

Architecture de l'application MXT

Organisation des services

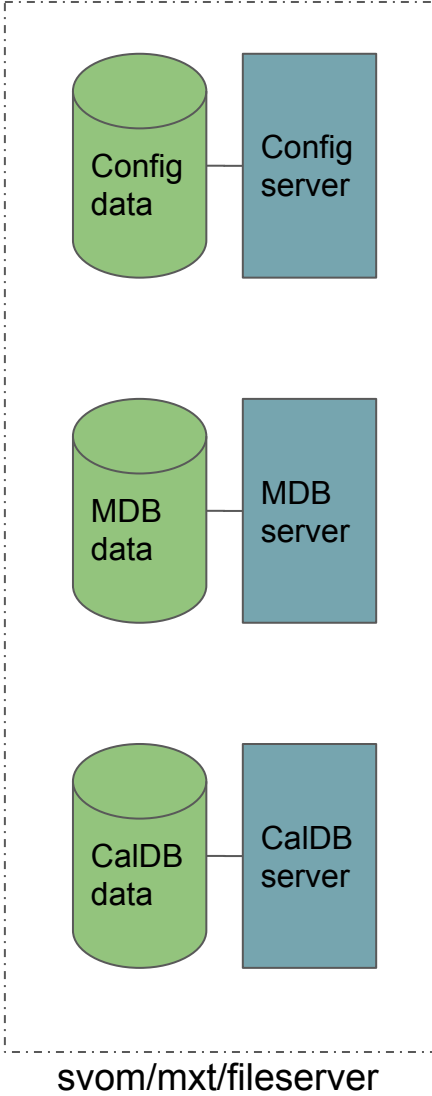
Persistence

Services

Frontend

Organisation des services

Persistence

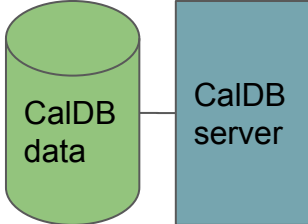
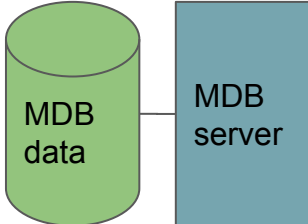
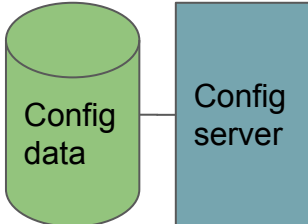


Services

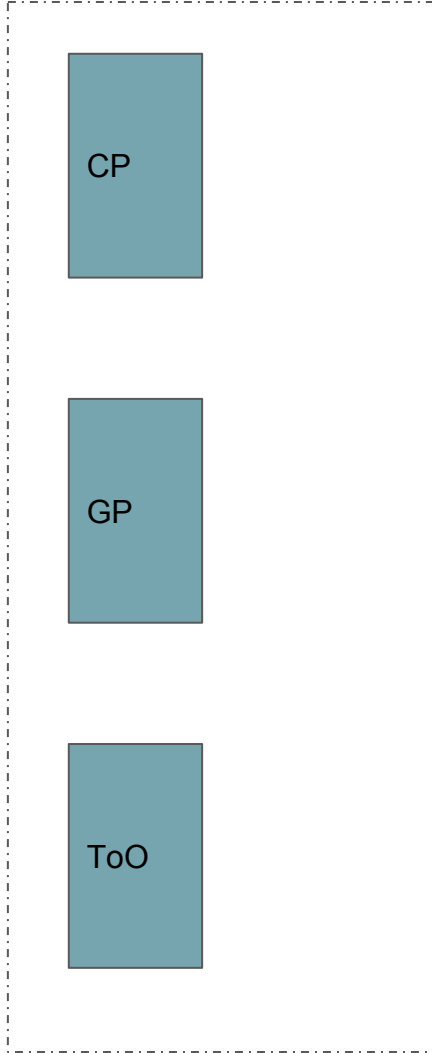
Frontend

Organisation des services

Persistence



Services

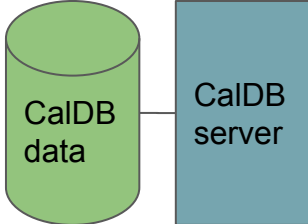
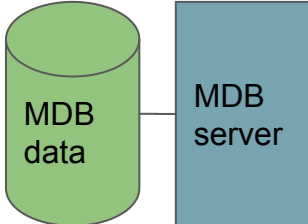
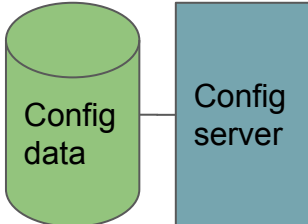


pipeline-bricks

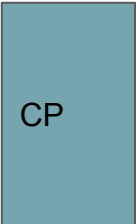
Frontend

Organisation des services

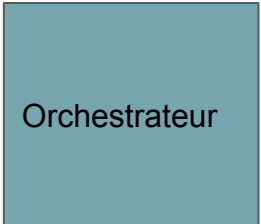
Persistence



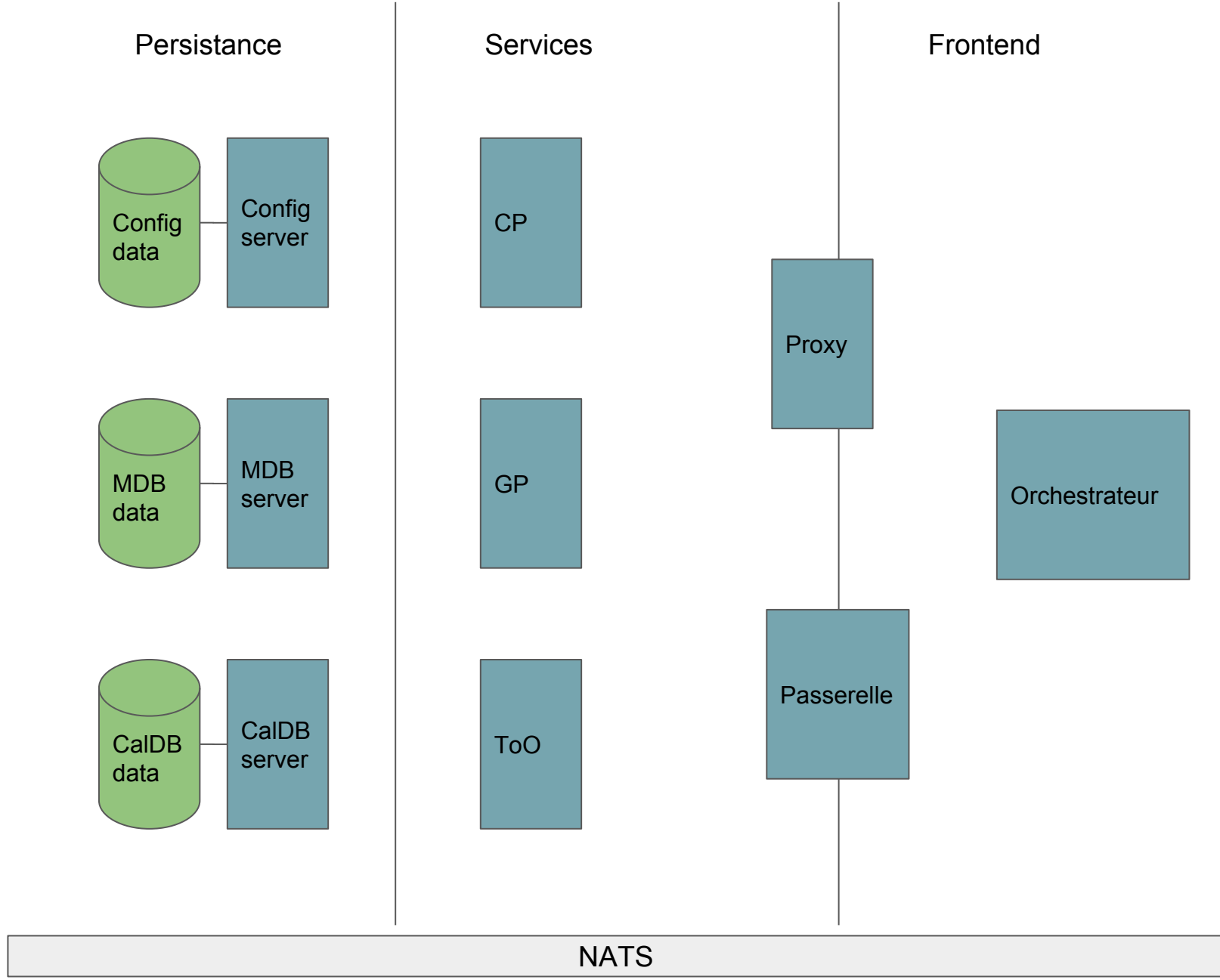
Services



Frontend



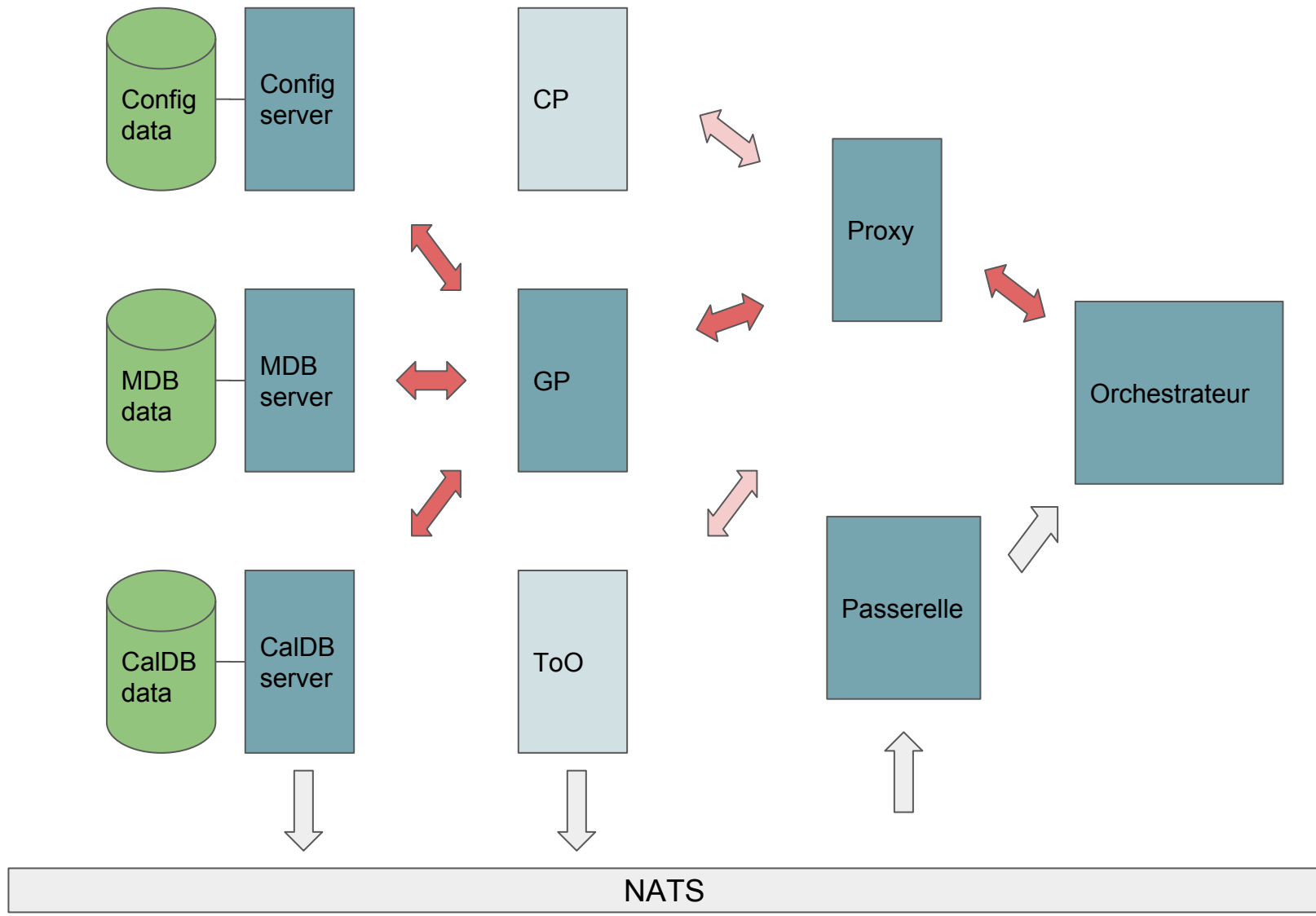
Communication entre les services



Communication entre les services

➡ NATS

↔ REST



Conclusion

- Architecture fonctionnelle
 - Lancer des traitements
 - Sauvegarder les données
 - Récupérer les logs
- Permet de gérer les traitements indépendamment de l'architecture
- Demande une intégration à l'architecture globale
- Reste à intégrer la physique

Quelques mots sur le Proxy

Utilisation de Nginx en mode **reverse proxy** -> projet [svom/mxt/proxy](https://svom.mxt.proxy)

Objectif:

- Minimiser les ports ouverts vers l'extérieur

Avantages:

- Un seul point d'entrée
- Adresses plus simples
- Déploiement facilité
- Plus de sécurité

proxy path	service	adresse
/mxt/	orchestrator	http://master:1024/
/mxt_configdb/	configdb	http://config:1024/
/mxt_missiondb/	mission db	http://missiondb:1024/
/caldb/	CALDB	http://caldb:1024/
/mxt_gp_pipeline/	General program	http://generalprogram:1024/
/mxt_cp_pipeline/	Core program	http://coreprogram:1024/
/mxt_too_pipeline/	Target of opportunity	http://too:1024/
/mxt_websocket/	websocket	ws://gateway:1024/

Quelques mots sur le Proxy

Difficultés :

- le proxy doit être démarré en dernier car il ne 'voit' pas les nouveaux services apparaître

Solution:

- Remettre à jour la config automatiquement
- [jwilder/nginx-proxy](#)
 - Surcouche a nginx
 - Connection au démon docker