OBSERVABILITY

Présenté par l'équipe
Dragon Eye

Ornel Zinsou-Ply : CSS, Chef de projet
Elvire Tossou Boco : CSS
Bassit Raïm : LD
Akim Traore: CSS
Aylone Gouthon: CSS

Encadrant

Amal Tiab

Client

Mr CHAVIN

Sommaire

Introduction

En tant que responsables de l'infrastructure pédagogique de l'ESEO,

Nous voulons disposer d'une plateforme unifiée capable d'observer en profondeur l'ensemble des machines physiques, virtuelles et services utilisés par les étudiants,

Afin d'améliorer la visibilité globale, accélérer le diagnostic des incidents et renforcer la qualité des enseignements en laboratoire.

Contexte du PFE

Infrastructure pédagogique

- Machines physiques + virtuelles
- Travaux : Cloud / Réseau / Cybersécurité

Problème

- ❌ Pas de vue centralisée
- ❌ Incidents difficiles à détecter
- ❌ Aucune anticipation des anomalies

Conséquences

- ⏱️ Diagnostics lents
- 🔥 Consommation énergétique non maîtrisée

Problematique du PFE

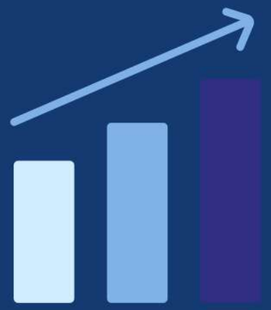
- Comment concevoir et déployer une solution d'observabilité centralisée pour surveiller l'ensemble de l'infrastructure de l'ESEO — machines physiques, virtuelles, services et applicatifs ?

- Comment y intégrer des capacités prédictives, des visualisations adaptées aux utilisateurs et une démarche écoresponsable ?

Objectif du PFE

Fournir une visibilité complète et en temps réel sur la santé du SI pour détecter et anticiper les défaillances.

- Centraliser les metriques et logs de l'ensemble des serveurs et services ✓
- Corréler et exploiter ces données pour détecter, analyser et anticiper les dysfonctionnements
- Fournir des indicateurs techniques et métiers via des Dashboard dynamique ✓
- Introduire une dimension écoresponsabilité en optimisant la consommation des ressources du SI ✓



Objectifs SMART

Objectifs	Spécifiques	Mesurables	Atteignables	Réalistes	Temporels
Centralisation	Déployer Prometheus, Grafana, Loki sur le centre d'observabilite	Tous les logs/métriques visibles depuis Grafana	Utilisation Docker Compose	Ressources disponibles (PC/serveurs test)	Avant fin du 2e mois
Dashboards	1 technique (CPU/RAM/logs) + 1 métier (SLA, dispo)	Données en <10s de délai	Grafana (dashboards JSON)	Indicateurs concrets	Mi-parcours
Détection & alertes	Configurer alertes (CPU, disque, erreurs)	Alerte reçue en <30 sec	PromQL + Alertmanager	Ciblé sur 3 cas simples	Mi-parcours
Prédictiion & écoresponsabilité	Prototype prédictif + mesure énergie	1 modèle ML + Watts/serveur affichés	Python (LD) + Scaphandre	Analyse simple/trend	Soutenance finale
Déploiement & documentation	Documentation + playbook Ansible	Déploiement <10 min	Ansible + Docker maîtrisés	Limité à 2–3 serveurs test	Rapport + soutenance

Analyse de l'existant

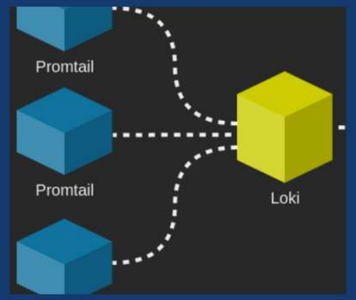
ZABBIX

Kibana

splunk>

Jenkins

Grafana Prometheus

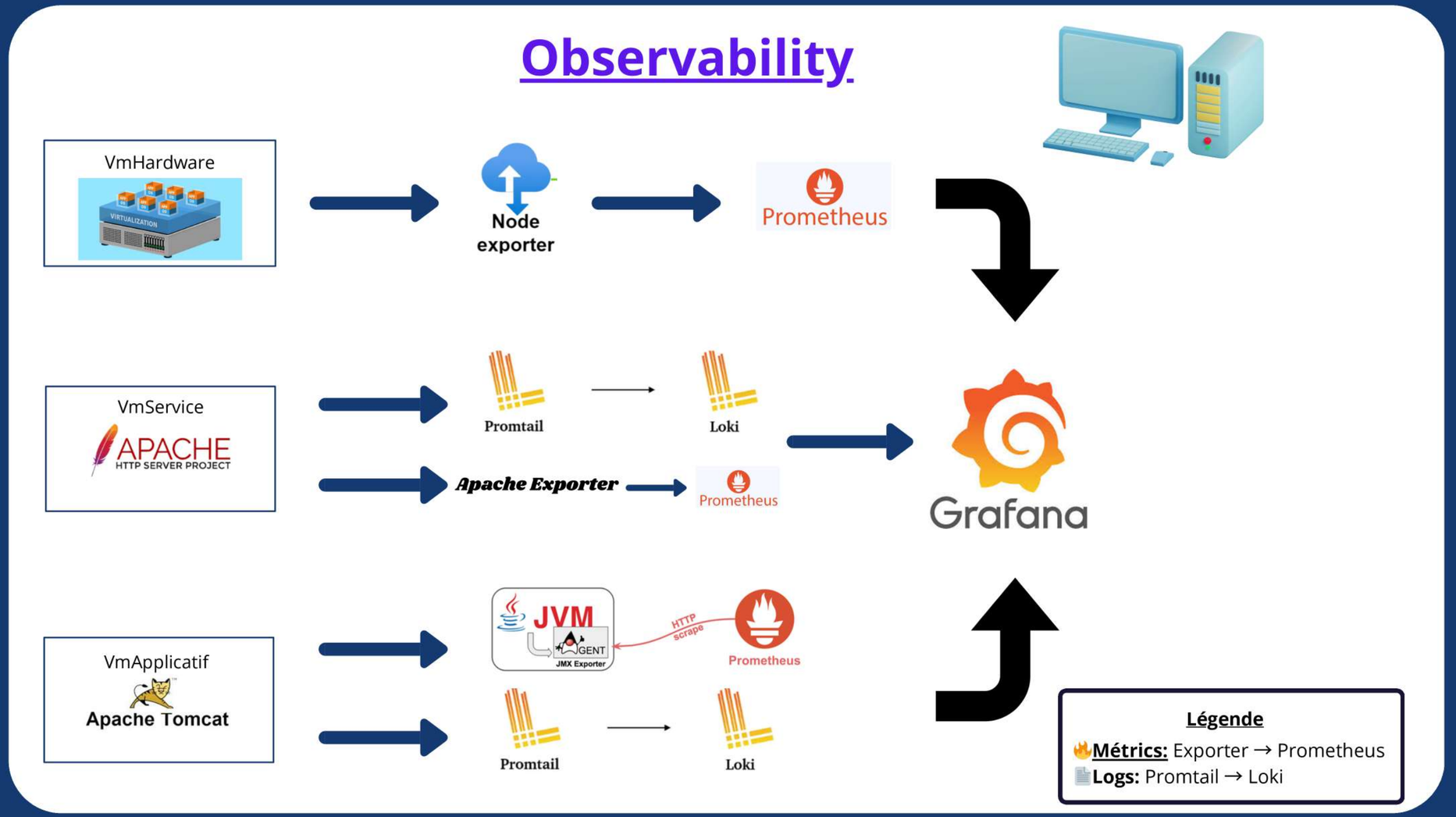


ANSIBLE

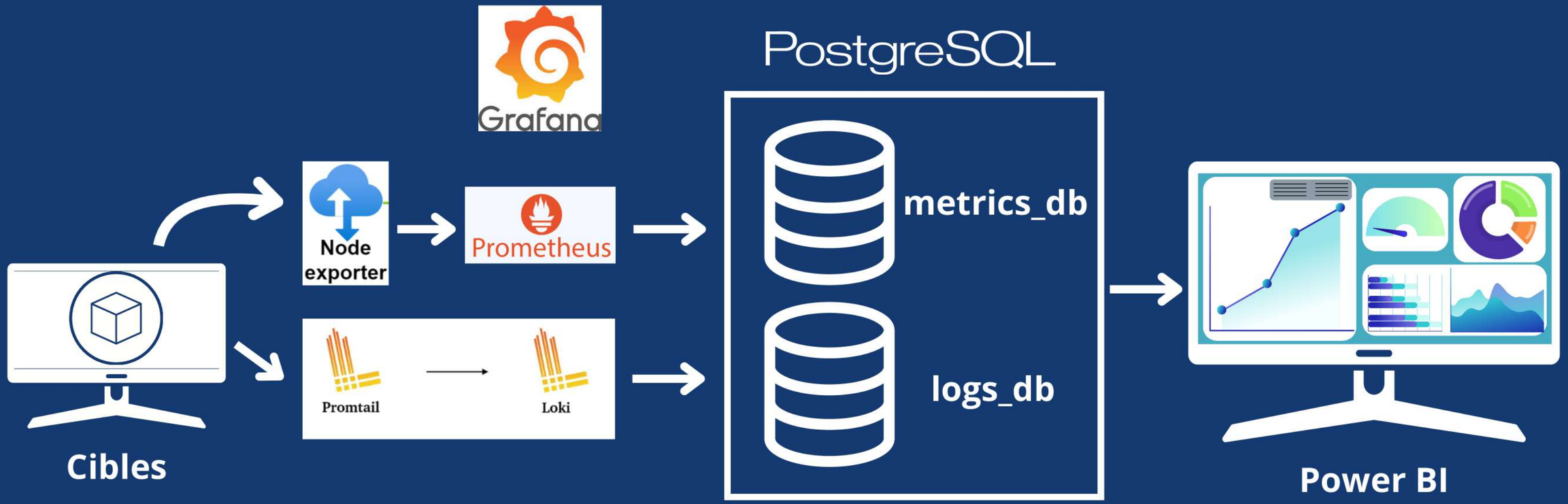
Labtainers

Power BI

Architecture du centre d'observabilité



Power Business Intelligence



Résultats obtenus lors du POC

Résultats techniques

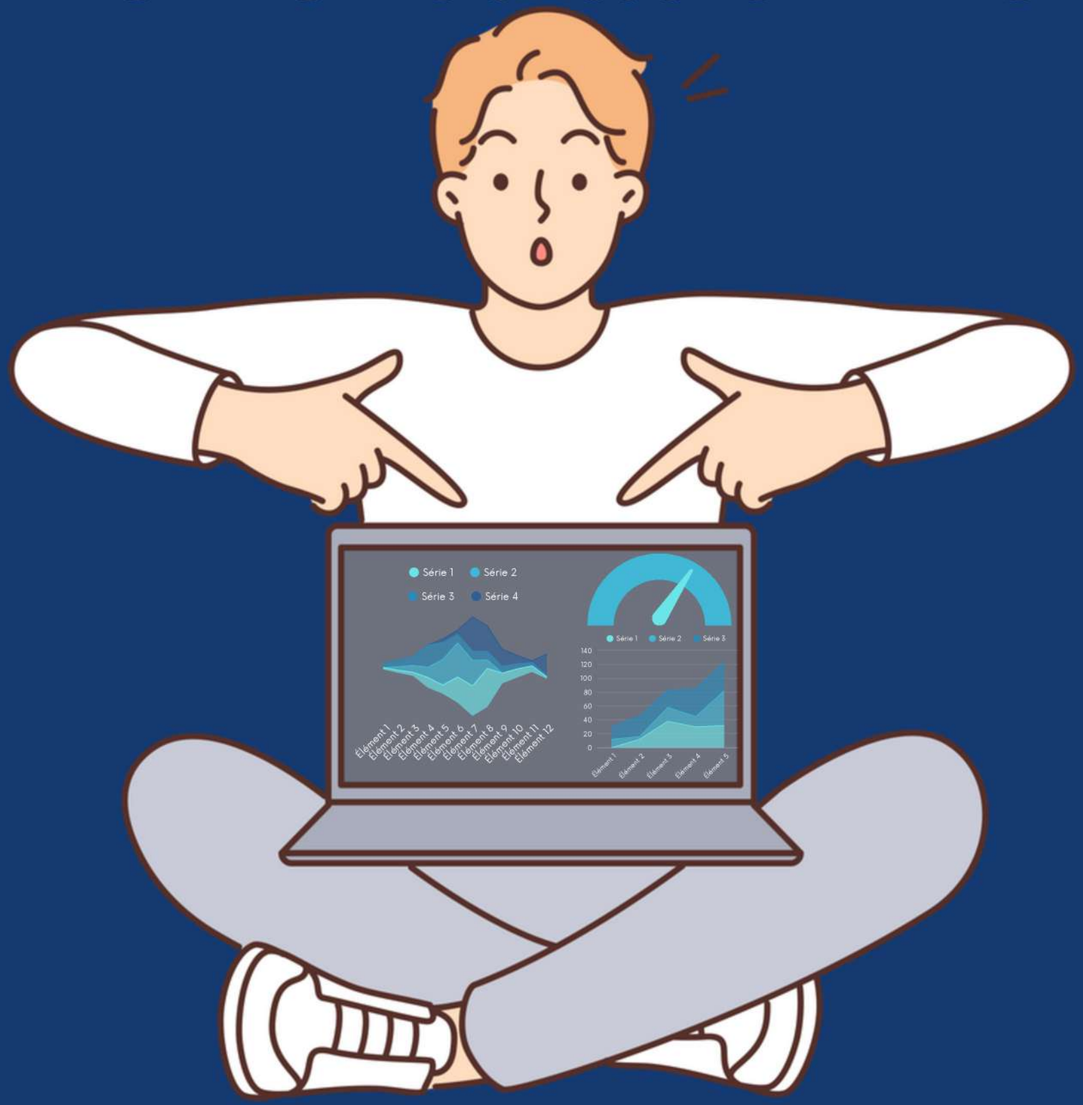
- Centre d'observabilité fonctionnel
- Collecte correcte des métriques système et applicatives
- Centralisation des logs en temps réel
- Dashboards Grafana opérationnels
- Corrélation métriques / logs possible
- Power BI

Résultats projet

- Validation du choix des outils
- Validation de l'architecture proposée
- Identification des limites techniques
- Réduction des risques avant migration
- Feu vert pour la migration vers un serveur réel

Démonstration POC

Grafana POC



Migration de la solution vers un serveur réel

Objectifs de la migration

- Déployer le centre d'observabilité sur un serveur réel
- Passer d'un environnement de test (POC) à un environnement stable
- Garantir une solution exploitable par le client

Motivations de la migration

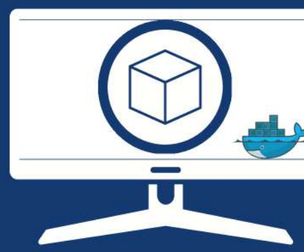
- Validation réussie du POC
- Besoin de supervision en conditions réelles
- Amélioration de la performance et de la disponibilité
- Centralisation des données sur une infrastructure dédiée

Étapes de la migration du centre d'observabilité

Gitea

branch test

branch main



Dockerfile

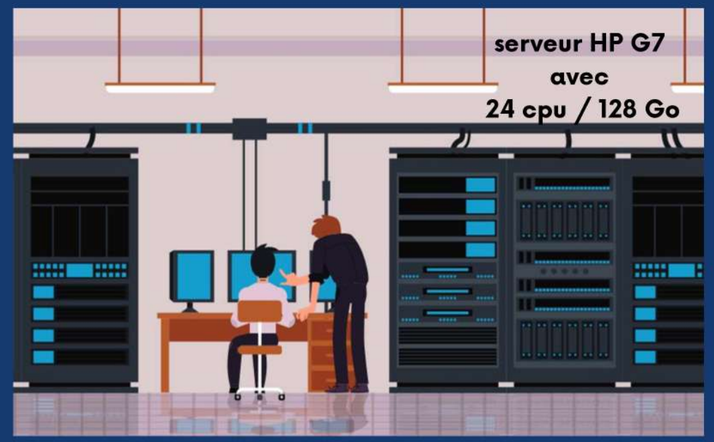
Configs file

centre d'observabilité
applicatif & service

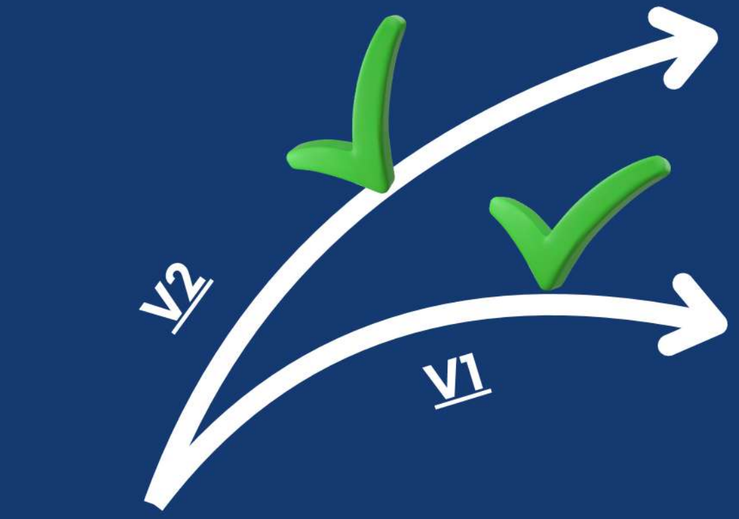
merge

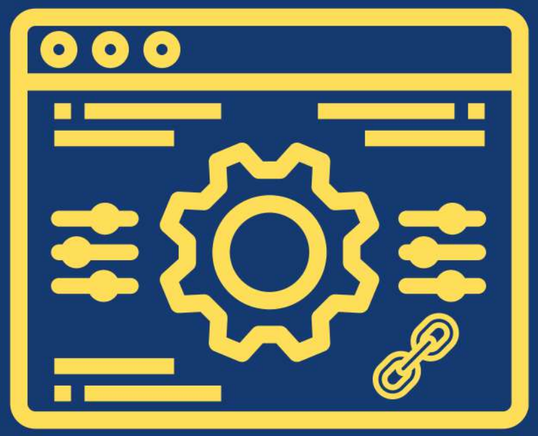
Phase 1: serveur de test

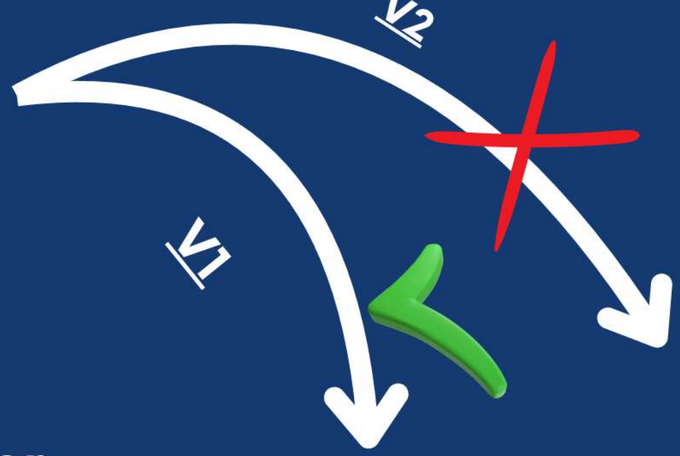
Phase 2: serveur de pre-production / production



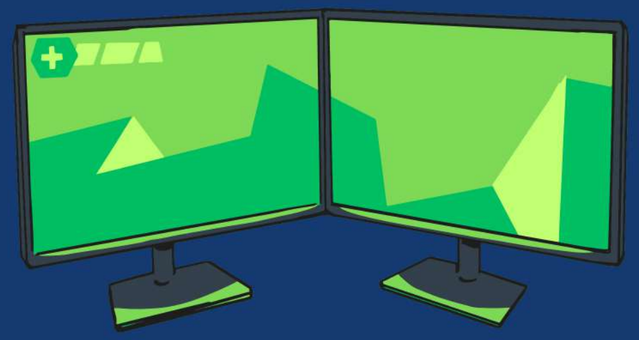
- Clone la branche main
- Redeploie conteneur:
1. centre observabilité
2. applicatif & service







Serveur de pré-production



Proof of Concept

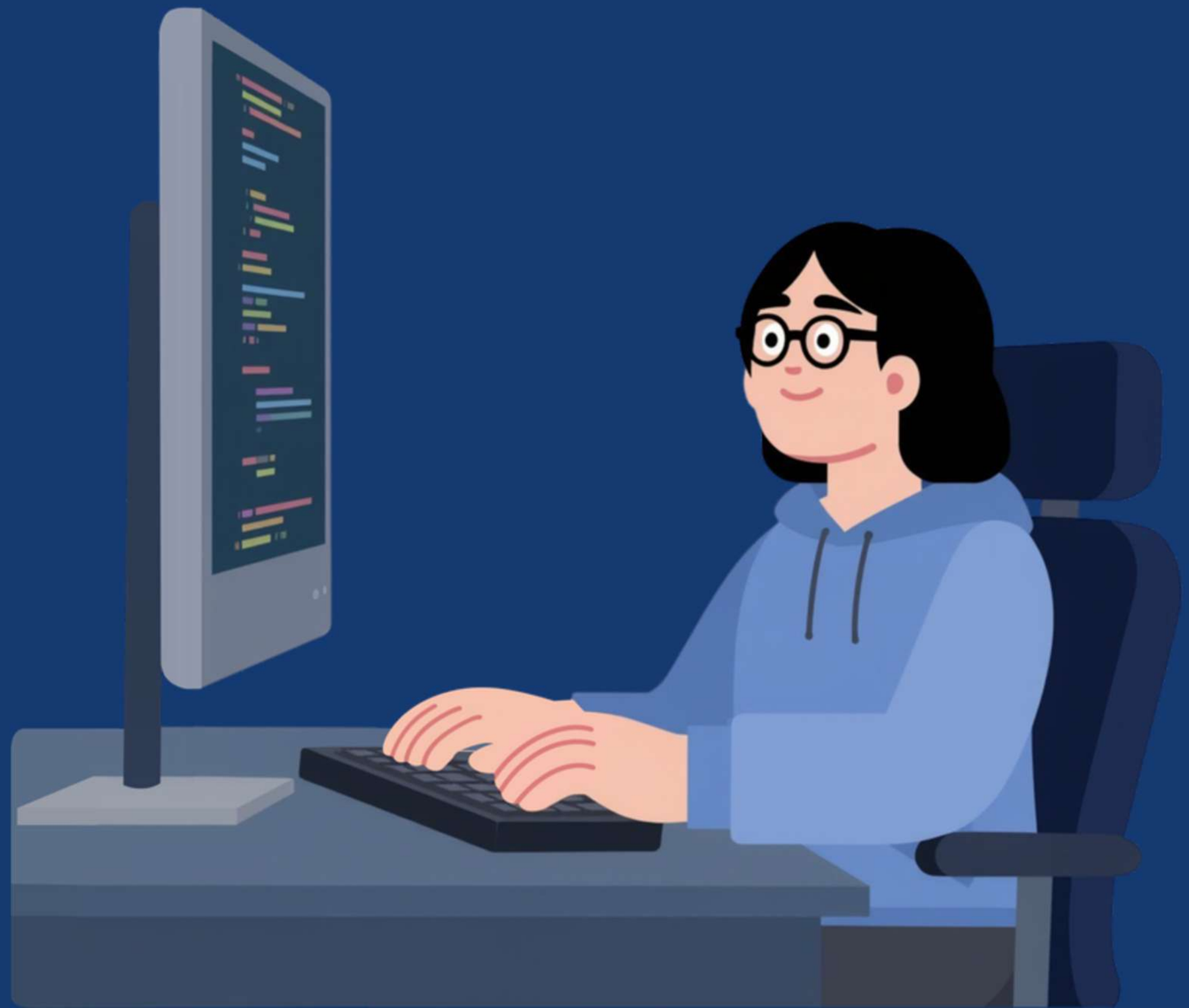
Serveur de production

Cycle de deploiement

Démonstration de pré-prod et de prod

Serveur de Pré-prod

Grafana



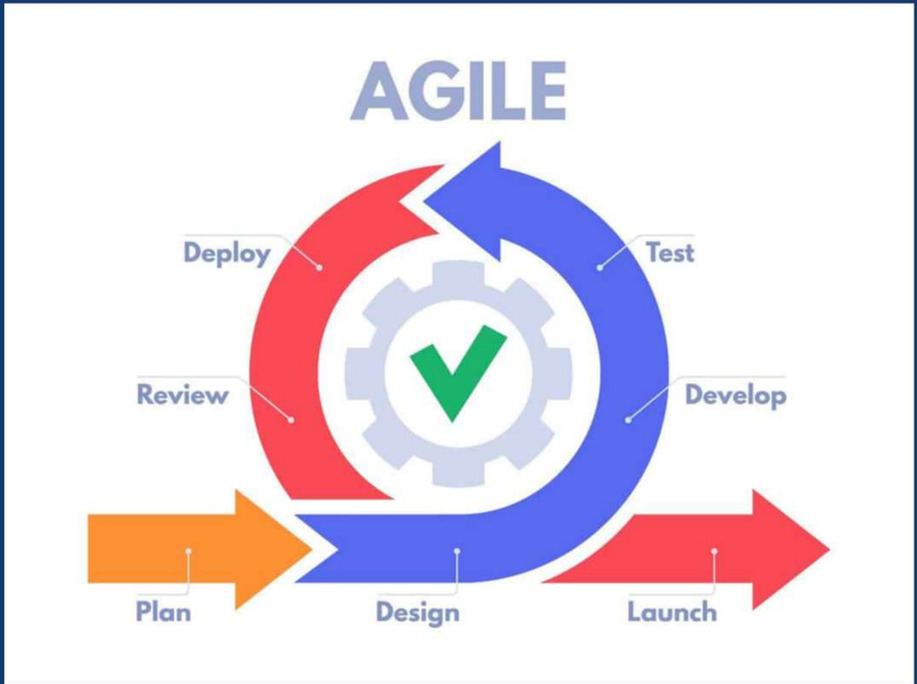
Serveur de Prod

Grafana

Prometheus

Gestion de projet

Méthode de travail

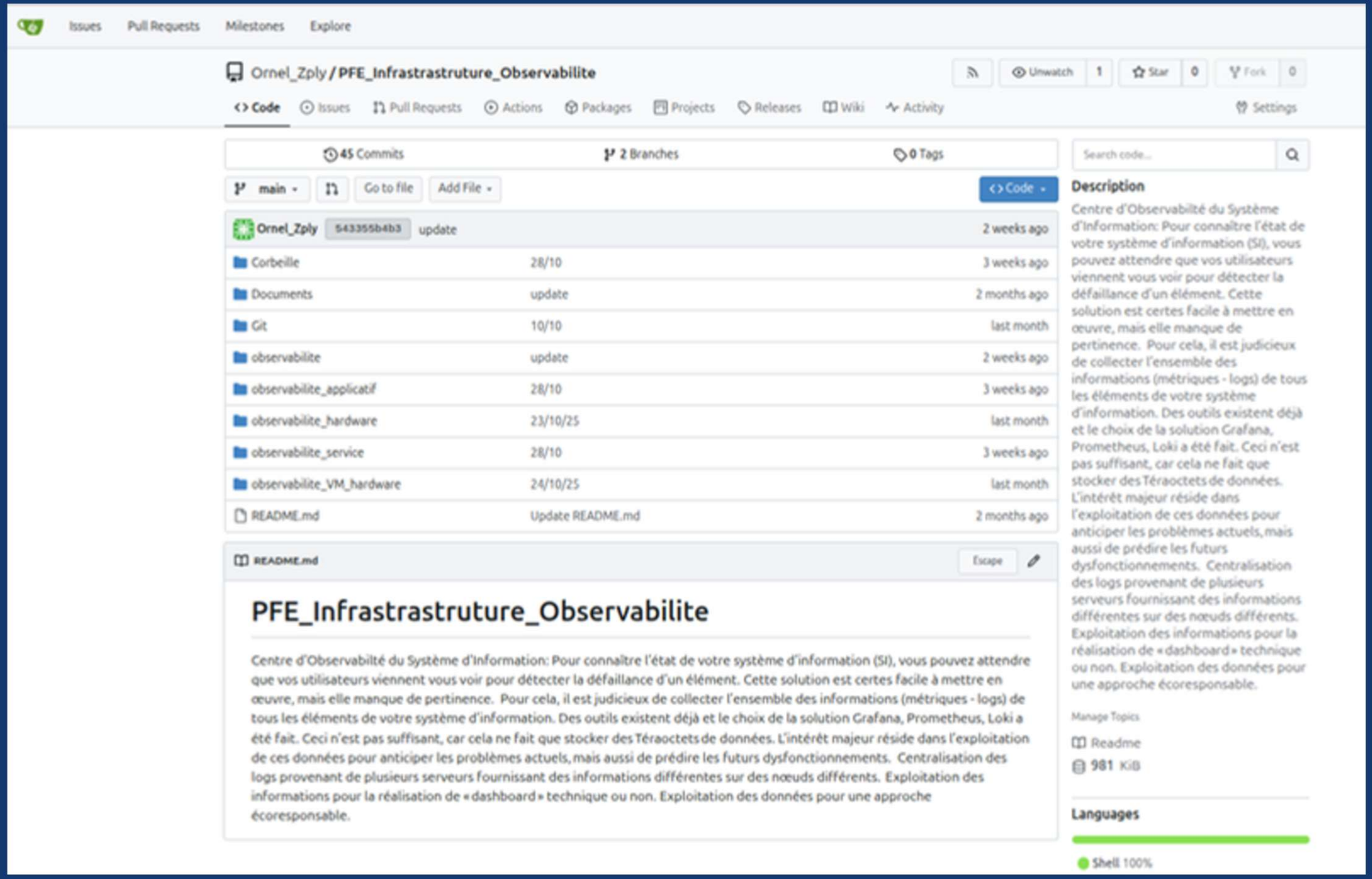


Organisation des réunions de suivi		
Type de réunion	**Fréquences**	**Objectifs principal**
Interne à l'équipe	Hebdomadaire	Suivi technique et répartition des tâches
Avec l'encadrant	En fonction des besoins	Validation de l'avancement global
Avec le client	En fonction des besoins	Orientation stratégique et technique

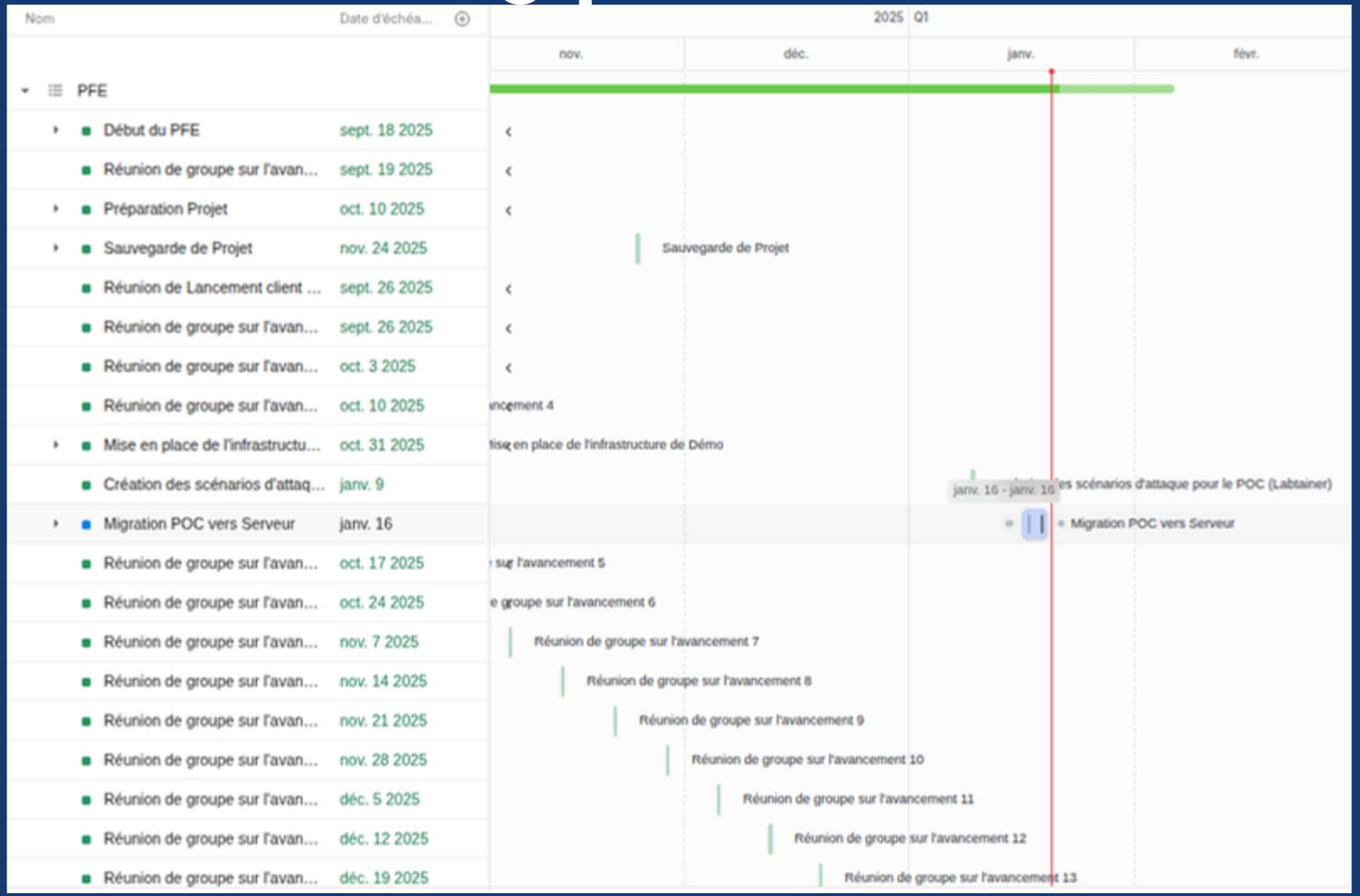
ClickUp login

Git login

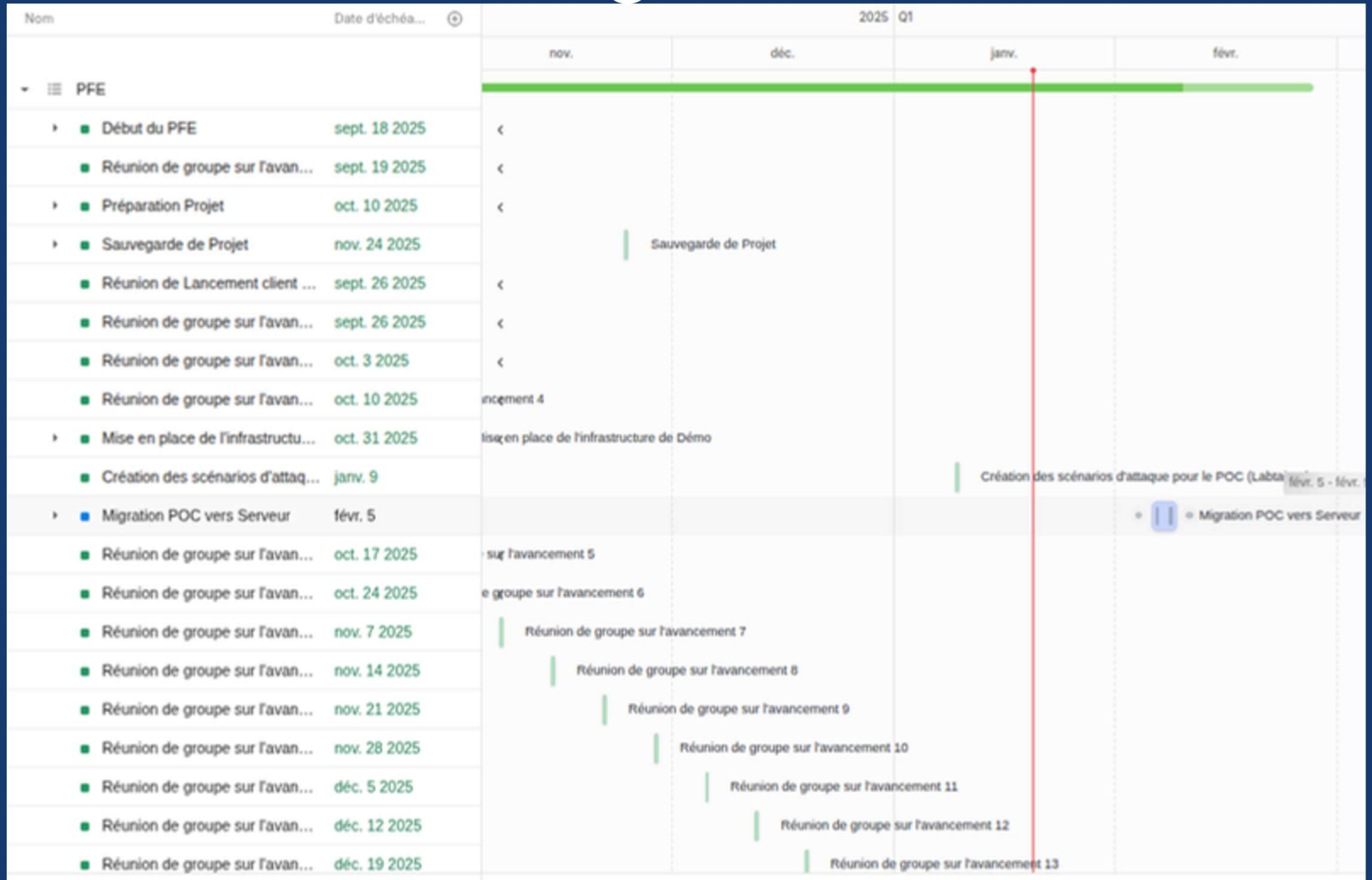
Gestion de projet



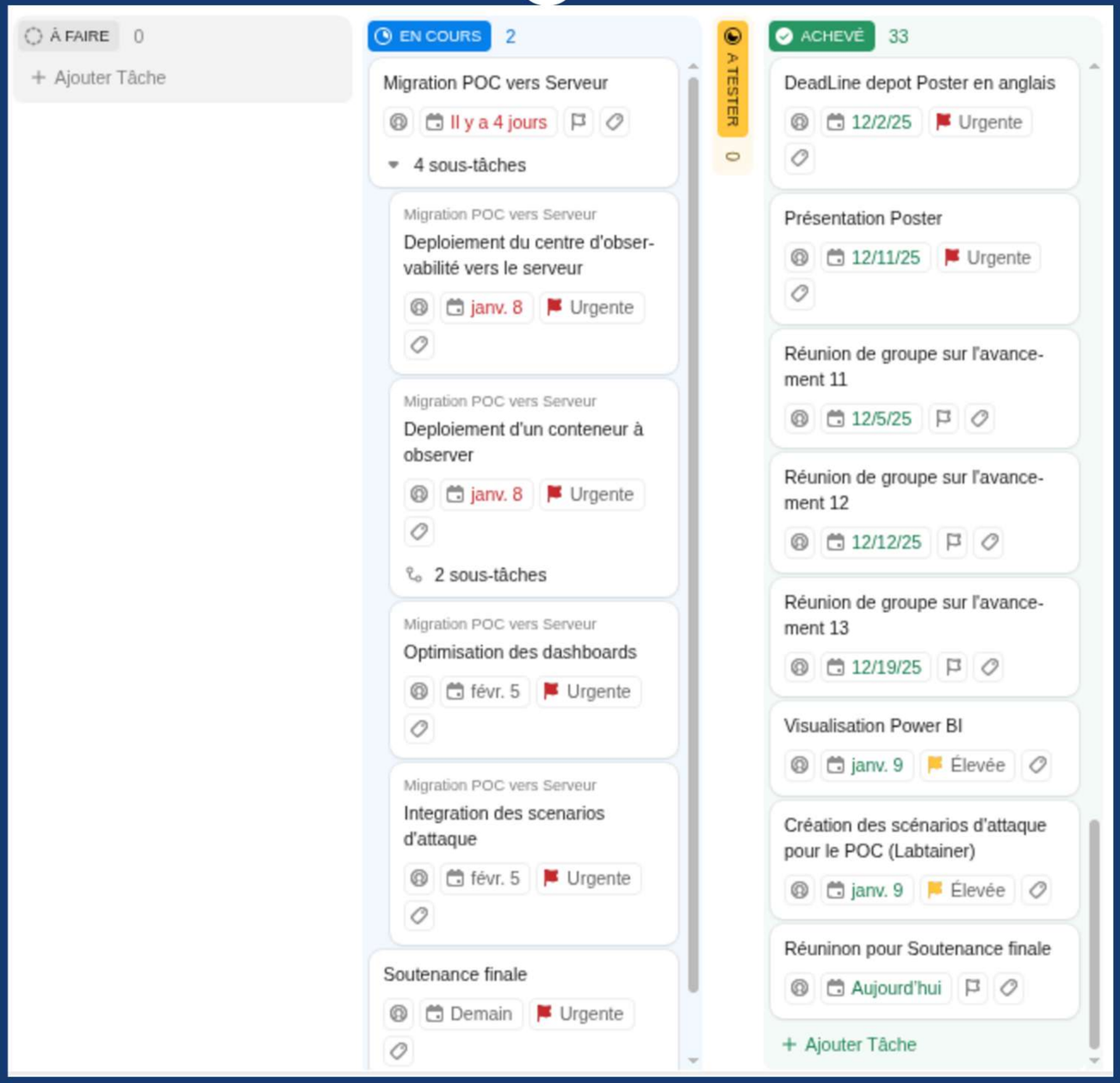
Planning prévisionnel



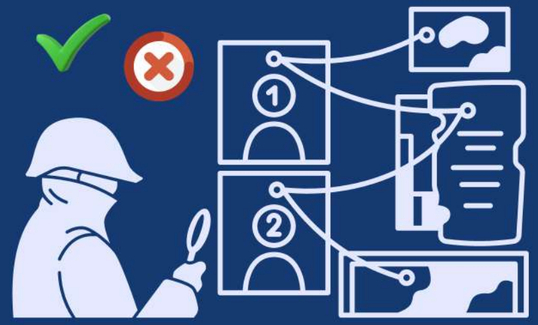
Planning effectif



Planning effectif



Demarche validation



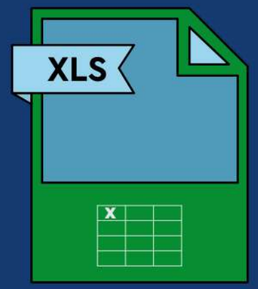
Identifier les exigences ou les hypothèses à valider.

Définir des critères de validations mesurables :

- Codes couleurs
- Messages

Choisir les moyens de validation :

- Excel
- Tests manuels



Demarche validation

	A	B	C	D	E	F	G	H
1	**IDTest**	**Objectifs**	**Préconditions**	**Valeurs en entrée**	**Résultats attendus**	**Testeur**	**Résultats obtenus**	**Remarque**
2	DEV-01	Vérifier le déploiement de api.war sur Tomcat	VM Tomcat accessible, fichier api.war prêt	api.war, accès SSH	L'application est déployée et accessible sur http://<vm>:8080/api/health avec code **200**	Bassit/Ornel	Validate	Vérifier permissions du fichier avant déploiement
3	DEV-02	Vérifier l'endpoint /health	api.war déployé, Tomcat démarré	curl -I http://<vm>:8080/api/health	Retourne **200** et JSON {"status":"UP"}	Bassit/Ornel	Validate	Tester dépendances (DB down)
4	DEV-03	Vérifier l'ingestion des logs applicatifs dans Loki	Promtail configuré sur la VM	Requête de test	Le log apparaît dans Loki sous 30s avec job=tomcat	Bassit/Ornel	Validate	Vérifier niveaux INFO/WARN/ERROR
5	DEV-04	Vérifier corrélation entre logs et métriques	Grafana connecté à Prometheus & Loki	Appel répété de l'endpoint	Logs et métriques se correspondent (même request_id)	Bassit/Ornel	Validate	Vérifier propagation du request_id
6	DEV-05	Vérifier conformité des logs	Format structuré JSON activé	Appels HTTP variés	Logs contiennent timestamp, level, request_id	Bassit/Ornel	Validate	Standardiser format JSON
7	DEV-06	Vérifier que l'API /utilisateurs rejette les tentatives d'injection SQL et que l'événement est logué dans Tomcat puis visible via Grafana/Loki.	L'API Java est déployée et accessible sur la VM Tomcat. Promtail est installé et envoie les logs Tomcat vers Loki. Grafana est configuré avec Loki comme source de données. L'endpoint testé /user?id=<valeur> existe et est censé accepter un entier (id).	http://<IP_VM>:8080/api/utilisateurs?id=1 OR 1=1	API renvoie 400 Bad Request Tomcat log : WARN - SQL injection attempt detected for parameter 'id' Grafana/Loki affiche le log Base de données inchangée	Bassit/Ornel	In Progress	L'API rejette la requête L'événement est logué Grafana affiche le log La base reste protégée
8	DEV-07							
9	DEV-08							
10	DEV-09							

Demarche validation

	A	B	C	D	E	F	G	H	I
1	IDTest	Objectifs	Précondition	Valeurs entrée/Action	Résultats attendus	Testeur	Résultat du test	Résultats obtenus	Remarques
2	OPS-1	Vérifier la collecte des métriques Prometheus sur tous les serveurs	Les VMs sont démarrées et Prometheus est en cours d'exécution. Les exporters sont configurés sur chaque VM	Requête Prometheus up{job="xxx"}	Tous les serveurs doivent apparaître avec up=1.	Bassit / Ornel / Elvire / Akim/ Aylone	Validate	Métriques disponibles pour tous les serveurs, services et applicatifs	//
3	OPS-2	Vérifier la centralisation des logs via Loki	Les agents Loki/Promtail sont installés sur chaque VM et configurés avec le bon endpoint Loki.	Commande logger	Log visible dans Loki <30s	Bassit / Aylone	Validate	Log disponible pour tous les serveurs, services et applicatifs	Vérifier label host
4	OPS-3	Vérifier la création de dashboards techniques.	Grafana opérationnel, données Prometheus disponibles.	Création d'un dashboard "Surveillance Système" avec CPU, RAM, et logs d'erreur.	Les dashbords techniques s'affichent correctement.	Bassit	Validate		//
5	OPS-4	Vérifier la corrélation métriques/logs.	Données Prometheus et Loki disponibles.	Sélection d'un pic d'erreur CPU dans Grafana.	Les logs associés s'affichent dans le même intervalle de temps.	Bassit	In Progress		//
6	OPS-5	Vérifier l'accès Grafana selon les rôles (Dev, Ops).	Comptes Grafana créés avec rôles attribués (Dev, Ops).	Connexion avec identifiants Dev / Ops	Le Dev ne modifie que les dashboards des applications, l'Ops ne modifie que ceux des infrastructures.	Bassit / Akim	Validate	Chaque rôle n'édite que les éléments qui lui sont attribués.	//
7	OPS-6	Vérifier résilience du stack observabilité	Docker Compose up & volumes persistants	Commande docker	Données (logs, dashboards) intactes	Bassit / Ornel	Validate	Dashbord disponible et claire	//
8	OPS-7	Vérifier rétention des logs Loki	Règles de rétention configurées				In Progress		En cours de réflexion
9	OPS-8	Vérifier le déclenchement d'alerte lorsque le CPU dépasse le seuil pendant un temps défini	Le serveur est monitoré par Prometheus. La règle d'alerte CPUHighLoad est configurée dans Prometheus Alertmanager. Le tableau de bord Grafana est opérationnel. le CPU est actuellement < 50 %	Simulation d'une montée du CPU à 90% pendant 6 minutes.	Prometheus détecte la métrique node_cpu_seconds_total anormale. L'alerte CPUHighLoad passe en état FIRING. Alertmanager envoie la notification (email, webhook, Teams, Slack...). L'alerte apparaît dans Grafana avec l'état Critical.	Akim / Bassit	In Progress	Prometheus détecte la métrique L'alerte apparaît dans Grafana avec l'état Critical.	//

Droit du numérique et gestion des ressources externes

Les données collectées (logs systèmes, métriques de performance, etc.) doivent respecter le cadre légal.

Si ces données contiennent des informations personnelles (par ex. adresses IP, identifiants utilisateurs, localisation précise...), elles entrent dans le champ du RGPD.

Il faut garantir :

La sécurité (stockage protégé, chiffrement si nécessaire),
La limitation de la conservation (ne pas garder indéfiniment des données inutiles),
La transparence (savoir quelles données sont collectées et pourquoi).

Ecoresponsabilité

Optimisation des ressources :

- minimum de consommation
- Durée de rétention optimisée

Redondance raisonnée : la sécurité est assurée sans multiplier les serveurs

Réduction de l'empreinte carbone :

- Infrastructure locale

Difficultés rencontrées

- Prise en main de Grafana, Loki, Prometheus
- Collecte des logs et métriques grâce aux agents tout en tenant compte des ports et adresses IPs
- Mise en place des dashboards personnalisés
- Mise en place des scénarios d'attaque
- Prise en main de scaphandre
- Mise en place de Power BI et des bases de données

Conclusion

Grâce à cette solution d'observabilité unifiée, l'ESEO dispose enfin d'une vision claire et instantanée de toute son infrastructure.

Les incidents sont détectés plus vite, les anomalies anticipées avant d'impacter les étudiants, et la consommation énergétique devient maîtrisable.

Nous mettons à disposition un livrable et un mode d'emploi pour permettre à un utilisateur ou une personne qui voudrais reprendre le projet de pouvoir prendre en main sans trop de problèmes notre centre d'observabilité et l'améliorer.

En bref : **nous transformons une infrastructure opaque en un environnement pédagogique fiable, réactif et durable.**

MERCI POUR VOTRE ATTENTION !