

PROJET MYCELIUM V2.0

Rapport de pré-étude et spécifications

21 novembre 2022

BOIZUMAULT Maïwenn
CAROFF Nicolas
JULIEN Léo-Paul
RAOUL Thibaut
TUMOINE Ivy
VILLE Grégoire

Sous la direction de PARLAVANTZAS Nikolaos.

Avec la participation de MOUREAU Julien et
LONGUEVERGNE Laurent (Université de Rennes 1).



Table des matières

1	Introduction	2
1.1	Contexte	2
1.2	Solution initiale	3
1.3	Objectifs de la nouvelle solution	4
2	État de l'art	5
2.1	LivingFog Platform Valence	5
2.2	Smart Street Lighting Montevideo	6
2.3	ConnecSens-2	7
2.4	WildCount	8
3	Première version et transfert de compétences	10
3.1	Technologies utilisées dans Mycélium 1.0	10
3.1.1	Fog Computing	10
3.1.2	ChirpStack	11
3.1.3	InfluxDB	12
3.1.4	OpenFaaS	13
3.2	Fonctionnalités existantes de Mycélium 1.0	13
3.2.1	Scénarios	13
3.2.2	Noeud SoLo	14
3.3	Prise en main de l'existant	16
4	Spécifications générales et fonctionnelles	18
4.1	Nouveaux scénarios	18
4.1.1	Normales saisonnières	18
4.1.2	Intempéries	18
4.2	Nouveaux types de capteurs	19
4.2.1	Recherche de faune sauvage : les flux vidéo	19
4.2.2	Capteur de mesure de l'eau avec connectivité GSM	19
4.2.3	Développement de la scalabilité sur le Cloud	19
4.2.4	Maintenabilité	20
5	Architecture globale	21
5.1	Architecture matérielle	21
5.2	Architecture du cluster	22
5.3	Architecture logicielle	24
6	Conclusion	28
7	Bibliographie	30

1 Introduction

1.1 Contexte

La nouvelle ligne B de métro rennaise facilite grandement la vie de ses habitants, leur permettant de circuler facilement au sein de la ville. Cependant, la construction de cette ligne a généré des émissions de carbone et une dénaturation, c'est pourquoi la ville de Rennes s'est engagée à compenser cet impact environnemental en renaturant certains espaces verts. Le projet Mycélium 2.0 se focalise sur la renaturation de la Croix Verte, située sur le campus de Rennes 1, face à l'INSA (*figure 1*). La restauration de cette zone vise à remonter l'eau d'une rivière souterraine vers trois points d'eau de la Croix Verte. En collaboration avec le laboratoire Géosciences Rennes et suite à leur demande, Mycélium 2.0 vise à équiper cette zone en capteurs, afin d'obtenir des statistiques sur la réponse du milieu au changement et au forçage climatique.

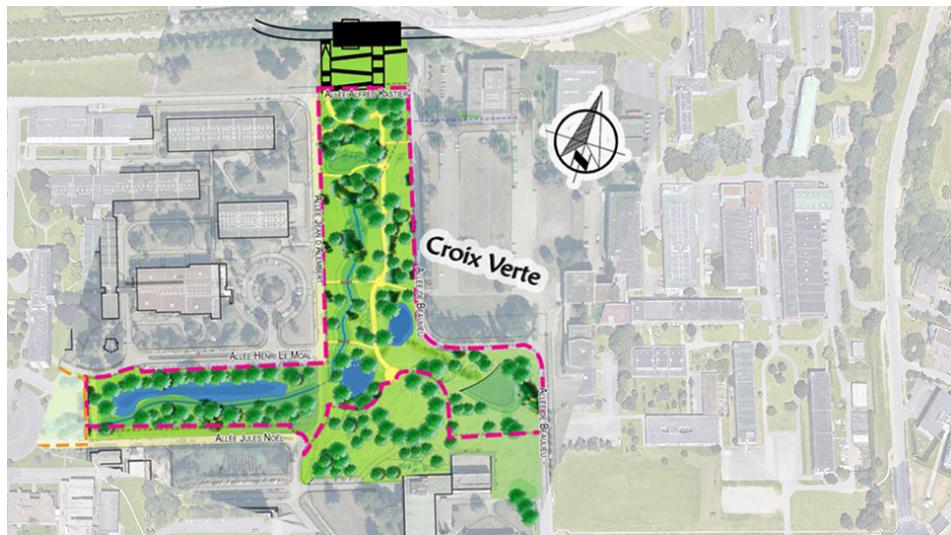


FIGURE 1 – Représentation cartographique de la Croix Verte

A plus grande échelle, notre travail s'intègre au vaste projet de suivi environnemental TERRA FORMA[1], coordonné par Laurent LONGUEVERGNE. Projet déployé nationalement, il permettra de surveiller notre capital sol et eau, ainsi que récolter des données liées à la biodiversité et à la pression chimique et entropique engendrée par l'être humain. Ultimement, TERRA FORMA cherche à partager les informations obtenues sur les différents sites d'observation, correspondant à différents milieux. Trois sites pilotes existent déjà : le col du Lautaret dans les Alpes, l'Auradé dans le Gers, et Ploemeur-Guidel dans le Morbihan. Douze autres, parmi lesquels figure la Croix Verte, sont en cours d'équipement.

1.2 Solution initiale

L'objectif du projet Mycélium 1.0 était de déployer un système de contrôle d'un réseau de capteurs intelligents et à basse énergie sur le site de la Croix Verte. Cela a dicté l'architecture globale de la première version de Mycélium (*figure 2*). Un seul nœud de capteurs a pour l'instant été déployé. Tout d'abord des capteurs basse consommation sont chargés de récolter des informations sur l'environnement de la Croix Verte. Ces capteurs répondent aux contraintes de l'environnement : ils sont étanches et fonctionnent dans les plages de températures de la région.

Ensuite, les mesures de ces capteurs sont agrégées et envoyées via le protocole de communication LoRaWAN, qui présente les avantages d'être peu consommateur en énergie et d'avoir une portée de plusieurs kilomètres. Ces données sont alors reçues par une gateway LoRaWAN puis transmises à un nœud de calcul et de traitement proche de la Croix Verte, ce qui permettra dans le cas d'une zone isolée d'être indépendant de la connexion internet. Cette gateway et le nœud forment un « Collecteur ».

Enfin, ce nœud de calcul et de traitement est connecté au Cloud pour les traitements plus complexes, comme l'envoi de notifications en cas de mesures particulières. Cette architecture correspond à une architecture dite « Fog », c'est-à-dire qu'elle étend le Cloud au plus proche de l'utilisateur [2].

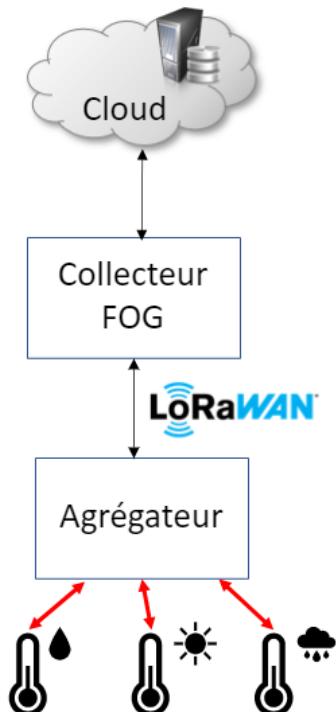


FIGURE 2 – Architecture globale de Mycélium 1.0

1.3 Objectifs de la nouvelle solution

L'objectif de Mycélium 2.0 est d'améliorer le système en place et de l'étendre en prenant en compte différentes contraintes. Une faible consommation est nécessaire pour que le système subsiste de manière indépendante, écologique et à longue durée. De plus, l'isolement du système permettrait de l'adapter et le déployer sur différents sites. En se basant sur le projet Mycélium mené l'an dernier, il utilisera un plus grand nombre de capteurs ainsi qu'un dispositif comprenant une caméra. Ce système permettra ainsi de renforcer les connaissances tant biologiques qu'abiotiques des chercheurs, de manière automatisée, en capturant des événements extrêmes mais aussi quotidiens liés à l'évolution de la Croix Verte.

La suite de ce document présente, dans un premier temps, l'état de l'art lié au projet. Dans un second temps, il se penche sur la prise en main de la première version du projet Mycélium. Ensuite, les spécifications générales et fonctionnelles sont exposées. Enfin, l'architecture générale du projet est présentée.

2 État de l'art

Cette partie regroupe une étude de projets faisant usage de technologies similaires à celles que nous envisageons d'utiliser, notamment des capteurs connectés et communiquant via un réseau LoRaWAN. Ces projets traitent des sections particulières du problème qui nous est posé mais pas l'intégralité de celui-ci.

2.1 LivingFog Platform Valence

LivingFog Platform[3] est une initiative du projet européen FogGuru[4] lancé en 2017 dans le cadre du programme Horizon 2020. L'objectif du programme est de monter une formation doctorale appliquée à l'industrie, capable de doter l'Europe de la prochaine génération d'experts en fog computing. LivingFog Platform est dirigé par M. Guillaume PIERRE, enseignant à l'Université de Rennes 1 et chercheur à l'IRISA.

Le projet a été déployé dans La Marina de Valence en Espagne afin d'aider les autorités portuaires à améliorer le service qu'ils proposent. Les capteurs installés au sein du port (*figure 3*) permettent de récolter des informations utiles telles que la qualité de l'eau, la météo, la hauteur des vagues dans le port, ou encore le nombre de personnes et de voitures entrant et sortant de la zone. En particulier, le traitement de ces données par diverses applications permet de :

- Mesurer la qualité de vie des propriétaires de bateaux ;
- Les aider au travers de l'assistance à la navigation ;
- Améliorer l'expérience vécue par les visiteurs ;
- Aider La Marina à surveiller la zone pour l'aménager au mieux.



FIGURE 3 – Emplacement des capteurs et passerelles LoRa au port de Valence

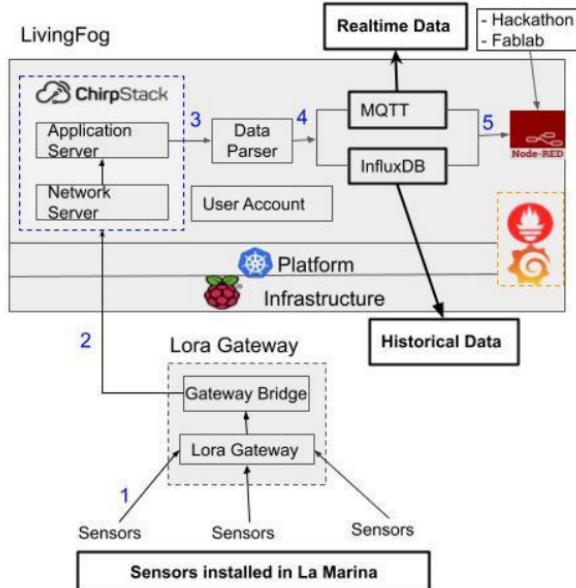


FIGURE 4 – Flux de collecte de données dans LivingFog

Comme détaillé dans la *figure 4* ci-dessus, ce projet utilise des capteurs utilisant les mêmes technologies que ceux que nous allons déployer.

2.2 Smart Street Lighting Montevideo

Le projet Smart Street Lightning[5] est un projet d'éclairage public intelligent. Porté par l'opérateur australien de la National Narrowband Network Company (NNNCo), il se base sur la plateforme IoT ThingPark Enterprise de l'entreprise européenne Actility. Il vise à fournir une couverture réseau à environ 70 000 lampadaires intelligents de la capitale uruguayenne, Montevideo. Ainsi, après remplacement des lampadaires actuels par des lampadaires intelligents à LED (*figure 5*), l'éclairage public de la ville pourra se faire à distance. Dans le contexte actuel de hausse des prix de l'énergie et de l'inflation, ce système permettra de réduire les émissions carbone liées à l'éclairage public de près de 80% et d'améliorer la sécurité publique et routière. Ce projet a aussi pour but d'incorporer d'autres applications comme la gestion des déchets et la conservation de l'eau.

Notre projet a lui pour but de déployer un réseau de capteurs intelligents et à basse énergie sur le site de Croix Verte.

Intelligent Street Light using LoRa

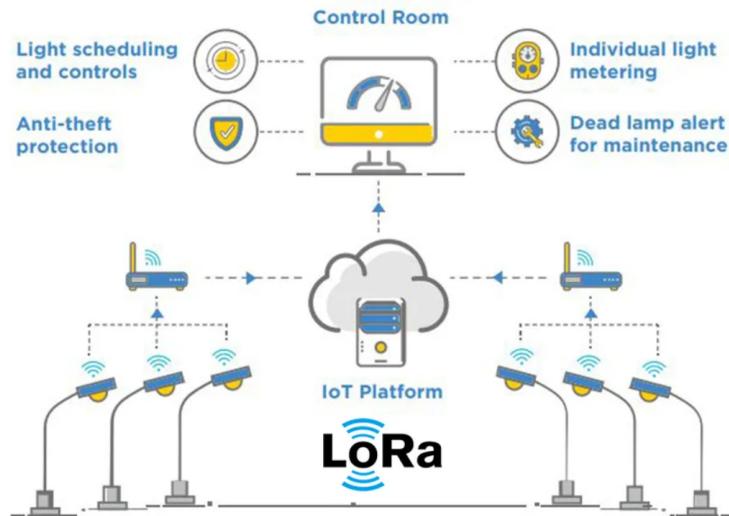


FIGURE 5 – Schéma de l’architecture de lampadaires intelligents utilisant le réseau LoRa[6]

2.3 ConnecSens-2

Le contexte scientifique du projet ConnecSenS-2[7] s’inscrit dans la continuité du projet ConnecSenS, présenté dans le rapport de pré-étude de Mycélium l’an dernier[8].

Il se positionne d’une part dans le domaine des révolutions des technologies numériques, comme les réseaux de capteurs sans fil, l’Internet des Objets, le web des données, les grilles de calcul, le nuage numérique, ou le grand volume de données.

D’autre part, il soutient la mobilisation de la communauté scientifique afin de mieux comprendre les dynamiques multi-échelles des écosystèmes et des territoires, mieux gérer les changements planétaires (climat, démographie, urbanisation, migrations) et concevoir de nouveaux modèles pour se tourner vers une démarche de développement durable.

Le projet ConnecSenS-2 vise à consolider l’instrumentation des sites de recherche avec des équipements complémentaires, mais également à ouvrir le champ scientifique d’intérêt pour ces réseaux de capteurs communicants déployés dans l’environnement (*figure 7*). De plus, après avoir centré l’intérêt scientifique sur la thématique de l’eau, ce nouveau projet s’intéresse également à celle du sol, par exemple pour modéliser le couplage sol-plante-atmosphère. Les différentes analyses sont effectuées sur quatre sites pilotes situés en Auvergne : le lac d’Aydat, le bras mort de la rivière Allier à Auzon, la zone atelier sur l’ancienne mine d’uranium à Roffin-Lachaux et le site de recherche et d’expérimentation en agrosystèmes de Montoldre. En complément de ces sites instrumentés du projet ConnecSenS, ConnecSenS-2 viendra alors équiper les prairies expérimentales de Laqueuille.

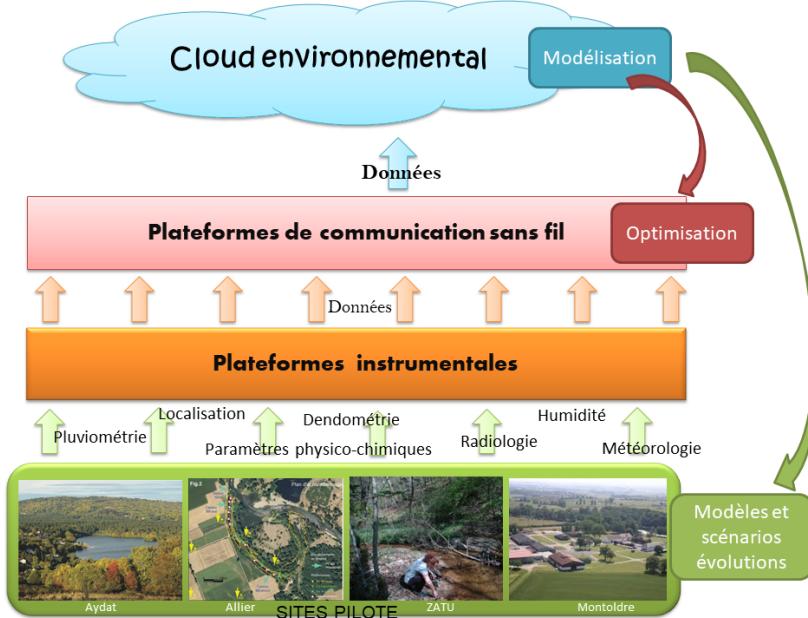


FIGURE 6 – Projets de monitoring environnemental via ConnecSenS-2

De multiples résultats sont attendus :

- Le déploiement d'une plateforme de collecte de données d'envergure servant de support à des recherches environnementales et s'inscrivant pleinement dans le cadre du CEBA (Cloud Environnemental pour le Besoin des Agriculteurs) ;
- Des avancées scientifiques sur les questions de protocole de communication bidirectionnel, un enjeu important pour l'Internet des Objets qui manque encore de données pour qualifier ces technologies ;
- La mise au point de méthodes de qualification de la qualité des données, qui est une question de recherche d'actualité importante ;
- L'investigation de propositions de requêtage de données au sein du Cloud mobilisant les technologies du web sémantique.

En ce qui concerne notre projet, nous allons déployer un système similaire à celui exploité par ConnecSenS et ConnecSenS-2. Ainsi, nous utiliserons les mêmes boîtiers de capteurs : les noeuds SoLo.

2.4 WildCount

Le projet Wildcount[9] est un projet porté par l'université Grenoble Alpes. Il s'inscrit comme notre projet dans celui de plus grande ampleur TERRA FORMA. Wildcount a pour objectif de développer un capteur d'edge computing peu onéreux afin de reconnaître et de compter de façon autonome les animaux dans les forêts ou zones protégées. L'edge computing est une méthode qui optimise le cloud computing afin de traiter les données au plus près de la source des données. Le comptage et la reconnaissance d'animaux sauvages sont basés sur un réseau de neurones entraînés

sur des bases de données de véhicules, d'humains et d'animaux. La grande partie du traitement des images se fait directement sur le capteur avant que les données transitent par LoRaWAN du capteur à InfluxDB, qui stocke les résultats afin qu'ils puissent être consultés par un humain sur Grafana ou que l'humain soit averti par des alertes (*figure 7*).



FIGURE 7 – Architecture du projet WildCount

Dans notre projet, un des objectifs est de pouvoir détecter et classer la faune environnante afin de répertorier la biodiversité, voire localiser un éventuel renard sur la Croix Verte.

3 Première version et transfert de compétences

Ce chapitre présente l'étude et la prise en main de la version 1.0 du projet Mycélium. Elle traite aussi du transfert de compétences entre l'équipe projet de l'année passée et l'équipe projet de cette année.

3.1 Technologies utilisées dans Mycélium 1.0

Comme dit dans l'introduction du rapport, un nœud de calcul et de traitement est chargé de récupérer les données et de prendre des décisions ainsi que de communiquer avec le Cloud. Ce nœud concentre plusieurs technologies qui vont être présentées dans cette partie[8].

3.1.1 Fog Computing

Le fog computing nécessite que les données soient traitées au plus proche de l'utilisateur final, afin de réduire la latence et les coûts et d'améliorer les performances. A ces fins, le nœud de calcul et de traitement est un **cluster de cinq Raspberry Pi 3** (*figure 8*). Il est utilisé pour déployer des conteneurs Docker qui contiennent les applications de Mycélium. Les Raspberry Pi sont des nano-ordinateurs qui présentent de bonnes performances et sont donc largement utilisés dans le domaine de l'informatique embarqué. Ce cluster permet de traiter les données au plus proche du noeud SoLo et ainsi d'économiser les coûts réseaux.



FIGURE 8 – Cluster de Raspberry Pi 3

Les différentes applications composant Mycélium sont donc déployées sur ce cluster. Chaque application est déployée dans un **conteneur Docker**, qui est un environnement d'exécution présentant l'avantage de nécessiter peu de ressources et de pouvoir communiquer avec d'autres conteneurs[10].

Cependant, les conteneurs seuls ne peuvent pas suffire à déployer Mycélium sur le cluster de Raspberry Pi de façon efficace et stable. Pour cela, **l'orchestrateur de conteneurs Kubernetes** est utilisé : il permet de gérer les pannes, d'équilibrer les charges et de distribuer les conteneurs entre les nœuds du cluster (*figure 9*).

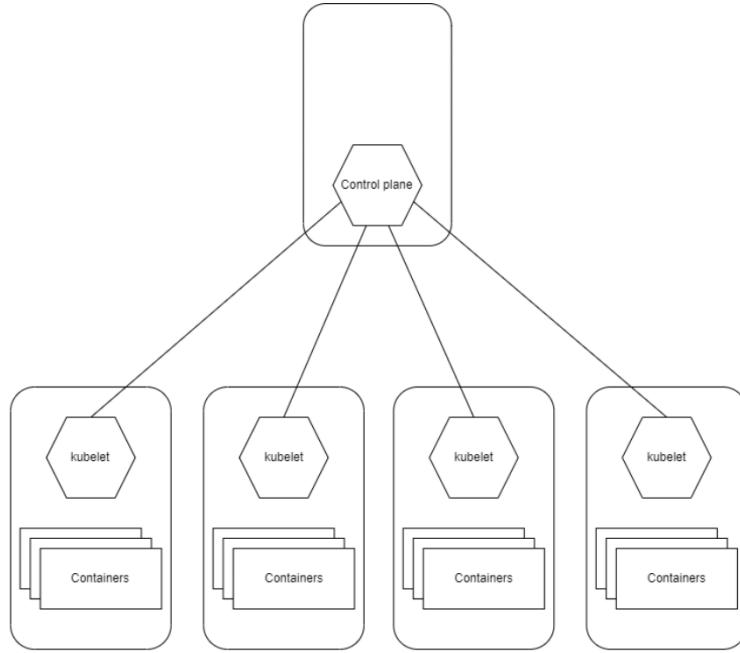


FIGURE 9 – Kubernetes dans le cluster

3.1.2 ChirpStack

ChirpStack est un ensemble de logiciels open source qui permet la gestion des appareils utilisant le réseau **LoRaWAN**. Une interface Web est mise à disposition pour gérer les appareils LoRaWAN et les APIs pour intégrer les logiciels qui traitent les données reçues (*figure 11*). L'appareil chargé de récupérer les données radios envoyées par les capteurs est une gateway. Celle que nous utilisons est la **gateway MultiTech**, elle est connectée au cluster par ethernet (*figure 10*).



FIGURE 10 – Gateway LoRaWAN MultiTech

Une fois la gateway connectée au cluster via ChirpStack, il est possible de visualiser les données qu'elle a reçues et de connaître le nombre de paquets en erreur.

FIGURE 11 – Configuration du réseau LoRaWAN

3.1.3 InfluxDB

Le système de base de données **InfluxDB** utilise des séries temporelles, c'est-à-dire que les données sont datées. Cela est très pratique dans notre cas car les données environnementales mesurées par le nœud SoLo varient en fonction du temps. Comme toute base de données, il est possible d'interagir avec cette dernière par requêtes, le langage de requêtes compatible avec les bases de données InfluxDB étant le **langage Flux**. Enfin InfluxDB présente également une interface Web pour visualiser les données (*figure 12*).

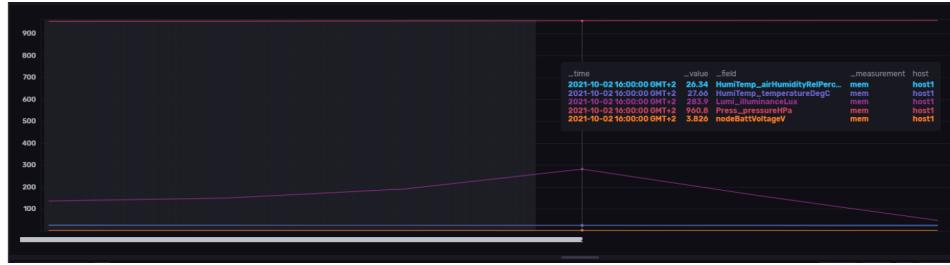


FIGURE 12 – Affichage InfluxDB

Cependant, l'affichage sur l'interface Web de InfluxDB reste simple et limité, c'est pourquoi le choix d'utiliser **Grafana**, un outil de visualisation de données très riche, a été fait. Cet outil permettra une meilleure observation des données et par conséquent une meilleure compréhension de la Croix Verte.

3.1.4 OpenFaaS

Une fois les données récupérées grâce à ChirpStack, il faut les décoder et les traiter pour prendre des décisions. Afin de faciliter le développement, l'équipe précédente a fait le choix de découper ces tâches en microservices. En effet, cela permet d'éviter un unique service trop complexe, qui nécessiterait à la fois un redémarrage à chaque changement et que les développeurs le maîtrisent en profondeur.

La méthode choisie pour implémenter ces microservices est le **serverless**. Chaque tâche sera codée par une **fonction**, la communication et l'exécution des fonctions est abstraite au développeur. Il existe actuellement plusieurs fonctions : pour décoder les données, pour les enregistrer dans InfluxDB, ou encore pour gérer les scénarios. Un système de subscriber/publisher permet au ChirpStack de publier les données et aux fonctions de s'abonner à ces publications. Le fonctionnement de ce système publisher/subscriber sera détaillé en partie 5.3. Ainsi, les fonctions sont exécutées lorsque le ChirpStack reçoit de nouvelles données et l'exécution de ces fonctions peut entraîner l'exécution d'autres fonctions. La gestion de ces fonctions est gérée par le logiciel **OpenFaaS**. OpenFaaS est un framework open-source permettant de déployer facilement des fonctions lancées par des événements, sur Kubernetes[11]

3.2 Fonctionnalités existantes de Mycélium 1.0

3.2.1 Scénarios

Un des objectifs du projet Mycélium est de développer un système capable de réagir à des situations inhabituelles, appelées scénarios. Ceux déjà créés par le groupe de l'année dernière sont [2] :

- **Scénario de luminosité**, utilisant les capteurs de luminosité et de température, créé uniquement pour tester les fonctions développées sur ces capteurs ;

- **Scénario neige**, lancé lorsque la température est inférieure à 0°C et que l'humidité dépasse un seuil. Il conduit à une augmentation de la fréquence de mesure des capteurs de température, humidité et pluviométrie ;
- **Scénario température anormale**, lancé lorsque la température dépasse un seuil défini ;
- **Scénario vol**, déclenché lors d'une brusque variation des mesures de l'accéléromètre ;
- **Scénario grand changement consécutif de données**, indiquant un changement soudain et inattendu dans les mesures.

Les deux premiers sont traités dans le **nœud SoLo** (voir sous-partie Nœud SoLo) et conduisent à une augmentation de la fréquence des mesures et de l'envoi de données des capteurs correspondants. Les trois derniers sont quant à eux traités dans le **cluster**, dont le fonctionnement sera explicité par la suite, et conduisent à l'envoi d'une notification liée au scénario sur le serveur discord.

Le système existant permet aussi de comparer les données fournies par les capteurs avec celles d'une API-météo grâce à la fonction OpenFaaS compare-general-api. Cela pourrait également devenir un scénario si une notification discord était envoyée lorsque les deux mesures à une date donnée diffèrent considérablement.

3.2.2 Nœud SoLo

Aperçu La détection de ces scénarios est faite grâce aux mesures du **nœud SoLo** [8] (*figure 13*). Il s'agit d'un boîtier de capteurs permettant d'effectuer des mesures de l'environnement de manière intelligente et de transmettre les données mesurées. Ce boîtier est à basse consommation (son autonomie est estimée à plusieurs mois), et sa couverture réseau est grande (10km dans le meilleur des cas) ; ce qui le rend très adapté à ce projet. Toutefois, un élément contraignant est qu'il ne peut qu'envoyer des données et non en recevoir.



FIGURE 13 – Nœud SoLo

Fonctionnement Le nœud SoLo contient des capteurs internes déjà présents initialement. Il est ainsi équipé d'un accéléromètre, d'un capteur de température et d'humidité, d'un capteur de luminosité et d'un capteur de pression. Mais il est également possible d'y ajouter des capteurs externes. Le groupe de l'année dernière l'a ainsi connecté à un pluviomètre (*figure 14*), également mis à disposition pour le projet Mycélium. Ces capteurs vont prendre des mesures périodiquement, les enregistrer et les envoyer par paquets à une fréquence définie.



FIGURE 14 – Pluviomètre connecté au nœud SoLo

Le nœud SoLo a donc besoin de stocker localement les données mesurées. Cela se fait au moyen d'une carte microSD de 8 Go, qui contient non seulement ces dernières au format CSV, mais aussi un fichier de logs et un fichier de configuration au format JSON. Dans ce dernier, il est possible de modifier simplement la fréquence de prise de mesure pour chaque capteur, mais aussi de créer un **mode alarme** permettant d'augmenter cette fréquence lorsqu'un événement spécifique à chaque capteur est déclenché, par exemple lorsque la quantité de pluie dépasse un seuil pour le pluviomètre.

Ces données sont envoyées au moyen d'un émetteur-récepteur utilisant la technologie LoRa et le protocole de communication **LoRaWAN**. Comme vu précédemment, ce réseau a une basse consommation et une longue portée. C'est lui qui confère ces bonnes propriétés au nœud SoLo, qui peut alors envoyer sur de longues distances des données de petite taille.

Le nœud SoLo est aussi équipé d'un GPS et d'une horloge Real-time clock (RTC). Le GPS sert non seulement à la localisation, mais aussi à la synchronisation de l'horloge.

Tous ces éléments s'articulent autour d'un micro-contrôleur que contient également le nœud, constitué des éléments essentiels d'un système embarqué : processeur, mémoires et périphériques d'entrée-sortie.

Nouvelles fonctionnalités du noeud SoLo Le groupe de l'année dernière a ajouté deux nouvelles fonctionnalités dans le noeud SoLo pour améliorer le suivi environnemental de la Croix Verte [2].

La première consiste, à chaque démarrage du noeud, à ce que tous les capteurs envoient une mesure, pour vérifier non seulement leur bon fonctionnement mais aussi que le noeud ne redémarre pas anormalement sans arrêt.

La deuxième partie du constat que les alarmes mentionnées précédemment sont insuffisantes pour détecter un scénario. En effet, ce dernier repose le plus souvent sur le dépassement d'un seuil pour plusieurs capteurs (événements multiples), alors que les alarmes concernent un événement unique. Pour pallier ce problème, **deux fonctions embarquées** ont donc été développées pour les deux scénarios de neige et de luminosité implémentés dans le noeud SoLo. Pour le scénario neige, la fonction associée permet d'augmenter la fréquence de prise de mesure du capteur de température et d'humidité et la fréquence d'envoi de mesures lorsque la température passe en-dessous de 0°C et l'humidité passe au-dessus de 88%. Pour le scénario de luminosité, la fonction associée permet aussi d'augmenter ces fréquences pour le capteur de température, qui mesure aussi l'humidité, et le capteur de luminosité lorsque la température passe au-dessus de 20°C et la luminosité passe au-dessus de 700 lux.

3.3 Prise en main de l'existant

Code existant Tout d'abord, il a fallu tenter de prendre en main le code déjà existant ainsi que les nouvelles technologies utilisées. Le code étant hébergé sur le GitLab de l'INSA Rennes, l'accès était relativement simple. Lors de la prise en main du code, nous avons été confrontés à un écueil. En effet malgré une documentation relativement conséquente du code de la part de l'équipe de l'année précédente, nous n'avons qu'effleuré la surface du projet de l'an dernier étant donné que la majorité des outils utilisés était inconnue de la plupart des membres du groupe. De plus, la complexité du code écrit par l'équipe de l'an dernier a, de manière prévisible, participé à l'échec de l'appropriation du code. En outre, Garden, plateforme permettant l'automatisation de développement logiciel et l'administration d'infrastructures informatiques pour Kubernetes, étant instable et gourmand en mémoire nécessaire à son utilisation du fait qu'il s'appuie sur Kubernetes, a été compliqué à gérer. Afin de remédier à cela, nous avons pris contact avec l'équipe précédente afin d'avoir plus d'explications relatives au code et à l'architecture générale du projet. De plus, lors de cette rencontre, ils nous ont recommandé de ne pas utiliser Garden pour poursuivre le projet cette année, pour la raison évoquée précédemment.

Gateway Après avoir essayé de reprendre le code par nous-même, nous avons tenté de connecter la gateway LoRaWAN avec un de nos ordinateurs personnels en utilisant la documentation fournie avec celle-ci. Cela ne fut pas concluant et un objectif plus abordable mais plus conséquent a été établi : connecter la gateway LoRaWAN au cluster de Raspberry Pi. Pour cela, il a fallu s'appuyer sur le manuel d'utilisation fourni, la documentation en ligne, ainsi que l'interface de la gateway. Cependant, les deux premiers ne nous paraissaient pas clairs quant à la manière de

procéder, et l'identifiant de celle-ci, très important dans la connexion entre celle-ci et le cluster, est très difficile à obtenir. Suite à cela, nous avons demandé conseil aux anciens responsables du projet à propos de la manière de procéder, et ceux-ci nous ont indiqué où se trouvaient les informations sur le git.

Connexion au cluster Enfin, nous nous sommes connectés au cluster de Raspberry Pi via le Wi-Fi IOTINFO afin de savoir quels logiciels fonctionnaient dessus. Nous avons pu remarquer que la capacité de stockage du cluster était déjà utilisée de moitié. Parmi la liste de logiciels installés dessus, nous avons trouvé ceux nécessaires au bon déroulement de notre projet : Docker, ChirpStack, Grafana et InfluxDB. Une seconde réunion avec un membre de l'équipe de Mycélium 1.0 nous a permis de découvrir en détail ce qui avait été réalisé l'an dernier. Suite à cette rencontre, nous avons pu lancer OpenFaaS depuis le cluster, nous amenant à lancer Grafana depuis ce dernier et à enregistrer des données dans InfluxDB pour les visualiser sur Grafana.

4 Spécifications générales et fonctionnelles

Aux fonctionnalités déjà développées dans l'existant du projet, de nouvelles vont devoir être développées : ajout d'un nœud sur le Cloud, d'un système de comptage de la faune avec une caméra et de nouveaux capteurs communiquant par GSM. La figure 15 résume les composantes à développer et leur placement dans l'architecture existante.

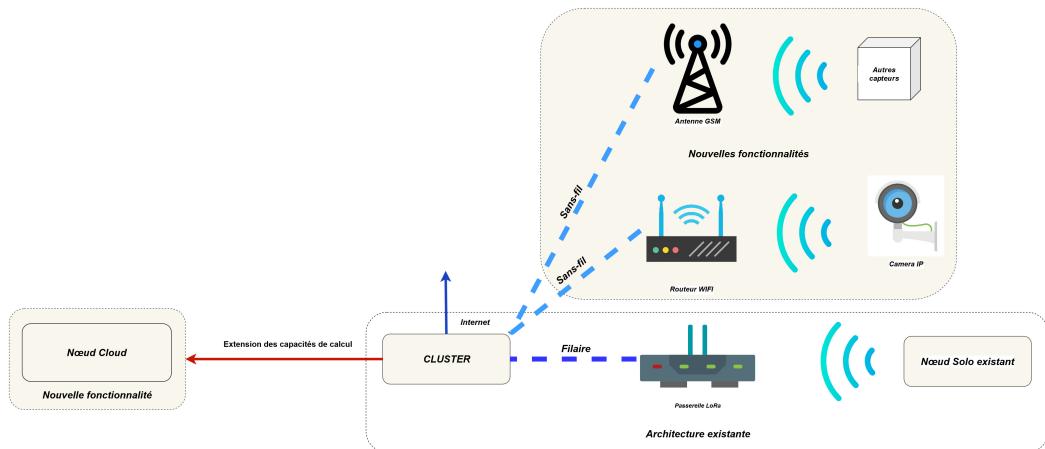


FIGURE 15 – Schéma simplifié des fonctionnalités à ajouter sur l'existant

4.1 Nouveaux scénarios

4.1.1 Normales saisonnières

Contexte Certains capteurs mesurent des grandeurs dont il existe une normale saisonnière. Le but est de pouvoir comparer chaque grandeur à sa normale saisonnière.

Réaction Lorsqu'une mesure, dont une norme de saison existe, est traitée par le nœud, une comparaison est effectuée avec la norme de saison correspondante grâce à une base de données fournie par un organisme météorologique. L'utilisateur final doit pouvoir facilement mesurer l'écart entre la mesure et la normale saisonnière correspondante. Les mesures seront effectuées avec plusieurs nœuds SoLo afin de pouvoir observer d'éventuelles variations localisées.

4.1.2 Intempéries

Contexte Il arrive que des épisodes orageux frappent la région bretonne et d'importantes quantités de pluie s'abattent sur le sol. Le but est de détecter ces épisodes.

Réaction Lorsque le nœud détecte que le pluviomètre enregistre un épisode de pluie important, une notification adéquate doit être envoyée aux utilisateurs de la

solution et des données pluviométriques, de température et de luminosité doivent être enregistrées.

4.2 Nouveaux types de capteurs

4.2.1 Recherche de faune sauvage : les flux vidéo

Contexte Dans un contexte de tentative de détection d'animaux et d'un éventuel renard, dans le secteur de la Croix Verte, une caméra va être installée et des images contenant des individus en mouvement seront envoyées au noeud de traitement.

Réaction Une classification des individus doit être effectuée et un rapport de ces décisions doit être mis à la disposition des utilisateurs, le tout à l'aide d'un compteur et d'un moyen d'accéder aux images détectées et classifiées. La détection d'un individu en mouvement par la caméra et la prise de vue doivent déclencher l'envoi et la réception par le noeud de traitement qui déclenchera la fonctionnalité de classification et de dénombrement.

4.2.2 Capteur de mesure de l'eau avec connectivité GSM

Contexte Le capteur mesurant les différentes propriétés de l'eau possède une connectivité GSM (Global System for Mobile communications). Il faut donc interconnecter ce dispositif avec notre infrastructure grâce à une interface de manière à pouvoir récupérer les données produites, les traiter et les afficher dans notre dispositif de monitoring.

Réaction Toute donnée produite par le noeud de capteurs doit arriver jusqu'au noeud de traitement des données. Il faut que l'arrivée d'une donnée déclenche le traitement de celle-ci et l'affichage et/ou la réaction adéquate, comme par exemple l'envoi d'une notification.

4.2.3 Développement de la scalabilité sur le Cloud

Contexte Dans un contexte d'accroissement du nombre de données et de type de traitement à effectuer sur les données, un accroissement de la charge sur le noeud de traitement est attendu.

Réaction Une répartition de la charge entre les composants du noeud de traitement doit permettre de subvenir aux nouveaux besoins de traitement. Lorsque le cluster détecte, par introspection, que la charge globale qu'il supporte est trop importante, un deuxième noeud doit pouvoir prendre le relai pour effectuer des tâches et ainsi maintenir une qualité de service adéquate et un temps de traitement relativement court. Ainsi, un scénario pourra être traité de manière transparente, sur un composant quelconque du noeud de traitement, sans que l'utilisateur final n'en soit informé.

4.2.4 Maintenabilité

Contexte Parce que le projet, dans sa forme la plus large, a pour objectif le déploiement de nœuds de capteurs et de traitement dans des environnements éloignés, la maintenance doit être effectuée avec une fréquence la plus basse possible. C'est pourquoi la stabilité doit être constamment améliorée.

Mise en œuvre Le nœud de traitement doit disposer de technologies et de fonctionnalités testées, éprouvées et approuvées afin que la stabilité soit assurée.

5 Architecture globale

Cette partie se porte sur l'architecture du projet aussi bien sur le plan matériel que sur le plan logiciel. Mycélium 2.0 reprenant le projet de 4INFO Mycélium en le déployant à plus grande échelle, l'architecture mise en place se basera sur celle du projet existant.

5.1 Architecture matérielle

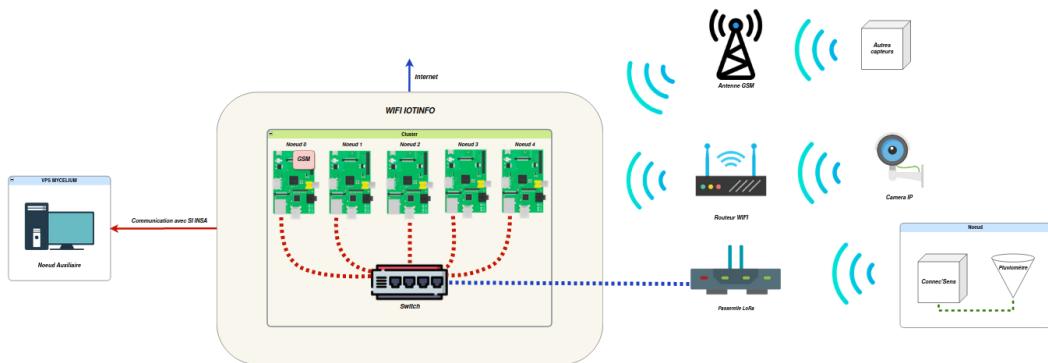


FIGURE 16 – Schéma de l'architecture matérielle

Cœur matériel Le cœur matériel du projet est le cluster de Raspberry Pi (*figure 16*), l'unité de traitement. Comme présenté précédemment, le cluster est composé de cinq noeuds. Ce choix matériel est motivé par la possibilité de distribuer la charge de travail et le fait d'avoir une consommation d'énergie faible par rapport à des ordinateurs classiques. Ces noeuds sont reliés entre eux avec un commutateur réseau permettant la communication. Le cluster est connecté au réseau spécifique au département informatique, IOTINFO, nous donnant accès à Internet. L'architecture logicielle implique une consommation très importante des ressources des noeuds, ce qui sera explicité par la suite. En prenant en compte ce que nous souhaitons développer, des limites vont rapidement être atteintes. C'est pourquoi, afin d'augmenter les capacités de notre système, nous avons souhaité avoir à disposition un noeud auxiliaire. Pour répondre à ce besoin, nous avons obtenu un VPS (Virtual Private Server) hébergé dans les serveurs de la DSI de l'INSA Rennes.

Le second composant matériel est le réseau de capteurs disposé sur la Croix Verte. Nous allons développer trois moyens de communiquer des données au cluster en lien avec trois ressources matérielles différentes.

- Le plus important est le **réseau LoRaWAN**. C'est grâce à celui-ci que les données d'humidité, de température, de pression ou encore de précipitation vont être transmises. Seront donc disposés sur le terrain des noeuds de capteurs et des boîtiers ConnecSenS, reliés à un pluviomètre externe, qui émettront des données à destination d'une gateway LoRa installée au département informatique. Elle servira de relais et retransmettra donc les données vers le

cluster, qui traitera ensuite ces données, via Ethernet. Ce système a déjà été mis en place l'année dernière sur un unique noeud.

- Le deuxième moyen de communiquer est le **Wi-Fi** : un routeur sera placé entre le cluster et le système de surveillance. Ce dernier consiste en une ou plusieurs caméra IP filmant la faune locale dans le but de la recenser.
- Enfin, le dernier mode de communication employé passera par **GSM**, le réseau mobile. Il sera utilisé par des capteurs ne pouvant communiquer par les moyens énoncés précédemment, tel que le capteur qui remontera diverses métriques liées au cours d'eau du site comme son débit.

5.2 Architecture du cluster

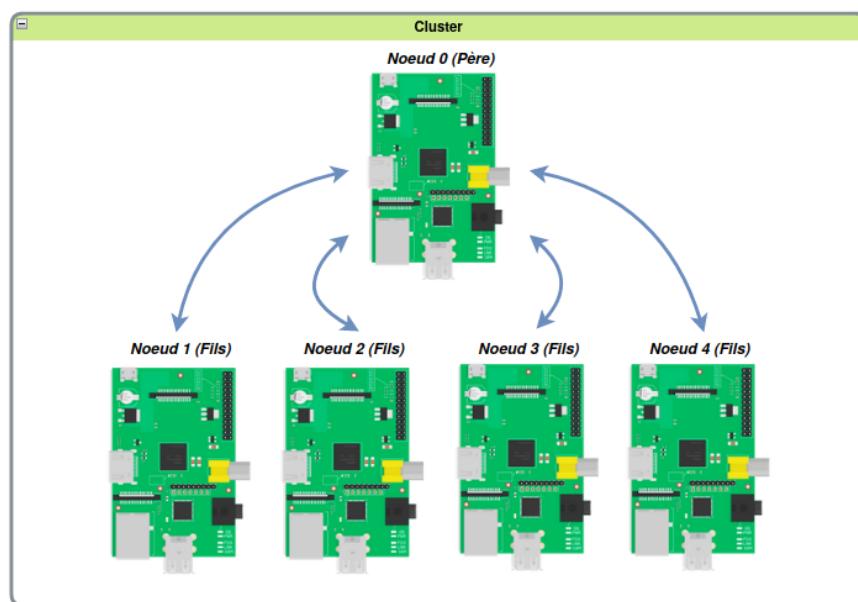


FIGURE 17 – Architecture générale du cluster et rôle des composants

Les noeuds du cluster n'ont pas tous le même rôle. Parmi les cinq noeuds, l'un d'eux est le noeud père, communiquant et contrôlant les quatre autres noeuds, désignés noeuds fils (*figure 17*). Les échanges de données au sein du cluster passent par ce noeud père. Ces deux types de noeuds ne contiennent pas les mêmes logiciels.

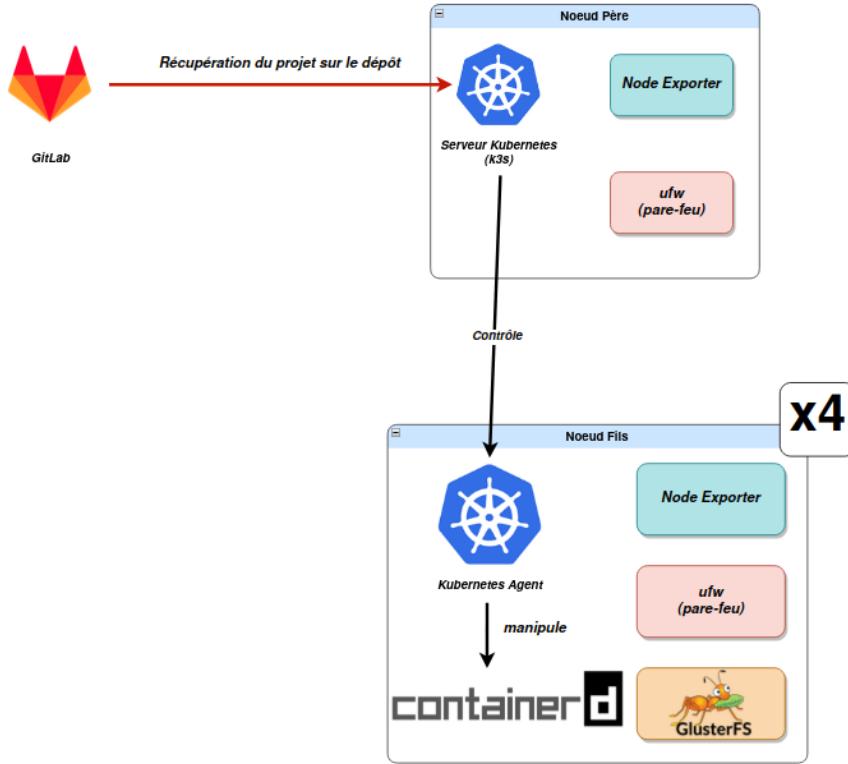


FIGURE 18 – Configuration logicielle des noeuds

En ce qui concerne le **noeud père**, il contient trois composantes (*figure 18*) :

- *Node Exporter*, qui expose les métriques du noeud ;
- *Ufw*, qui permet de configurer le pare-feu du noeud ;
- Le *serveur Kubernetes* déployé, le différenciant des noeuds fils[11]. Il s’agit du cerveau du cluster qui va permettre de manipuler les noeuds fils. Le serveur va récupérer le projet sur le dépôt GitLab.

Un **noeud fils** contient quant à lui un *agent Kubernetes* échangeant avec le serveur et contrôlant lui-même le fonctionnement de containers. Ce dernier permet de gérer les conteneurs déployés sur le noeud. Les conteneurs contiennent les différents composants de l’application du projet : par exemple, chaque fonction OpenFaaS[12] est dans un conteneur distinct. Un autre composant spécifique des noeuds fils est *GlusterFS* qui permet d’avoir un système de fichiers distribué sur le cluster. Cela permet d’obtenir des performances satisfaisantes même lors de l’exécution de tâches gourmandes en ressources sur plusieurs machines.

5.3 Architecture logicielle

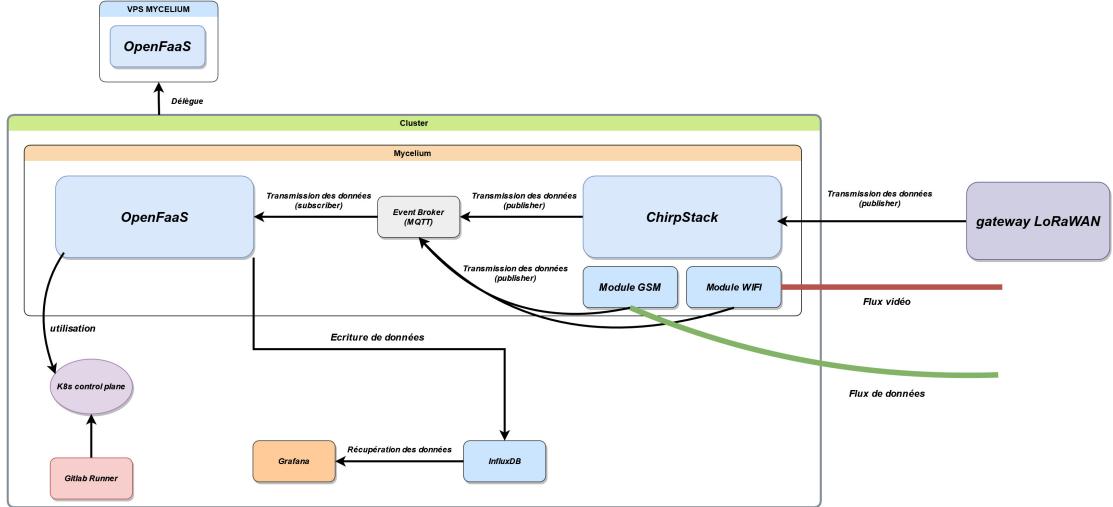


FIGURE 19 – Interfaçage entre le cluster et la passerelle LoRa

Cette section se concentre sur la partie logicielle à l'échelle du cluster. La **gateway LoRa** constitue un principal point d'entrée vers le cluster (*figure 19*). Le ChirpStack gateway bridge contenu dans celle-ci va envoyer les données provenant des capteurs à destination du Network Server du ChirpStack[13]. Les messages sont envoyés en suivant un protocole standard en IoT (Internet of Things) : MQTT (Message Queueing Telemetry Transport). Ce dernier fonctionne sur un principe de publisher/subscriber : entre les parties prenantes de la communication, un broker effectue un relais, ici il fait partie intégrante du ChirpStack. Plus précisément, le broker va stocker les données dans des topics, c'est-à-dire un espace mémoire unique à chaque flux de données. Un dispositif va publier des données sur ce broker et un dispositif va souscrire à ce broker pour récupérer les informations. Ici cela correspond respectivement à la gateway LoRa et à un composant interne du ChirpStack adressé par la suite, le Network Server. Toujours en utilisant le même protocole, cette fois le ChirpStack va publier les données dans le broker externe, Event Broker. Cela fait le pont avec OpenFaaS qui va y souscrire pour traiter les données. C'est l'arrivée de nouvelles données dans un topic qui va généralement lancer un appel d'une fonction. Un exemple de fonction est celle qui permet de stocker les données traitées dans la base de données InfluxDB. En ce qui concerne le traitement des données, on l'a dit précédemment nous avons besoin d'un noeud dans le Cloud et plus précisément sur un VPS INSA. C'est pour cela que l'on a une instance de OpenFaaS dessus qui exécute des fonctions qui ne peuvent l'être sur le cluster.

Afin d'avoir une interface visuelle sur les données remontées par les capteurs, **Grafana** est déployé sur le cluster et va récupérer les données écrites dans la base de données InfluxDB pour les afficher sous la forme de graphiques. Pour ce qui est de l'exécution d'application et notamment des fonctions OpenFaaS, on passe par le

K3s control plane appelé par OpenFaaS pour prendre les grandes décisions au niveau du cluster en termes de planification et pour répondre aux événements du cluster. C'est le GitLab Runner qui va exécuter les briques applicatives du projet contenues dans le dépôt GitLab et qui va utiliser les pipelines d'intégration continue.

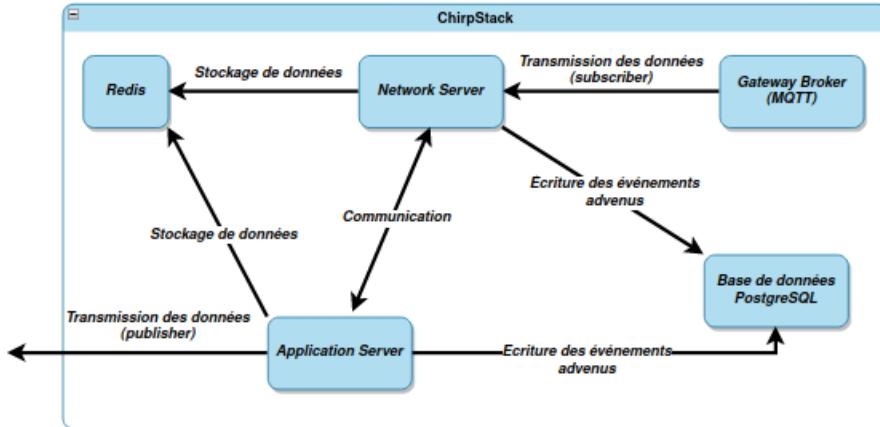


FIGURE 20 – Architecture du serveur ChirpStack

On va à présent se concentrer sur ce qui se passe au niveau du **ChirpStack** (*figure 20*). Ce dernier va comme expliqué précédemment comporter un broker, le Gateway Broker, qui va recevoir les données de la gateway LoRaWAN. C'est le Network Server qui va souscrire au broker pour récupérer les données des gateways. Ce dernier a pour but de rassembler les messages reçus des différentes sources, d'authentifier les sources puis de communiquer avec l'Application Server. Ce composant va pouvoir exposer les données reçues de différentes manières, mais deux d'entre elles sont particulièrement intéressantes dans le cadre du projet.

- La première est l'utilisation du **protocole MQTT** avec un broker, ici Event Broker, dans lequel l'Application Server publie les données liées aux événements survenus sur le terrain, comme le passage d'un renard devant la caméra.
- La seconde est l'utilisation du **protocole HTTP** via une API REST qui était envisageable, mais étant donné que l'interlocuteur sera une instance d'OpenFaaS et qu'un composant existe déjà pour gérer les entrées en MQTT dedans, il était préférable de ne pas mettre en place cela en plus du reste.

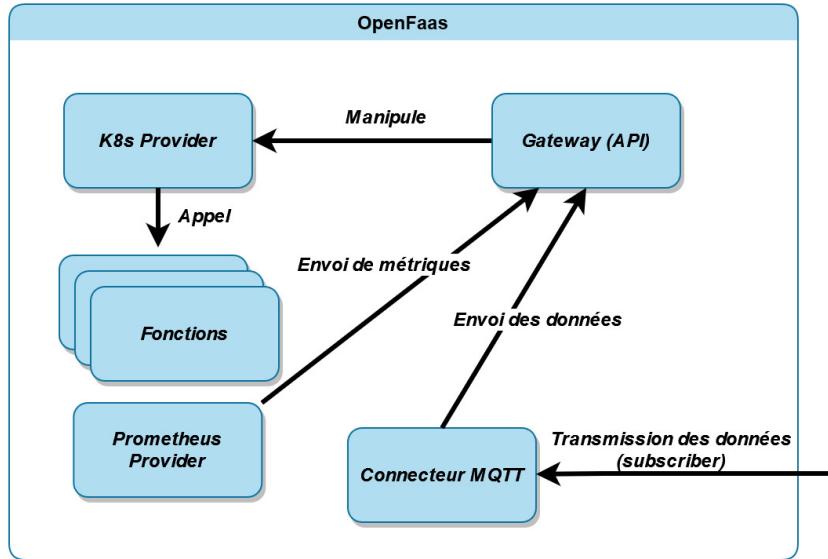


FIGURE 21 – Architecture OpenFaaS

Dans OpenFaaS, un connecteur MQTT souscrit donc au broker pour avoir ces données, mais l'outil **Prometheus** récupère aussi des métriques de deux sources différentes à savoir les nœuds du cluster et OpenFaaS (*figure 21*). Prometheus est un outil permettant de surveiller, suivre des événements et de gérer des alertes en s'appuyant sur une base de données en séries temporelles. On pourra donc récupérer l'occupation des ressources des nœuds comme la mémoire. Dans le cas d'OpenFaaS, on a les ressources demandées par chaque fonction en cours d'exécution. Cela permet de répartir la charge de travail sur l'ensemble du système. Ces deux composants vont envoyer les données vers l'API gateway permettant de manipuler le Kubernetes Provider qui gère principalement l'appel des fonctions OpenFaaS comme les fonctions liées aux différents scénarios (intempéries, vérification des normales saisonnières, ...).

Comme vu précédemment, d'autres points d'entrée sur le nœud de traitement seront installés. Le premier est une entrée pour des flux vidéo en provenance de caméras déployées sur la Croix Verte. Le transfert de données ne passera pas par le ChirpStack mais directement par l'Event Broker grâce à la création d'un topic spécifique. Toutefois, un module de pré-traitement serait tout de même nécessaire afin d'exposer à OpenFaaS les données au format et au contenu adéquats. La communication entre le cluster et la caméra se feront sur un réseau Wi-Fi dans un souci de simplicité. En ce qui concerne le second point d'entrée, il s'agit d'un flux de données GSM qui fonctionnera de la même manière, sauf lors de la récupération des données sur le cluster, nécessitant un module hardware spécifique.

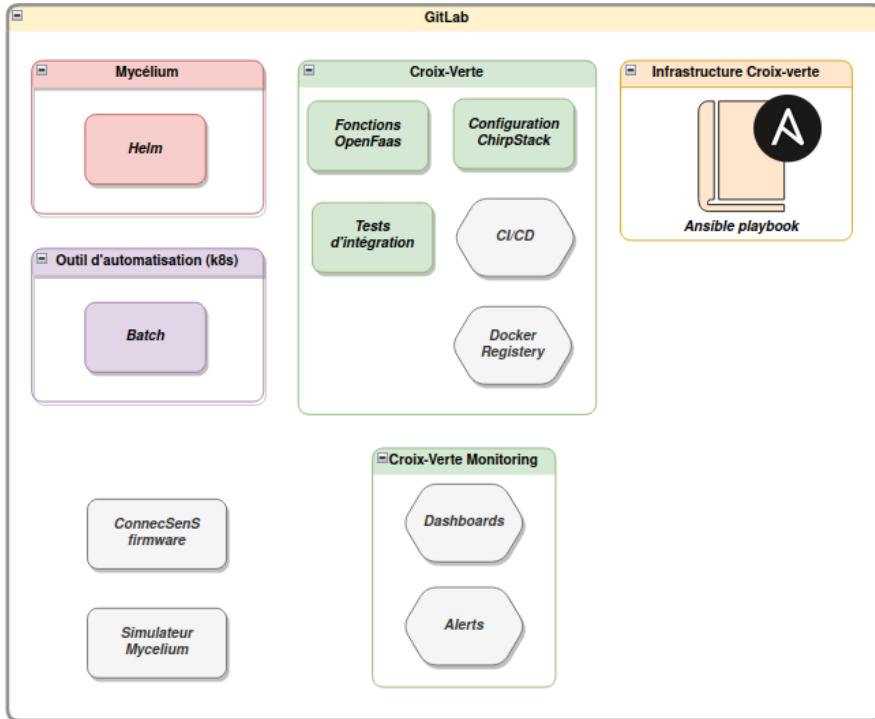


FIGURE 22 – Architecture du GitLab du projet

Le dépôt **GitLab** du projet est séparé en plusieurs sous-projets dont le principal est Croix Verte, qui contient les fonctions OpenFaaS qui vont se traduire dans le Docker Registry par une fonction associée à une image (*figure 22*). Ce dernier a donc la main sur les différentes images Docker du projet. Ce sont des modèles servant au déploiement de conteneurs, c'est-à-dire de l'environnement d'une application sur le cluster. Les fichiers de tests d'intégration et de la configuration du ChirpStack y sont aussi répertoriés. La mise en place de pipelines, à savoir des étapes d'exécution et de jobs, sont faites par le composant CI/CD (intégration continue/livraison continue).

Un autre sous-projet moins conséquent est le projet Infrastructure Croix Verte, qui sert à la maintenance de l'infrastructure présentée précédemment. Cela passe notamment par l'outil Ansible et plus précisément le playbook qui contient un ensemble de tâches de configuration et de déploiement d'application sur le système. Ensuite, le projet Croix Verte Monitoring correspond à la configuration de Grafana. Le projet Mycélium contient essentiellement ce que l'on appelle des charts Helm.

Helm est un outil de configuration de l'application Kubernetes, et les charts sont des fichiers décrivant les ressources Kubernetes en établissant leur environnement d'utilisation. Enfin un ultime projet répond aux besoins d'automatisation de tâches de déploiement ou de tests sur les différents sous-projets du projet.

Le projet de l'année précédente utilisait l'outil Garden, mais nous avons établi qu'il ne correspondait pas à nos besoins, c'est pourquoi il sera remplacé par un ou plusieurs batchs, autrement dit une série d'instructions à exécuter en boucle.

6 Conclusion

Ce document a donc introduit le contexte dans lequel le projet Mycélium 2.0 s'inscrit. Il présente l'état de l'art des technologies utilisées, la première version du projet Mycélium ainsi que ses limites. Cela permet alors de projeter des améliorations qui sont décrites dans les spécifications générales et fonctionnelles. Enfin, une architecture globale du projet est exposée, permettant de comprendre le système pour agir et mettre en œuvre les améliorations prévues.

L'objectif global du projet est ainsi de déployer un réseau de capteurs, pour réaliser un suivi automatique de l'environnement de la Croix Verte, s'inscrivant dans un projet de suivi environnemental participatif national. Techniquement, des alertes correspondant à des scénarios anormaux seront lancées suite à une détection par plusieurs nœuds SoLo, contre un seul l'an dernier, et une caméra. Ces données seront envoyées grâce au réseau LoRaWAN à un cluster de Raspberry Pi, en passant par une gateway connectée à ChirpStack. La gestion de ces données sera ensuite réalisée par des applications sur ce cluster, dans des conteneurs Docker, gérés par Kubernetes, et avec des fonctions FaaS.

Passer outre Garden et utiliser plus de capteurs divers, dont une caméra, ajoute de nouveaux enjeux, et un véritable incrément par rapport au travail réalisé l'an dernier. Bien que nous ayons fait face à de nombreux obstacles lors de la reprise du projet, nous avons fixé des objectifs clairs pour la suite de notre travail, qui s'inscrit dans une démarche environnementale importante et valorisante.

Table des figures

1	Représentation cartographique de la Croix Verte	2
2	Architecture globale de Mycélium 1.0	3
3	Emplacement des capteurs et passerelles LoRa au port de Valence	5
4	Flux de collecte de données dans LivingFog	6
5	Schéma de l'architecture de lampadaires intelligents utilisant le réseau LoRa[6]	7
6	Projets de monitoring environnemental via ConnecSenS-2	8
7	Architecture du projet WildCount	9
8	Cluster de Raspberry Pi 3	10
9	Kubernetes dans le cluster	11
10	Gateway LoRaWAN MultiTech	12
11	Configuration du réseau LoRaWAN	12
12	Affichage InfluxDB	13
13	Noeud SoLo	14
14	Pluviomètre connecté au noeud SoLo	15
15	Schéma simplifié des fonctionnalités à ajouter sur l'existant	18
16	Schéma de l'architecture matérielle	21
17	Architecture générale du cluster et rôle des composants	22
18	Configuration logicielle des noeuds	23
19	Interfaçage entre le cluster et la passerelle LoRa	24
20	Architecture du serveur ChirpStack	25
21	Architecture OpenFaaS	26
22	Architecture du GitLab du projet	27

7 Bibliographie

Références

- [1] L. LAHMAR, “TERRA FORMA : un nouveau paradigme pour l’observation des territoires.” www.insu.cnrs.fr.
- [2] G. CHAUVEAU, P. IACONE, F. DABAT, E. JUVÉ, Y. TIAN, i. M. ELOSO RODRIGUES, and H. ZHANG, “RAPPORT FINAL MYCELIUM.”
- [3] “The LivingFog platform – FogGuru project.” www.fogguru.eu.
- [4] “FogGuru project – training the next generation of european fog computing experts.” www.fogguru.eu.
- [5] “Montevideo, one of the most ambitious smart street lighting projects in the world.” wellnesstg.com.
- [6] raghulraj, “Intelligent street light using LoRa.” www.instructables.com.
- [7] L. de Physique de Clermont, “ConnecSenS-2.” clrwww.in2p3.fr.
- [8] G. CHAUVEAU, F. DABAT, P. IACONE, JUVÉ, Y. TIAN, V. M. VELOSO RODRIGUES, and H. ZHANG, “PRÉ-ÉTUDE ASSOCIÉE À LA PHASE d’ANALYSE.”
- [9] “WildCount · GitLab.” gitlab.com.
- [10] “Docker : qu’est-ce que c’est et comment l’utiliser ? le guide complet.” datascientest.com.
- [11] “Kubernetes documentation.” kubernetes.io.
- [12] “Openfaas documentation.”
- [13] “Chirpstack documentation.” www.chirpstack.io.
- [14] L. ROYER, “Réseau LoRaWAN sur l’ETNA.” journees-lpwan-2021.limos.fr.
- [15] T. Gurus, “The LivingFog platform.” www.fogguru.eu.
- [16] G. CHAUVEAU, F. DABAT, P. IACONE, E. JUVÉ, Y. TIAN, V. M. VELOSO RODRIGUES, and H. ZHANG, “Mycelium.” projets-info.insa-rennes.fr.
- [17] “Optimiser la gestion de l’eau par le fog computing (informatique géodistribuée) | www.univ-rennes1.fr.” www.univ-rennes1.fr.
- [18] “Qu’est-ce que l’edge computing ?” www.redhat.com.
- [19] “Laboratoire de physique de clermont - projet ConnecSenS.” www.lpc-clermont.in2p3.fr.
- [20] G. CHAUVEAU, F. DABAT, P. IACONE, JUVÉ, Y. TIAN, V. M. VELOSO RODRIGUES, and H. ZHANG, “RAPPORT DE SPÉCIFICATION FONCTIONNELLE.”