

authors :

- S. LAVAZAIS
- J. SPICHT
- A. QUÉRÉ

sources:

- <https://excalidraw.com>
- <https://nx.dev>

PRÉSENTATION NX

Par



SOMMAIRE

1. Qu'est-ce qu'un mono-repository ?
2. Cas appliqué : un projet micro service
3. Et NX dans tout ça ?

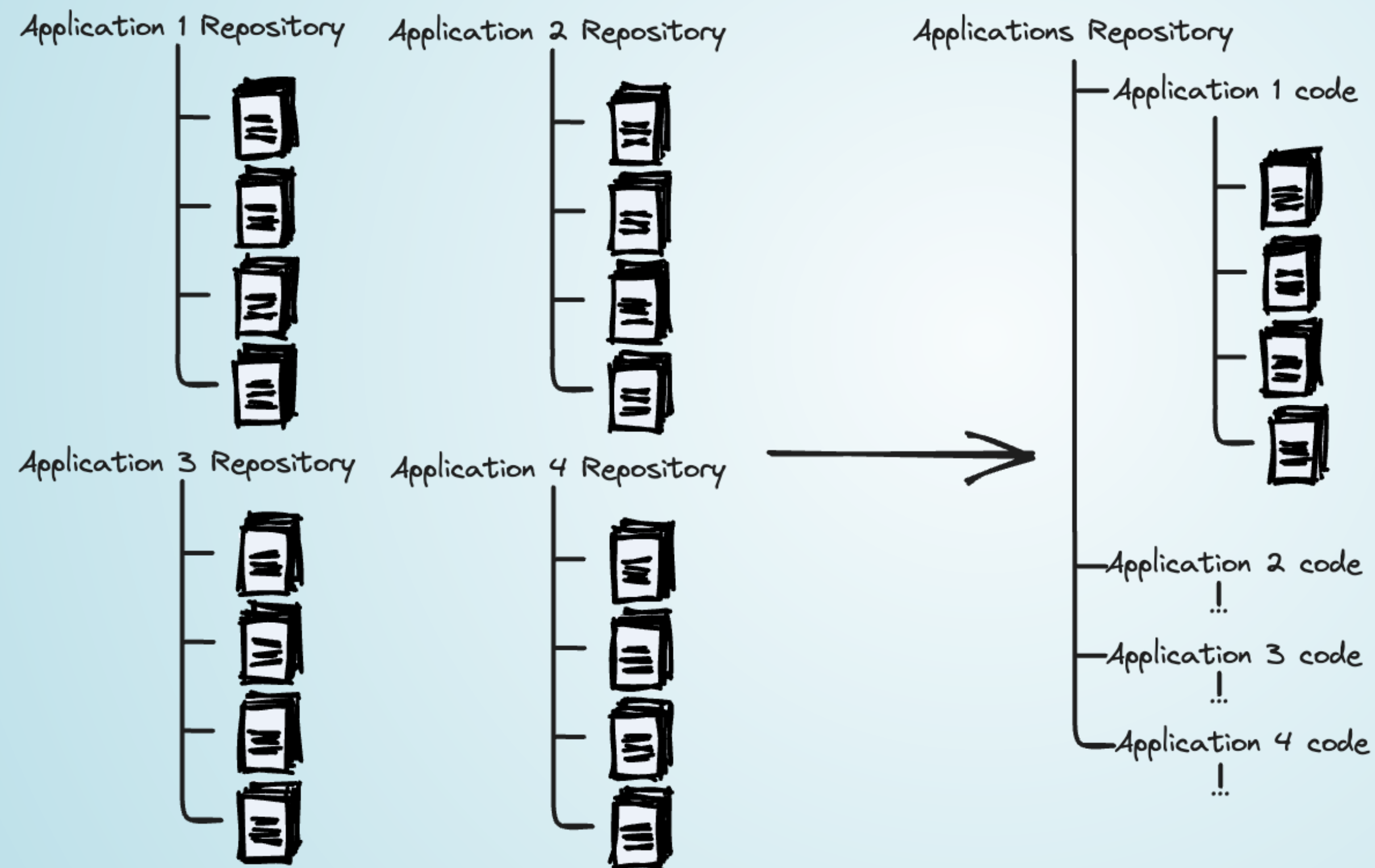
1. Qu'est-ce qu'un mono-repository ?
2. Cas appliqué : un projet micro service
 - présentation du cas
 - le cycle de dev
 - en multi-repo
 - en mono-repo
 - l'organisation des releases
 - en multi-repo
 - en mono-repo
3. Et NX dans tout ça ?
 - décrire le principe du fonctionnement de NX
 - parler de ce qui a été réalisé sur le projet DavLab
 - Demo !
4. Questions
5. Remerciements

Lorsque l'on gère le code d'un projet type "micro-service", on range souvent le code dans plusieurs dépôts de code. (figure de gauche)

Le principe de base d'un mono-repository est de mettre le contenu de tous ces dépôts dans un seul dépôt de code afin de garder une seule et même arborescence. (figure de droite)

QU'EST-CE QU'UN "MONO-REPOSITORY" ?

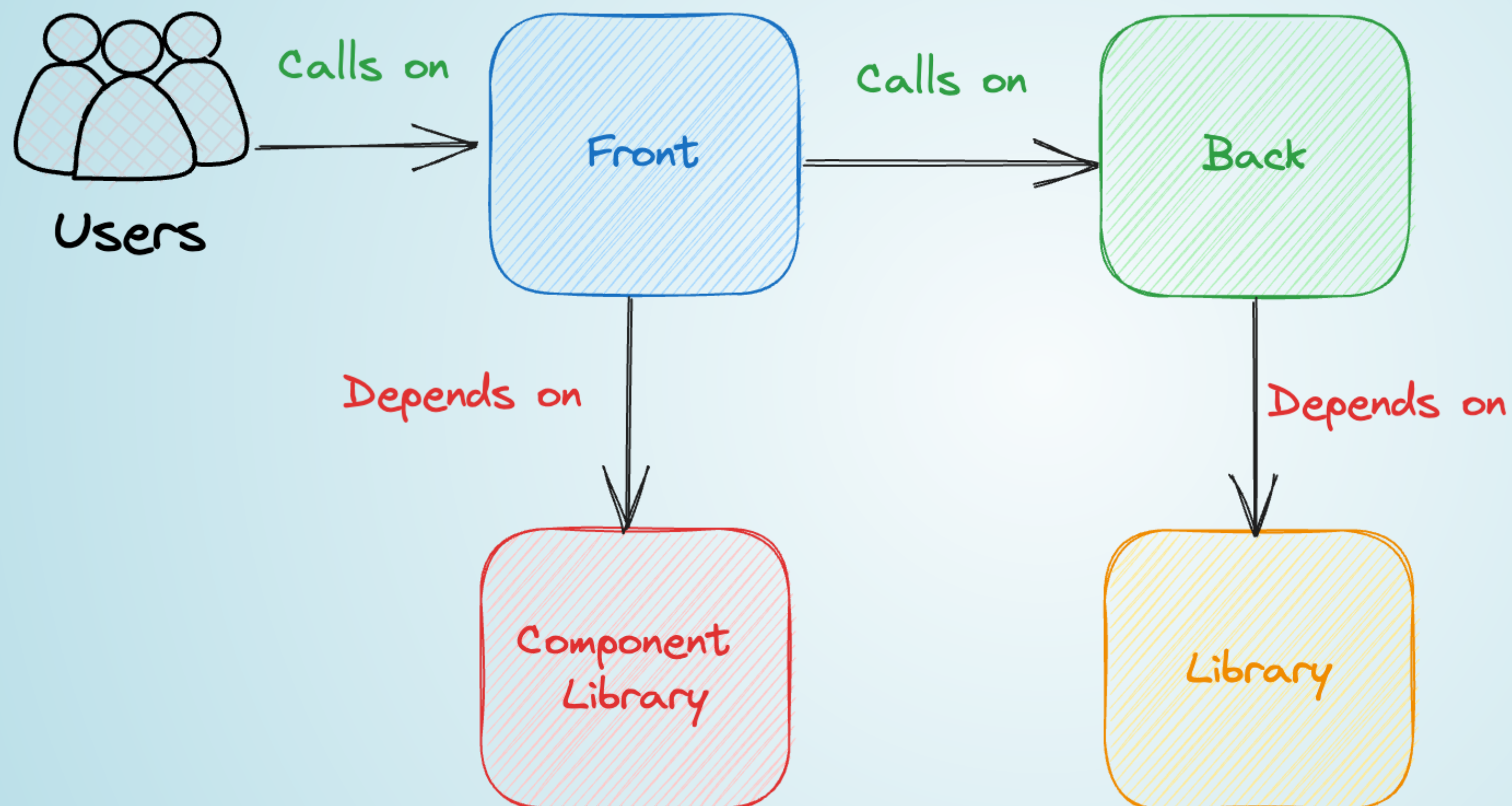
Le principe de base



Exemple en tête, un projet de micro-service typique, avec un back, un front, une librairie pour le back et une librairie de composant pour le front.

Maintenant, on va voir la différence concrète entre une organisation de code "classique" et une organisation en mode "mono-repository"

CAS APPLIQUÉ : UN PROJET MICRO SERVICE

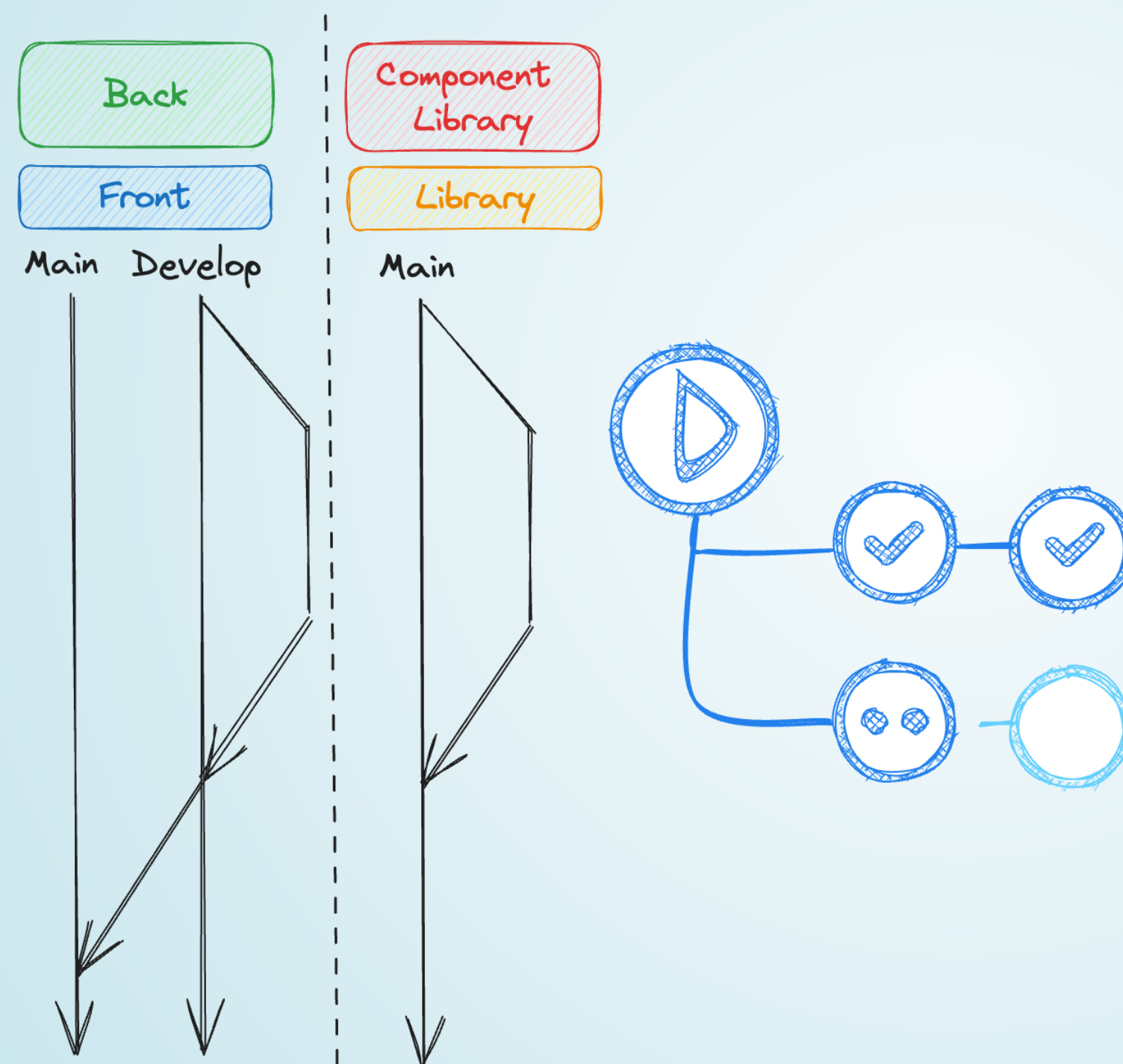


Comme on a pu le voir précédemment, le code est donc découpé en quatre dépôts (un pour chaque partie de ce micro-service).

Et Chacun de ces composants va avoir son propre cycle de développement

CAS APPLIQUÉ : UN PROJET MICRO SERVICE

Le cycle de développement - multi



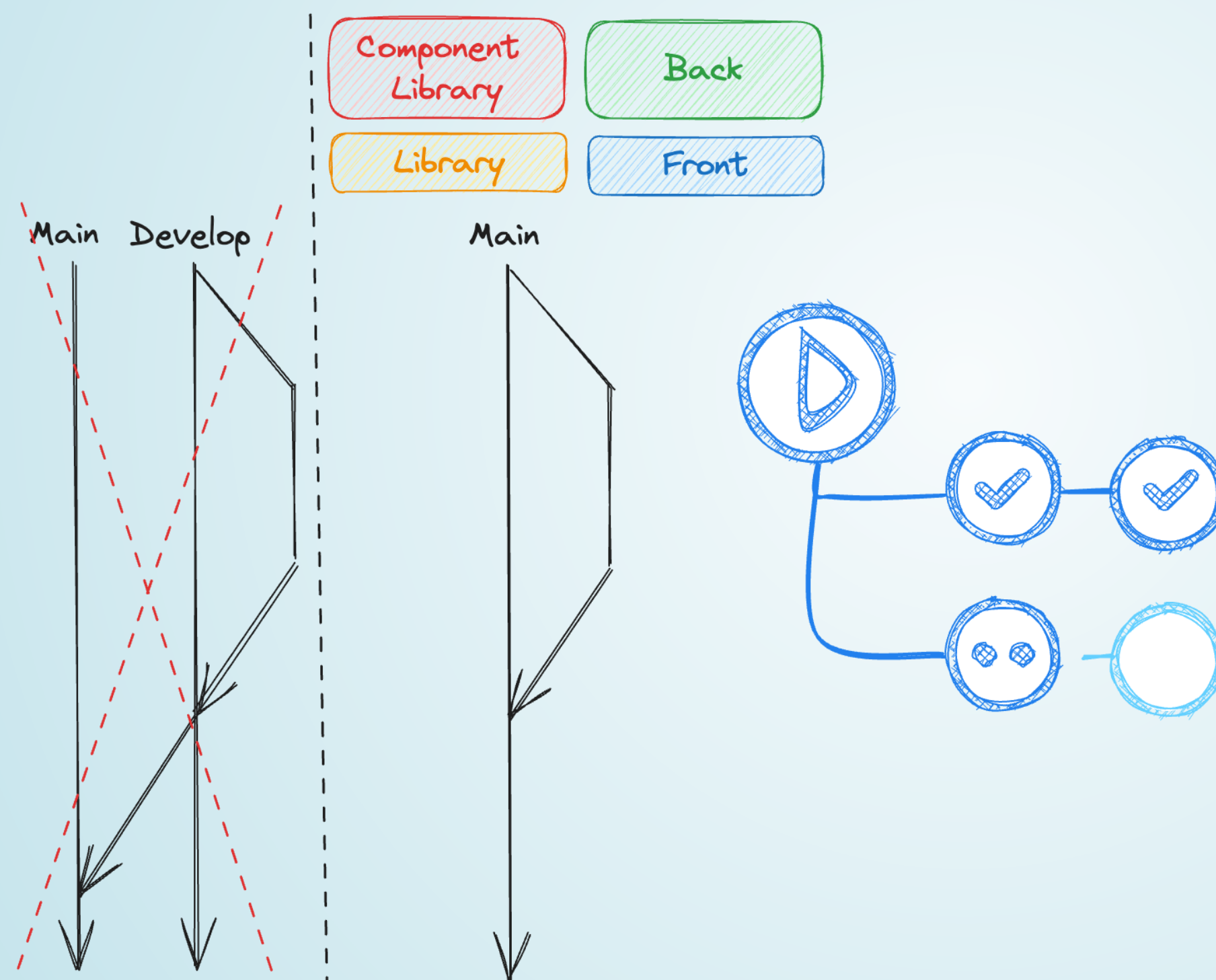
Dans le cas d'un mono repository, on ne peut pas avoir deux cycles de développements différent.

Il faut donc en choisir un et faire en sorte que chaque composant s'inscrive dans ce nouveau cycle de développement.

Par soucis de simplicité, on peut choisir le trunk base, mais sur le papier, le git flow fonctionne également.

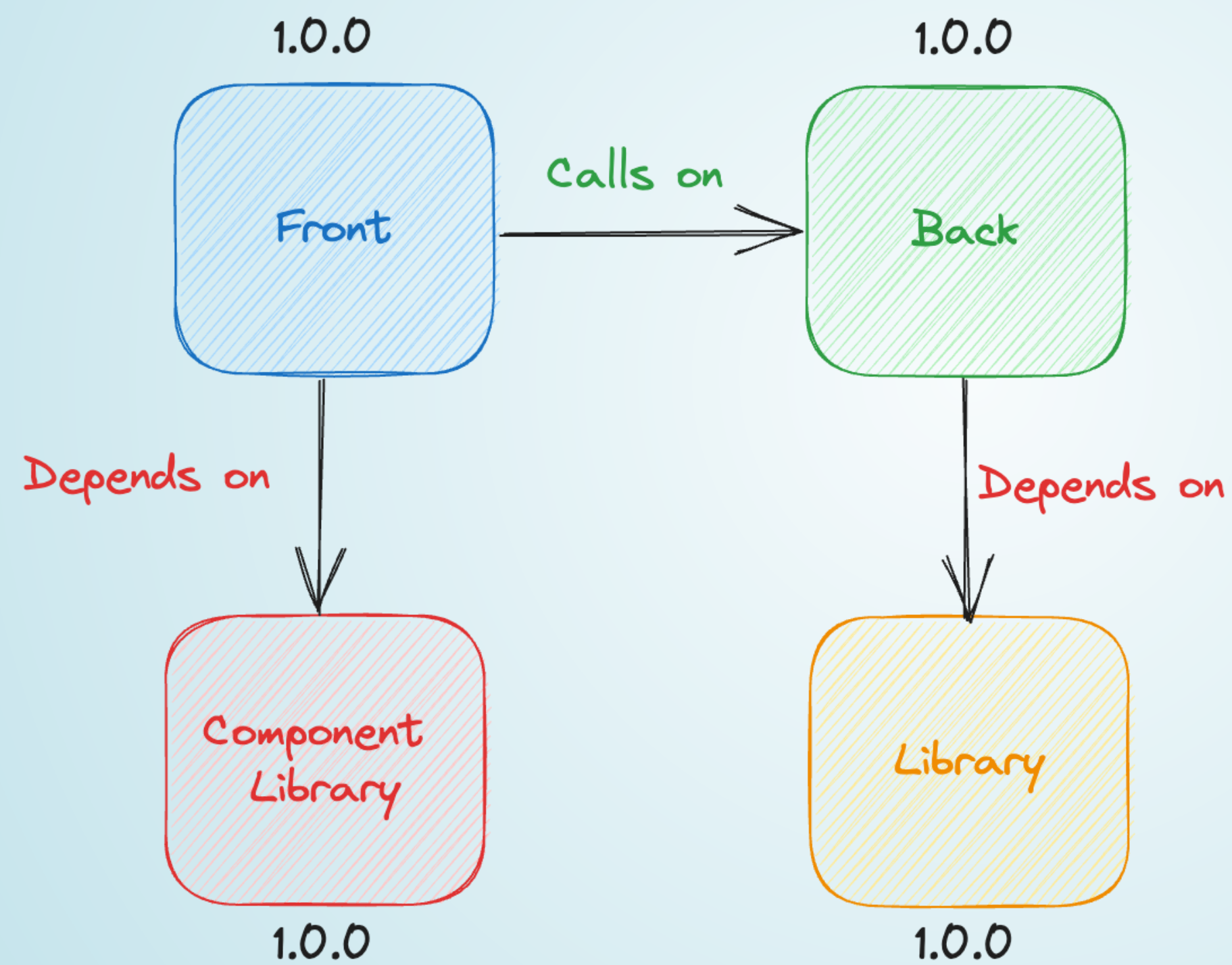
CAS APPLIQUÉ : UN PROJET MICRO SERVICE

Le cycle de développement - mono



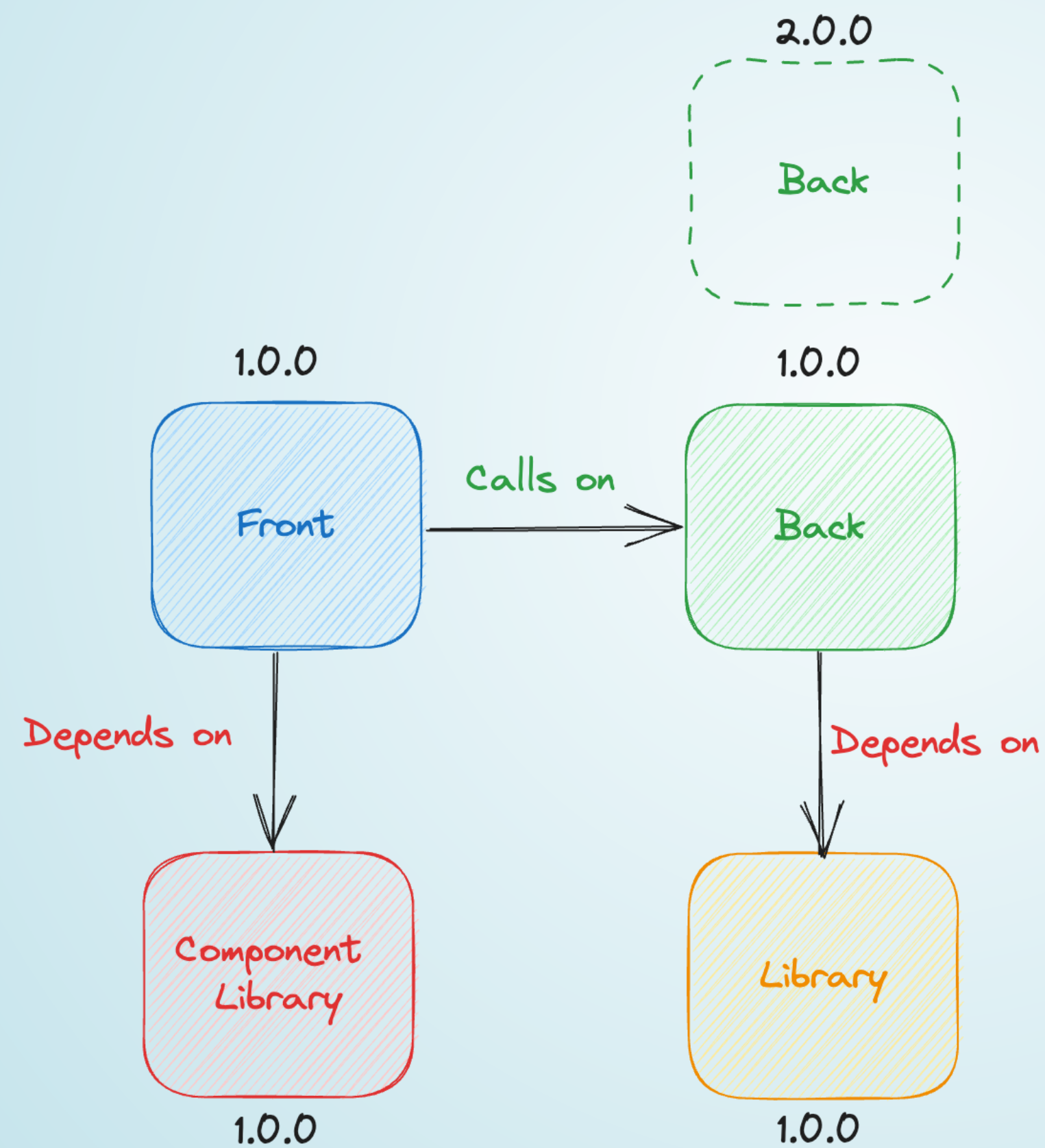
CAS APPLIQUÉ : UN PROJET MICRO SERVICE

L'organisation des releases - multi



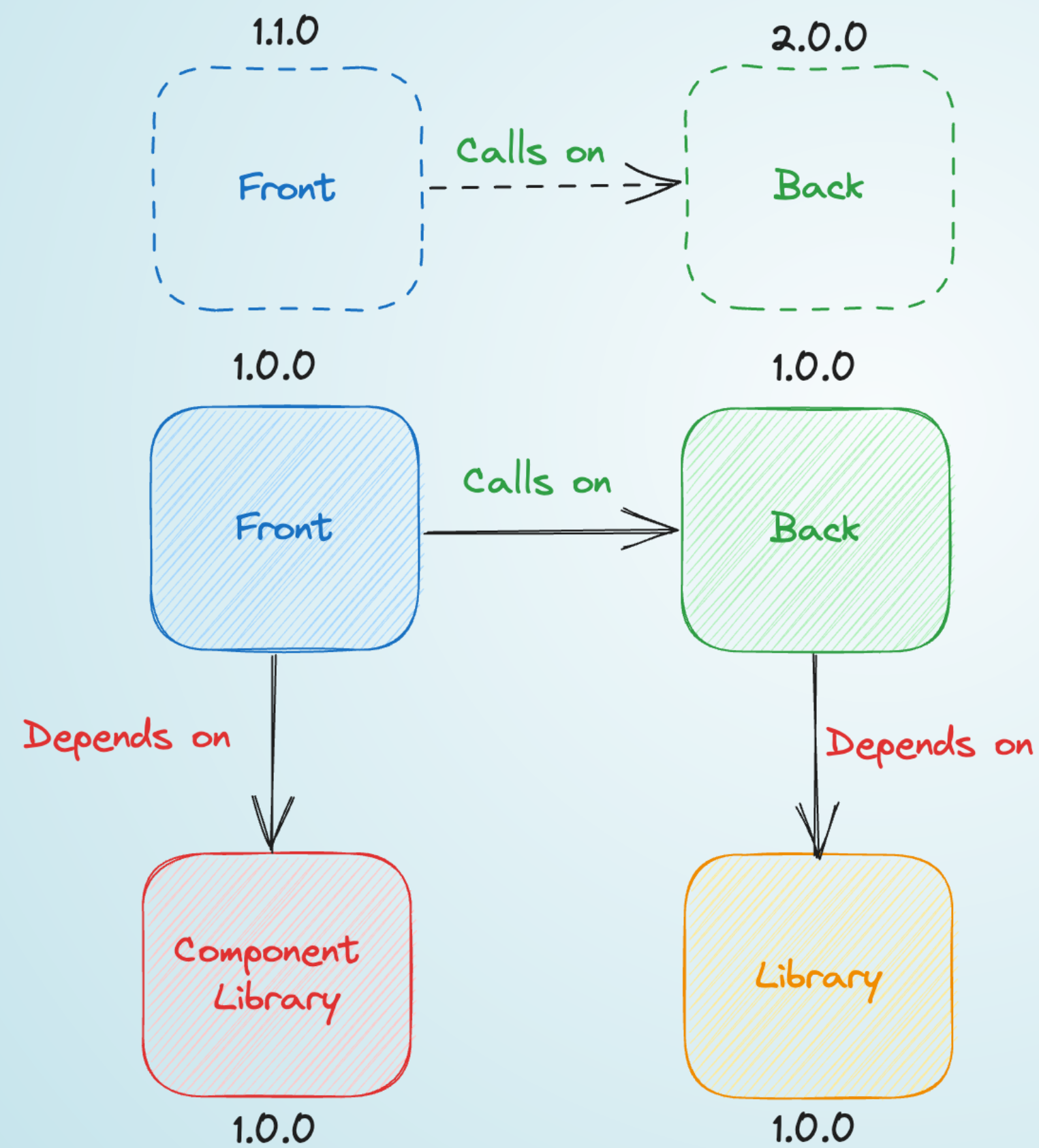
CAS APPLIQUÉ : UN PROJET MICRO SERVICE

L'organisation des releases - multi



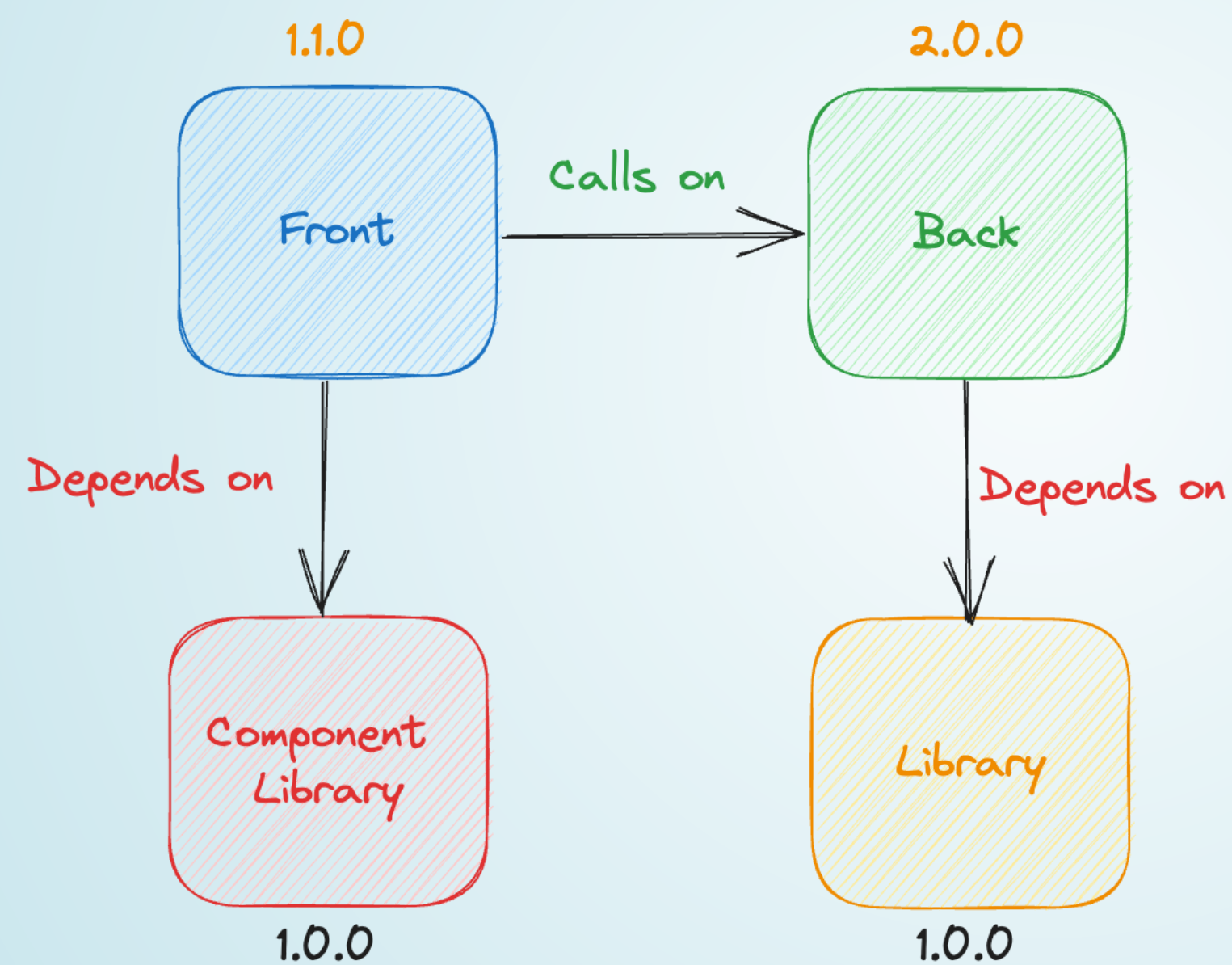
CAS APPLIQUÉ : UN PROJET MICRO SERVICE

L'organisation des releases - multi



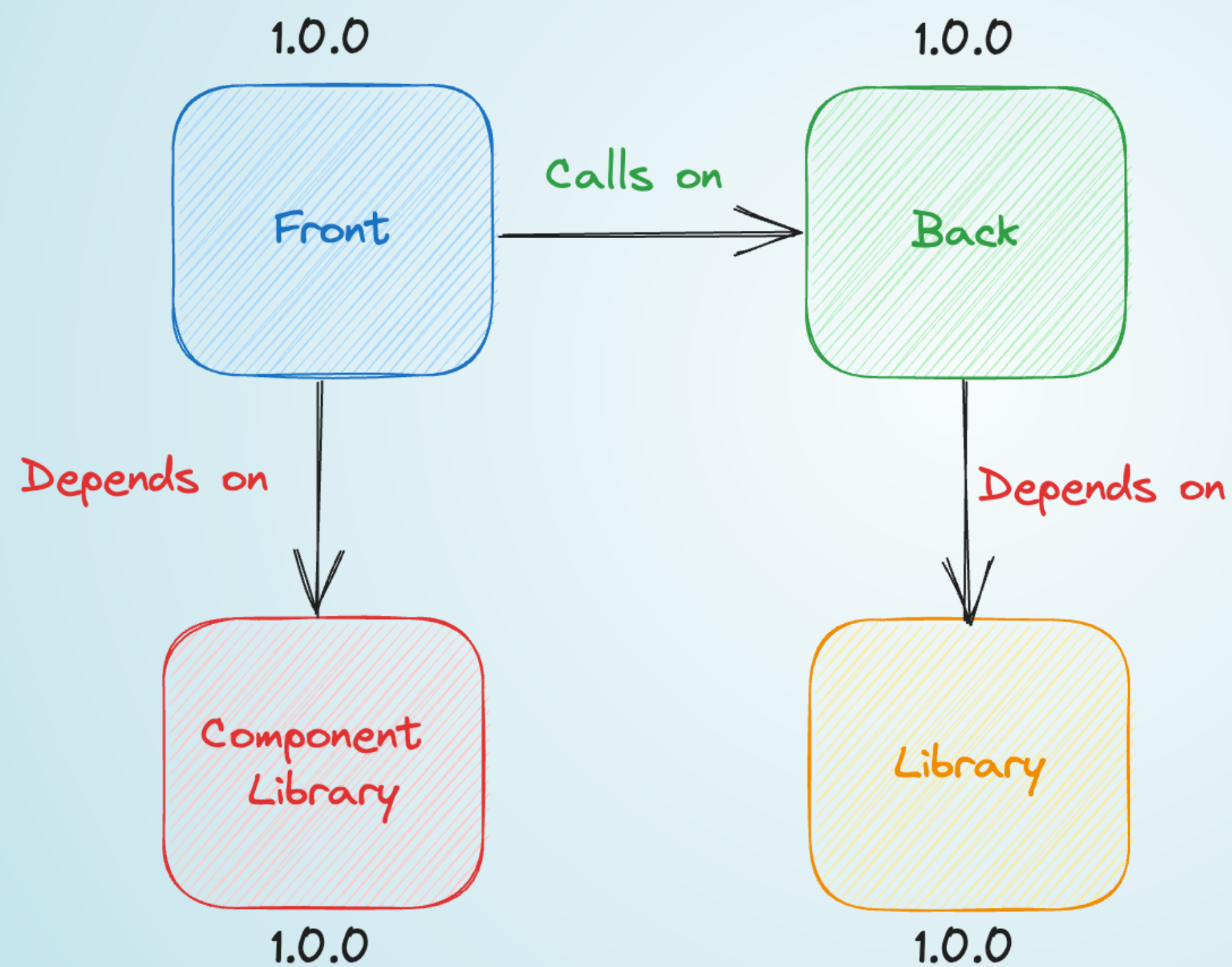
CAS APPLIQUÉ : UN PROJET MICRO SERVICE

L'organisation des releases - multi



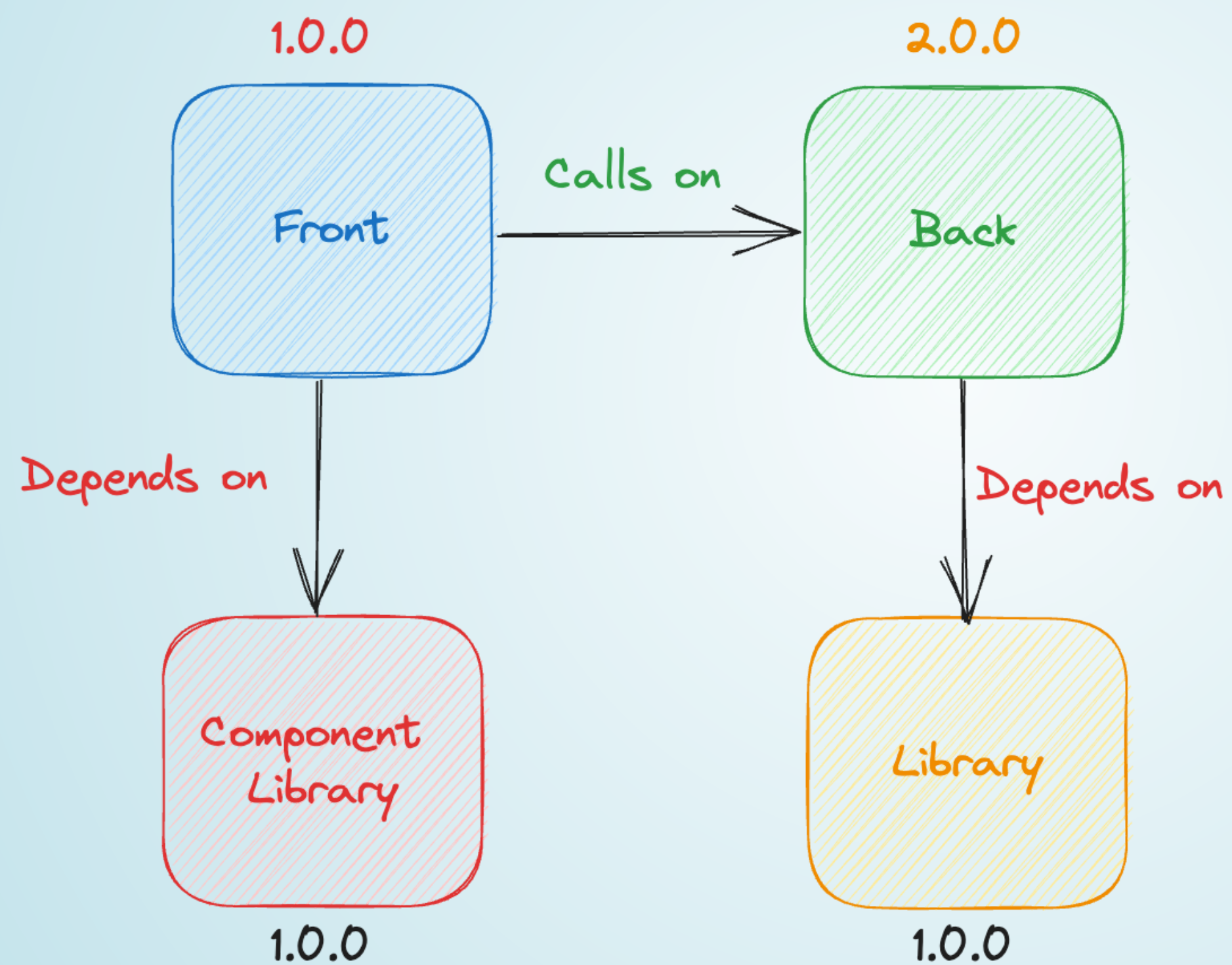
CAS APPLIQUÉ : UN PROJET MICRO SERVICE

L'organisation des releases - mono



CAS APPLIQUÉ : UN PROJET MICRO SERVICE

L'organisation des releases - mono



NX est un framework conçu par "NRWL" (narval) comportant des outils et des techniques simples permettant d'accélérer grandement la productivité et la rendre plus simple du point de vue de l'expérience développeur.

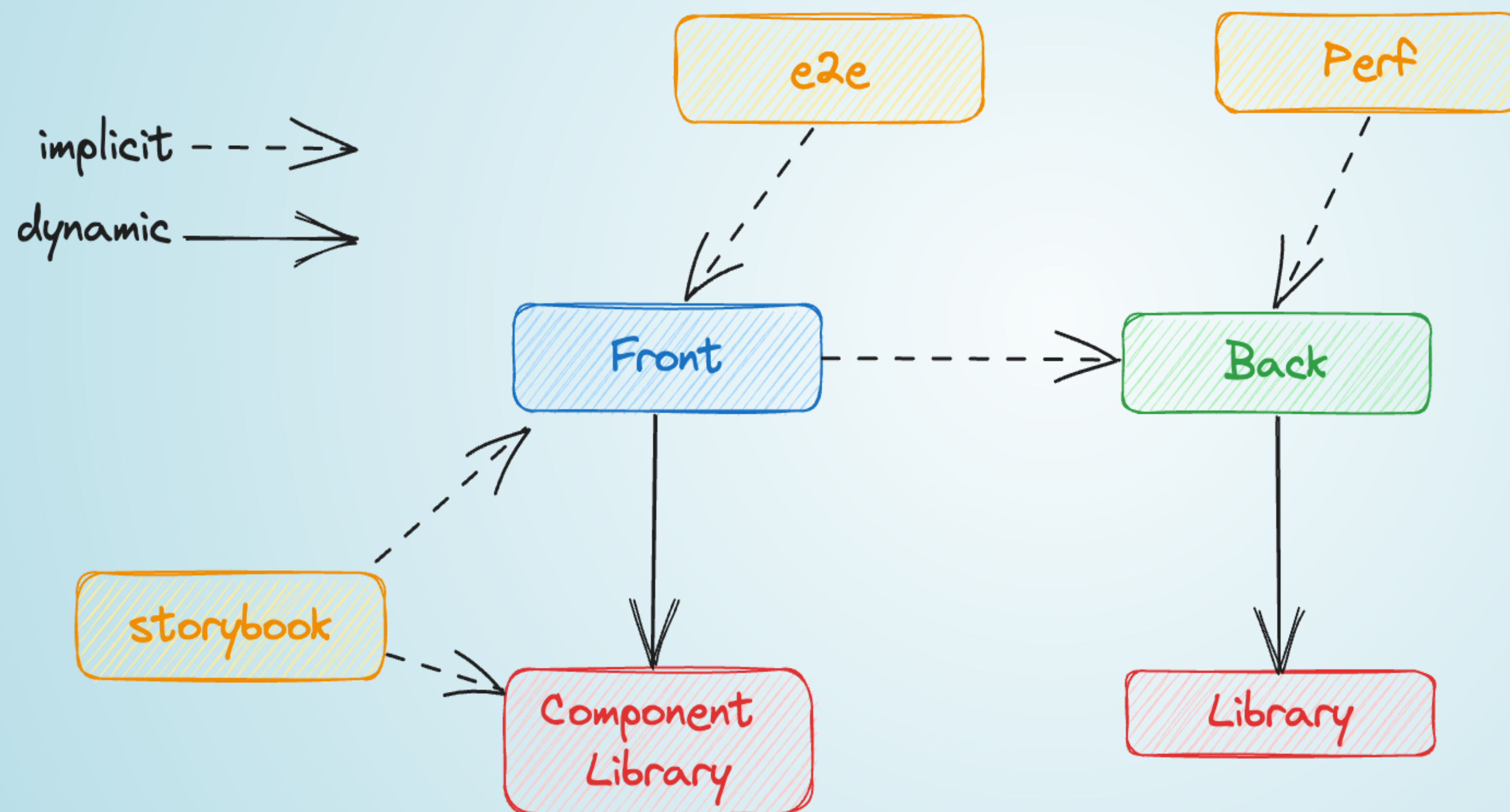
Dans notre cas de projet micro-service NX est l'outil idéal pour mettre en pratique un mono-repository presque parfait

ET NX DANS TOUT ÇA ?



ET NX DANS TOUT ÇA ?

Le graph de dépendance

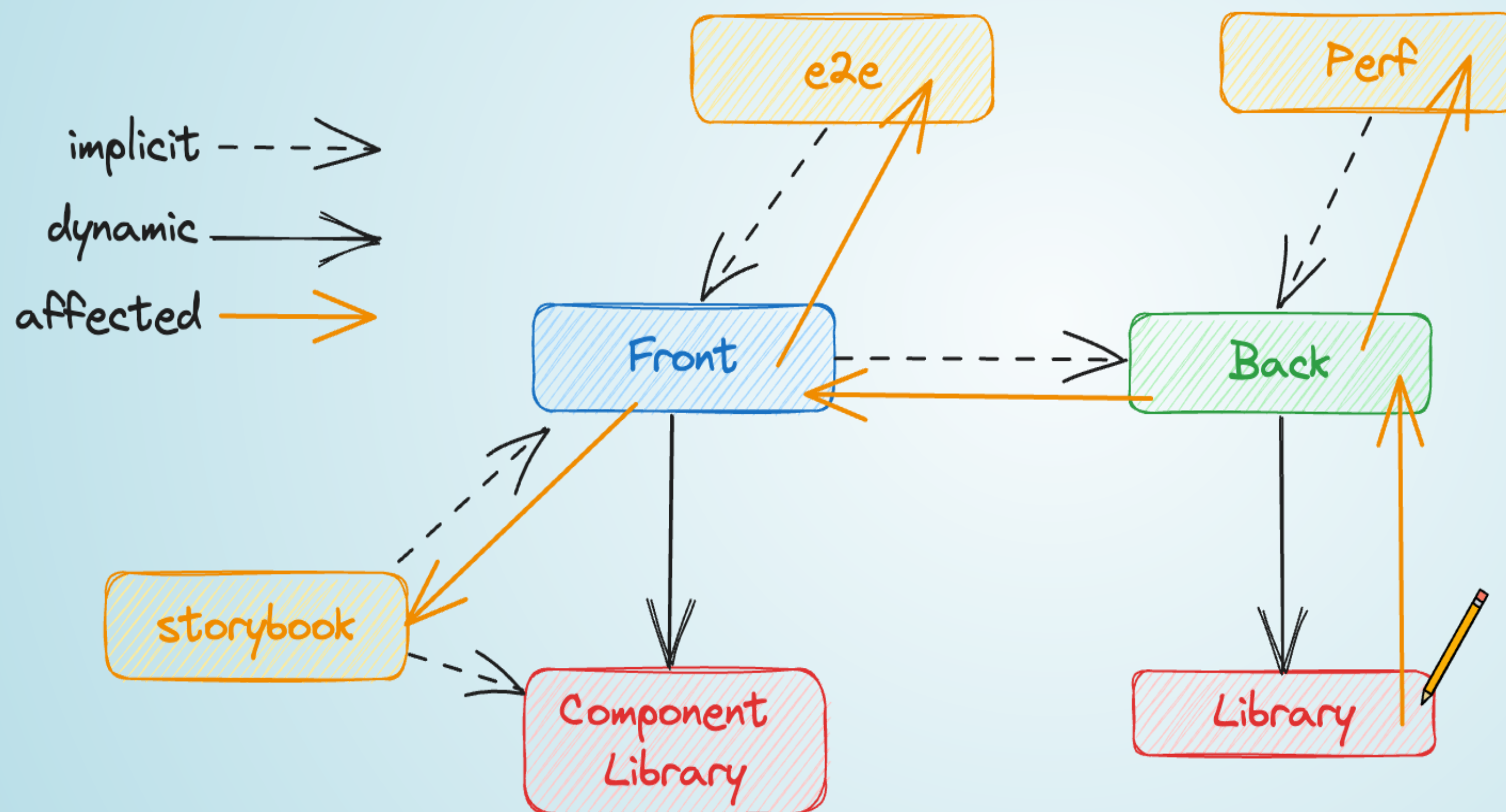


NX est capable de déduire les projets impactés par des modifications. À partir des dépendances entre projets, il est capable de reconstituer toute la chaîne de projets impactés.

Grâce à ce mécanisme, NX est capable de garantir la cohérence de l'ensemble du mono-repo

ET NX DANS TOUT ÇA ?

Le graph de dépendance - les "affected"



ET NX DANS TOUT ÇA ?

Les langages



ET NX DANS TOUT ÇA ?

La Démo

Speaker notes

Pour la démo, pour faire les différentes parties, j'ai retravaillé un peu l'ordre, je vous propose ce scénario :

1. Présenter le repository de Démo Départ : back, lib back, lib front

2. Présenter un générateur Générer l'app Next avec `nx g app`

Modifier le contenu de la page principale (ctrl+c/v) Montrer l'import de la librairie Montrer l'utilisation de l'API

3. Lancer Nx graph Lancer `nx graph`

Montrer le lien entre le back et la lib du back Pareil pour le front

Montrer l'absence de lien entre l'app front et l'app back

4. Rajouter la dépendance implicite entre back et front Modifier le `project.json`

5. Montrer l'intégration dans le graph Relancer `nx graph` Montrer le nouveau lien dans le graphs

6. Montrer 2, 3 commandes commit les changements sur la main 🤖 `git add .` && `git commit -m "feat(myapp-front): create myapp-front"`

Créer une nouvelle branche `git checkout -b my-feat`

Montrer les tests en affected : rien ne se passe `nx affected --target=test`

Modifier la lib

Relancer les tests et regarder les prjets affected `nx affected --target=test`

Montrer lancer un executeur sur un projet specific `nx run myapp-front:serve`

formater tout le code `nx format`

7. NX Cloud (si il reste du temps) Montrer une PR avec des erreurs dans les tests

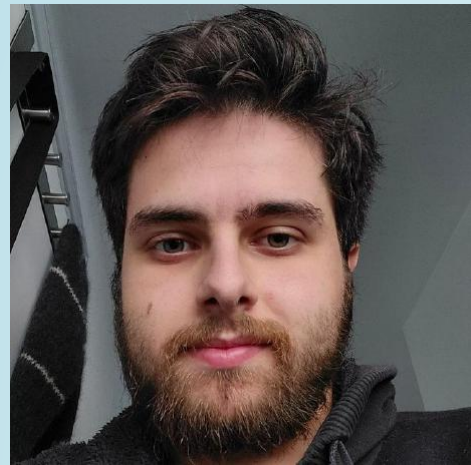
Montrer le rapport de NX sur la PR

Montrer les logs sur la console

Montrer le cache sur la console

MERCI POUR VOTRE TEMPS!

Présenter par



Anthony QUÉRÉ Jules SPICHT Sylvain LAVAZAIS



Sources: