

Documentation de déploiement

PROJET WILD ADVENTURES

Benoît Aubin / Nicolas Bouème
OPEN CLASSROOMS | B.AUBIN95@GMAIL.COM

Table des matières

1	Rappel sur le déploiement	2
2	Environnement d'intégration continue :	4
2.1	Généralités	4
2.2	Machine de déploiement :	4
2.2.1	Démarrer la machine Wild adventures	5
2.2.2	Accéder à la machine par son nom de domaine	6
2.2.3	Accéder à la machine en SSH.....	7
2.3	Démarrer les outils d'intégration continue :	8
2.3.1	Jenkins	8
2.3.2	SonarQube.....	9
3	Déploiement.....	11
3.1	Prérequis	11
3.2	Organisation du déploiement	11
3.3	Lancement du déploiement d'un composant	12

1 Rappel sur le déploiement

L'application Wild adventures est composée en deux briques logicielles distinctes :

- Une partie frontend développée en React JS
- Une partie backend structurée en micro services.

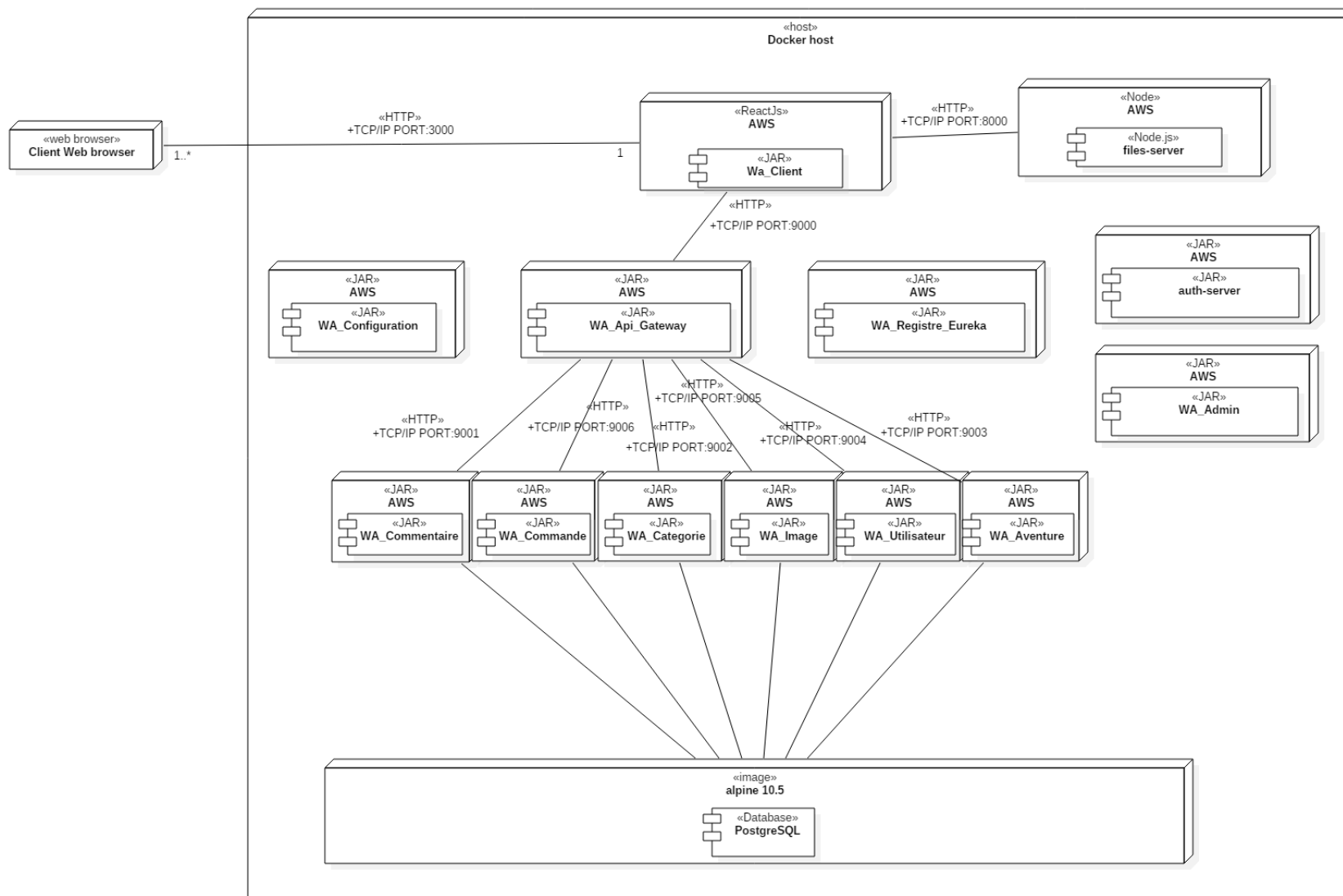
Les micro services sont réparties en deux parties :

- Les micro services métiers contenant la logique métier de l'application (exposition des services REST, règles de gestion...).
- Les micro services edge contenant les outils permettant de rendre l'API scalable, d'assurer la sécurité des accès et d'administrer plus simplement l'API.

Les grands principes de déploiement (port, protocole et différentes briques de l'architecture micro service...) sont détaillés dans le diagramme UML de déploiement ci-dessous.

Brique logiciel	Port par défaut	Rôle	Nom du build
Serveur de configuration	9101	Récupère les fichiers de configurations hébergés dans le cloud sur github. Permet ainsi la mutualisation de la configuration	WA-MSEConfig
Serveur d'annuaire Eureka	9102	Répertorie les instances des différents micro services démarrés	WA-MSEEureka
Gateway Zuul	9000	Gestion des accès et load balancing	WA-MSEGateway
Serveur d'authentification	9104	Gestion de l'authentification des utilisateurs	WA-MSEAuth
Serveur d'administration	9103	Administration de l'API (monitoring)	WA-MSEAdmin
MSM ¹ Gestion des aventures	9003	Appel CRUD et règles de gestion pour les aventures	WA-MSMAdventure
MSM ¹ Gestion des catégories	9002	Appel CRUD et règles de gestion pour les catégories	WA-MSMCategory
MSM ¹ Gestion des commentaires	9001	Appel CRUD et règles de gestion pour les commentaires	WA-MSMComment
MSM ¹ Gestion des images	9005	Appel CRUD et règles de gestion pour les images	WA-MSMImage
MSM ¹ Gestion des commandes	9006	Appel CRUD et règles de gestion pour les commandes	WA-MSMOrder
MSM ¹ Gestion des utilisateurs	9004	Appel CRUD et règles de gestion pour les utilisateurs	WA-MSMUser
Serveur d'upload d'image		Gestion des uploads d'images	WA-Node-Server
Interface client	3000	Interface utilisateur front	MSC-React-Front

¹ Micro service Métier



2 Environnement d'intégration continue :

2.1 Généralités

L'application a été développée et déployée à l'aide d'un environnement d'intégration continue constitué d'outils simplifiant sa compilation, son déploiement et assurant la qualité du code et la sécurité de l'application. Son déploiement et sa mise en service sont assurés sur l'outil de PAAS Amazon web services. Un nom de domaine a été réservé pour l'application il s'agit de wild-adventures.fr. Les outils utilisés sont les suivants :

- Jenkins : plateforme d'intégration continue par excellence jenkins récupère le code sur Github, pilote le build et le lancement des tests unitaires, transmet les rapports d'exécution à SonarQube et lance le déploiement des différentes briques logicielles à la fin de leur build respectif.
- SonarQube : Plateforme de reporting de la qualité du code ainsi que de celle des tests, sonarqube est un outil d'analyse puissant permettant de détecter failles de sécurité, amélioration de code possible (répétition, erreur, simplification...) ainsi que zone de code non testé.
- Github : Outil de versioning en ligne, github permet d'assurer la mutualisation du code sur une plateforme unique. Couplé à un développement git en local cette plateforme simplifie le développement et la communication entre les différents développeurs du projet. Le code de l'application est réparti dans deux repository : Le premier nommé wild-adventures-app contient le code à proprement parlé de l'application tandis que le second wild-adventures-conf contient les différents fichiers de configuration des micro services.
- AWS : Outil de Platform as a Service, Amazon web services a été utilisé pour héberger l'application ainsi que les différents outils d'intégration continue décrit plus haut.

Outils	Port	Adresse complète d'accès
Jenkins	8080	http://www.wild-adventures.fr:8080
SonarQube	9099	http://www.wild-adventures.fr:9099
Github Application	(Externe à AWS)	https://github.com/nboueme/wild-adventures-app
Github Configuration	(Externe à AWS)	https://github.com/nboueme/wild-adventures-conf

2.2 Machine de déploiement :

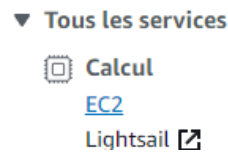
Comme vu précédemment la machine de déploiement est la même que la machine hébergeant l'environnement d'intégration continue. Pour assurer les deux rôles cette machine doit posséder des capacités hardware assez élevées. Ainsi l'application est hébergée sur une machine AWS de type t2.xlarge :

Caractéristiques de la machine	Valeur
Nombre max de vCPU	4
RAM	16GB
Stockage	50Go
Performances réseau	Modéré

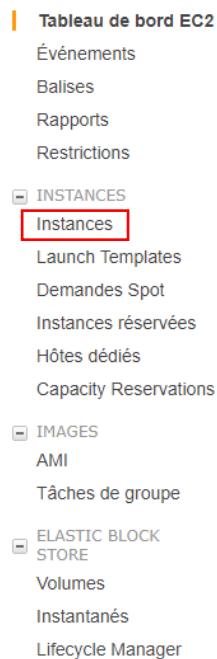
2.2.1 Démarrer la machine Wild adventures

Pour démarrer la machine wild adventures de déploiement et d'intégration continue il suffit de se connecter sur le compte AWS et de suivre les instructions suivantes :



- Dans la console AWS cliquer sur **Ec2** dans les services de calcul.



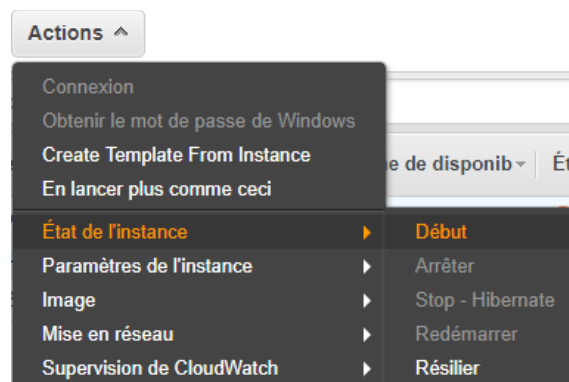
- Dans le tableau de bord choisissez **instances** dans la barre d'outils à gauche.



- La liste d'instances s'affichent alors sélectionner l'instance nommée WildAdventures

	WildAdventures	i-074a4233f626b37c1	t2.xlarge	eu-west-3c	 stopped
---	----------------	---------------------	-----------	------------	---

- Si l'état de l'instance est stopped il faut la démarrer en cliquant sur l'onglet actions -> état de l'instance -> début.



- Confirmer le démarrage de la machine et attendé que l'état de l'instance passe de pending à running.

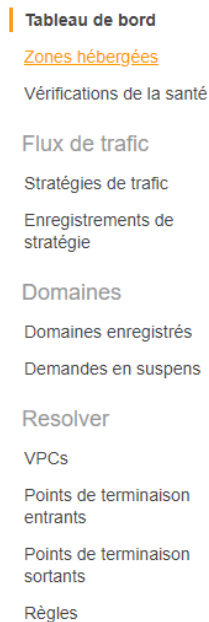
2.2.2 Accéder à la machine par son nom de domaine

Pour accéder à la machine via le nom de domaine wild-adventures.fr il faut préciser sur quel adresse ip Publique le nom de domaine doit pointer. Pour cela copier l'ip publique IPV4 de la machine dans l'onglet de description de l'instance.

- Rendez-vous ensuite dans la console AWS et choisissez route 53



- Sur le tableau de bord Route 53 choisir dans la barre d'outil à gauche l'onglet zones hébergées



- Dans la liste des zones hébergées qui s'affichent cliquer sur le nom de domaine wild-adventures.fr

Nom de domaine	Type	Nombre de jeux d'enregistrements
wild-adventures.fr	Public	4

- Dans la page qui s'ouvre sélectionner le nom [www.wild-adventures.fr](#)

Nom	Type	Valeur	Évaluer l'état de la cible	ID de vérification de l'état	Durée de vie	Région	Poids	Géolocalisation
<input type="checkbox"/> wild-adventures.fr	NS	ns-1930.awsdns-49.co.uk. ns-592.awsdns-10.net. ns-318.awsdns-39.com. ns-1065.awsdns-05.org.	-	-	172800			
<input type="checkbox"/> wild-adventures.fr	SOA	ns-1930.awsdns-49.co.uk. awsdns-hostmaster.ama	-	-	900			
<input type="checkbox"/> tech.wild-adventures.fr	A	52.47.194.163	-	-	300			
<input checked="" type="checkbox"/> www.wild-adventures.fr	A	35.180.135.113	-	-	300			

- Dans la barre d'outils à droite coller l'IP publique de la machine wild adventures et cliquer sur enregistrer le jeu d'enregistrements.

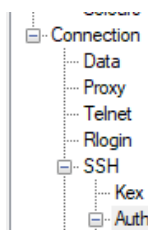
- La machine est maintenant accessible via l'url www.wild-adventures.fr

2.2.3 Accéder à la machine en SSH

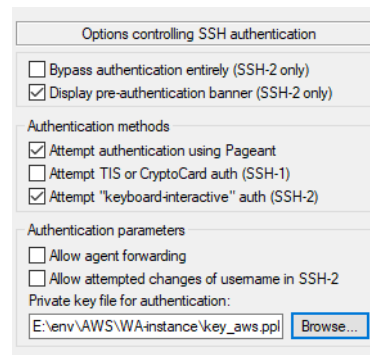
Pour accéder à la machine en ligne de commande sous windows installer l'outil putty. Suivre les étapes suivantes pour configurer la connexion en SSH :

- Démarrer putty.
- Rentrer dans le champs Host Name ec2-user@www.wild-adventures.fr et 22 dans le champ port. Enfin sélectionner le type de connexion SSH.

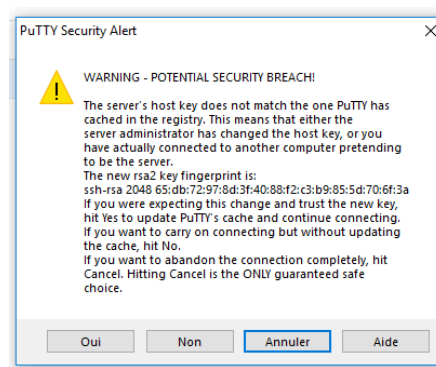
- Pour accéder à la machine AWS utilise un système de clés publiques clés privés. Pour configurer l'authentification se rendre dans la barre d'outils de gauche de putty et sélectionner **connection -> SSH -> auth**



- Dans la fenêtre principale de putty dans le champ **Private key file for authentication** sélectionner le fichier de clé privé sur votre poste



- Retourner ensuite dans l'onglet session de putty et cliquer sur **open**.
- Une fenêtre de sécurité s'ouvre. Ignorer là en cliquant sur **oui**.



- La fenêtre de putty se transforme alors en terminal linux dans lequel vous pouvez écrire des commandes unix qui seront exécuté sur la machine AWS...

```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Fri Dec 28 07:27:18 2018 from 1fbn-1-3099-75.w90-79.abo.wanadoo.fr

 _ _ | _ _ )
 _ | ( _ /
 _ | _ _ |
      Amazon Linux AMI

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
[ec2-user@ip-172-31-42-66 ~]$ ps aux | grep java
```

2.3 Démarrer les outils d'intégration continue :

2.3.1 Jenkins

Jenkins par défaut démarre avec la machine. Néanmoins lors d'un premier démarrage il peut être nécessaire de démarrer le service jenkins en ligne de commande. Pour ce faire il suffit de suivre les étapes suivantes :

- Se connecter à la machine en SSH (voir partie précédente).
- Vérifier que jenkins n'est pas démarré avec la commande : **ps aux | grep jenkins**. Si le résultat contient la figure suivante jenkins est déjà déployé

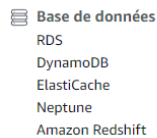
```
/usr/lib/jenkins/jenkins.war
```

- Pour démarrer le service jenkins taper la commande : **sudo service jenkins start**
- Répéter l'étape précédente pour vérifier que jenkins est bien démarré. Vous pouvez alors vous connecter à jenkins à l'adresse <http://www.wild-adventures.fr:8080> .

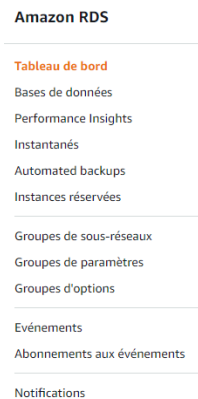
2.3.2 SonarQube

2.3.2.1 Prérequis -> Base de données mysql

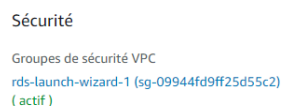
- Copier l'ip privée de la machine ec2 dans l'onglet de description de la machine.
- Rendez-vous dans la console AWS et cliquer sur RDS dans la partie base de données



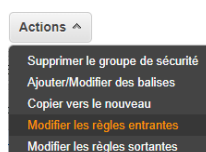
- Dans la barre d'outils de gauche du tableau de bord RDS sélectionner base de données.



- Dans la liste des bases qui s'affiche cliquer sur sonarqube-sonar
- Vérifier que la base de données est marquée comme disponible sinon cliquer sur **actions -> démarrer**.
- Une fois la base démarrée dans la section **connectivité -> sécurité -> Groupes de sécurité vpc**



- Dans la fenêtre qui s'ouvre cliquer sur **actions-> modifier les règles entrantes**



- Dans le champ source de la ligne de description sonarqube changer l'ip par l'ip de la machine que vous avez copiée.

Modifier les règles entrantes ✕

Type ⁱ	Protocole ⁱ	Plage de ports ⁱ	Source ⁱ	Description ⁱ
MYSQL/Auror	TCP	3306	Personnali: 90.79.116.75/32	Par exemple, SSH for Admin De ✕
MYSQL/Auror	TCP	3306	Personnali: 172.31.42.66/32	Sonarqube ✕

[Ajouter une règle](#)

REMARQUE : Les modifications apportées à des règles existantes se traduiront par la suppression de la règle modifiée et par la création d'une nouvelle règle avec les nouveaux détails. Le trafic lié à cette règle sera alors abandonné pendant un temps très limité jusqu'à ce que la nouvelle règle puisse être créée.

[Annuler](#) [Enregistrer](#)

Cette procédure est nécessaire dans le cas où la machine a été redémarré et que son IP privé a changé.

2.3.2.2 Démarrage de sonarqube

Sonarqube ne démarre pas avec l'instance ec2 il sera surement nécessaire de le démarré manuellement avec la procédure suivante :

- Se connecter à la machine en SSH (voir partie précédente).
- Vérifier que sonarqube n'est pas démarré avec la commande : **ps aux | grep sonar**. Si le résultat contient la figure suivante sonarqube est déjà déployé

```
sonar 4854 0.0 0.0 17872 1792 ? S1 22:24 0:00 /opt/sonar/bin/linux-x86-64/./wrapper
```

- Pour démarrer le service sonarqube taper la commande : **sudo service sonar start**
- Répéter l'étape précédente pour vérifier que sonarqube est bien démarré. Vous pouvez alors vous connecter à sonarqube à l'adresse <http://www.wild-adventures.fr:9099> .

2.3.2.3 Configurer sonarqube :

Pour changer la configuration de connexion à la base de données de sonarqube ou pour modifier le port utiliser par sonarqube il peut être nécessaire de modifier le fichier de propriétés sonar.properties (/opt/sonar/conf/sonar.properties). Ceci peut être réaliser simplement en utilisant l'outil vi en ligne de commande : **sudo vi /opt/sonar/conf/sonar.properties**

2.3.2.4 Accéder aux logs de Sonar.

Pour consulter les logs généraux de sonar le fichier sonar.log (/opt/sonar/logs/sonar.log) peut être affiché à l'aide de la commande : **sudo cat /opt/sonar/logs/sonar.log**

Pour consulter les logs du serveur web de sonarqube le fichier web.log (/opt/sonar/logs/web.log) peut être affiché à l'aide de la commande : **sudo cat /opt/sonar/logs/web.log**

Le niveau de log de sonar peut être adapté à tout moment dans le fichier sonar.properties mais occasionnera un redémarrage du service sonar.

3 Déploiement

3.1 Prérequis

Pour fonctionner l'application repose sur une base de données packagé dans une image docker. Cette image doit être démarré et opérationnelle avant d'effectuer les opérations de déploiement (les micro services métiers notamment ne démarreront pas sans cela). Le démarrage de cette base n'est pas automatisé étant donné que le déploiement de chaque composant de l'application se fait indépendamment des autres. Pour démarrer l'image docker de la base de données suivre la procédure suivante :

- Se connecter à la machine en SSH (voir précédemment).
- Se placer dans le répertoire **/home/ec2-user/déploiement/wild-adventures-app** (avec la commande **cd**).
- Vérifier que le répertoire est à jour avec la commande avec la commande **git pull origin master**
- Se placer dans le répertoire **/home/ec2-user/déploiement/wild-adventures-app/docker** (avec la commande **cd**).
- Pour démarrer la base de données lancer la commande **docker-compose up -d**
- Pour arrêter l'image de la base de données si besoin il suffira de lancer la commande **docker-compose rm -v**.

3.2 Organisation du déploiement

Le déploiement des différents composant s'effectue via Jenkins. Le build jenkins si ce dernier se déroule sans erreur va déployer le composant correspondant. Il nécessite un ordre plus ou moins établi puisque la plupart des micro services nécessite le démarrage préalable d'un ou plusieurs micro services edge.

Il est ainsi nécessaire de déployer en premier lieu les micro services edge de configuration et d'annuaire (Eureka) pour garantir un démarrage optimal des autres micro services. Voici un exemple d'ordre de déploiement type :

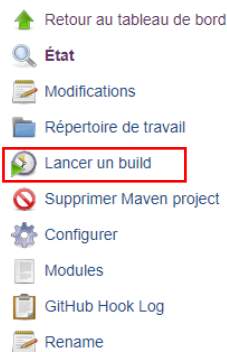
Numéro de l'étape	Nom de l'étape	Brique logiciel déployé	Nom des builds à appeler
1	Déploiement du serveur de config	ConfigServer.jar	WA-MSEConfig
2	Déploiement du serveur eureka	EurekaServer.jar	WA-MSEEureka
3	Déploiement des microservices métiers	Waadventure.jar, Wacategory.jar, Wacomment.jar, Waimage.jar, Waorder.jar, Wauser.jar	WA-MSMAdventure, WA-MSMCategory, WA-MSMComment, WA-MSMImage, WA-MSMOrder, WA-MSMUser
4	Déploiement des microservices edge restant	AuthServer.jar, ZuulServer.jar, AdminServer.jar	WA-MSEAuth, WA-MSEGateway, WA-MSEAdmin
5	Déploiement du serveur d'upload node	Server.js	WA-Node-Server
6	Déploiement du front	App.js	MSC-React-Front

Cet ordre est conseillé mais n'est pas obligatoire. En effet les micro services peuvent fonctionner indépendamment les uns des autres. Ceci est juste un ordre logique permettant d'éviter des erreurs peut parlantes au déploiement qui peuvent être dû à des erreurs de contact d'un autre micro service.

3.3 Lancement du déploiement d'un composant

Pour lancer le build et donc le déploiement d'une brique il suffit de se logger sur jenkins à l'adresse suivante <http://www.wild-adventures.fr:8080> et de suivre la procédure suivante :

- Se connecter à jenkins
- Sur la page d'accueil sélectionner le build à exécuter en cliquant sur le nom du projet
- Dans le menu de gauche du récapitulatif du projet cliquer sur lancer le build :



- Une fois le build achevé le composant devrait être déployé si tout s'est bien passé (un build effectué avec succès est marqué d'un point bleu).

 **#13** 28 déc. 2018 07:31

- Pour vérifier qu'un composant est bien déployé on peut utiliser la commande suivante : **ps aux | grep NOM_DU_COMPOSANT**