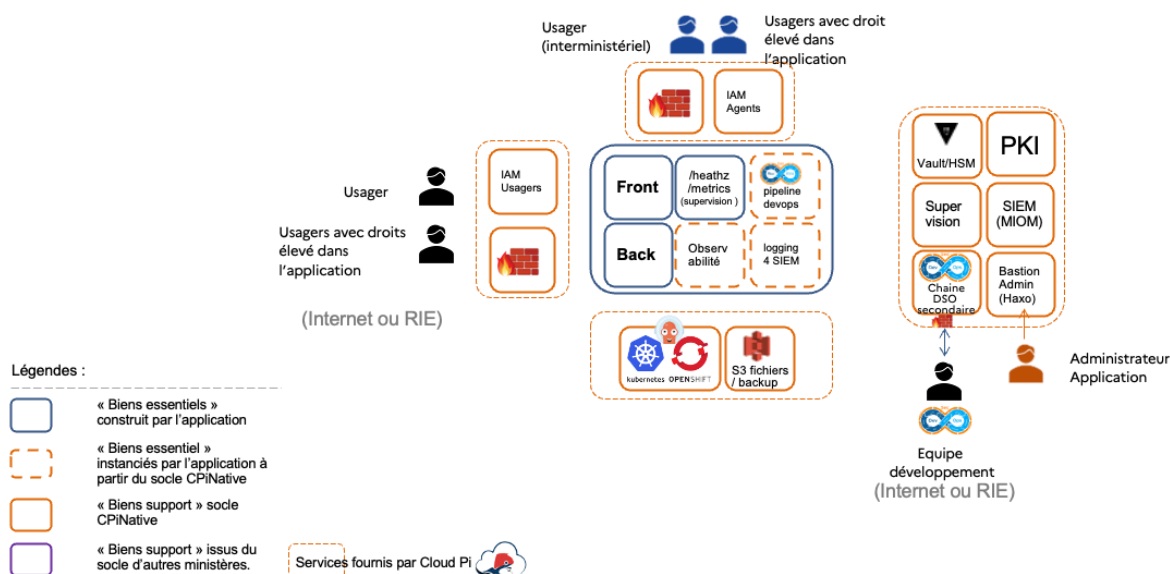
L'équipe intégrée est invitée à mener une activité constante de refactoring du code produit et du suivi des vulnérabilités de sécurité. La chaîne secondaire est susceptible de bloquer les déploiements si la qualité d'ensemble du code est en baisse ou que le scan fait remonter des Cves critiques. L'équipe est invitée à vérifier et prendre en compte également le résultat des scans de vulnérabilité après la dernière version stable déployée de l'application et de corriger les Cve critiques et importantes.

L'équipe prend en compte que si le suivi des plans d'action n'est pas mis en oeuvre et que de surcroît des vulnérabilités critiques sont détectées depuis le dernier déploiement stable, et que l'équipe projet n'a pas pris en compte les injonctions de correction, l'application sera susceptible d'être suspendue jusqu'à la remédiation pour garantir l'intégrité et la protection de ses données

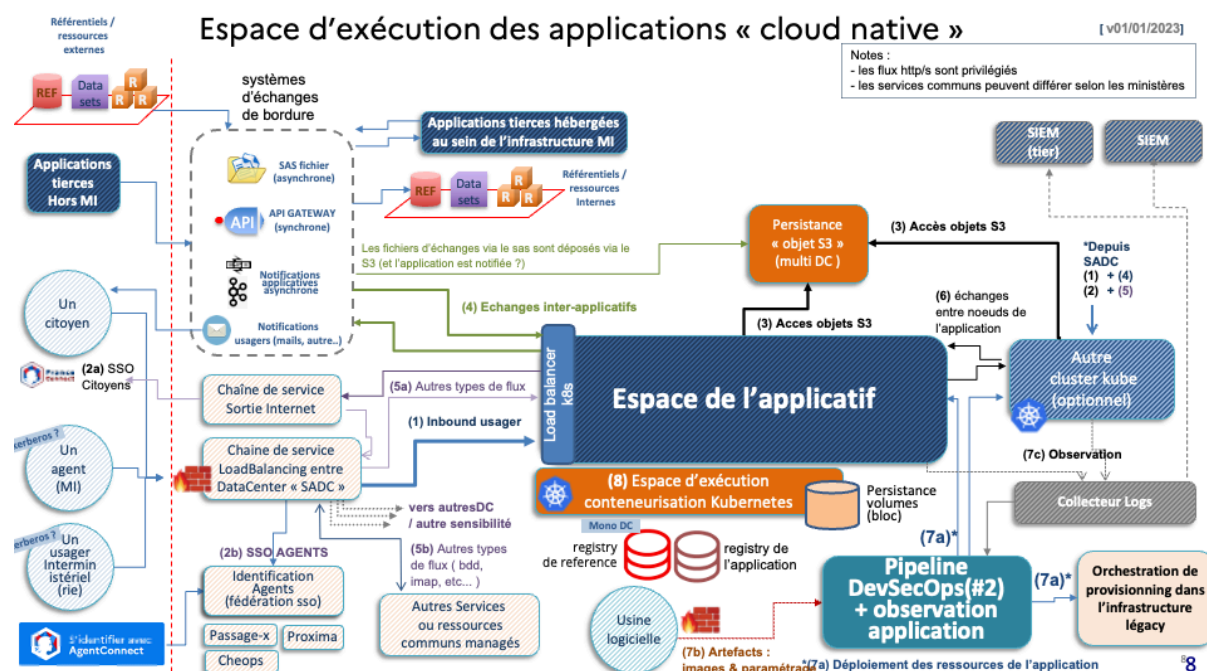
Modèle d'intégration d'une application dans le cadre Cloud Native

Le schéma ci-dessous précise le cadre général d'intégration d'une application. Des variantes sont possibles entre les ministères, elles sont précisées directement auprès des équipes concernées. Le respect de cadre permet à la direction d'application d'accéder à un socle de sécurité accélérant les homologations, l'ouverture automatique des segments réseau et l'homologation en continu.

Vision logique d'ensemble et services de socles et du modèle de sécurité



Architecture d'intégration réseau et flux typiques



- (1) Inbound usager : accès à l'application des usagers https / websockets (depuis RIE ou Internet)
- (2a) SSO Citoyens + (2b) SSO AGENT : authentification des usagers (OIDC / SAML V2)
- (3) Accés objets S3 : accès à la persistance objets de l'application
- (4) Echanges inter-applicatifs (bordure externe de l'application) : permet d'échange entre des applications de porteurs différentes, selon plusieurs modalités possibles : API restful synchrone, Asynchrone , fichiers
- (5a) Autres types de flux : autres types d'échange, sortie vers internet, vers d'autres zone d'hébergement, ou entre des zones de sensibilité différentes
- (5b) Flux d'accès à des services communs ou ressources managées (de protocoles plus variés)
- (6) échanges entre noeuds de l'application : permet la réplication de l'application entre 2 data centers au même niveau sensibilité de données
- (7a) Déploiement des ressources de l'application : gestionnaire & console DEVSECOPS / le pipeline interagit avec le/les clusters kubernetes et les gestionnaires d'infrastructures utilisés (ouverture de flux réseaux, etc...)
- (7b) Artefacts images & paramétrage : ensemble des ressources liées à une application ou communes (ex : sources d'images de référence)
- (7c) Observation : permet de collecter les données liées à l'usage pour la mise au point de l'application ou données de vie.
- (8) Kubernetes, sous la forme d'un ou plusieurs namespace(s) isolés ou couplés : fournis l'espace d'exécution de l'application et la gestion des volumes pour le stockage bloc.

4 - Présentation de l'offre interMinistérielle Cloud Pi Native et de ses évolutions pressenties

L'offre Cloud PI native répond aux exigences du CCT à travers un ensemble organisationnel et technique. Elle propose une offre Cloud régaliennne, souveraineté, sécurisée et isolée de toute problématique juridique extra-européenne.

La composition de l'offre est amenée à évoluer en termes de catalogue de service selon la demande et les financements disponibles, ces évolutions permettent la diminution de la quantité de code produit par les équipes de développement et l'accélération des performances, typiquement : fonctions as services, services managés, gpu;

Les offres d'hébergement compatibles avec les applications « Cloud Native » du ministère de l'intérieur sont :

- Hébergement de l'application sur les infrastructures internes infogérées ;
- Hébergement de l'application sur des infrastructures cloud externes ;
- Hébergement de l'application sur des infrastructures gérées par l'application.

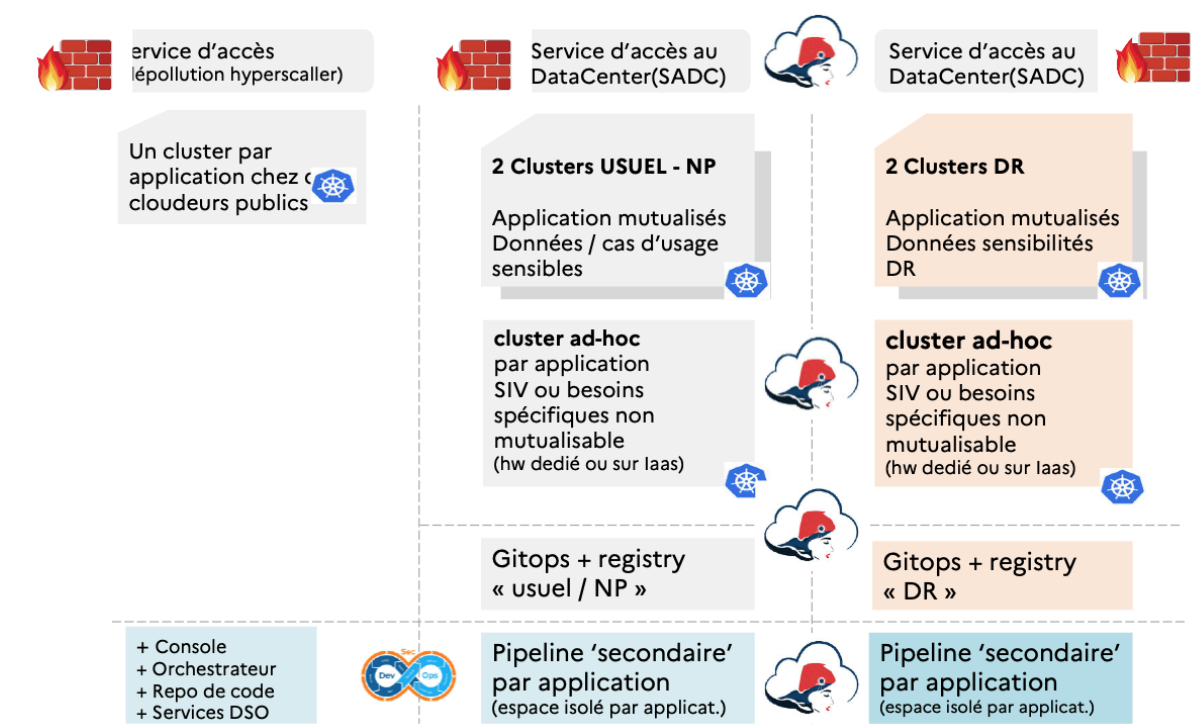
L'ensemble de l'administration technique de la plateforme et des infrastructures est automatisée, pilotée par le développeur/concepteur via l'orchestration DevSecOps avec mise en œuvre d'un principe dit *gitops*: la plateforme de production "tire" le déploiement.

Pour rappel, le développeur n'accède pas directement à l'environnement de production. Toute correction ou évolution suit le processus de déploiement automatisé passe via le principe "gitops" évoqué ci-dessus. Le développeur dispose d'un accès libre à ses environnements (via un poste bastion si l'environnement est situé côté ministériel) il dispose également d'un accès proxifié aux indicateurs de la production. La service team qui l'accompagne, avec les accréditations nécessaires, dispose quant à elle d'un poste dédié lui permettant d'accéder directement via un bastion aux services d'observabilité et faciliter le débogage en production. (note modalité en cours d'évaluation incrémentale)

L'ensemble du code source de l'offre Cloud PI Native et sa documentation sont disponibles sur demande en open-source sous la licence Apache2. Toute contribution est bienvenue.

La politique de segmentation d'hébergement est présentée ci-dessous :

Focus : Politique de segmentation de l'hébergement



Note : chaque région est autonome dans son fonctionnement. Seul le service de stockage objet de type S3 est accessible sur l'ensemble des régions ministérielles. (réplication en proximité dans le datacenter).

Les magasins de composants kubernetes et d'image de base

Associée avec l'offre Cloud pi Native, des magasins de composants Kubernetes sont mis à disposition incrémentalement selon les besoins des applications cela inclut celui de l'éditeur RedHat. Le développeur peut dès aujourd'hui s'appuyer sur un catalogue porté par l'Insee autour de son produit Onyxia : <https://github.com/InseeFrLab/helm-charts>

L'équipe produit est fortement invitée à l'utiliser ces composants courants et ne pas refabriquer une version dédiée dont le cycle ne sera pas en adéquation avec les besoins de réactivité en mcs.

A propos des images de base nues, il est recommandé d'utiliser les versions dites "LTS", certifiées et maintenues selon un processus qualité au sein des communautés ou éditeurs pour la construction des pods/conteneurs. Debian, Redhat, Ubuntu font partie des communautés les plus attentives.

Sur les besoins "classiques" de persistance : postgres, redis, mariadb, mongodb, elastic, etc... l'approche recommandée de s'appuyer sur les opérateurs kubernetes disponibles et

les objets statefullSet. Sur le cluster de production les opérateurs sont déployés par l'hébergeur car généralement ils requièrent les droits globaux.
L'équipe projet doit vérifier les versions disponibles lors de la conception de son projet.

6 - Introduction à l'offre de service du ministère

Les produits & services constituant l'offre Cloud Pi Native

Compute	Exposition réseau	Accélération de la construction applicative et autonomisation des équipes
<p>Namespace as a service au sein de clusters kubernetes/openshift mutualisés et managés *</p> <p>Mise à disposition cluster kubernetes/openshift dédiés et managés *</p> <p>Stockage objet S3 (repliqué entre sites)</p> <p>Pour besoin internes : environnements clouds avec GPU (OVH / Scaleway)</p> <p>En construction : namespace as a service sur cluster baremetal avec GPU managés</p>	<p>Exposition et protection de l'applications sur le RIE (Miom ou Interministériel)</p> <p>Exposition et protection des l'applications sur Internet.</p> <p>CDN / Anti-ddos renforcé (internet)</p> <p>Services pour le maintien en disponibilité de l'application.</p> <p>Supervision applicative H24 7/7 et exécution de fiches réflexes**</p> <p>Supervision de sécurité (MIOM)**</p> <p>Mise à disposition de 2 pc d'administration à distance**</p>	<p>Console cloud pi native / chaîne de construction applicative DevSecOps et monitoring de l'état de la menace CVE accessible depuis le RIE.</p> <p>Environnement d'accélération projet + chaîne primaire accessible de l'Internet (sur offre SNC)</p> <p>Accès à la base de connaissance, design de référence, charts helm & operators K8s avec composants usuels. (ex: stack d'observabilité)</p> <p>Cadre de cohérence technique Cloud Pi Native</p> <p>Accompagnement au succès, à l'autonomisation et à l'usage du cloud native des développeurs</p> <p>En construction : mise en place d'une certification pour les "champions / référents" de l'offre (Etatique et ESN)</p>

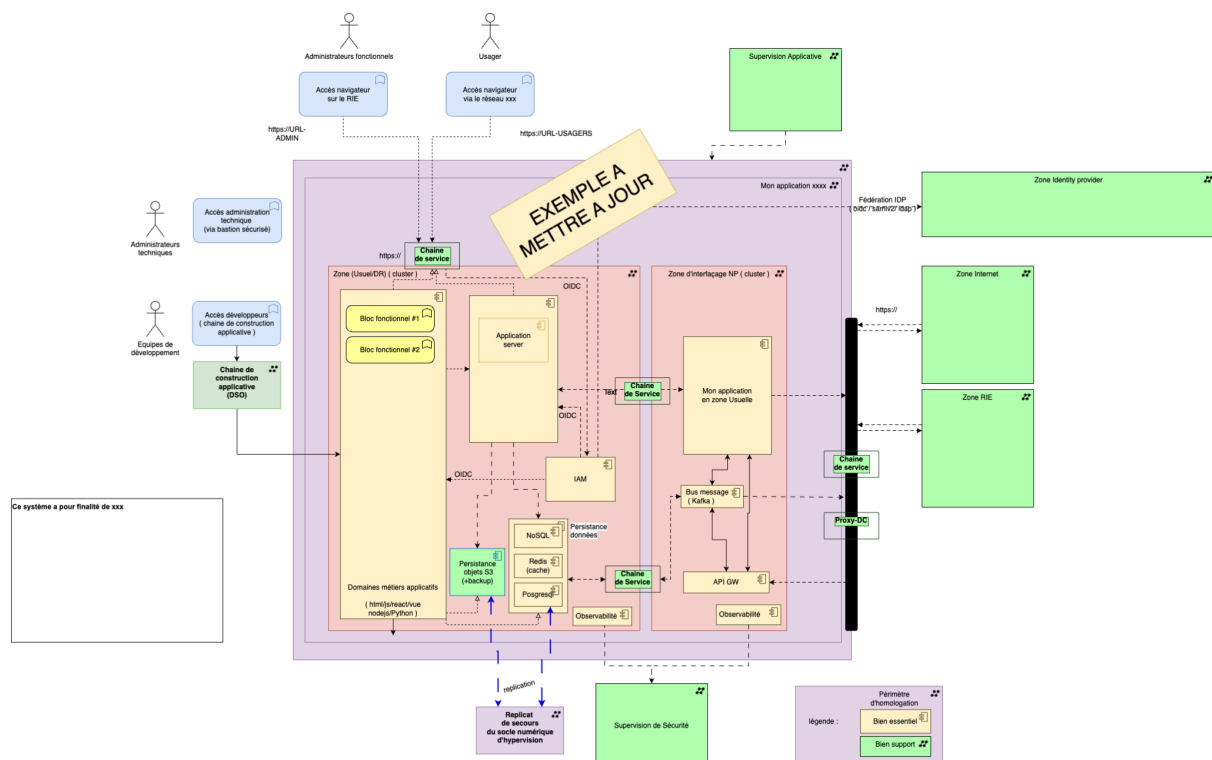
*On prem pour niveau Usuel et/ou DR
Offre par défaut : 2 environnements production + 1 hors production
** selon configuration opérationnelle de l'équipe pour gérer le run

Lien pour demander un hébergement :

<https://www.demarches-simplifiees.fr/commencer/cloud-pi-native>

Lien vers un exemple de DAG (exemple à mettre à jour pour le projet)

<https://app.diagrams.net/?url=https%3A%2F%2Fraw.githubusercontent.com%2Fcloud-pi-native%2Fcct-cloud-native%2Fmain%2FDAG-exemple.drawio>



5 - Référentiel d'exigences et modalités d'usage

Les exigences du CCT sont classées en 2 niveaux d'exigence (périmètre du Ministère de l'Intérieur) :

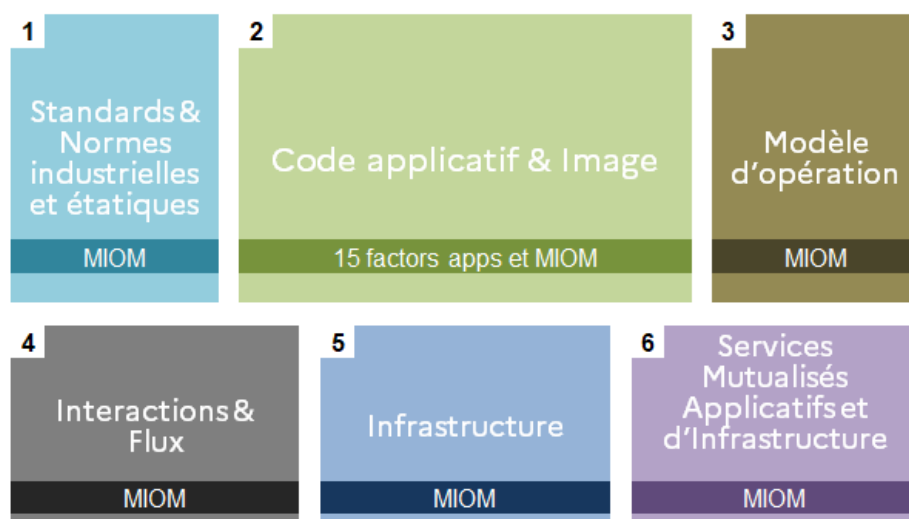
- Primordial : L'exigence est impérative et traitée administrativement.
- I – Important : Exigence prise en compte pour la notation technique de la solution

Précisions sur le cas de l'exclusion administrative (périmètre du Ministère de l'Intérieur) :

- La non-conformité au cadre de norme entraîne l'exclusion administrative lors du dépouillement et la mise en œuvre des actions de remédiation du marché lors de l'exécution du marché.
- La non-conformité aux exigences d'architecture entraîne l'impossibilité d'utilisation du socle de sécurité associé à l'offre Cloud Native

Par défaut les règles du CCT s'imposent. Elles peuvent être précisées dans le cas d'un appel d'offres dans le règlement de consultation pour le dépouillement des offres et dans le CCAP pour l'exécution du marché. Une demande de dérogation est possible. (cf paragraphe ad hoc)

Pour information les exigences sont organisées telles que décrites ci-dessous.



1. **Standards & Normes industrielles et étatiques** : ensemble des exigences relatives aux normes de niveau supérieur à respecter pour toute application étatique
2. **Code applicatif & Image** : exigences issues des “15 factors” pour garantir la conception d’une application “Cloud Native”, associées aux exigences minimales permettant de s’intégrer au contexte du Ministère de l’Intérieur
3. **Modèle d’opération** : voir le chapitre précédent
4. **Interactions & Flux** : exigences d’intégration et d’interaction inter-applicatives dans le contexte étatique et du Ministère de l’Intérieur
5. **Infrastructure** : exigences et prérequis concernant l’infrastructure sous-jacente (notamment Kubernetes)
6. **Services mutualisés Applicatifs et d’Infrastructure** : exigences d’intégration aux services centralisés du Ministère de l’Intérieur, permettant une homogénéisation de la production, un meilleur contrôle et une maîtrise de la dette technique

6 - Annexes

Les normes industrielles, institutionnelles applicables

La conception de système d'information dans le cadre de l'État est encadrée par un ensemble de recommandations ou règles à mettre en œuvre. Ces normes sont citées ci-dessous. Le lecteur est invité à vérifier qu'il dispose des versions les plus à jour.

Norme industrielle	Kubernetes : https://kubernetes.io/fr/ ArgoCD : https://argo-cd.readthedocs.io/en/stable/
Guides & outils pour la conception	DSFR : Design System FR. La charte internet de l'État (qui intègre le RGAA) https://www.systeme-de-design.gouv.fr/ Guide d'éco conception : https://ecoresponsable.numerique.gouv.fr/publications/referentiel-general-ecoconception/ Divers guides de conceptions logiciels: https://guides.etalab.gouv.fr https://catalogue.numerique.gouv.fr https://schema.gouv.fr https://code.gouv.fr
Cadres de pratiques de conception et de conduite de projet agile	https://www.numerique.gouv.fr/actualites/guide-pour-allier-agilite-et-securite-numeriques/
Logiciel libre	Socle InterMinistériel des Logiciels Libres (SILL) de par sa fonction de source pour le référentiel de produits du CCT Ministériel : https://sill.etalab.gouv.fr/fr/software
Normes interMinistérielles de conception de solutions	Doctrine cloud de l'état : https://www.legifrance.gouv.fr/circulaire/id/45205 Référentiel Général d'Accessibilité pour les Administrations : https://accessibilite.numerique.gouv.fr/
Référentiel Général de Sécurité, en association avec le règlement européen et l'EIDAS.	https://www.ssi.gouv.fr/entreprise/reglementation/confiance-numerique/liste-des-documents-constitutifs-du-rgs-v-2-0/

Référentiel Général de Gestion des Archives	https://www.ssi.gouv.fr/entreprise/reglementation/confiance-numerique/le-reglement-eidas/
Référentiel Général de Gestion des Archives	https://francearchives.fr/fr/circulaire/R2GA_2013_10
règlement européen sur la protection des données personnelles	https://www.cnil.fr/fr/reglement-europeen-protection-donnees

Liens vers autres contenus utiles(informatif)

<https://kubernetes.io/fr/>

<https://www.rancher.com/products/k3s>

<https://www.redhat.com/en/technologies/cloud-computing/openshift>

<https://argo-cd.readthedocs.io/en/stable/>

<https://www.redhat.com/en/topics/microservices/what-is-a-service-mesh>

<https://www.redhat.com/en/topics/devops/what-is-gitops>

<https://www.cloudcomputingpatterns.org/>

<https://12factor.net/fr/>

<https://tanzu.vmware.com/content/blog/beyond-the-twelve-factor-app>

<https://www.techworld-with-nana.com/devops-bootcamp>

<https://ecoresponsable.numerique.gouv.fr/publications/bonnes-pratiques/bonnes-pratiques/#bonnes-pratiques-services-numeriques>

La documentation sur le CloudPI :

<https://pi.interieur.rie.gouv.fr/> (accessible via le rie uniquement)

<https://cloud-pi-native.fr/>

<https://github.com/Cloud-pi-Native/>

Glossaire

Terme	Description / définition
Agile	Une méthode agile est une méthode de développement informatique permettant de concevoir des logiciels en impliquant au maximum le demandeur (client), ce qui permet une grande réactivité à ses demandes. Les méthodes agiles se veulent plus pragmatiques que les méthodes traditionnelles. Elles visent la satisfaction réelle du besoin du client, et non d'un contrat établi préalablement. La notion de méthode agile est née à travers un manifeste signé par 17 personnalités (parmi lesquelles Ward Cunningham, l'inventeur du Wiki), créateurs de méthodes ou dirigeants de sociétés. (Source : https://www.techno-science.net/definition/743.html)
ADR	Enregistrement des décisions d'architecture suivant le modèle MADR
API	Une API est le moyen « standard » désormais, par lequel est exposée une ressource, afin d'en permettre l'accès. Le qualificatif « RESTFULL » renvoie à la conformité de l'API au modèle « REST » qui est un modèle de représentation de l'URL de l'API. Une API est assortie d'un contrat de service qui décrit son fonctionnement. Ce contrat doit être conforme au standard « OPEN API V3 » et accessible aux développeurs.
API Management	Processus de gestion de la totalité du cycle de vie d'une API, de son idée jusqu'à son retrait de service. Décrit dans la Stratégie d'API. Une plateforme d'exposition d'API existe à la DNUM : api.minint.fr , ainsi qu'une autre, de niveau interMinistérielle : api.gouv.fr
BATN	Bureau Appui à la Transformation Numérique
CaaS	Les CaaS ou Containers en tant que Service (Containers as a Service en anglais) sont une catégorie de services Cloud permettant aux développeurs de logiciels de télécharger, d'organiser, d'exécuter, de gérer, de mettre à l'échelle et d'arrêter des containers en utilisant l'interface web ou l'API d'un fournisseur. Source : www.lebigdata.fr
CCAP	Cahier des Clauses Administratives Particulières
CCT	Cadre de Cohérence Technique du MIOM.
CCTP	Cahier des Clauses Techniques Particulières
CCU	Cadre Commun d'Urbanisation

CERFA	Centre d'Enregistrement et de Révision des Formulaires Administratifs
CHAP	Challenge Hash Authentication Protocol
CI/CD	<p>CI/CD signifie « Continuous Integration/Continuous Delivery », ou intégration continue/livraison continue ; c'est une méthode de mise en œuvre logicielle utilisée par les équipes de développement pour apporter des modifications de code plus fréquentes et plus fiables. Le CI/CD englobe deux ensembles de pratiques complémentaires, chacune reposant fortement sur l'automatisation.</p> <p>(Source : https://www.splunk.com/fr_fr/data-insider/what-is-ci-cd-pipeline.html#:~:text=CI%2FCD%20signifie%20%C2%AB%20Continuous%20Integration,plus%20fr%C3%A9quentes%20et%20plus%20fiables.)</p>
Client	Dans une architecture client-serveur, le client est celui qui est à l'initiative des requêtes faites au serveur
Cluster	Cluster (grappe) : plusieurs systèmes sont interconnectés soit pour augmenter la puissance de calcul (on parle alors de cluster de performance), soit pour offrir une tolérance de pannes accrue par la redondance des composants unitaires (on parle alors de cluster de haute disponibilité). Dans les deux cas, pour bénéficier de l'architecture en grappe, il faut que les applications aient été conçues en conséquence ou que le système d'exploitation, le compilateur et les logiciels sous-jacents (bases de données, middlewares, etc.) prennent en charge les fonctions adéquates de parallélisation des traitements ou de reprise sur incident.
Conteneur	<p>Les conteneurs sont des unités exécutables de logiciel dans lesquelles le code d'application est empaqueté, avec ses bibliothèques et ses dépendances, de manière commune, afin qu'il puisse être exécuté n'importe où, que ce soit sur un ordinateur de bureau, dans un système informatique traditionnel ou dans le cloud.</p> <p>(Source : https://www.ibm.com/fr-fr/cloud/learn/containers#:~:text=Les%20conteneurs%20sont%20des%20unit%C3%A9s,traditionnel%20ou%20dans%20le%20cloud.)</p>
CVE	<p>Common Vulnerabilities and Exposures.</p> <p>Common Vulnerabilities and Exposures ou CVE est un dictionnaire des informations publiques relatives aux vulnérabilités de sécurité. Le dictionnaire est maintenu par l'organisme MITRE, soutenu par le département de la Sécurité intérieure des États-Unis.</p>

DevSecOps (DSO)	Le DevSecOps inclut la composante sécurité (security) dans l'approche DevOps, qui lie elle-même le développement (développement) et l'exploitation (opérations). Le DevSecOps s'instancie par des mesures de formation, organisationnelles et des ajouts de système de vérification à chaque fois que l'application est construite.
DINUM	Direction Interministérielle du Numérique
DITP	Direction InterMinistérielle de la Transformation Publique (placée sous l'autorité du ministre de l'Action et des Comptes publics, chargée de la réforme de l'État).
DNUM	Direction du Numérique
DSFR	Design System FR. La charte internet de l'État.
ENT(A)	Environnement Numérique de Travail (de l'Agent)
FIP	Factory Instrumental Protocol (Flux d'Information Processus). Actuellement FIP est une norme française (NF C46 601 à NF C46 607) et une norme européenne (EN 50170-3). La promotion et une part d'assistance technique de ce réseau sont effectuées par l'organisation WorldFIP dont le siège se trouve en France. La cible privilégiée de WorldFIP est l'interconnexion de capteurs, actionneurs et automates dans les systèmes automatisés. Comme la quasi-totalité des réseaux de terrain WorldFIP a une structure en trois couches. (https://www.i3s.unice.fr/~map/Cours/LPIREEL/COURS3FIP.pdf)
Gitops	L'approche GitOps repose sur l'utilisation de référentiels Git comme unique source de vérité pour distribuer l'infrastructure en tant que code. Le code envoyé vérifie le processus d'intégration continue, tandis que le processus de distribution continue vérifie et applique les exigences relatives à certains aspects, comme la sécurité, l'infrastructure en tant que code (IaC), ou toute autre limite fixée pour le framework d'application. Toutes les modifications apportées au code font l'objet d'un suivi, ce qui facilite les mises à jour et le contrôle de versions en cas de restauration. (Source : https://www.redhat.com/fr/topics/devops/what-is-gitops#:~:text=Le%20GitOps%20peut%20%C3%AAtre%20consid%C3%A9r%C3%A9,les%20configurations%20de%20l'infrastructure.)
Hébergement	L'hébergement, dans son sens le plus générique, est un service par lequel des ressources de stockage et de calcul sont fournies à une personne ou à une organisation pour l'hébergement et la maintenance d'un ou plusieurs sites Web et services connexes.

	(Source : https://definir-tech.com/hebergement/)
IAM	Gestion des identités et des accès (Identity and Access Management)
Java	Java est un langage de programmation et une plate-forme de calcul lancé par Sun Microsystems en 1995. (Source : https://www.java.com/fr/download/help/whatis_java.html)
Kubernetes	Kubernetes est une plate-forme open-source extensible et portable pour la gestion de charges de travail (workloads) et de services conteneurisés. Elle favorise à la fois l'écriture de configuration déclarative (declarative configuration) et l'automatisation. C'est un large écosystème en rapide expansion. (Source : https://kubernetes.io/fr/docs/concepts/overview/what-is-kubernetes/#:~:text=Kubernetes%20est%20une%20plate%2Dforme,large%20%C3%A9cosyst%C3%A8me%20en%20rapide%20expansion.)
Logiciel Libre	Un logiciel libre est un logiciel dont la licence dite « libre » donne à chacun le droit d'utiliser, d'étudier, de modifier, de dupliquer, de donner et de vendre ledit logiciel sans contrepartie. La notion de logiciel libre ne doit se confondre ni avec celle de logiciel gratuit (freeware ou graticiels) ni avec celle de shareware (partagiciels). De même, les libertés octroyées par la licence d'un logiciel libre sont plus étendues que le simple accès au code source, ce qu'on appelle parfois logiciel « à sources ouvertes ».
MCO	Maintien en Condition Opérationnelle
MI	Ministère de l'intérieur
NFR	Non Functional Requirement. Exigence non fonctionnelle que doit embarquer l'équipe. Généralement des exigences d'architectures, de sécurité, etc...
MIOM	Ministère de l'intérieur et des Outre-Mer
Node	NodeJS est une plateforme construite sur le moteur JavaScript V8 de Chrome qui permet de développer des applications en utilisant du JavaScript. Il se distingue des autres plateformes grâce à une approche non bloquante permettant d'effectuer des entrées/sorties (I/O) de manière asynchrone. (Source : https://grafikart.fr/tutoriels/nodejs-intro-792)
Open API	Une API ouverte, parfois appelée API publique, est une interface de programmation d'application (Application Programming Interface) qui permet au développeur d'accéder à une application logicielle propriétaire par voie de programmation. (https://www.lemagit.fr/definition/API-ouverte#:~:text=Une%20API%20o

	ouverte%2C%20parfois%20appel%C3%A9e,propri%C3%A9taire%20par%20voie%20de%20programmation.)
Openshift	OpenShift est un service de plate-forme en tant que service de la société Red Hat qui permet de déployer des projets dans des containers. Pour ce faire, OpenShift utilise les technologies Docker et Kubernetes. (https://fr.wikipedia.org/wiki/OpenShift)
Open Source	La désignation Open Source (« source ouverte » en français) s'applique aux logiciels dont la licence respecte des critères précisément établis par l'Open Source Initiative, c'est-à-dire la possibilité de libre redistribution, d'accès au code source, et de travaux dérivés. Les logiciels Open Source et les logiciels libres désignent les mêmes logiciels, ceux dont la licence est reconnue libre par l'Open Source Initiative ou la Free Software Foundation. Le terme « Open Source » est en concurrence avec le terme « logiciel libre » (Free Software) recommandé par la FSF. Le terme Freeware (graticiel) désigne des logiciels gratuits qui ne sont ni nécessairement ouverts, ni libres. (Source : https://fr.wikipedia.org/wiki/Open_source)
Objet Métiers	Représentation schématique d'un concept métier, instanciée sous la forme d'une donnée gérée par un système d'information maître et de l'organisation qui à la charge de maintenir cette donnée à jour et exempte d'erreur.
Pods	Un pod représente une collection de conteneurs d'applications et de volumes fonctionnant dans le même environnement d'exécution. Les pods, et non les conteneurs, sont les plus petits artefacts déployables dans un cluster kubernetes. Les applications s'exécutant dans le même pod partagent la même adresse IP et le même espace de nom réseau. Source : Kubernetes maîtrisez l'orchestrateur des infrastructures du futur
PP	
Proxy	Un serveur proxy est une sorte de pont qui vous relie au reste d'Internet. Normalement, lorsque vous naviguez sur Internet, vous vous connectez directement au site Web qui vous intéresse. Un proxy établit à votre place la communication avec le site Web. (Source : https://www.avast.com/fr-fr/c-what-is-a-proxy-server#:~:text=Un%20serveur%20proxy%20est%20une,communication%20avec%20le%20site%20Web.)

Python	<p>Le langage Python est un langage de programmation open source multi-plateformes et orienté objet. Grâce à des bibliothèques spécialisées, Python s'utilise pour de nombreuses situations comme le développement logiciel, l'analyse de données, ou la gestion d'infrastructures.</p> <p>(Source : https://www.futura-sciences.com/tech/definitions/informatique-python-19349/)</p>
RACI	<p>L'acronyme RACI (responsable, accountable, consulted et informed) ou RAM (responsibility assignment matrix) désigne dans le domaine du management une matrice des responsabilités. Elle indique les rôles et les responsabilités des intervenants au sein de chaque processus et activité. La matrice RACI donne une vision simple et claire de qui fait quoi dans le projet, en permettant d'éviter une redondance de rôles ou une dilution des responsabilités. (Source : https://fr.wikipedia.org/wiki/RACI)</p>
R2GA	Référentiel Général de Gestion des Archives sur le portail national des archives
Restful API	<p>Une API REST (également appelée API RESTful) est une interface de programmation d'application (API ou API web) qui respecte les contraintes du style d'architecture REST et permet d'interagir avec les services web RESTful. REST (Representational State Transfer).</p> <p>(Source : https://www.redhat.com/fr/topics/api/what-is-a-rest-api#:~:text=Une%20API%20REST%20(%C3%A9galement%20appel%C3%A9e,avec%20les%20services%20web%20RESTful.))</p>
RGAA	Référentiel Général d'Accessibilité pour les Administrations
RGI	Référentiel Général d'Interopérabilité
RGPD	<p>Le sigle RGPD signifie « Règlement Général sur la Protection des Données ». Le RGPD est le règlement européen sur la protection des données, articles de lois sur l'accessibilité des données à des fins de recherche</p>
RGS	Référentiel Général de Sécurité, en association avec le règlement européen eIDAS

Rootless	<p>Les conteneurs rootless font référence à la possibilité pour un utilisateur non privilégié de créer, d'exécuter et de gérer des conteneurs. Ce terme inclut également la variété d'outils autour des conteneurs qui peuvent également être exécutés en tant qu'utilisateur non privilégié.</p> <p>"Utilisateur non privilégié" dans ce contexte fait référence à un utilisateur qui n'a aucun droit d'administrateur et qui n'est "pas dans les bonnes grâces de l'administrateur" (en d'autres termes, il n'a pas la possibilité de demander que plus de privilèges lui soient accordés à eux, ou pour les progiciels à installer).</p> <p>(Source : https://rootlesscontaine.rs/)</p>
SADC (CDS)	Service d'accès au data center. Correspond à la chaîne de protection et de pollution des des flux entrants vers le data center. Cela peut être appelé parfois "chaîne de service" (CDS).
SDID (ex SDITN)	Sous-Direction à l'Innovation et Données de la Direction de la transformation numérique du MIOM. (ex: sous direction de l'innovation et de la transformation numérique)
Shift left	<p>Le Shift Left décrit un principe qui consiste à rendre les flux de travail des entreprises plus efficaces, grâce à des tests et avec des suivis précoces. Cette méthode vous permet de transmettre les connaissances de votre service d'assistance rapidement et facilement à tous les employés de votre entreprise. (Source : https://freshservice.com/fr/shift-left-blog/#:~:text=Le%20Shift%20Left%20d%C3%A9crit%20un,les%20employ%C3%A9s%20de%20votre%20entreprise.)</p>
SI	<p>Selon la définition restreinte donnée par Joël de Rosnay, « Un système est un ensemble d'éléments en interaction dynamique, organisés en fonction d'un but ». Le système d'information n'échappe pas à cette définition. Il est un ensemble dont les éléments sont les constituantes de toute organisation (entreprise, administration, association, groupement, ...). Ces éléments sont de plusieurs natures : organisationnelle, informationnelle, métier, technique, technologique. Tous ces éléments forment un tout (plus ou moins cohérent) et participent à la réussite de l'organisation dans son objectif.</p>
SIC	Systèmes d'Information et de Communication
SIEM	Security information and event management. Dans le domaine de la sécurité informatique, les produits et service logiciels de SIEM combinent la gestion des informations de sécurité (SIM) et la gestion des événements de sécurité (SEM) pour fournir des alertes en temps réel.

	Cet outillage est mis en œuvre un centre de supervision de la sécurité (Security Office Center : SOC)
SILL	Socle interministériel de logiciels libres. Il regroupe l'ensemble des logiciels libres préconisés au sein des ministères. Il est alimenté par des agents publics volontaires Ministériels, sous le contrôle de la DINUM
SPOC	Single Point Of Contention : littéralement « point individuel de contention ». Consiste dans un système à identifier, pour les différents composants (matériels et/ou logiciels), l'existence de points constituant un goulet d'étranglement. Ce composant est alors considéré comme un SPOC pour le système.
SPOF	Single Point Of Failure : littéralement « point individuel de défaillance ». Consiste dans un système à identifier, pour les différents composants (matériels et/ou logiciels), l'existence de points de défaillance pouvant générer un dysfonctionnement du système de par l'impossibilité de redonder ce composant ou de par le choix de ne pas le redonder. Ce composant est alors considéré comme un SPOF pour le système.
ST(SI) ²	Service des Technologies et des Systèmes d'Information de la Sécurité intérieure. Le ST(SI) ² est chargé de concevoir, de piloter et de conduire les projets liés aux systèmes d'information, de communication et de commandement pour l'ensemble des policiers et des gendarmes. Il contribue à la définition de l'action, de la stratégie et de la politique de sécurité du ministère de l'intérieur en matière de système d'information et de communication. Il coordonne les services SIC de proximité de la police et de la gendarmerie. Il anime la politique d'innovation technologique du ministère dans ce domaine. (Source : https://fr.linkedin.com/company/stsisi)
Swagger	<p>Une définition Swagger spécifie un ensemble de métadonnées qui décrivent une API REST.</p> <p>Si vous avez un fichier Swagger définissant une API REST, vous pouvez l'ajouter à votre projet en tant que source de synchronisation externe. Cette source peut être synchronisée avec le projet.</p> <p>(Source : https://www.ibm.com/docs/fr/rtw/9.0.1?topic=testing-swagger-definitions)</p>
TCP	TCP (Transmission Control Program) est un protocole permettant l'ouverture de circuits virtuels entre applications

VM	<p>Une machine virtuelle, ou « virtual machine », est « le client » créé dans un environnement informatique, « l'hôte ». Plusieurs machines virtuelles peuvent coexister sur un seul hôte. Les principaux fichiers qui constituent une machine virtuelle sont un fichier journal, un fichier de paramètres de RAM non volatile, un fichier de disque virtuel et un fichier de configuration. (Source : https://www.vmware.com/fr/topics/glossary/content/virtual-machine.html)</p>
VPN	<p>Virtual Private Network, réseau privé virtuel (RPV) : Le principe du RPV consiste à créer un réseau privé au sein d'un réseau public. Cette démarche existe depuis longtemps : les opérateurs s'en servent pour gérer les lignes privées de leurs clients au sein des mêmes « tuyaux ». Aujourd'hui, on parle surtout de réseaux privés virtuels sur Internet. Les RPV mettent en œuvre des mécanismes de contrôle d'accès (authentification des utilisateurs) et assurent la confidentialité des données (cryptographie). Le terme de réseau privé virtuel s'applique aussi au réseau téléphonique : les opérateurs font ainsi transiter sur le réseau public des services évolués de téléphonie jusque-là cantonnés au réseau privé de l'entreprise appel en numérotant uniquement l'extension, renvoi d'appel, conversation à plusieurs, etc. Cette technologie s'étend aussi aux mobiles.</p>
Windows Server	<p>Windows Server est un système multi-tâches, multi-utilisateurs qui dans ses fonctionnalités peut se comparer au système UNIX/Linux. Il présente l'avantage que certains logiciels soient moins chers que leur équivalent fonctionnant sous UNIX, et plus rarement, Linux. Par ailleurs, la quasi-totalité des éditeurs propose des versions de leurs produits pouvant tourner sur serveur Windows.</p>

--- fin du document ---

7 - Référentiel d'exigences applicables au CCT Cloud Pi Native

Note: le terme développeur est générique et fait référence à l'individu ou l'organisation pluridisciplinaire qui est chargée de produire et maintenir : la base de code, le corpus de tests et les fichiers de description d'infrastructure et les documentation technique et usager.

Il est responsable de l'adéquation et de la qualité de la solution au besoin des usagers en collaborant de manières étendues avec les autres acteurs impliqués.

ID	Type	Exigence	Catégorisation
Version		<p>Version : liste des exigences, mise à jour au 27.5.2024</p> <p>change management :</p> <p>liste dé-doublonnée, triée, renumérotée</p> <p>ajout exigence MCS et variabilité du déploiement</p> <p>Précision sur les exigences de la construction des images "from"</p> <p>Précision sur les exigences liés à la résilience intra cluster</p> <p>Ajout respect des 12 factors</p> <p>Exigence P : primordiale, le non respect entraîne un rejet administratif de l'offre lors d'un appel d'offre et un plan de remédiation obligatoire lors de l'exécution du marché</p> <p>Exigence I : importante, permet de maximaliser la qualité de la solution</p>	-

EXI-G-1	I	Dans le cadre d'un appel d'offre, en cas d'incohérence entre les documents et hors mentions explicites, le cct et la liste des exigences associées sont de niveaux supérieures.	hiérarchie des normes
EXI-G-2	I	Le respects des exigences du CCT sont limitées à la durée du marché pour un opérateur économique, mais permanent pour la direction d'application métier.	applicabilité
EXI-G-3	I	<p>Respect des standards et des normes applicables industrielles, européennes et étatiques, pour la conception de solutions numériques hébergées dans le cloud native (kubernetes), Design Système de l'État, RGAA, RGS, RGI, doctrine Cloud au centre. cf paragraphe du volet de CCT :</p> <p>Les normes industrielles, institutionnelles applicables</p>	Cadre de normes

EXI-G-4	I	Respect du guide d'éco-conception. La conception frugale vis à vis des ressources d'infrastructures consommées et l'impact vis-à-vis du terminal de l'utilisateur.	Qualité de la solution
EXI-G-5	P	Choix d'un hébergement adapté à la nature des données manipulées (usuel / DR) et selon le cadre légal adapté. Cloud Pi (on-prem), Cloud public, Infrastructure dédiée ou mixte.	Infrastructure
EXI-G-6	P	Si l'application est exposée sur internet un service de CDN / anti-ddos externalisé sera mis en place.	Infrastructure

EXI-ORG-1	P	Conformité au modèle de responsabilité Cloud Pi Native : les développeurs/concepteurs sont responsables de la partie développement, du maintien en continu d'une qualité constante de la solution et l'absence de vulnérabilités exploitables notamment avant toute mise en production.	Modèle d'Opération
EXI-ORG-2	I	Les équipes contribuent à la remontée et à l'enrichissement de la chaîne l'offre Cloud Native. Signale les éventuels défauts ou besoin d'évolution, il soumet le cas échéant une rectification ou une évolution sur le dépôt de code après l'avoir testé	Modèle d'Opération
EXI-ORG-3	P	Conformité au modèle de responsabilité "you built it - you run it - you secure it - you support it" de l'offre cloud pi native. Les développeurs/concepteurs sont responsables d'organiser le bon fonctionnement de l'application en production et du soutien aux usagers. Un modèle de collaboration est établi dans la convention d'utilisation de l'offre avec l'hébergeur avec le modèle de responsabilité de chacun des acteurs. Cela peut inclure la mise en place d'autres conventions avec d'autres service tels que la supervision applicative et le SOC ministériel. (les norme d'interfaces sont à demander lors de la contractualisation de l'hébergement)	Modèle d'Opération
EXI-ORG-17	I	L'équipe est formée aux bonnes pratiques des technologies Cloud Native (kubernetes), à la construction et au maintien en qualité d'une base de code, à l'agilité et au devsecops, à l'intégration continue. (fournir les certifications en preuve)	Compétence
EXI-ORG-16	I	L'équipe s'appuie de préférence sur un ou plusieurs coachs & experts certifiés Cloud Pi Native pour l'initialisation du programme et une revue régulière des pratiques	Compétence

EXI-ORG-6	I	Intégration technique de l'usine logicielle primaire (responsabilité du développeur) à la chaîne DevSecOps, maintien par le développeur en condition de disponibilité et de sécurité du point de vérité du logiciel et du code d'infrastructure.	Modèle d'Opération
EXI-ORG-7	P	Le développeur est responsable du pipeline de la chaîne secondaire de construction applicative. (pipeline gitlab) Il s'appuie sur les outils et les exemples proposés par la documentation. Il met en œuvre les vérifications nécessaires portant sur la qualité du code, les vérifications fonctionnelles automatiques, les mesures spécifiques de sécurité issues de la démarche d'homologation.	Modèle d'Opération
EXI-ORG-8	P	Seul le mode de déploiement en GitOps est mis en œuvre. Le développeur versionne son code applicatif et d'infrastructure. Les déploiements par manifest "à la main" ne sont pas autorisés.	Modèle d'Opération
EXI-ORG-9	I	Le développeur met en place un principe de livraison continue et par lot de taille réduite.	Modèle d'Opération
EXI-ORG-10	I	Couverture et capitalisation des tests, respect des métiers (non régression): La couverture des tests unitaires du code visé pour le back-end doit être de 100% du code. La mise en place d'une approche test drive développement est fortement recommandée.	Code Applicatif & Image
EXI-ORG-11	I	La couverture fonctionnelle des tests du front end permet de s'assurer de non-régression majeure. Elle permet de vérifier des séquences d'usage principale, (login, actes métiers fréquents, etc...)	Qualité de la solution
EXI-ORG-12	I	Les bugs détectés donnent lieu d'abord à l'implémentation d'un test automatique avant l'écriture du correctif. Cela permet d'augmenter la base des cas de tests et de renforcer la performance des tests de non régression.	Qualité de la solution

EXI-ORG-13	P	<p>L'équipe produit intégrée s'engage à monitorer les nouvelles vulnérabilités qui seraient susceptibles d'être découvertes et d'y remédier après le dernier déploiement stable. Elle met en œuvre les plans d'exploitation de sécurité et de maintien en condition de sécurité décrite notamment dans le dossier d'homologation.</p> <p>L'équipe prend en compte que la chaîne de construction applicative est susceptible de bloquer les déploiements si la qualité d'ensemble du code est en décroissance ou si des vulnérabilités critiques sont détectées</p> <p>L'équipe prend en compte que si le suivi des plans d'action n'est pas mis en œuvre et que de surcroît des vulnérabilités critiques sont détectées depuis le dernier déploiement stable et que l'équipe projet n'a pas pris en compte les injonctions de correction, l'application sera susceptible d'être suspendu jusqu'à la remédiation pour garantir l'intégrité et la protection de ses données. Pour améliorer le "signal sur bruit" sur le plan des cve se référer à l'exigence sur la construction des images.</p>	Modèle d'Opération
EXI-ORG-14	P	<p>L'équipe projet ou équipe produit met en œuvre une activité continue de refactoring du code produit et réserve une capacité d'environ de 20% de la capacité d'ensemble avec des principes organisationnels tel que le peer review, un processus de maîtrise des changements (ex: lors des pull requests).</p> <p>La qualité du code mesurée ne peut pas être décroissante dans le temps.</p> <p>L'équipe fournit les preuves que des tests de sécurité, de qualité, de robustesse des algorithmes ont été mis en œuvre, et qu'ils n'ont pas remonté de vulnérabilités ou d'erreurs majeures.</p> <p>Ces preuves seront annexées au dossier d'homologation, et les actions suivies au sein du plan d'action.</p>	Qualité de la solution
EXI-ORG-15	I	<p>Dans l'objectif que le développeur soit notifié au plus tôt de défauts en modalité "shift-left", alerte de supervision l'équipe de développement mettra à disposition un point d'interface automatisé compatible avec les capacités d'alerting de la solution kibaba.</p>	Modèle d'Opération
EXI-ORG-16	I	<p>Sont proposés des intégrations avec Mattermost et Tchap. S'il souhaite aller loin dans l'automatisation par exemple pour l'intégration dans les flux de travail, le développeur peut également proposer la mise en place d'un WebHook.</p>	Modèle d'Opération
EXI-ORG-17	P	<p>L'orchestration de la sauvegarde est de la responsabilité de l'équipe de développement application</p>	Modèle d'Opération
EXI-ORG-18	I	<p>L'équipe projet ou équipe produit teste régulièrement sa capacité à restaurer l'ensemble de l'architecture et des données de l'application avec un scénario de test connu à l'avance</p>	Modèle d'Opération

EXI-ARCH-1	I	<p>Choix du langage et du framework :</p> <p>Sauf mention explicite dans le cahier des charges, il est recommandé d'utiliser le ou les langages de programmation notés au sein de la liste des langages suivants :</p> <ul style="list-style-type: none"> - Typescript / nodejs, python, Java(attention à la licence Oracle, privilégier OpenJDK) - Ruby (normalisation ISO, framework de test et modules), Php (point d'attention sur le déploiement dans une image conteneurisée car nécessite un server web) - Go (pour l'automatisation de service cloud fortement supporté par Google) - RUST pour le typage / performance mémoire - Scala pour la performance en datascience / bigdata. - Variante de C <p>Lors du choix de langage, il est conseillé de prendre en compte la capacité de maintien de la base de code, le maintien des compétences nécessaires sur le temps long > 5 ans.</p> <p>Le choix et la pertinence du ou des frameworks utilisées doit prendre en compte la couverture d'utilisation réelle du framework.</p> <p>Typiquement un framework entraine un couplage fort de la base de code avec celui-ci et entrainer un coût non négligeable de refactoring à chaque montée de version.</p> <p>Si un framework est faiblement utilisé il sera préférable d'isoler la fonction utilisée et de ne pas dépendre du framework.</p> <p>Sont préférés les écosystèmes de langages qui limitent les erreurs de programmation (typage et framework de test) avec des IDE et scan de qualité. supportés par une communauté large et active dans le domaine d'application visé.</p> <p>Pensez à vous appuyer sur des libraires ou des domaine de langage spécifique (DSL en anglais) qui permettent la prise en compte de fonctionnalités complexes uniquement par du paramétrage de fichier et non par de l'algorithmie de codage pur.</p> <p>Dans le cas de conception de système à haute intensité transactionnelle, des langages compilés ayant une gestion efficace de la mémoire seront préférés. (typiquement Rust et C)</p> <p>Lors de la candidature à un appel d'offre, la motivation d'utilisation de tel ou tel langage sera</p>

Langages

		argumentée.	
EXI-ARCH-2	P	<p>L'architecture interne de l'application et l'organisation sont conçus pour s'assurer que la durée maximale d'interruption acceptable (DMIA) désirée par le métier est atteinte, en combinant des mesures telles qu'un déploiement sur 2 datacenters (résilience assurée au niveau applicatif), une architecture modulaire avec des principes de "graceful degradation" et les mesures organisationnelles. Pour la résilience intra datacenter, mettre des règles d'anti-afinités lors de la constitution des manifests / charts afin que les composants de la solutions sont bien distribués sur des AZ différentes. (lorsque le cluster commandé le permet)</p> <p>Vérifier que les composants de résilience de la solutions sont bien distribués au sein du cluster étendu sur des AZ différentes lors des déploiements.</p> <p>L'équipe doit confirmer que ses processus et les mesures en place atteignent l'objectif visé par des exercices réguliers.</p> <p>*notes:</p> <ul style="list-style-type: none"> - lors de la contractualisation avec l'hébergeur par défaut seront proposés 2 environnements de production, 1 environnement hors production et 2 buckets S3 (prod/hors production) - la résilience applicatif est assurée au niveau applicatif actif / passif ou actif-actif selon le besoin métiers - utilisation de composant tel que https://cloudnative-pg.io/ permettent facilement la mise en place de HA pour la persistance de donnée - utilisation du S3 permet de mettre en place une disponibilité entre datacenter - graceful dégradation est un mécanisme qui permet à un module, lorsqu'une dépendance n'est pas accessible de fournir une expérience dégradée mais restant acceptable à un usager, généralement une valeur par défaut basée sur un historique ou bien un affichage dégradé pour un navigateur ancien. 	Architecture
EXI-ARCH-3	I	<p>L'architecture de la solution est modulaire, le couplage organisationnel et technique est lâche entre les composants par l'usage d'une interface normée et documentée (ex: API).</p> <p>Les composants peuvent être déployés indépendamment les uns des autres sans nécessiter une coordination entre les équipes. Les services de haut niveau ne dépendent pas de l'interface offerte de composants de niveau inférieur (principe d'inversion de dépendances).</p>	Architecture
EXI-ARCH-4	P	<p>Le point de présence et de vérité de la sauvegarde est assuré par un service de type S3 indépendant et résilient de l'architecte de production</p>	Architecture

EXI-ARCH-5	P	<p>Pour l'usage de composants d'infrastructure logicielle open source (ex: nginx, keycloak, airflow, https://quarkus.io/, etc...) il est recommandé de s'appuyer sur un repository communautaire de template ayant un cycle de mco/mcs fréquents et les charts helms et des images doivent être compilées rootless et de préférence multiarchitectures (cela augmente la surface de test). Vérifiez systématiquement la date de dernière mise à jour et des vulnérabilités portées et l'accès au code source.</p> <p>Quelques principes fortement recommandés :</p> <ul style="list-style-type: none"> - La configuration (valeur & secret) doivent être traités dans des fichiers à part, gérés en version tel que le mécanisme values.yaml des charts helm. <p>Une façon pratique (et testée) consiste à placer les secrets et valeur de configuration liés à l'environnement directement au sein du namespace pour qui surcharge les valeurs par défauts, via une configMap ou un secret.</p> <p>Un exemple illustratif sur le service nginx https://github.com/bitnami/charts/blob/main/bitnami/nginx/values.yaml :</p> <p>Les paramètres de nginx du "serveurblock" peuvent être surchargés au runtime par un volume monté au lancement du pod. (deploymentset)</p> <ul style="list-style-type: none"> - name: nginx-server-block <p>mountPath: /opt/bitnami/nginx/conf/server_blocks</p> <p>Tip: comme les pods tournent en readonly fs par défaut, pensez à monter explicitement un volume /tmp emptydir :</p> <ul style="list-style-type: none"> - name: empty-dir <p>mountPath: /tmp subPath: tmp-dir</p>	Code Applicatif & Image d'infrastructure applicatif
EXI-ARCH-6	P	<p>Le choix d'image de base est un élément important et structurant pour le projet, plus l'image est 'fine' plus vous augmenterez le "signal sur bruit" de la détection de vulnérabilité spécifique à votre code.</p> <p>Pour du code "custom", il est recommandé d'utiliser des images de base (FROM) dans l'ordre :</p> <ul style="list-style-type: none"> - Images custom à partir d'une distribution sans aucun tooling installé (distroless ou minideb par exemple) - Images issues de communautés reconnues, régulièrement mises à jour, et adaptées à un usage 	Services Mutualisés

	<p>sécurisé OpenShift (rootless, readonly FS, securityContext, etc.) comme bitnami pour les composants d'infrastructures (ex:nginx) ou si aucune image distroless n'est disponible.</p> <ul style="list-style-type: none"> - Pour faciliter les travaux d'intégration, il peut être utile pour l'équipe de disposer d'une version "debug" incluant quelques outils courants tel que curl, tail, vi + un shell avancé. Dans ce cas 2 versions d'images sont maintenues : "version" et "version-debug". <p>Lors de la construction des images custom, vous pouvez utiliser la methode de construction à 2 containers qui ne recopie que les fichiers utiles</p> <p># exemple pour créer une image python/flask en production à 2 étapes.</p> <pre>FROM python:3.8-slim as builder # Set the working directory WORKDIR /app # Copy the current directory contents into the container at /app COPY app.py /app # Install the dependencies RUN pip install --no-cache-dir flask boto3 gunicorn # Use a distroless Python image as a base FROM gcr.io/distroless/python3:nonroot # Set the working directory WORKDIR /app # Copy the dependencies and the app from the builder stage COPY --from=builder /app /app COPY --from=builder /usr/local/lib/python3.8/site-packages /app/site-packages COPY --from=builder /usr/local/bin/gunicorn /usr/local/bin/gunicorn ENV PYTHONPATH /app/site-packages VOLUME ["/tmp"] # Command to run the application CMD ["/usr/local/bin/gunicorn", "-b", "0.0.0.0:5000", "app:app"]</pre> <p>Note / tips :</p> <ul style="list-style-type: none"> - Les conteneurs fonctionnent en rootless et readonly, il impératif de placer un volumeMount "empty dir" aux emplacements clés ou l'application est susceptible d'écrire, tel que /tmp (à voir en fonction de votre framework/langage) 	
--	---	--

		<p>volumeMounts:</p> <ul style="list-style-type: none"> - name: empty-dir <p>mountPath: /tmp</p> <p>subPath: tmp-dir</p> <p>Pour le débogage, il n'est pas possible "d'écouter" le trafic au sein de l'infrastructure. Pour faciliter l'intégration pensez à mettre en place dans votre code custom un login avancé, vous pouvez envisager de mettre en place des composants d'intermédiation (temporaire?) tel qu'un reverse proxy pour produire ces logs</p> <p>cf. article sur le build en multi-stage : https://docs.docker.com/build/building/multi-stage/.</p>	
EXI-ARCH-7	P	<p>Les certificats racines "CA" des PKI internes à l'organisation ou autres nécessaire doivent être intégrés aux images, afin que les services consommant des end-points https puissent valider les certificats de celui-ci.</p> <p>(le certificat du end point ne doit pas être intégré puisqu'ils est amenés à être renouvelé, et généralement au mauvais moment)</p> <p>Tips : L'ensembles des langages et frameworks mette en oeuvre des comportements différents sur l'usage des CA racines, il a été constaté 2 cas généraux :</p> <ul style="list-style-type: none"> - Le cas idéal : les runtimes ou applications permettent de monter les CA depuis fichier ou un répertoire au lancement. Dans ce cas les fichiers ou volumes contenant les certs .crt ou .pem peuvent être montés via un volume à partir d'un objet configmap ou d'un secret. - Le cas complexe : le service utilise le CA-bundle de l'os, nécessitant que la commande "update-ca-certificates" soit être utilisée pour inclure les CA de l'organisation. Cette action ne peut qu'être réalisée à la création de l'image, par exemple par un processus multi-build. (cf exigence sur la construction des images custome) 	
EXI-ARCH-8	I	<p>Le déploiement est réalisé via gitops (argoCD)</p> <p>L'architecture de déploiement d'application est organisée pour déployer indépendamment les modules de l'application, sans couplage entre-eux.</p> <p>Typiquement l'application est découpée en une application maître d'ensemble référençant le déploiement de sous-applications.</p> <p>L'ensemble est géré en version dans un git.</p> <p>Chaque changement est réalisé via un git push ou via pull request si l'équipe a mis en place une</p>	

		relecture collective du code de version	
EXI-ARCH-9	P	<p>L'équipe met en place une stratégie gitops et "d'immutability" de la configuration de l'infrastructure. Chaque nouveau déploiement recrée un environnement nouveau ("écrase" l'ancien).</p> <p>Le déploiement met en oeuvre une stratégie de reprise de données applicative à chaque nouveau déploiement.</p> <p>(la sauvegarde des données applicatives doit être effectuée régulièrement, juste avant le nouveau déploiement)</p> <p>A titre d'exemple, l'opérateur cnpg de postgres (https://cloudnative-pg.io/) permet de mettre en place ce type de pratique.</p>	
EXI-ARCH-10	P	<p>Consommation systématique des services socles. (chaîne de service, IAM, Vault, etc...)</p> <p>Dans le cas où plusieurs services sont susceptibles d'être utilisés, privilégier les services dont l'enrôlement est automatisé.</p>	Services Mutualisés
EXI-ARCH-11	P	<p>Identification utilisateur : l'application doit obligatoirement utiliser le SSO Agent disponible et/ou France connect pour les citoyens</p>	Architecture
EXI-ARCH-12	P	<p>L'application doit pouvoir être monitorée techniquement et fonctionnellement (dont healthcheck) au travers de service de télémétrie mis à disposition par l'application. (obligatoire sous kubernetes)</p> <p>Mise à disposition obligatoire au sein de la solution des API de supervision auto descriptive /health (json+problem) et /metrics (prométhéus) et Intégration obligatoire de la solution dans la (ou les) chaîne(s) de supervision</p>	Code Applicatif & Image
EXI-ARCH-13	I	<p>Les services applicatifs d'observabilité (logs et télémétrie), registre d'image et gestion des secrets mis à disposition par l'offre Cloud Pi Native doivent être utilisés.</p>	Services Mutualisés
EXI-ARCH-14	I	<p>L'application prend en compte dans sa conception : la séparation des flux usagers, de services et d'administration, l'utilisation privilégiée du protocole https et l'authentification mutuelle des composants.</p> <p>Pour minimiser les spofs et la surcharge de trafic inutiles des équipements de protection,</p>	Architecture

		l'interconnexion entre applications mutuellement connues au sein d'un même niveau de sensibilité ne requiert pas, le transit des flux via une passerelle ou un composant d'API management. (ex: chaîne de service)	
EXI-ARCH-15	I	Les modules de l'application permettent un fonctionnement en mode dégradé, si une dépendance ne devait plus être disponible (ex arrêt d'un autre module). "Graceful degradation"	Architecture
EXI-ARCH-16	P	L'architecture et les développements garantissent l'idempotence des transactions	Architecture
EXI-ARCH-17	I	L'application minimise le besoin d'échange synchrone. Les échanges asynchrones sont réalisés via un bus de message favorisant de découplage les modules et l'ajout facilité de service en écoute.	Architecture
EXI-ARCH-18	P	<p>Les images de conteneurs sont conçues pour s'exécuter "rootless" sur des ports réseaux internes sont > 1024 et ne nécessite pas d'élévation de privilèges.</p> <p>L'image est minimaliste, concentrée sur une fonction primaire ou atomique du système.</p> <p>Elles sont conçues pour démarrer très rapidement avec le moins de dépendance possible (dans l'intérêt de la résilience et du bon fonctionnement de l'orchestrateur)</p> <p>L'image n'embarque pas de services exposant des services réseau non essentiels à son fonctionnement tel que SSHD.</p> <p>L'image ne se modifie pas elle-même au lancement. (ce qui souvent nécessite des élévations de privilèges)</p> <p>L'écriture dans les pods n'est possible qu'avec le montage d'un volume.</p>	Code Applicatif & Image

		(note : L'exécution du pod bloquée si ces règles ne sont pas respectées). cf. https://docs.openshift.com/ , et en particulier https://docs.openshift.com/container-platform/4.15/openshift_images/create-images.html (toujours vérifier la dernière version)	
EXI-ARCH-19	P	Les pods applicatifs stateless doivent pouvoir être redémarrés rapidement et à tout instant, sans affinité de session par l'orchestrateur. L'application est conçue pour minimiser les pertes de données, de transactions en cours ainsi que l'idempotence de celle-ci en cas de bascule de noeuds.	Code Applicatif & Image
EXI-ARCH-20	P	Les modules de l'application s'exécutent sans état (Stateless) et s'appuient sur la résolution de nom interne de kubernetes pour communiquer entre-eux. Les processus/microservices sont indépendants les uns des autres. Les états sont stockés dans un magasin de données centralisé. Cela pour garantir une mise à l'échelle et continuité de service de l'application (Principe Stateless)	Architecture
EXI-ARCH-21	I	L'application est conçue pour un déploiement en continu, la conception du modèle de données permet à la version future et l'actuelle de coexister sans impact fonctionnel.	Architecture
EXI-ARCH-22	P	**La configuration de l'application ne doit jamais être codée en dur** . La configuration de l'application doit provenir de l'environnement (principe Config des 12 facteurs : https://12factor.net/config). note : l'utilisation de charts helm avec un fichier values.yaml permet de couvrir l'exigence.	Déploiement
EXI-ARCH-23	I	L'ensemble des services applicatifs/processus de l'application, et ses services de supports génèrent leurs propres flux d'événement bruts pour apporter une visibilité sur son comportement. La norme de nommage des flux d'événement est, sans condition, respectée. (Principe d'observabilité) Les logs doivent obligatoirement être redirigés vers la sortie standard.	Architecture

EXI-ARCH-24	P	La performance de toutes les transactions réalisées par l'application doit être mesurable en temps réel pour contrôler la qualité de services et soutenir les investigations en cas de dysfonctionnement. Il est préconisé d'assurer la surveillance de l'application à minima au travers de : Flux d'événement de performance applicative, flux d'événement et donnée pour analyse et reporting métier, journaux d'intégrité et du système (Principe de Télémétrie)	Architecture
EXI-ARCH-25	I	L'application doit respecter dans sa conception les 12 facteurs d'une application Cloud Native : https://12factor.net/fr	Architecture
EXI-ARCH-26	I	Les déploiements au sens kubernetes ne doivent pas imposer d'éléments sur les securityContexts. Sont notamment interdits : runAsUser, runAsGroup, fsGroup, allowPrivilegeEscalation, add capabilities, etc. De la même façon, il n'est pas autorisé de monter des ressources depuis le node (hostPath par exemple)	Architecture