



Dossier Technique

Anthony Jover

Stage MCO Capgemini

4 Novembre au 24 Janvier

Table des matières

Table des matières.....	3
Remerciements.....	5
Abstract.....	6
Compétences mises en œuvre	7
Introduction.....	8
Présentation entreprise	8
Organigramme	9
Le projet	10
Présentation poste.....	10
Organisation.....	11
Safe.....	11
Les équipes	12
PI Planning.....	13
Release Management.....	14
Equipe Matrice.....	14
La MCO.....	16
Périmètre Applicatif	17
ICE	17
Présentation.....	17
Architecture	17
Eboard	21
Présentation.....	21
Architecture	21
Pages des environnements.....	24
Front.....	25
Persistance	25
Scripts	26
Réalisations	27
Capitalisation	27
Résolution d'incidents.....	27
Développements Ansible	28

Modification Page angular.....	30
Modification des liens sur pageenv	30
Modification Scripts surveillance	32
Ajout surveillance Téléfact	33
Développement d'un Job Jenkins.....	36
Insertion Table Postgres.....	38
Renouvellement des certificats apaches Fweb.....	40
DEVOPS et Outils	41
Outils organisationnels.....	41
Jira.....	41
Confluence	42
Vagrant box.....	42
Ansible	43
Jenkins	46
Linux/Scripts sh	48
Sonar (contrôle qualité).....	49
Nexus repositories.....	50
GIT	50
Intégration continue	53
Delivery.....	54
Perspectives :	56
Interface pour visualiser le suivi de l'état d'une MEP implémentant l'envoie de mail de CR.56	56
Spécifications.....	56
Automatisation de la Vabf ICE avec UIpath.....	56
Bilan.....	57
Bilan du projet	57
Bilan Personnel	57

Abstract

During my internship , I worked for Capgemini on the AMS branch on mutualized Maintenance in Operationnal condition (MCO).

I joined the MCO teams which brings together the personnel responsible for maintaining some thirty application for the needs of one of the group's main customers.

So I worked un the Matrix team, oboard on agile train, in charge of several missions for the team, such as ensuring the responsibility of 2 web applications.

Resolve incidents and organize regular deployment of new versions of the application using DEVOPS Practices.

I took part on the meetings with the customer on the various workside organized concerning my perimeter.

So I had to perform various tasks from coding on different programming languages like Yaml used by Ansible application-deployment tool , Groovy for Jenkins pipelines, Html CSS as part of AngularJS Framework, Linux Bash Scripts, analyzing the log files and various other parameters on servers to solve malfunctions.

A significant part of the internship was used to transfer skills and knowledge on the job, such as application architecture, maintenance tasks that have to be done.

The team ensure avaibility of applications from 8AM to 6PM according a larger third-party applications maintenance contract.



Compétences mises en œuvre

Activité type 1 : Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

- 1 Maquetter une application
- 2 Développer une interface utilisateur de type desktop
- 3 Développer des composants d'accès aux données
- 4 Développer la partie front-end d'une interface utilisateur web
- 5 Développer la partie back-end d'une interface utilisateur web

CCP2 : Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

- 6 Concevoir une base de données
- 7 Mettre en place une base de données
- 8 Développer des composants dans le langage d'une base de données

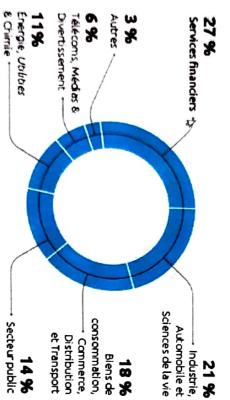
CCP3 : Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

- 9 Collaborer à la gestion d'un projet informatique Et à l'organisation de l'environnement de développement
- 10 Concevoir une application
- 11 Développer des composants métier
- 12 Construire une application organisée en couches
- 13 Développer une application mobile
- 14 Préparer et exécuter les plans de tests d'une application
- 15 Préparer et exécuter le déploiement d'une application

Introduction

Présentation entreprise

Capgemini est la première entreprise de services du numérique (ESN) en France et employant 200 000 collaborateurs et présent dans plus de 40 pays.



Basée à Paris, la société fait partie du CAC 40 à la Bourse de Paris et génère un chiffre d'affaires de 13,2 milliards d'euros en 2018.

- Quelques dates clé:

- Crédit par Serge Kampf le 1er octobre 1967 à Grenoble (France) sous le nom de Sogeti (Société pour la gestion de l'entreprise et traitement de l'information)
- 1974 : Les premières acquisitions démarrent avec l'achat de deux concurrents: CAP (France) et Gemini Computer Systems (USA).
- 1985 : Cap Gemini Sogeti fait son entrée à la Bourse de Paris.
- Depuis 2000 : Acquisitions stratégiques Kanbay, CPM Braxis, Igate :
- 2020 : Suite à son OPA, Capgemini détient plus de 50% du capital d'Altan

- Sites à Toulouse :
 - 109 Avenue du Général Eisenhower, 31100 Toulouse
 - 8 Rue Paul Mesplé, 31100 Toulouse
 - AEROPARK, 3 Chemin de Laporte, 31300 Toulouse

Le groupe Capgemini est depuis sa création attaché a 7 valeurs :

- Honnêteté
- L'audace,
- La confiance
- La liberté
- Le plaisir
- La modestie
- La solidarité

Organigramme

Business Unit Manager : Charlotte Lagauzere.

Chef de projet responsable de la MCO/Tutrice : Cedrine Delrieu.

Le projet

J'ai eu la responsabilité avec mon équipe du maintien en condition opérationnelles de deux applications :

- **L'application Internet Collectivités et Entreprises (ICE)**, portail qui fournit un ensemble de service destinés aux segments de clientèle Entreprise (Industrie, PME, PMI et Professionnels), permettant de suivre les données des services et de faire des demandes (Gestion Relation Client).
- **L'application Eboard** accessible depuis le portail ICE. Permettant aux clients ayant souscrit le service de faire des prises de position sur les marchés sur les produit fournis.

Présentation poste

Je travaillais avec d'autres développeur chargés de la MCO dans l'espace mutualisé.

L'entreprise m'a fourni un ordinateur portable sous Windows 10 équipé d'un processeur i5 et de 8go de RAM. J'avais également un écran supplémentaire de 22 pouces.

On m'a fourni une liste de logiciels nécessaires dans le cadre de mon travail ainsi que les différents accès nécessaires (Serveurs des applications, Gitlab, Jenkins etc...)

Afin d'accéder aux différentes zone réseau contenant les différents serveurs hébergeant les briques applicatives, je dispose d'accès VPN vers le réseau du client ainsi que vers le réseau Capgemini.

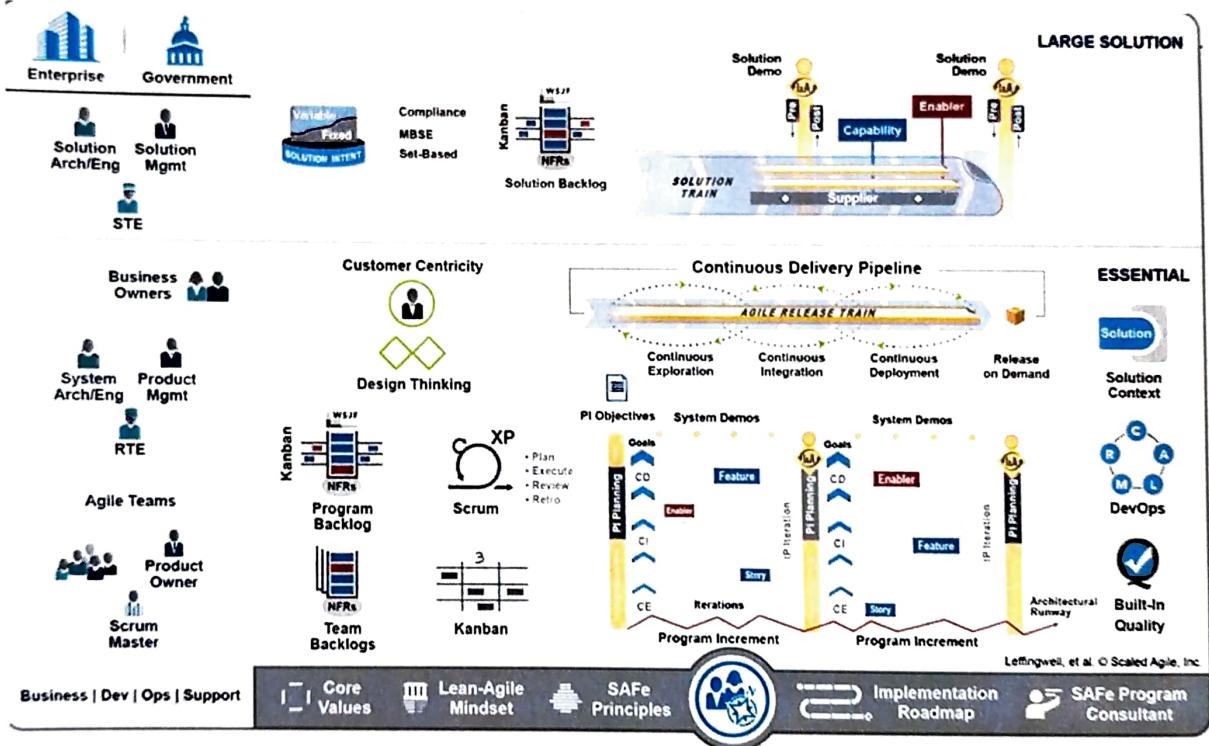
Dans le cadre d'un contrat TMA avec un gros client j'ai été formé à la MCO (Maintien en Condition Opérationnelle) au sein des équipes dans le centre de services.

Organisation

Safe

Les applications sont gérées selon une organisation Agile Safe Framework.

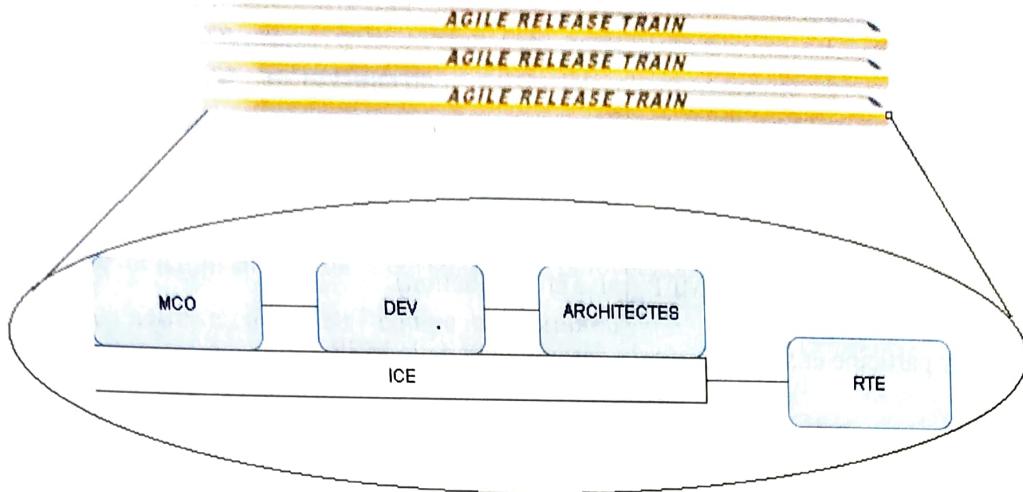
Les applications maintenues font partie d'un ensemble d'applications permettant de répondre aux problématiques du client et qui évoluent en parallèle.



Les équipes

- **Les DevTeams :**
 - Les équipes s'organisent par elles-mêmes,
 - Elles sont composées de 3 à 9 personnes,
 - Regroupant tous les rôles : Architecte, concepteur, développeur, spécialiste IHM, testeur, etc..
- **Le Scrum Master (SM) :**
 - Responsable de la mise en œuvre des valeurs et des pratiques de SCRUM,
 - Élimine les obstacles,
 - S'assure que l'équipe est entièrement fonctionnelle et productive,
 - Permettre la coopération entre les divers rôles et fonctions,
 - Protège l'équipe de toutes les interventions extérieures,
- **Le Product Owner (PO) :**
 - Responsable du retour sur investissement,
 - Choisi la date et le contenu de la release,
 - Définit les fonctionnalités du produit,
 - Définit les priorités dans le Backlog (Sprint Backlog) en fonction de la valeur "métier" (Business Value),
 - Ajuste les fonctionnalités et les priorités à chaque Sprint, si nécessaire.
 - Accepte ou rejette les résultats produits par la DevTeam.

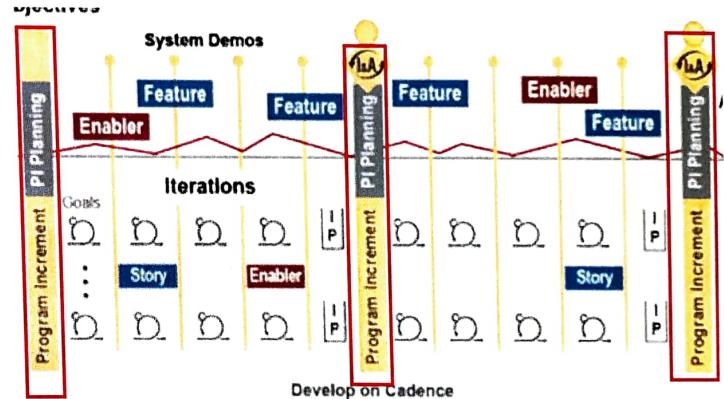
Chaque équipe de développeur travaille sous la supervision du Scrum Master avec des Sprint de 2 semaines à l'issue desquelles les nouvelles fonctionnalités développées sont présentées.



PI Planning

Toutes les 10 semaines, un PI est organisé et rassemble les Scrum master des équipes du train sous la supervision d'un Scrum de Scrum.

Cet événement est réalisé en 2 jours de réunion.



Objectifs:

- Identifier collectivement les objectifs du prochain Program Increment.
- Aligner les équipes du train de release pour s'engager sur l'atteinte de ces objectifs.

Il permet :

- D'aligner l'équipe autour d'une vision partagée sur laquelle elle s'engage.
- De mieux identifier et gérer les dépendances (internes et externes au train).

Release Management

Tous les Jeudi, une réunion de Release management est organisée, où est passé en revue le périmètre candidat à la prochaine MEP (mise en production).

Mon équipe y participe et sera chargée du déroulement de la MEP.

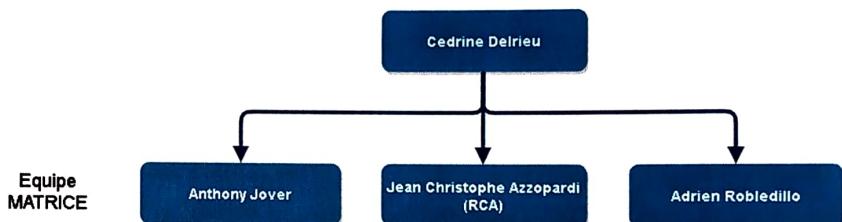
Equipe Matrice

J'ai travaillé au sein de l'équipe Matrice chargée de la MCO et embarquée dans le train.

Composée de 3 membres au mois de février 2020 supervisé par Cedrine Delrieu, responsable des équipes MCO au sein du centre de service.

Organisation au sein de

Capgemini :



La mutualisation de la MCO permet d'assurer la disponibilité des ressources et une meilleure attribution du temps alloué. Par exemple, l'application Eboard ne nécessitant pas un développeur travaillant à 100% de son temps. Cela permet aussi une meilleure synergie entre les développeurs confrontés à des problématiques similaires.

L'équipe MCO n'a pas à développer de fonctionnalités sur l'application, mais a également adopté une organisation agile avec des cérémoniaux :

- Daily le matin avec explications succinctes sur :
 - Nos avancées de la veille.
 - Qu'allons-nous faire aujourd'hui et en combien de temps nous estimons finir la tâche ?
 - Les points bloquants rencontrés.

Lors des différentes réunions avec le client, nous récoltons le besoin, évoluant avec l'avancée des différents chantiers.

Nous en dégageons des tâches, que nous pouvons découper

Pour chaque tâche nous désignons un développeur responsable et un autre chargé de la review.

Une partie du temps disponible a été consacré à la réversibilité sur le poste du nouveau RCA et des deux autres membres de l'équipe moi compris.

La MCO

Déroulement

Chaque Matin à 7h 30, la priorité est d'effectuer une VABF (vérification d'aptitude au bon fonctionnement) où nous testons le fonctionnement des différentes parties de l'application en production, via le navigateur web.

Si nous constatons des anomalies, nous contactons un infogérant et après analyse, lui indiquons les actions à effectuer sur les machines. (Seul les infogérants chez le client ont les droits admin sur les serveurs) Par exemple : redémarrage serveur d'application, modification d'un fichier de configuration etc.

Par la suite, la priorité est de s'assurer du bon fonctionnement des environnements hors production notamment IA d'intégration continue, et des environnements de travail des équipes de développeurs.

Par la suite nous effectuons les différentes tâches inhérentes à l'organisation du processus de MEP en cours, ainsi qu'aux différents chantiers selon la priorité des tâches.

En cas d'incident sur un environnement de production, notre priorité est sa résolution

Périmètre Applicatif

ICE

Présentation

L'application Internet Collectivités et Entreprises (ICE) est une application offrant les fonctionnalités de portail internet pour les clients et les professionnels.

Elle est construite de plusieurs couches logicielles, réparties sur plusieurs serveurs et faisant appel à différents partenaires via des appels vers des API.

Architecture

L'application ICE s'appuie sur l'architecture générique des portails Internet comprenant des serveurs web (Apache) d'application (Jboss), et base de données (Postgres).

En production le projet repose sur plusieurs briques applicatives réparties sur plusieurs serveurs (eux-mêmes sous forme de machines virtuelles (VM))

4 se trouvent sur le même zone réseau :

- Serveur FWEB Contenant la briques :

- **Apache 1 :**

- Contrôle toutes les requêtes entrantes.
 - Routage des requêtes vers APPs pour les contenus transactionnels, ou le PUBLISHER pour les contenus éditoriaux. (via une instance AEM Dispatcher)



- Serveur APP Contenant:



- Un serveur d'application JBoss avec l'application sous forme de fichiers EAR.

- **Base de donnée applicative**, Un serveur de base de données PostgreSQL pour les données de configuration et d'évènements



**Adobe
Experience
Manager**

- Serveur AUTHOR : Contenant une instance d'AEM Authoring permet de modifier les contenus éditoriaux.
- Serveur PUBLISH Contenant une instance d'AEM Publishing Le publishing permet d'exposer au public les contenus éditoriaux

Sur un réseau distinct se trouvent 2 serveurs supplémentaires (fournissant également les services à d'autres applications du cloud) :

- FSSO Contenant la brique :

- Apache 2
 - Protège l'accès au serveur Open AM (JBoss SSO).
 - Routage des requêtes utilisateurs vers l'apache 1 (pour les accès à l'application)



- SSO Contenant les briques

- **Jboss SSO**
 - Traite les requête d'authentification, soit en s'appuyant sur l'annuaire GARDIAN (application partenaire) pour les utilisateurs externes, soit en s'appuyant sur un annuaire LDAP pour les utilisateurs externes
 - Traite les requêtes envoyées par l'agents SSO déployés sur l'apache 1



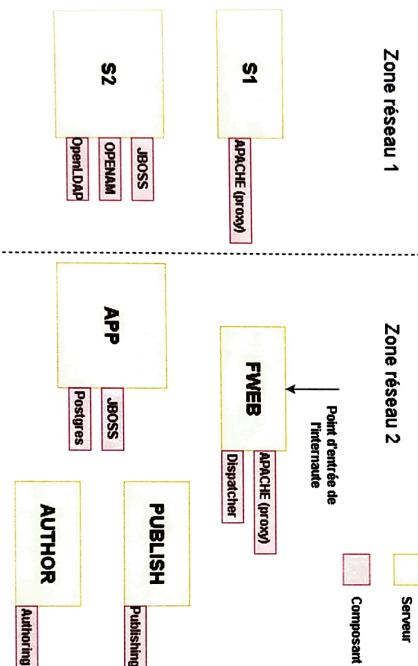


- Annuaire OpenLDAP Contient les informations et sur les utilisateurs de l'application. Il est le référentiel d'identité des utilisateurs du portail ICE. Ce référentiel est utilisé par le JBoss SSO pour authentifier et/ou autoriser les utilisateurs sur le portail de l'application.



Maquette architecture

Schémas simplifié (sans les flux de données)



Seules les briques Apache sont directement exposées au réseau internet. Les échanges de données entre les différentes briques sont fait via des protocoles sécurisés (HTTPS, SMTP...). L'application, pour ses besoins, accède également à diverses applications partenaires, accessible sur la même couche réseau (par exemple accès à un annuaire GUARDIAN) ou externes (par exemple pour téléchargement de factures du client).

A l'environnement de production , S'ajoutent pour chaque serveurs, 2 branches en hors production, contenant chacun 7 instances pour les besoins de développement et tests :

Production		6 Serveurs virtuels	
Hors Production	B1	6 serveurs x 7 environnements:	la(env d'intégration continue) , pp (Préprod) , ra(Recette) , va , sa , ta , ua
	B2	6 serveurs x 7 environnements:	ib , cp (isoProd) , rb , vb , sb , tb , ub

L'équipe matrice dispose pour ses besoins de 3 environnements pour l'application ICE

Les environnements Hors production CP et Recette sont également synchronisés avec les environnements équivalents des applications partenaires.

Eboard

Présentation

L'application Eboard est accessible via un lien de l'application ICE. Présent si les clients ont souscrit au service

Elle Permet à de gros clients qui ont souscrits à un contrat spécifique de faire des prises de position sur les marchés financiers selon le produit fourni.

L'application permet de faire des simulations, de sélectionner les différents contrats du client, de visualiser et exporter l'historique des cours etc.

Architecture

L'application est constituée de différentes briques hébergées sur un serveur.

- D'une instance Apache pour le point d'entrée de l'internaute.
- Une instance PostgreSQL pour les données de configuration
- Du projet Springboot de l'application, cœur de l'application Eboard
- D'un projet SpringBatch récupérant les données des marchés chaque jour auprès d'application partenaire.

Pour l'authentification, les utilisateurs sont redirigés vers les briques SSO partagées avec l'application ICE.

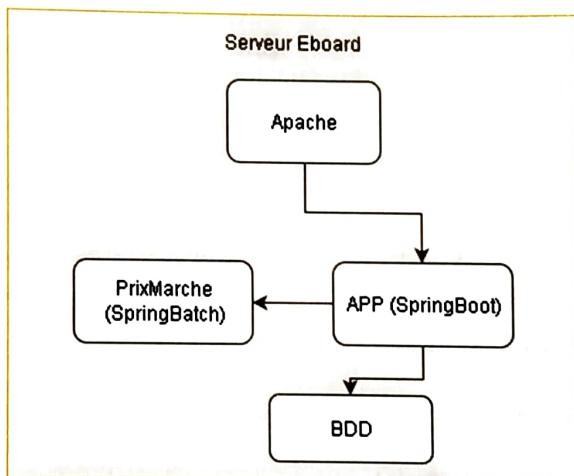
La même architecture est déployée sur un second serveur virtuel iso prod et sur un troisième contenant l'environnement de recette + 6 environnements pour les besoins des développeurs

		Environnement
Production	1 serveurs x 1 environnement	PD
Hors Production	1 serveurs x 1 environnement:	PP
	1 serveurs x 7 environnements:	R1, R2, R3, Q1, Q2, Q3

L'équipe matrice dispose de 3 environnements sur Eboard

Maquette architecture

Schémas simplifié :



Base de donnée Eboard

La base de donnée interne à l'application Eboard assure la persistance du paramétrage utilisateur, la sauvegarde des données techniques de l'application et la persistance des données des marchés issues du SpringBatch.

Aucune donnée sensible n'est présente en BDD .

Les données stockées dans la base Eboard sont les suivantes :

- Données de l'internaute sur le parcours (Identifiant, date de dernière connexion...)

- Données de paramétrage fonctionnel (Activation/désactivation des notifications de connexion d'un client, Activation/désactivation des différents onglets par client pour chacun des contrats / chacune des offres)
- Les données publiques du prix du marché

Tables

Les tables présentes ne représentent que des données partielles du système de données et nécessaires au fonctionnement de l'application. Aucune donnée de client n'est persistant sur ces tables.

- marche (19)
- pposition (22)
- t_newsletter (4)
- t_user_admin (2)
- t_user_experience (5)
- user_fav (9)
- user_history (4)
- user_preferences (12)
- user_settings (2)
- version_cg (2)

Pages des environnements

L'équipe MCO a développé 2 pages web afin de surveiller le bon fonctionnement du périmètre applicatif dont il à la charge.

Les deux pages fonctionnent sur le même principe.

Les pages sont déployées sur le serveur apache Fweb de notre environnement hors production B1 pour ICE, et sur R7 notre environnement de recette sur Eboard, donc non accessibles depuis le réseau internet public.

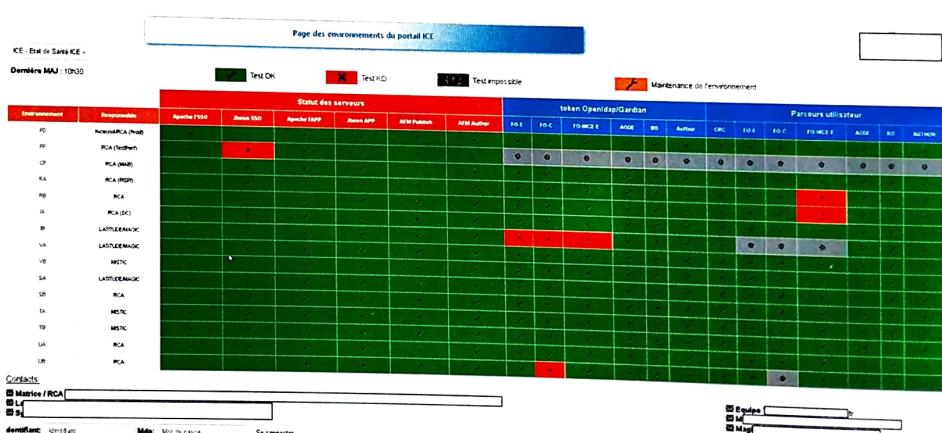
Des scripts linux sont exécutés toutes les 10 min via la crontab.

Ils effectuent des appels curl pour tester les différentes briques de l'application, et stockent le résultat dans des fichiers json réécrits toutes les 10 min.

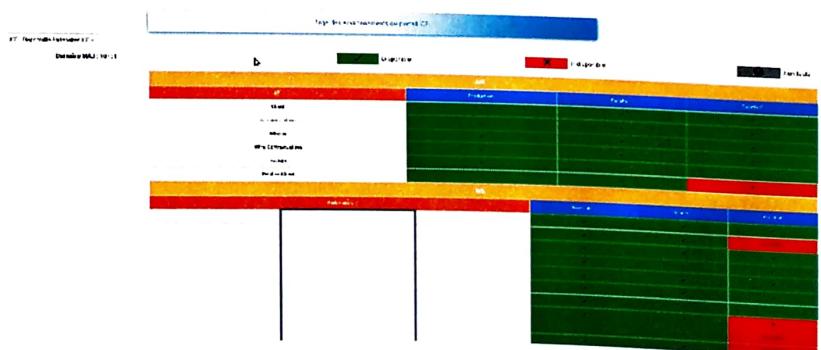
Une page angular se charge de parcourir les fichiers json et affiche l'état selon les données dans le fichier json (OK KO 404..)

Aperçu :

Surveillance de
l'application (tous les
environnements)



Surveillance des
applications
partenaires (production et
recette):



Front

Page angular

(Tableaux html) :

```
<!-- TABLE API -->


| API / URL    |                                       |            |                                                                                                             |
|--------------|---------------------------------------|------------|-------------------------------------------------------------------------------------------------------------|
| Titre URL    | URL                                   | Statut     | Actions                                                                                                     |
| Titre URL_01 | <a href="#">http://www.google.com</a> | OK         | <span class="glyphicon glyphicon-remove"></span> <span class="glyphicon glyphicon-exclamation-sign"></span> |
| Titre URL_02 | <a href="#">http://www.yahoo.com</a>  | Production | <span class="glyphicon glyphicon-remove"></span> <span class="glyphicon glyphicon-exclamation-sign"></span> |
| Titre URL_03 | <a href="#">http://www.bing.com</a>   | Recette 1  | <span class="glyphicon glyphicon-remove"></span> <span class="glyphicon glyphicon-exclamation-sign"></span> |
| Titre URL_04 | <a href="#">http://www.bing.com</a>   | Recette 6  | <span class="glyphicon glyphicon-remove"></span> <span class="glyphicon glyphicon-exclamation-sign"></span> |


```

Persistance

Exemple fichier Json :

```
[{"env":"pd",
"owner":"Kicecool/RCA (Prod)",
"apache_s1":"OK",
"jboss_s2":"OK",
"statut_publish":"OK",
"token_opendap_fo_e":"OK",
"APPS":"OK",
"parcours_F0_E":"OK",
"token_opendap_fo_c":"OK",
"parcours_F0_C":"OK",
"token_opendap_aode":"OK",
"parcours_AODE":"OK",
"token_opendap_bo":"OK",
"parcours_BO":"OK",
"token_opendap_author":"OK",
"parcours_AUTHOR":"NA",
"timestamp":"12h40"}]
```

Le composant en Javascript parcours les fichiers Json et affiche la case du tableau concernée en rouge ou vert selon l'état renseigné dans l'objet concerné dans le Json. Par exemple ici la case correspondant à l'apache S1 apparaîtra en vert.

Si un composant est KO, un mail d'alerte est envoyé aux membres de l'équipe MCO ainsi qu'au PO:

/!\ ATTENTION /!\\

La surveillance des environnements indique un KO en production :

-JBoss S2

[Lien page des environnements](#)

Bonne journée à tous,

L'équipe CA ICE

Scripts

Différents scripts lancés sur les serveurs en crontab ou manuellement nous permettent de faciliter les actions.

Nous maintenons actuellement 11 scripts + Les scripts de surveillance de la page des environnements.

Ceux-ci nous permettent : De sauvegarder ou purger les bases de données, l'annuaire LDAP, les contenus AEM.

Réalisations

Capitalisation

Ayant pu couvrir lors de mon stage un processus de MEP complet, j'ai pu créer un article détaillé rappelant le déroulement et les étapes nécessaires à l'équipe Matrice tout au long d'une MEP. (Configuration des Variables Ansible, modifications Jenkins nécessaires, Livrables à fournir etc.

Partie 1: Initialisation des articles associés à la MEP - RFC

Contenu des outils spécifiques à la MCO, (scripts MCO, page des environnements...)	Process de release
Système Team	Organisation MEP
Contenu des outils nécessaires à la compilation, aux tests, au packaging et au déploiement de l'application de manière automatisée (Code Ansible, Jenkinsfile, etc...)	Sur XE en réalité des MEP à la demande (Delivery on demand)
	• Une réunion de Release Management (RM) est organisée avec le team agile. Un représentant de chaque équipe projet, ainsi que le maître et les RAs sont présents.
	• Un représentant de chaque Feature / UAT fait un brief instantané de la fonctionnalité proposée. Le code modifié a préalablement été validé et merge sur release candidate du projet (Si associé). Rapport.
	• Nous pouvons avoir des fonctionnalités à plusieurs lors de la réunion de RM, par exemple lors de migration / MAJ certificats, dépendance avec les équipes de dev... important : dans les tâches doivent être marquées en déplacement (order pour apparaître dans le panier lors de la PA).
	• Un ticket sur Jira est associé à cette Release (Dans l'ordre : ticket : on peut déjà se faire une idée des projets qu'il faut améliorer que nous allons utiliser)
	• Le DO est déclenché après conciliation entre les différentes équipes lors de la PA.
	• Un mail est envoyé aux équipes
	• Les personnes sont alertes et disponibles à l'installation
	• Le numéro de la version qui sera déployée (ex: 19.0.0.15)
	• Le planning du déploiement de MEP : Créditeur branche release : TNR - MAB - MEP

Création des articles

- Feuille de route
- PTT procédure technique d'installation

J'ai également pu créer une page d'état des lieux des serveurs et composants hors prod en vue d'un chantier en cours de réinstallation sur de nouveaux serveurs aux performances améliorées.

Cet article continue d'être alimenté par les différents acteurs du chantier.

Résolution d'incidents.

En cas de KO sur prod ou hors prod, analyse (recherche de la brique applicative ko et analyse des log pour trouver le problème) puis rédaction Procédure et résolution effectuée par infogérant(Si problème en Serveur de production) qui possède les droits en écriture sur les serveurs de production.

Exemple de problème rencontré :

- Echec d'un pipeline d'installation automatique sur environnement IA, causant l'indisponibilité de l'application sur cet environnement. Relancement de l'installation.

- Echec d'un démarrage automatisé d'un Jboss sur l'environnement de recette. L'application ne répond pas et la page des environnements nous indique la brique JBoss APP KO. Après analyse des logs, il s'avère qu'il n'y avait plus d'espace disponible dans le système de fichier où se situent certains logs (du -sh* pour voir le fs le plus rempli) causant le refus du démarrage du serveur d'application. Résolution du problème après suppression du fichier de log incriminé modification d'un script de rotation des log (diminution du temps de rétention des fichiers de log incriminés). Solution : Purge du log et relance du serveur.

Développements Ansible

- Ajout option verify_checksum

Le pipeline Jenkins Delivery fait toujours appel à certains rôles communs à tous les modules tel que le téléchargement des livrables Nexus, Classés dans un dossier de rôle COMMON/Getlivrables (Voir chapitre DEVOPS/Ansible).

L'une des tâches appelées téléchargeait systématiquement depuis nexus les livrables demandés dans un répertoire cible. En se basant sur l'option « state : présent » nous pouvions gagner du temps, la tâche ne téléchargeait pas les nombreux livrables si un livrable portant le même nom était présent. Seulement, si le livrable avait le même nom mais le fichier est différent, la tâche ne le téléchargeait pas et ce n'était pas le comportement désiré.

- Après des recherches dans la documentation j'ai proposé d'utiliser l'option verify_checksum, permettant de vérifier l'intégrité du fichier. Ainsi la tâche ne téléchargera pas le livrable uniquement si le livrable identique n'est pas déjà présent et non modifié, gagnant du temps lors du packaging. Dans le cas contraire, nous gagnons en sécurité en nous assurant que les livrables ne sont pas corrompus entre notre dépôt nexus et notre Installation.
exemple :

```

- name: "Download artifacts from Nexus release or snapshot repos to local folder"
  maven_artifact:
    repository_url: "https://si-devops-nexus[REDACTED]/repository/ice-ice-maven-["
    group_id: '{{ item.group_id }}'
    artifact_id: '{{ item.artifact_id }}'
    extension: '{{ item.extension | default() }}'
    version: '{{ item.version | default() }}'
    classifier: '{{ item.classifier | default() }}'
    username: '{{ nexus_login }}'
    password: '{{ nexus_pass }}'
    dest: '{{ dossier.livraison }}/{{ module }}/{{ item.file }}'
    verify_checksum: always
  loop: "{{ livrables[ module ][ 'artifact' ] | default([]) }}"
  register: download_files

```

- Montée de version Ansible (2.6.7)

Le prochain agent Jenkins exécutant notre code Ansible utilisera une version plus récente d'ansible.

J'ai effectué une veille sur la documentation principale d'ansible

https://docs.ansible.com/ansible/latest/porting_guides/porting_guide_2.6.html

Et constaté que le mot clé with_item utilisé pour faire passer les variables doit être remplacé par loop dans les nouvelles versions.

L'option loop n'acceptant que les listes, il a fallu procéder aux modifications lorsque nécessaire.

Exemple modification effectuée :

```

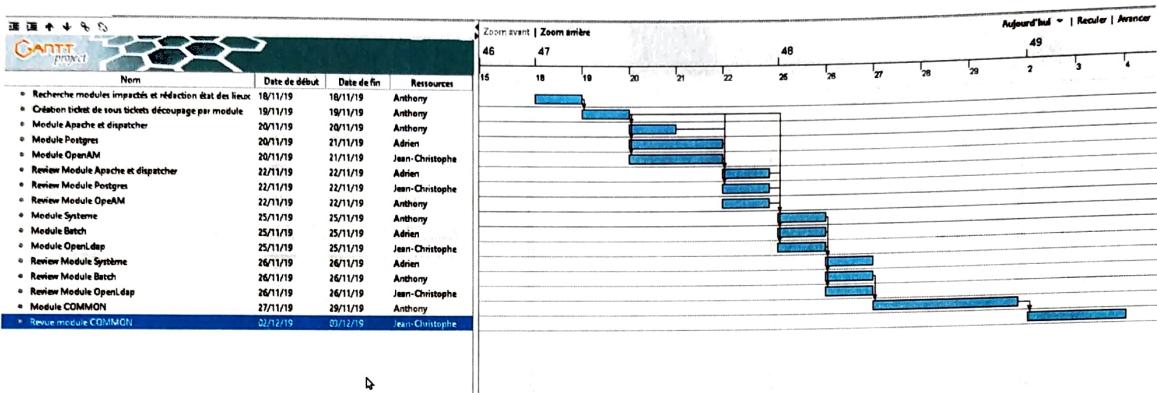
15 -   with_items:
16 -     - "{{ livrables[ composant ][ 'artifact' ] }}"
17 -     - "{{ livrables[ composant ][ 'raw' ] | default([]) }}"
15 +   loop: "{{ livrables[ composant ][ 'artifact' ] + livrables[ composant ][ 'raw' ] | default([]) }}"

```

Planification :

Le mot clé à modifié étant présent dans les taches jouées pour tous les composants, il a fallu le modifier et tester le code en appelant toutes les tâches et donc déployer tous les composants dans diverses conditions.

Pour cela, le travail a été réparti dans les équipes selon le planning suivant (tenant compte des réunions et priorités données aux incidents).



Cette tâche m'a permis d'analyser en profondeur le découpage fin du code ansible utilisé sur mon périmètre.

Nous avons travaillé sur une branche tirée depuis la release continue.

Nous avons découpés les modifications à effectuer par module.

Modification Page angular

Modification des liens sur pageenv

- La page des environnements fourni les liens vers les outils DEVOPS (Nexus Jenkins...)
J'ai dû y ajouter les liens vers nexus et NexusIQ manquants.

Aperçu code source :

```

<table style="font-family :Helvetica;font-size: 10pt;color: #3333ff; width:100%;margin-top:8px">
  <tr>
    <td style="text-align: left">
      <ul>
        <li><a href="#" target="_blank">[REDACTED] /confluence Devops</a></li>
        <li><a href="#" target="_blank">[REDACTED] Dashboard.jspa>JIRA</a></li>
        <li><a href="#" target="_blank">[REDACTED] /gitLab/</a></li>
        <li><a href="#" target="_blank">[REDACTED] /project>SonarQube</a></li>
      </ul>
    </td>
    <td style="text-align: left">
      <ul>
        <li><a href="#" target="_blank">[REDACTED] /jenkins/job/application/">JENKINS 2</a></li>
        <li><a href="#" target="_blank">[REDACTED] /zebra-eocard/job/application/">JENKINS 4</a></li>
        <li><a href="#" target="_blank">[REDACTED] /NEXUS</a></li>
        <li><a href="#" target="_blank">[REDACTED] /NexusIQ</a></li>
      </ul>
    </td>
  </tr>
</table>

```

Aperçu résultat :

Liens Devops :

- Confluence Devops
- JIRA
- GitLab
- SonarQube
- JENKINS 2
- JENKINS 4
- NEXUS
- NexusIQ

- Elle fournit également les coordonnées des équipes du projet. Une équipe étant ajoutée et une autre changeant de nom, j'ai également ajouté ces modifications à la page des environnements

Aperçu code source :

```

<div ng-hide="showContacts()">
  <h3> Contacts : </h3>
  </div>
  <table style="width:100%">
    <tr>
      <td style="text-align:left;"><span class="glyphicon glyphicon-envelope"></span><strong> Matrice / RCA :</strong> geoffrey-externe.bois@edf.fr</td>
      <td style="text-align:left;"><span class="glyphicon glyphicon-envelope"></span><strong> System :</strong> [REDACTED]@sys</td>
      <td style="text-align:left;"><span class="glyphicon glyphicon-envelope"></span><strong> Equipe :</strong> [REDACTED] fr</td>
    </tr>
    <tr>
      <td style="text-align:left;"><span class="glyphicon glyphicon-envelope"></span><strong> [REDACTED] :</strong> [REDACTED]</td>
      <td style="text-align:left;"><span class="glyphicon glyphicon-envelope"></span><strong> [REDACTED] :</strong> [REDACTED]</td>
    </tr>
  </table>

```

Aperçu Résultat :

<u>Contacts:</u>	<input checked="" type="checkbox"/> Matrice / RCA : <input type="text" value="anthony-"/> <input type="checkbox"/> L : <input type="text"/> <input checked="" type="checkbox"/> Système : <input type="text"/>	<input checked="" type="checkbox"/> Equipe : <input type="text" value="fr"/> <input checked="" type="checkbox"/> Mis : <input type="text"/> <input checked="" type="checkbox"/> Ma : <input type="text"/>
------------------	--	---

Planification :

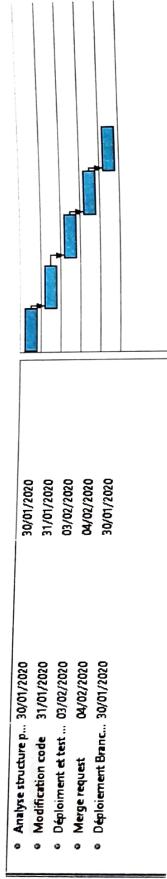
Analyse structure page des environnements

Modification code

Déploiement et test branche 2:

Merge request

Déploiement Branché



Modification Scripts surveillance

- Mise à jour des mails des destinataires

Les scripts de surveillance des pages des environnements envoient des mails d'état de santé de nos applications

J'ai rajouté mon mail et celui des autres nouveaux membre de l'île sainte-marguerite dans la liste contacts dans le menu déroulant.

- ## • Découpage des destinataires

Dans le cadre d'un chantier de la brique d'authentification, une nouvelle équipe a été rajoutée et elle doit reévoquer les mails uniquement pour cette brique. Ajout d'une condition dans le script existant et d'une nouvelle liste à concaténer.

Ajout surveillance Téléfact

Le service Téléfact permet aux clients de payer leurs factures.

J'ai ajouté la surveillance de la disponibilité de ce service à la page des environnements dans l'onglet état de santé

API	Cloud	Database
-----	-------	----------

Schémas Maquette :

Partenaires	Produits	Partenaires	Vert/Rouge

J'ai ajouté une fonction dans le script Surveillance_partenaires, celle-ci étant appelée à la fin du script.

partenaire :

Aperçu du Json

Aperçu du Json

```
[{"partenaire": "Telefact",  
 "statut_prod": "OK",  
 "timestamp": "12h31"}]
```

```
function check_partenaires()  
{  
    // appel partenaires  
    // Appel partenaires refacto  
    result= curl https://[REDACTED]  
    // Construction json  
    echo "[" > $resultpartenaires  
    echo "{'partenaire': 'Telefact'}" >> $resultpartenaires  
    if echo "$result curl" | grep 'OK'  
    then  
        echo "{'statut_prod': 'OK'}" >> $resultpartenaires  
        ANTHONY=$(date '+%d-%m-%Y %H:%M')  
        echo "{'statut_prod': 'OK', 'date': '$ANTHONY'}" >> $resultpartenaires  
        echo "{'datestamp': '$date'}" >> $resultpartenaires  
        echo "]" >> $resultpartenaires  
        chmod 740 $hom espacio/pageAngular/json/recap_partenaires  
    fi  
}
```

Aperçu du script

Surveillance :

Après ajout du tableau et du service \$http.get nécessaire dans le fichier app.js ainsi que des balises ajoutant un tableau supplémentaire dans le fichier etatSantePartenaires.html, la page AngularJS est en mesure de consommer le json et afficher la case du tableau correspondant au partenaire Telefact en rouge ou en vert selon la disponibilité de celui-ci.

Extraits app.js

```
app.controller('tableauController', function($scope, $http, $filter) {  
  
    $scope.login = false;  
    $scope.changementEnv = true;  
    $scope.listeEnv = [];  
    tabProd = 0;  
    tabEnt = 0;  
    ligenthes = 0;  
    tabETAT_Sante = 0;  
    tabCheckAPI = 0;  
    tabCheckPartenaires = 0;  
  
    $scope.listeEnv = [  
        {"name": "Liens", "env": ""},  
        {"name": "ICE - Internet Collectivité Entreprise", "env": "ICE"},  
        {"name": "ICE - Version Relais", "env": "EC"},  
        {"name": "ICE - Etat de Santé ICE", "env": ""},  
        {"name": "ICE Disponibilité Partenaires ICE", "env": ""}  
    ];  
    $scope.nouveauEnv = false;
```

```
$http.get('pageAngular/json/recap_partenaires').then(function(res) {  
    $scope.recapPartenaires = res.data;  
});
```

```
app.directive('partenaires', function() {  
    return {  
        template: '

[[etatSantePartenaires]]

',  
        controller: 'tableauController'  
    };  
});
```

Extrait etatSante-partenaires.html :

Aperçu du résultat :



Développement d'un Job Jenkins

Tous les matins, un redémarrage applicatif est programmé sur les environnements d'hors production. Ce redémarrage était séquentiel via un projet freestyle.

Il m'a été demander d'écrire un nouveau pipeline effectuant ce redémarrage en parallèle afin de gagner du temps.

application > 3_Services_Auto

Saisissez un nom

Redemarrage_hors_prod_quotidien » Champ obligatoire

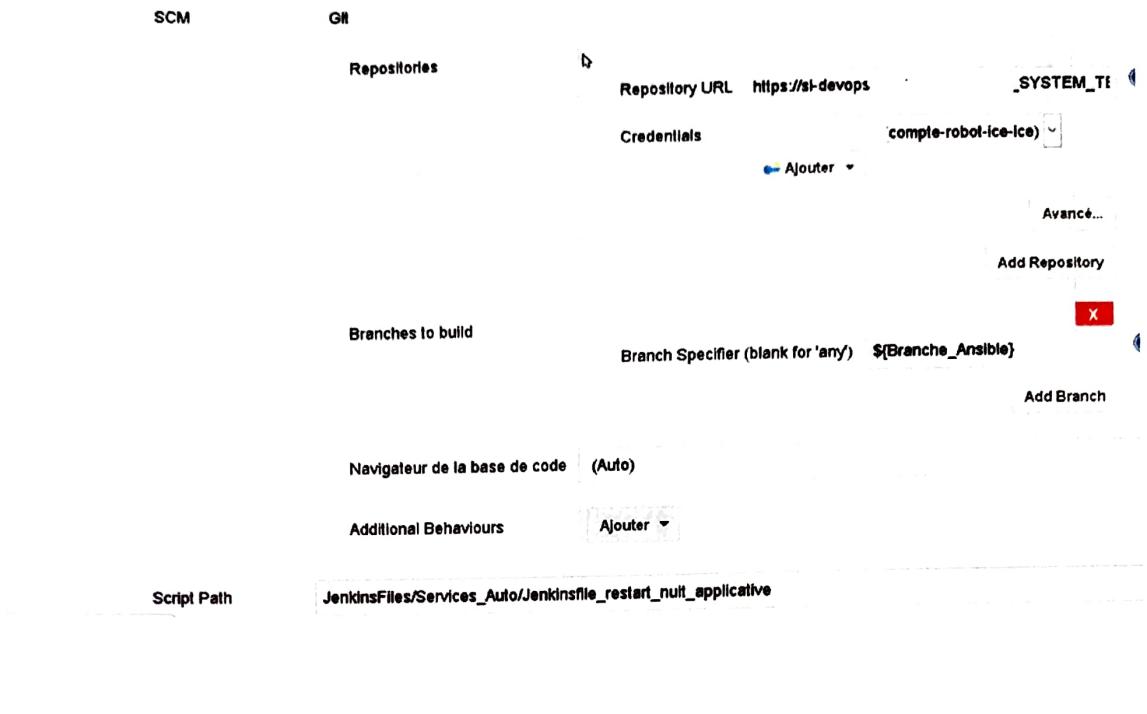
 **Construire un projet free-style**
Ceci est la fonction principale de Jenkins qui sert à builder (construire) votre projet. Vous pouvez intégrer tous les outils de gestion de version avec tous les systèmes de build. Il est même possible d'utiliser Jenkins pour tout autre chose qu'un build logiciel.

 **Construire un projet maven**
Construit un projet avec maven. Jenkins utilise directement vos fichiers POM et diminue radicalement l'effort de configuration. Cette fonctionnalité est encore en bêta mais elle est disponible afin d'obtenir vos retours.

 **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

 **Tâches automatisées**

Configuration du pipeline



Aperçu Utilisateur :

Pipeline 5 / Redémarrage hors production quotidien

Ce build nécessite des paramètres :

Environnement	<input type="text" value="pp,cp,ib,va,vb,sa,sb,ta,tb"/>
Environnements HorsProduction pp,cp,ra,rb,ja,ib,va,vb,sa,sb,ta,tb,ua,ub	
Mode	<input type="button" value="restart"/>
Sélectionner un mode	
Branche_Ansible	<input type="text" value="release/continue"/>
Composants	<input checked="" type="checkbox"/> fssso_apache <input checked="" type="checkbox"/> sso_jboss <input checked="" type="checkbox"/> sso_opendap <input checked="" type="checkbox"/> fapp_apache <input checked="" type="checkbox"/> app_jboss <input checked="" type="checkbox"/> app_postgres <input checked="" type="checkbox"/> publish_aem <input checked="" type="checkbox"/> author_aem
<input type="button" value="Build"/>	

Insertion Table Postgres

Lors d'une Mep sur Eboard, il nous a été demandé de jouer un script de création et d'insertion dans une table de la base de donnée PostgreSQL, ajouts nécessaires au fonctionnement d'une nouvelle feature.

La table t_auction Comporte 4 champs id , date, value et year.

Avec le dictionnaire de données suivant :

code	signification	Type	
Id	Identifiant	Nombre entier	
Date		Date et heure	
Value	Valeur associée	Nombre réel	
Year	année	Nombre entier	

Script de création (modèle physique) :

```
CREATE TABLE public.t_auction
(
    id bigint NOT NULL DEFAULT nextval('t_auction_id_seq'::regclass),
    date timestamp without time zone,
    value real NOT NULL,
    year integer NOT NULL,
    CONSTRAINT "t_auctionPK" PRIMARY KEY (id)
)
```

Le script d'insertion :

```
INSERT INTO public.t_auction(
```

```
    id, date, value, year)
VALUES (1, '2016-12-15 00:00:00', 9.9998, 2017),
(2, '2017-11-09 00:00:00', 9.631, 2018),
(3, '2017-12-14 00:00:00', 9.3753, 2018),
(4, '2017-12-14 00:00:00', 12.9998, 2019),
(5, '2018-03-08 00:00:00', 18.5, 2019),
(6, '2018-04-26 00:00:00', 18.2409, 2019),
(7, '2018-06-21 00:00:00', 18.5002, 2019),
(8, '2018-09-13 00:00:00', 18.5007, 2019),
(9, '2018-10-18 00:00:00', 16.77, 2019),
(10, '2018-12-13 00:00:00', 18.0457, 2019),
(11, '2019-03-21 00:00:00', 20.0009, 2020),
(12, '2019-05-16 00:00:00', 20.0024, 2020),
(13, '2019-06-27 00:00:00', 22.382, 2020),
(14, '2019-09-12 00:00:00', 20.0008, 2020),
(15, '2019-10-17 00:00:00', 17.7797, 2020),
(16, '2019-12-12 00:00:00', 16.5839, 2020);
```

Nous avons donc placé le script sur un emplacement du serveur nous étant réservé, et avons fourni à l'infogérant la procédure en ligne de commande suivante (en s'étant assuré de ses droits sur les fichiers et application)

```
psql -h localhost -p "*****" -U "****" -d "****" -a -q -f /livraison/mco/postgres/insert-capacite.sql
```

Renouvellement des certificats apaches Fweb.

J'ai eu à renouveler les certificats permettant à l'application eboard de communiquer avec un partenaire.

Pour cela j'ai généré le certificat au format jks nécessaire, avec l'outil keystore explorer et le fichier cer fourni.

Après exécution et tests sur environnement hors prod, nous avons déposé le certificat sur le serveur nous avons planifié une plage horaire et fourni la pti suivante à appliquer par l'infogérant : (noms et chemins modifiés)

en root:

```
cd « Chemin vers certificats »/certificats/  
cp -p partenaire.jks partenaire.jks.old  
cp /livraison/mco/certif/eleneo.jks .  
chown utilisateurApplicatif:GroupeApplicatif partenaire.jks  
chmod 750 partenaire.jks  
service eboard restart  
rm partenaire.jks.old
```

Après exécution nous avons testé l'application sur environnement prod et constaté le bon fonctionnement du service

DEVOPS et Outils

Afin d'assurer la livraison en continu de nouvelles fonctionnalités, les équipes travaillent avec une approche Devops.

Outils organisationnels

Afin de faciliter nous utilisons différents outils organisationnels :

Jira

Nous utilisons le tableau Kanban disponible sous notre espace Jira, dans lequel nous avons 4 colonnes : A faire, En cours, Bloqué/revue, Done.

Exemple de tableau kanban utilisé :

Tableau Kanban

FILTRES RAPIDES: Mes demandes Récemment mis à jour

À FAIRE 4 EN COURS 4 BLOQUÉ / REVUE 6 TERMINÉ 12 SUR 156 Publisher...

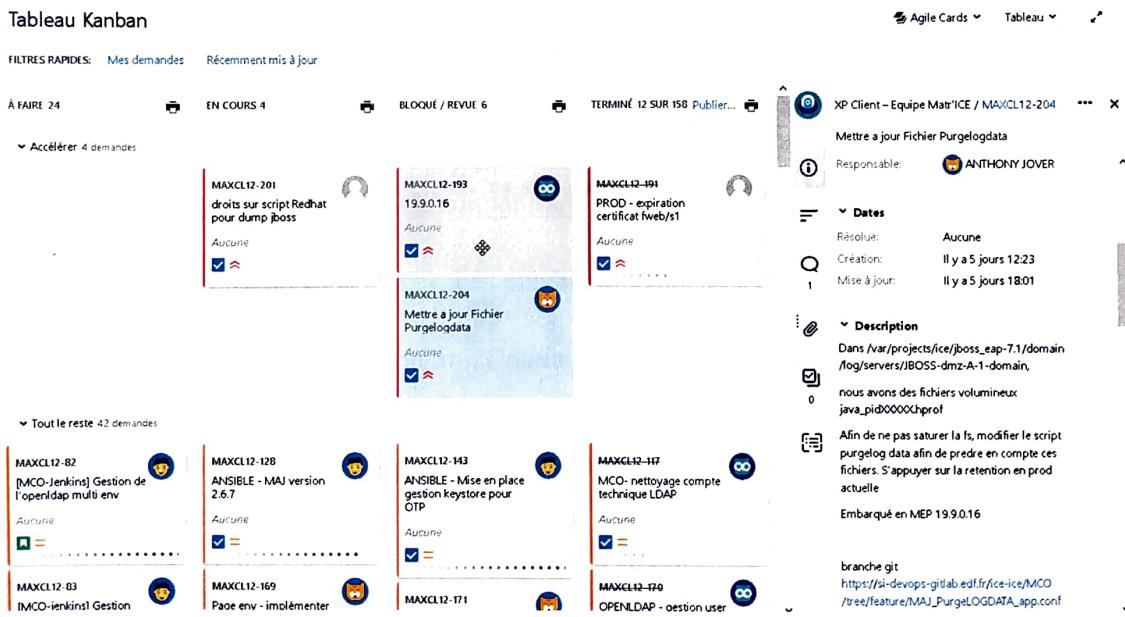
▼ Accélérer 4 demandes

MAXCL12-201 droits sur script Redhat pour dump jboss Aucune MAXCL12-193 19.9.0.16 Aucune MAXCL12-191 PROD - expiration certificat lweb/s1 Aucune MAXCL12-204 Mettre à jour Fichier PurgeLogdata Aucune

XP Client – Équipe Matrice / MAXCL12-204 Responsable: ANTHONY JOVER Dates Résolu: Aucune Crédit: Il y a 5 jours 12:23 Mise à jour: Il y a 5 jours 18:01 Description Dans /var/projects/ice/jboss_eap-7.1/domain /log/servers/JBOSS-dmz-A-1-domain, nous avons des fichiers volumineux java.pidXXXXXXchprof Afin de ne pas saturer la fs, modifier le script purgeLog data afin de prendre en compte ces fichiers. S'appuyer sur la retention en prod actuelle Embarqué en MEP 19.9.0.16 branche git https://si-devops-gitlab.edf.fr/ice-ice/MCO /tree/feature/MAJ_PurgeLOGDATA_app.conf

▼ Tout le reste 42 demandes

MAXCL12-82 [MCO-Jenkins] Gestion de l'ldap multi env Aucune MAXCL12-128 ANSIBLE - MAJ version 2.6.7 Aucune MAXCL12-143 ANSIBLE - Mise en place gestion keystore pour OTP Aucune MAXCL12-117 MCO - nettoyage compte technique LDAP Aucune MAXCL12-03 [MCO-jenkins] Gestion Aucune MAXCL12-169 Paie env - implémenter Aucune MAXCL12-171 OPENLDAP - gestion user Aucune



Confluence

L'équipe est amenée à effectuer des procédures variées utilisant de nombreux outils.

Nous utilisons un espace dédié à notre équipe dans l'application Confluence, nous permettant de sanctuariser et capitaliser :

- Les informations nécessaires sur l'architecture des applications.
- Les procédures d'exploitation
- Les informations sur les outils logiciels utilisé

The screenshot shows a Confluence page with a sidebar containing a navigation tree. The main content area is titled "Process de release". It includes sections for "Voir: process de release", "Organisation MEP", and "Partie 1: Initialisation des articles associés à la MEP - RFC". The "Organisation MEP" section contains several bullet points describing the Release Management process, including roles like "Demande d'habilitation", "Procédures d'administration", "Procédures d'exploitation", and "Procédures MCO". The "Partie 1" section provides a step-by-step guide for initializing articles associated with the MEP, involving tasks such as creating tickets, sending emails, and preparing Go documents.

Vagrant box

Box vagrant contenant une distribution debian avec ansible, avec un point de montage configuré pointant vers les dépôts locaux des projets eboard et ice sur notre poste.

Cela nous permet de lancer des commandes ansible en s'affranchissant de Jenkins (nous permet de tester de nouveaux paramètres, tester des modifications du code ansible).

Les développeurs n'utilisent que Jenkins pour leurs déploiements. Seul la MCO à la charge du maintien du code ansible.

```
Vagrant@vagrant:~/media/ansible/template_projet_ansible
(.ICE) vagrant@vagrant:~/media/ansible/template_projet_ansible$ cd ~/media/ansible/template_projet_ansible/
(.ICE) vagrant@vagrant:~/media/ansible/template_projet_ansible$ ls
ansible.cfg  .gitignore  install.sh  nom_playbook.retry  playbook  _staging.ini
glossaire.md  install_dev.sh  nom_playbook.yml  play  test1.txt
(.ICE) vagrant@vagrant:~/media/ansible/template_projet_ansible$
```

Ansible

Ansible permet de réaliser des déploiements, l'exécution de tâches et la gestion de configuration sur plusieurs machines en même temps. Il est **agent-less** et utilise **SSH** pour mettre en place les actions à réaliser, elles-mêmes écrites en **YAML**.

Pour fonctionner, ansible utilise plusieurs éléments :

Les tâches : plus petite entité dans ansible. (Exemple : copie ou suppression d'un fichier, modification des droits d'un dossier, téléchargement d'un livrable...) les tâches peuvent utiliser des variables.

Différents modules sont disponibles permettant de couvrir l'ensemble des actions possibles sur un serveur. (Par exemple module copy pour la copie d'un fichier avec différents paramètres).

(Liste de tous les modules sur

https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html)

Playbook : Liste ordonnée de tâches. Les playbook peuvent aussi utiliser des variables.

Inventaire : Fichier avec extension .ini. Contient les informations sur nos machines distantes, tel que l'ip (nécessaire pour effectuer la connexion ssh). Nous pouvons organiser des groupes dans l'inventaire afin de toucher plusieurs machines.

Exemple de tache :

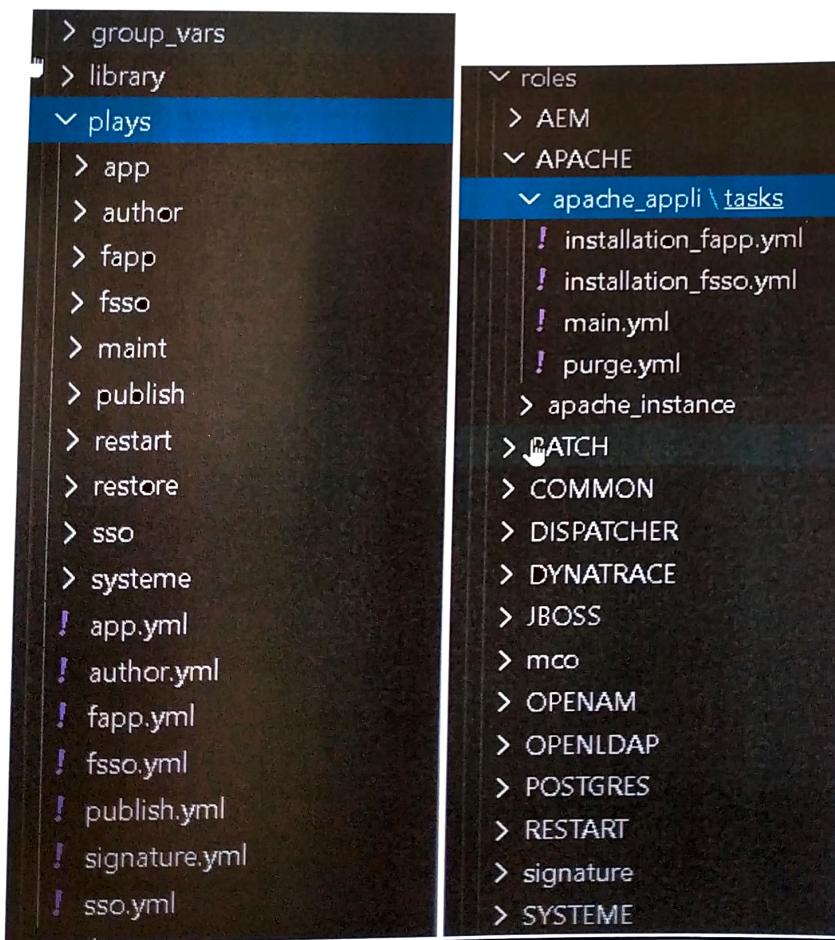
Utilisée lors du déploiement
serveur apache ICE

```
- name: "Copie recursive apache configuration folder - SSL certificates"
  synchronize:
    src: "{{ item.livraison }}"
    dest: "{{ item.destination }}"
    recursive: yes
    checksum: yes
    group: no
    perms: no
    delete: no
    times: no
    owner: no
  delegate_to: "{{ inventory_hostname }}"
  with_items:
    - "{{ livrables[ 'apache' ][ 'artifact' ] }}"
    - "{{ livrables[ 'apache' ][ 'raw' ] }}"
    - "{{ livrables[ 'apache' ][ 'npm' ] }}"
```

Nous utilisons une arborescence plus poussée.

Groups vars rassemble les variables utilisées dans les play et les roles.

Arborescence :



Via Jenkins ou directement sur notre environnement de développement vagrantbox, la commande est lancée avec les paramètres. Il est possible d'utiliser des tags pour ne sélectionner que les play désirés.

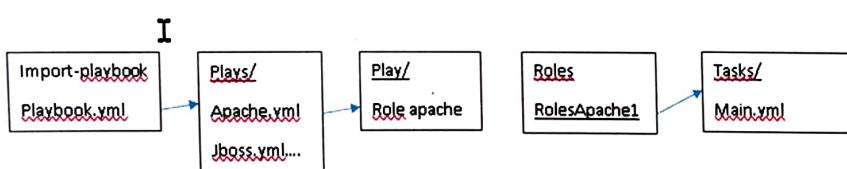
Exemple de commande permettant de réinstaller tout le composant apache sur le serveur FWEB de l'environnement RA :

```
export HOST : ra
```

```
ansible-playbook "../installation.yml" -i "./${InventaireHorsProd.ini}" -e  
ansible_ssh_user=${Nom Utilisateur du serveur} -e ansible_ssh_pass=${Mot de passe de  
l'utilisateur} -e ansible_become_pass=${Mot de passe associé } -e nexus_login=${login  
dépôts nexus} -e nexus_pass=${mot de passe associé } --tags 'fapp-apache' -l ${HOST} -  
vvvv
```

InventaireHorsProd est un fichier d'inventaire. : Contient toutes les adresses IP cible
Il contient les alias des serveurs Chaque alias de serveur est associé à la variable ansible:
« ansible_ssh_host »: où l'on renseigne l'IP utilisée pour la connection SSH à la machine
"../installation.yml" Point d'entrée Ansible, contient la liste des plays à exécuter.

Schémas :



Chaque tâche est exécutée et renvoie 4 états possibles :

- ok
- changed
- unreachable (La connexion à échoué avec le serveur)
- failed (La tâche à échoué)

Nous avons un récapitulatif des tâches lancées en fin de sortie de la commande execplaybook

Exemple de sortie :

```
[11:20:21] COMMON/common_getLivrables : Set variable if_getlivrable_done ...Monday 30 December 2019 11:20:21 +0
| MOOB2S1BIS_IB | SUCCESS | 1430ms
{
  - changed: False
  - ansible_facts: {
    - if_getlivrable_done: apache
  }
}
[11:20:22] COMMON/common_getLivrables : include_tasks ...Monday 30 December 2019 11:20:22 +0100 (0:00:01.492)
| MOOB2S1BIS_IB | SKIPPED | 75ms
[11:20:26] system ... | -- Play recap --
MOOB2S1BIS_IB : ok=10  changed=6  unreachable=0  failed=0
Monday 30 December 2019 11:20:26 +0100 (0:00:03.343)      0:00:25.965 *****
=====
COMMON/common_getLivrables : Unarchive compressed files ----- 4.74s
COMMON/common_getLivrables : include_tasks ----- 3.34s
COMMON/common_getLivrables : Suppression /tmp - Creation /tmp et dossier livraison --- 3.07s
COMMON/common_getLivrables : Create needed unarchived folders ----- 2.64s
COMMON/common_getLivrables : Download artifacts from Nexus release or snapshot repos to local folder --- 2.31s
COMMON/common_getLivrables : delete needed unarchived folders ----- 2.19s
COMMON/common_getLivrables : Sets attributes recursively on distant folder --- 2.11s
Gathering Facts ----- 1.90s
COMMON/common_getLivrables : Download raw files from Nexus raw repo to local folder --- 1.75s
COMMON/common_getLivrables : Set variable if_getlivrable_done ----- 1.49s
COMMON/common_getLivrables : include_tasks ----- 0.12s
COMMON/common_getLivrables : Download npm files from Nexus npm repo to local folder --- 0.09s
```

Jenkins

Jenkins est un serveur Open source permettant l'intégration continue. Il fournit une interface graphique permettant de lancer des jobs exécutant du code écrit en différents langages tels que les scripts linux, du code ansible et s'interface facilement avec Git.

Lors du lancement d'un job Jenkins avec ansible, un agent docker est lancé. Le code ansible utilisé est celui versionné dans la branche SYSTEM_TEAM dont nous avons la charge.

Les jobs peuvent être paramétrés depuis l'Ihm ou directement écrits dans un Jenkins File utilisant le langage Groovy.

Paramètres de déploiement

Nom version applicative	
Ex: MAXCL12_33.SNAPSHOT	
Environnement cible	Nom de la branche sensible
IA	release/continue
Maintenance	Yes
Sauvegarde	Yes
Script MCO	Yes

Version des livrables

- Projet

Composants à installer

FSSO	FAPP	SSO	APP	AUTHOR	PUBLISH
<input type="checkbox"/> fssso_apache	<input type="checkbox"/> fapp_apache	<input type="checkbox"/> sso_jboss	<input type="checkbox"/> app_jboss	<input type="checkbox"/> author_aem	<input type="checkbox"/> publish_aem
	<input type="checkbox"/> fapp_dispatcher	<input type="checkbox"/> sso_openam	<input type="checkbox"/> app_postgres	<input type="checkbox"/> author_appli	
		<input type="checkbox"/> ...	<input type="checkbox"/> ...	<input type="checkbox"/> ...	<input type="checkbox"/> ...

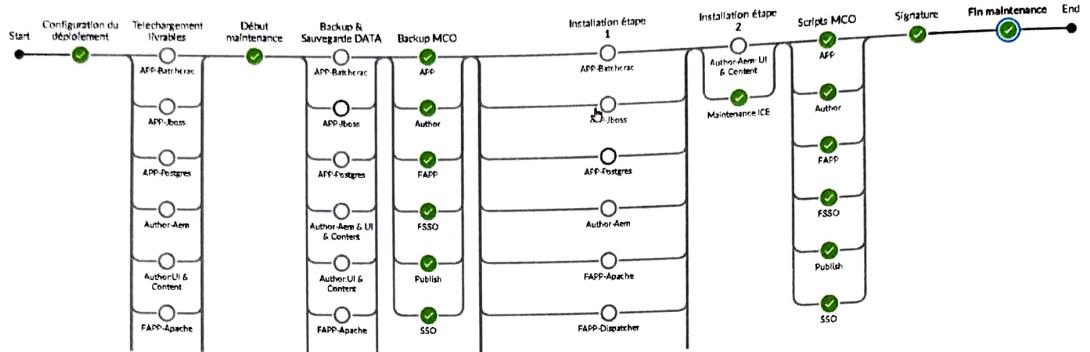
Jenkins va ainsi exécuter commandes ansible execPlaybook selon les paramètres renseignés.

Extrait du Jenkinsfile associé :

Les Variables {Composants} {Environnement} etc. correspondent à celles renseignées dans l'interface

```
//----- Récupération livrable -----
stage("Téléchargement livrables") {
    parallel {
        stage("FSSO-Apache") {
            when {
                expression "${Composants}" == 'fssso_apache'
            }
            steps {
                execPlayBook("${Environnement}-fssso","livrables","mco",env.ExtraVars,"${ansibleUsername}","${ansiblePassword}",execplaybook_vars)
            }
        }
        stage("SSO-Jboss") {
            when {
                expression "${Composants}" == 'sso_jboss'
            }
            steps {
                execPlayBook("${Environnement}-sso","livrables","sso-openam,sso-batchcrac,sso-batchgestionneur,sso-batchsuppressioncomptes,sso-openid,mco",
            }
        }
    }
}
```

Nous pouvons suivre l'avancée via le module Blue Océan :



Linux/Scripts sh

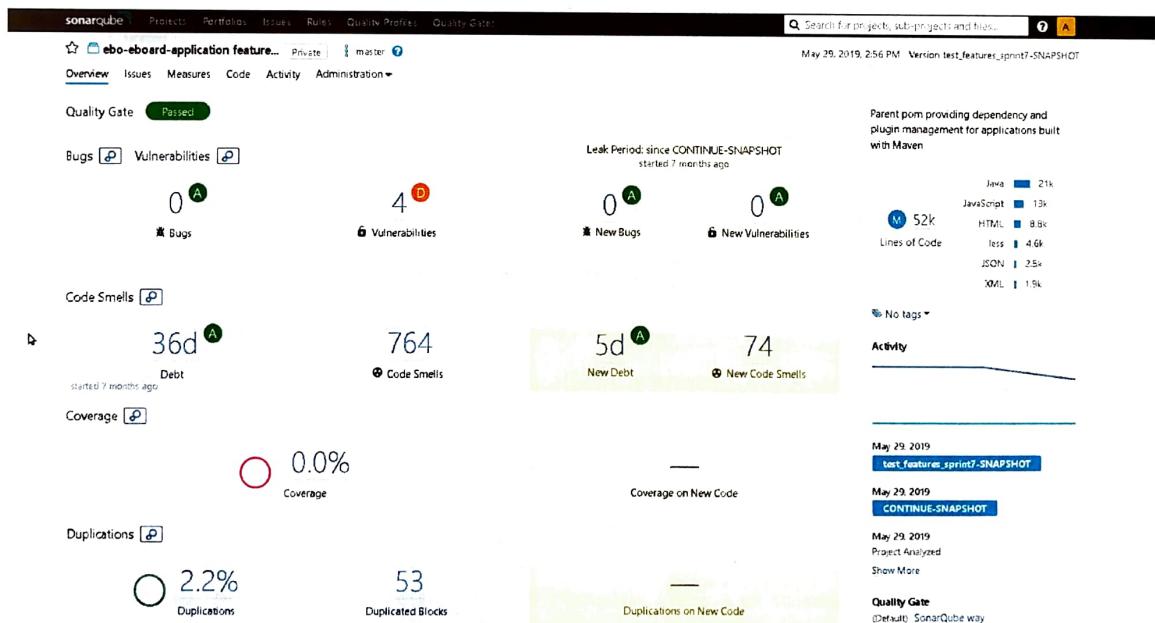
- Page des environnements :
Des scripts lancent des commandes curl permettent de vérifier le fonctionnement (présence d'une réponse).
- D'autres scripts nous permettent de simplifier des tâches quotidiennes (import/export des annuaires ldap, sauvegarde ou purge de la base de donnée...S etc)

Sonar (contrôle qualité)

Lorsque l'on effectue un merge sur la branche release/continue, une analyse sonar du code (duplication de code, vulnérabilités connues etc) et packaging sont lancée via un job jenkins par un trigger, ainsi que Si les analyses sont OK ,

The screenshot shows a Jenkins pipeline stage named "Merge branch 'feature/MAXCLI-3...' into release/continue". The stage has three steps: "Analyse Sonar", "Compilation", and "Dépot des livrables", all of which have been successfully completed (indicated by green checkmarks). The commit hash "a87cfdb3" is also visible.

Page d'accueil de sonarqube :



Nexus repositories

Les livrables sont hébergés sur un dépôt nexus accessible depuis le réseau interne pour les utilisateurs autorisés.

The screenshot shows the Sonatype Nexus Repository Manager interface. On the left, the 'Browse' view displays a tree structure of artifacts under the 'ice-ice-maven-releases' repository. The tree includes common, etc, ice, ice-test, ice-pair, ice-pair-env, and ice-pair-tag nodes, with various sub-folders and files like 'ice-page-env-19.9.0.16-pageEnv.zip'. On the right, a detailed view of the 'ice-page-env-19.9.0.16-pageEnv.zip' artifact is shown in a modal window. The details include:

- Repository:** ice-ice-maven-release
- Format:** element2
- Component Group:** [REDACTED]
- Component Name:** ice-page-env
- Component Version:** 19.9.0.16
- Path:** [REDACTED]
- Content type:** application/zip
- File size:** 18 MB
- Date created:** Thu Dec 19 2019 16:04:00 GMT+0100 (heure normale d'Europe centrale)
- Date updated:** Thu Dec 19 2019 16:04:03 GMT+0100 (heure normale d'Europe centrale)
- Last downloaded:** has not been downloaded
- Last modified:** 19.9.0.16
- Blob references:** maven-repository@E3E9C9F [REDACTED]
ES-4162-4253-5b3d8c184476
- Containing repos:** ice-ice-maven-release
- Uploader:** FADEV007
- Uploader's IP Address:** 10.29.32.118

GIT

Les Projets sont hébergés sur un serveur interne Gitlab.

The screenshot shows the GitLab interface for the 'ice-ice/_SYSTEM_TEAM_' repository. The left sidebar shows project navigation options like Project, Repository, and Merge Requests. The main area displays the repository structure and commit history. A 'Switch branch/tag' dropdown is open, showing available branches and tags. The commit history table includes columns for Last update, Commit ID, and Author. Key commits shown include:

Last update	Commit ID	Author
1 week ago	ee385716	[REDACTED]
1 week ago	feature/MAXCLL2-104_parallelRobotProd	[REDACTED]
1 week ago	feature/DLATLAT-45_unique_uid	[REDACTED]
2 years ago	release/continue	[REDACTED]
2 years ago	release/19.9.0.16	[REDACTED]
2 years ago	feature/MAXCLL2_128_M3_GTP	[REDACTED]
1 year ago	feature/MAXCLL2_128_M3_Available	[REDACTED]
1 year ago	ajout d'un cas d'exception	[REDACTED]
2 years ago	gitignore	[REDACTED]
2 years ago	[REDACTED] Add specific user agent for replications tasks	[REDACTED]

L'équipe Matrice est en charge de 2 repositories pour l'application ICE et 2 pour l'application Eboard.

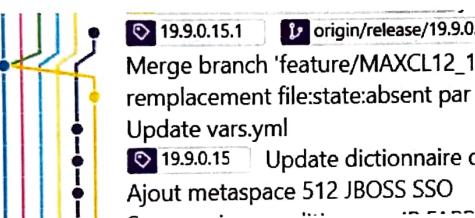
- ICE : SystemTeam, MCO
- Eboard : Applications, Devops

Lorsqu'on développe une feature, on crée une branche en local à partir de la branche release/continue, que l'on nomme feature/{numero du ticket}_{Description brève de la feature}

Par la suite, une fois notre feature développée nous poussons notre branche sur le dépôt distant, puis nous mettons notre ticket Jira associé dans la colonne review et préparons une merge request en vue de merger notre branche dans release continue si elle passe les tests.

Un autre développeur de l'équipe s'occupe de vérifier notre fonctionnalité et si tout est ok, il effectue le merge.

Exemple Sur Release Continue :

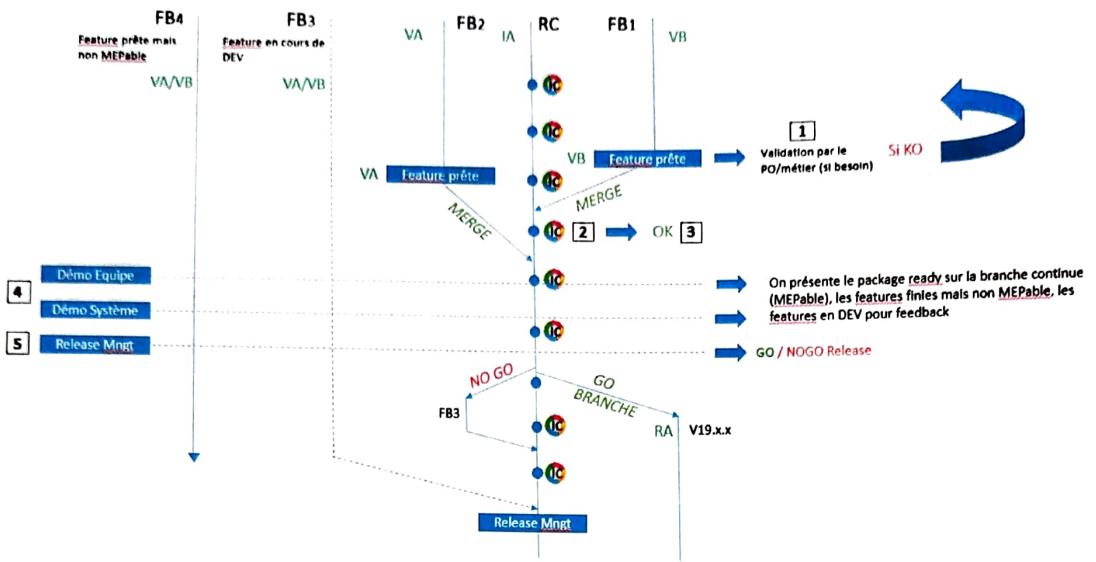


19.9.0.15.1	origin/release/19.9.0.15	Update dictior	4 déc. 2019 11:22	jenkins <jenkins-2-devop>
Merge branch 'feature/MAXCL12_167_verify_checksum'		remplacement file:state:absent par verify_checksum de	4 déc. 2019 09:06	JEAN-CHRISTOPHE AZZO <antho>
		Update vars.yml	4 déc. 2019 09:06	ANTHONY JOVER <antho>
19.9.0.15		Update dictionnaire des versions ICE	3 déc. 2019 04:53	GEOFFREY BOIS <geoffre>
Ajout metaspace 512 JBOSS SSO			3 déc. 2019 11:44	jenkins <jenkins-2-devop>
			3 déc. 2019 10:30	bois geoffrey <geoffrey-e>

Gitflow

Lors d'une phase de MEP (mise en production), Tous les projets impactés créent une branche release-{numero version} que nous installons sur l'environnement de recette via un job Jenkins en version -{numero version}-BETA1. Si nous constatons des régressions ou des bugs, les correctifs sont apportés et nous installons à nouveau une version BETA2...etc (Cf Intégration continue). Lorsque tout est ok nous installons la version release finale d'abord sur l'environnement de recette, puis lors d'une mep a blanc sur l'environnement CP (copie prod) et enfin sur l'environnement de production. Lors des différentes phases, les job Jenkins créent des tags dans git permettant de retrouver les versions BETA et release finale précédentes.

Si nous avons effectués des nouveaux commits sur la branche release afin d'effectuer des correctifs, nous pouvons les rapatrier sur la branche release (par exemple via des cherry-pick) afin de les embarquer par la suite.



EXPLICATION DÉTAILLÉE DU PROCESSUS

1) Le PO (Product Owner) valide les features dès qu'elles sont prêtes:

- Il effectue ses tests d'acceptance
- Il s'assure que les TU (tests unitaires) ont été joué
- Il vérifie que la feature répond au besoin métier (si besoin ou doute, il demande une validation du métier)

Une fois ces critères remplis, il donne le **GO** pour le merge sur la **Release Continue** si la feature est éligible pour la prochaine MEP.

2) L'intégration continue, qui tourne pour le moment tous les jours, à heures fixes, va tester la qualité du code récemment mergé (dette technique, performances, tests de non-régressions).

3) Une fois que l'intégration continue a testé et validé le code (pas de KO dans les jobs Jenkins), la feature est **DONE**.

4) On présente en démo:

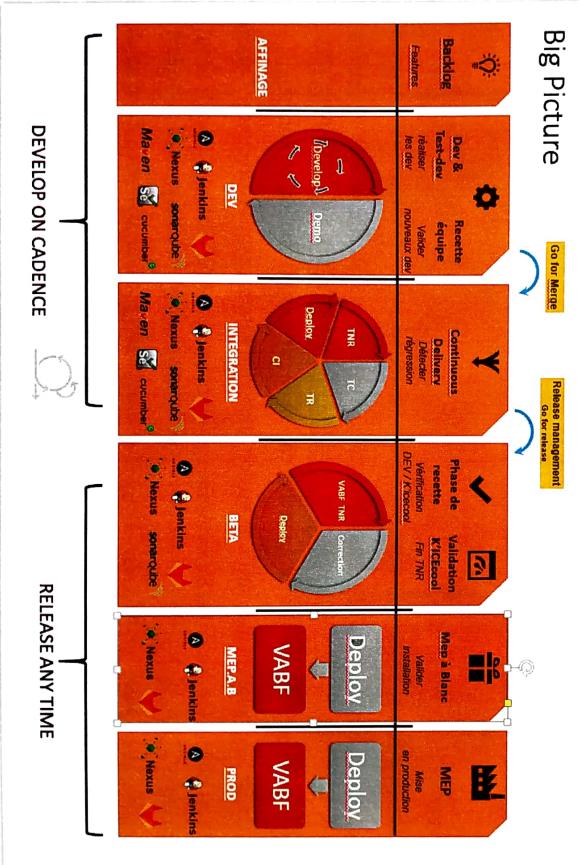
- Ce qui est fait et mergé => sur la release continue, environnement IA

- Ce qui est fait mais pas mergé (on ne veut pas envoyer en PROD pour des raisons commerciales par exemple) => sur un environnement de DEV, VA/VB
- Ce qui est en cours de développement (on souhaite un retour PO/Métier) => sur un environnement de DEV, VA/VB

5) En release management, une seule question: GO for PROD?

- On présente ce que l'on a sur la release continue:
 - Si les features présentes sont suffisantes, **GO for PROD**.
 - Si manque des choses, ou bien certaines choses sont à revoir, **NO GO**, pas de MEP => on refera un point lors de la prochaine release management (1 par semaine).

Intégration continue



Tout les matins à 6h, un job Jenkins se charge d'installer sur les environnements IA d'intégration continue.

les livrables (release-SNAPSHOT) créés à partir de release continue, et on lance toute une série de tests:

- des TNR automatisés pour détecter les régressions fonctionnelles.
- des tests de qualité de code
- des tests de sécurité

L'objectif de ce processus de merge est de détecter le plus tôt possible les régressions fonctionnelles, les régressions de performance et les régressions de qualité de code.

Quand l'intégration continue est KO, il y a besoin de réactivité pour corriger et stabiliser l'environnement de qualité.

L'équipe matrice à la responsabilité de la disponibilité de l'environnement d'intégration IA.

A ce niveau, si les TNR sont passants alors l'incrément est releasable.

Delivery

Lors de la release management, toutes les features DONE et présentes sur l'environnement d'intégration continue sont listées. Le métier choisit de partir en production ou non, si l'incrément de qualité est jugé suffisant ou doit attendre d'autres features/corrections. Lorsque le GO est donné, chaque projet impacté crée une branche release (cf GitFlow) à partir de la branche Release Continue.

Notre équipe récupère le périmètre à installer et nous créons les livrables en version *version_release-BETA* et les déployons sur l'environnement de recette RA. S'en suit une période de tests de 10jours selon un calendrier que nous établissons

Notre Branche _SYSTEM_TEAM_ est systématiquement impactée, car elle Contient les outils nécessaires à la compilation, aux tests, au packaging et au déploiement de l'application

de manière automatisés (Code Ansible, JenkinsFiles, etc...) ainsi que les variables ou sont renseignées les versions des briques logicielles livrées.

	L	M	M	J	V	S	D	L	M	M	J	V	S	D	L	M	M
Planning																	
GO pour MEP																	
Merge et création des branches releases																	
MR7																	
TNR																	
Tag release																	
MEP BLANC																	
VABF MOA																	
MEP																	
VABF MOA																	

Les bugs sont détectés pendant une période de 10 jours et nous créons une nouvelle Beta à chaque livraison de correctif (Beta2, Beta3, etc). Une fois que tout est corrigé, nous créons le livrable de production, la release finale.

Lors d'une dernière phase nous livrons une dernière fois la release finale sur RA (via Jenkins). En cas de bug bloquant (ce qui ne devrait pas être le cas grâce à l'intégration continue), le PM peut arrêter le processus et donner le NO GO pour la MEP. C'est le dernier moment où il peut le faire, si nécessaire.

MEP à Blanc

On répète la PTI d'installation On effectue une VABF poussée de l'application pour vérifier que tout est OK. Le jour de la MEP, les mêmes actions seront répétées.

MEP

En dehors de la plage horaire de disponibilité de l'application, nous installons notre release finale en production avec un infogérant (via le même job Jenkins) et on veille à ce qu'il effectue les bonnes actions.

Nous effectuons ensuite une dernière fois les vérifications nécessaires (VABF)

Nous envoyons un compte rendu une fois les actions terminées.

Perspectives :

Interface pour visualiser le suivi de l'état d'une MEP implémentant l'envoie de mail de CR

Spécifications

- Affiche les différentes étapes de la Mep
- On peut saisir le numéro de la version et ça met à jour le titre
- On peut saisir le numéro de la beta et ça met à jour le label de l'étape version de beta
- Les données sont stockées dans un json

Automatisation de la Vabf ICE avec UIpath

Lors d'un Technical Meeting (organisé à Capgemini tous les Jeudi) il nous a été présenté l'outil UIPath , permettant d'automatiser la navigation sur un site web. Il a été décidé et validé le choix d'automatiser la VABF ICE en utilisant cet outil.

Bilan

Bilan du projet

Les développements continuent.

- Décommissionnement des 2 applications prévu pour début 2021.
- Participation de l'équipe MCO aux prochains travaux en vue du décommissionnement.
Ex: Fusion de certaines briques applicatives avec d'autres projets.

Bilan Personnel

Première expérience professionnelle dans l'informatique, ce stage m'a permis une montée en compétence à la fois dans le développement et également côté opérationnel.

Ayant de l'appétence pour cette double compétence j'ai très apprécié d'être intégré dans une équipe multidisciplinaire. En effet certains membres du bureau étant plus spécialisé développement, et d'autre en gestion de base de donnée, et d'autres administration serveurs cela permet une véritable synergie.

L'entreprise m'a fait confiance et proposé un contrat d'embauche en CDI. C'est donc avec joie que je souhaite continuer l'aventure au sein de cette équipe.

BRADLEY

BRADLEY