PRÉSENTATION NX

Par



Speaker notes

authors:

- S. LAVAZAIS
- J. SPICHT
- A. QUÉRÉ

sources:

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



SOMMAIRE

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

Speaker notes

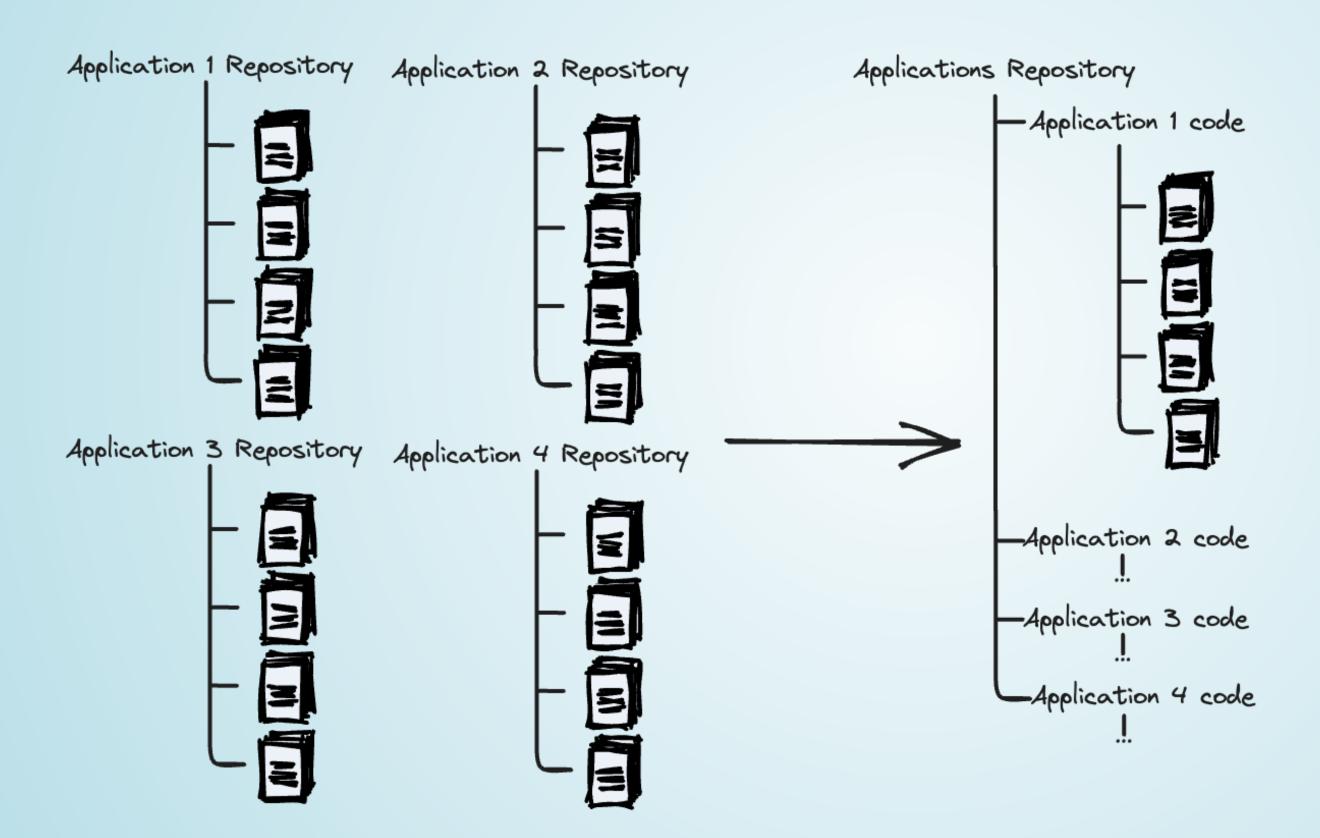
le plan de la présentation

- 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 réaliser sur le projet DavLab
 - Demo!
- 4. Questions
- 5. Remerciements



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

Le principe de base

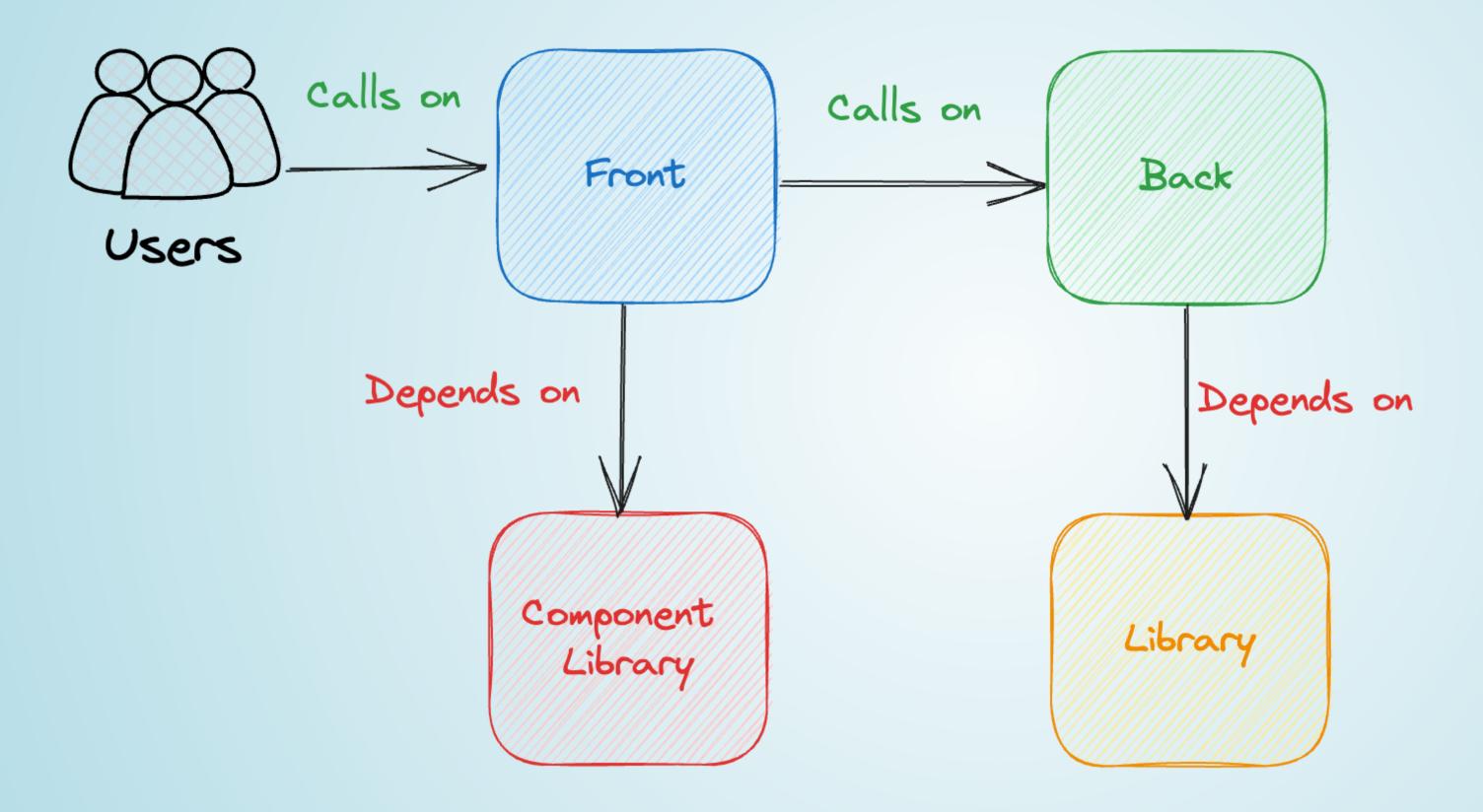




Speaker notes

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)



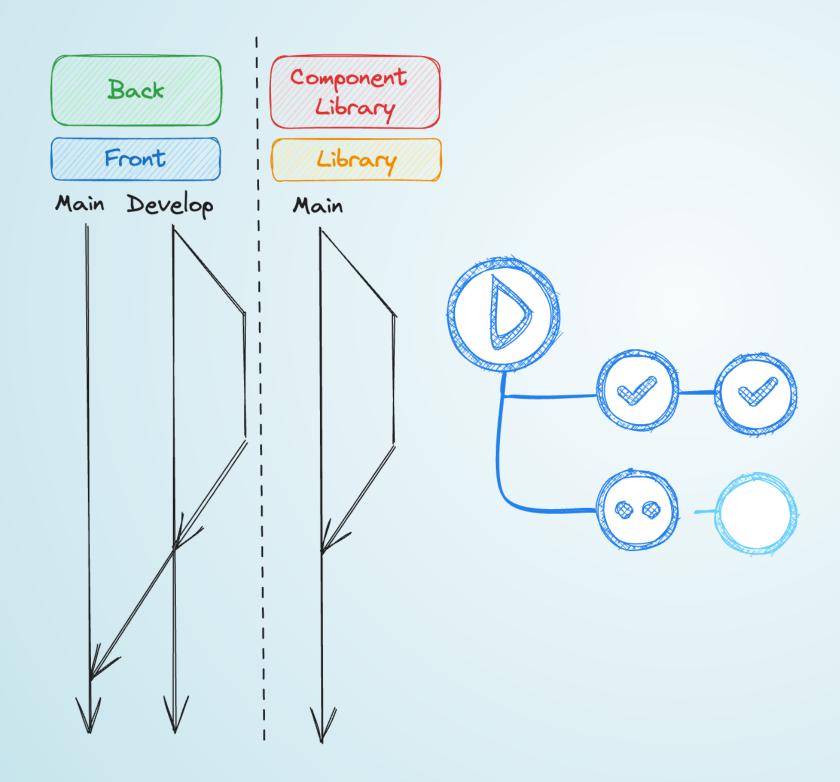


Speaker notes

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"

Le cycle de développement - multi



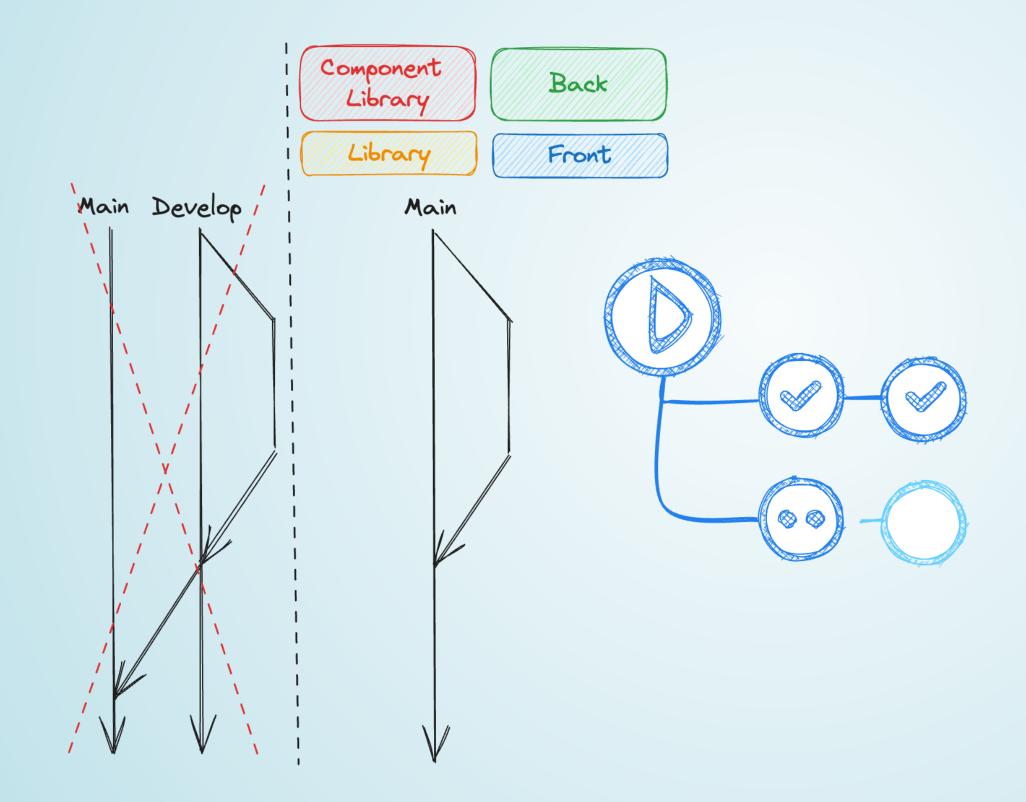


Speaker notes

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

Le cycle de développement - mono





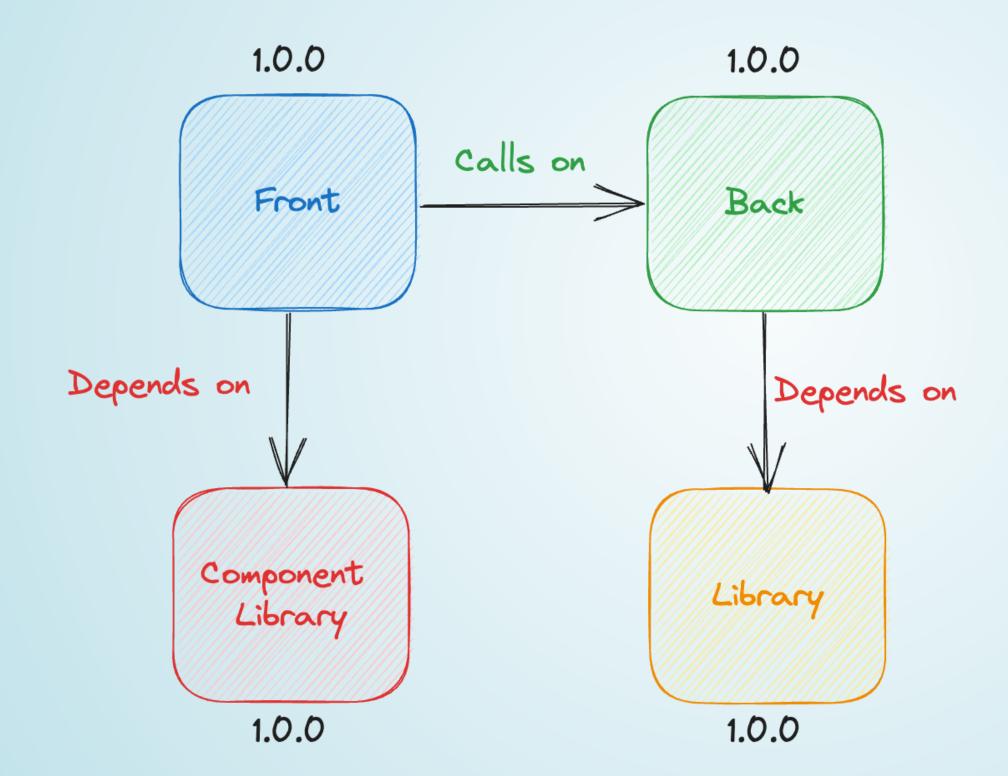
Speaker notes

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

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.

L'organisation des releases - multi

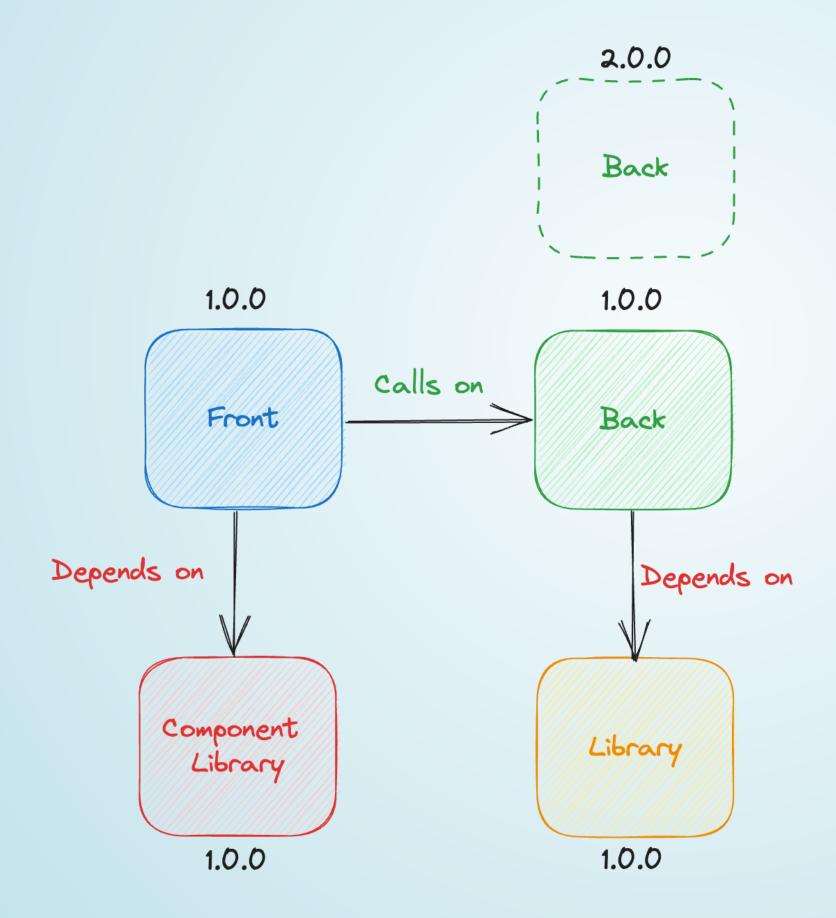




Speaker notes

Sur la partie livraison de composants, chacun des dépôts livre sa propre version.

L'organisation des releases - multi

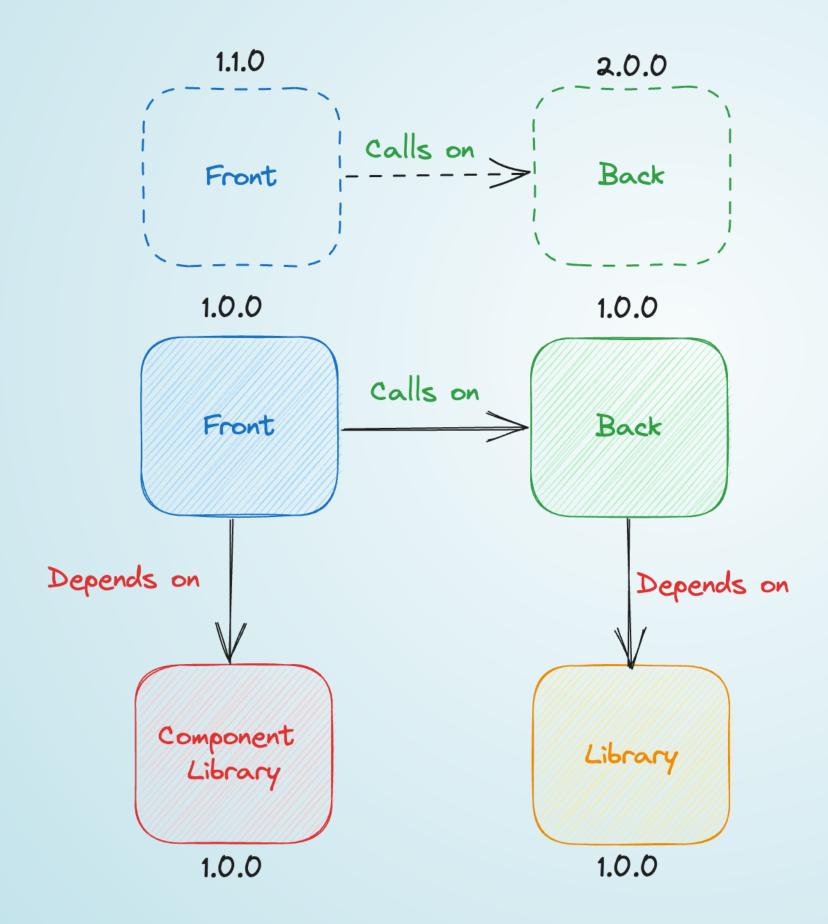




Speaker notes

Par conséquence, lorsqu'un composant crée une dépendance vis à vis d'un autre, alors, on se retrouve à devoir mettre à jour les liens de la dépendance en plusieurs fois.

L'organisation des releases - multi

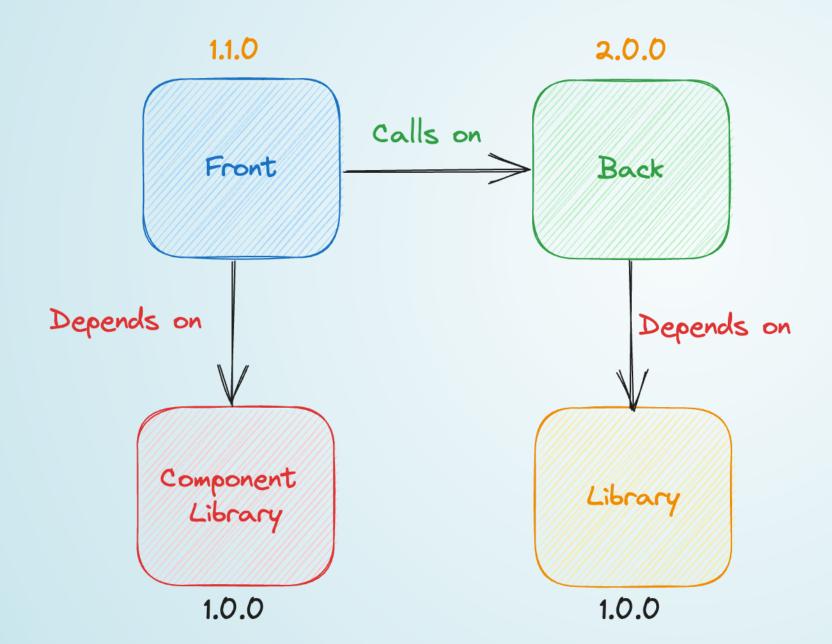




Speaker notes

ce qui oblige donc à maintenir pendant une période de temps plusieurs versions de composants

L'organisation des releases - multi

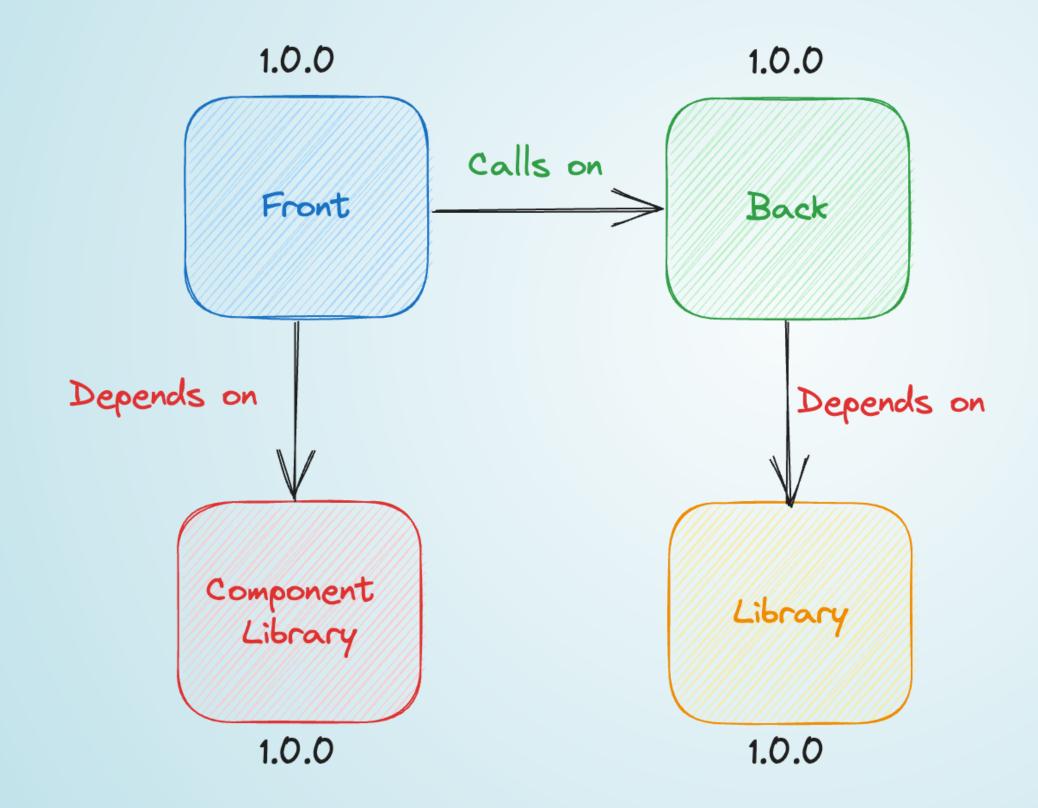




Speaker notes

On obtient à la fin, un cycle de vie qui fonctionne en plusieurs temps.

L'organisation des releases - mono

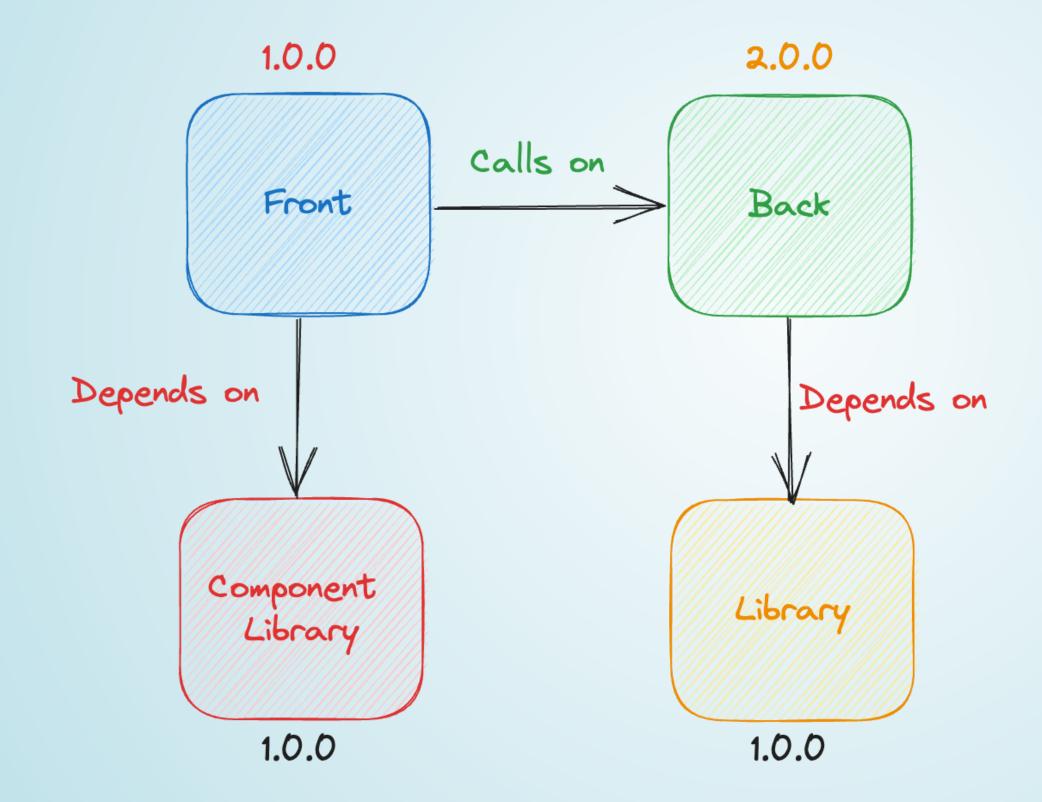




Speaker notes

Dans une mono-repository toute cette mécanique est beaucoup plus simple, car la cohérence entre les composants est systématiquement vérifiable à chaque tentative de livraison.

L'organisation des releases - mono





Speaker notes

Dans une mono-repository toute cette mécanique est beaucoup plus simple, car la cohérence entre les composants est systématiquement vérifiable à chaque tentative de livraison.



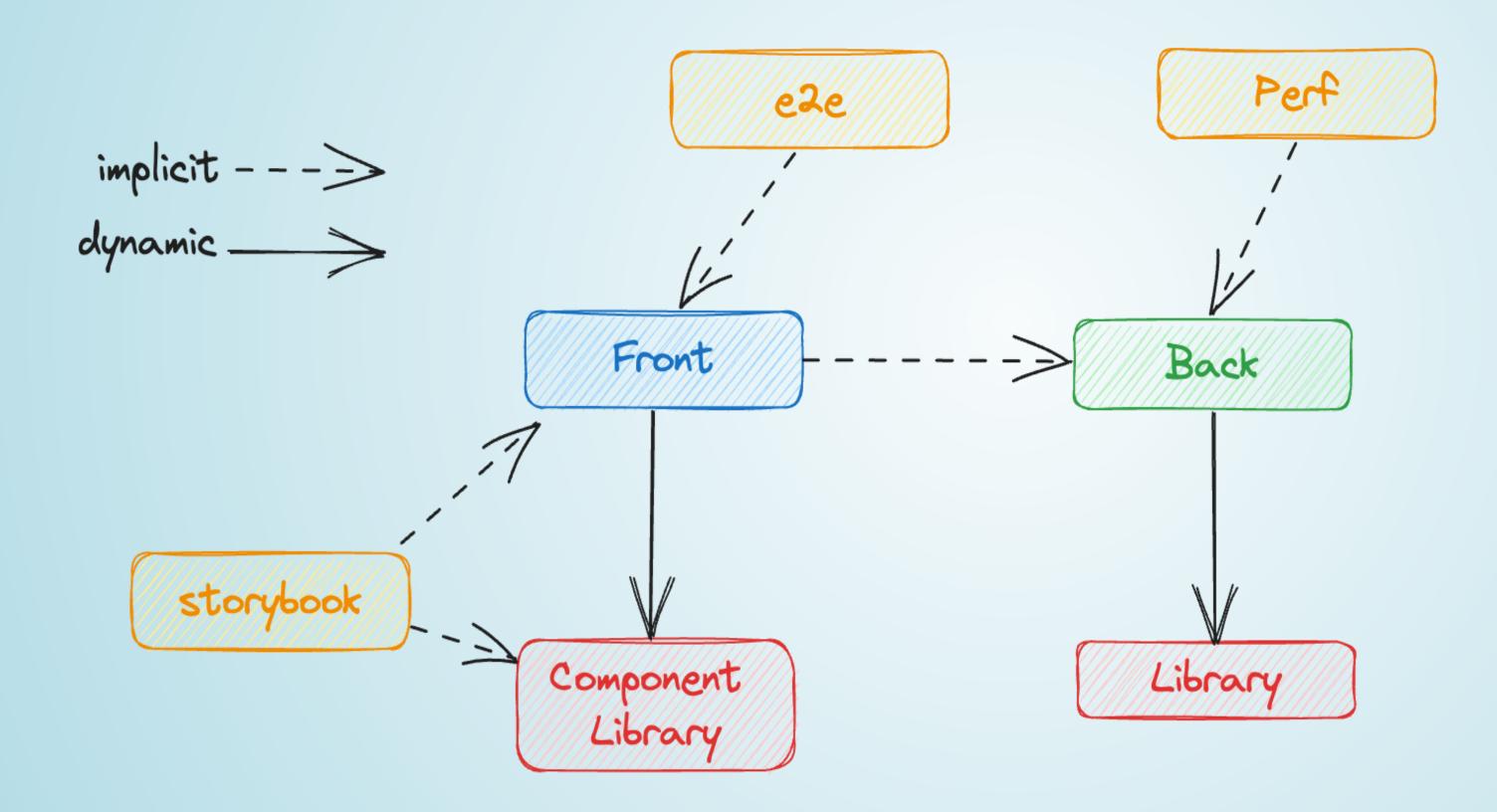


Speaker notes

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

Le graph de dépendance

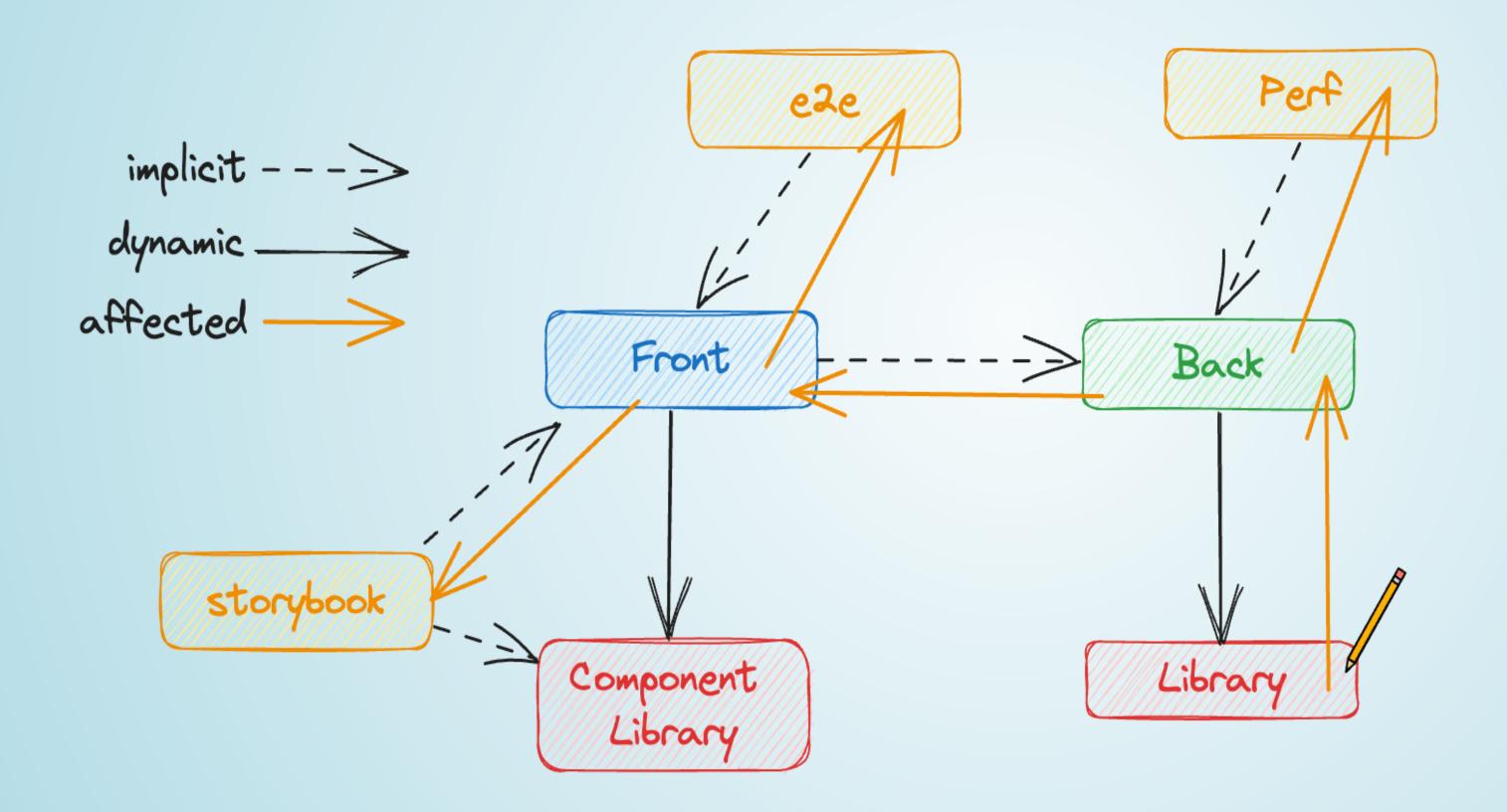




Speaker notes

NX construit un graph entre les applications / les librairies / les tests E2E

Le graph de dépendance - les "affected"





Speaker notes

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

Les languages

















Speaker notes

NX supporte nativement l'écosystème node, et est capable de supporter tout type de langages via des plugins communautaires, ou par implémentation directe.

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 librarie 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'abscence 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 projets affected nx affected --target=test

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

formatter 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

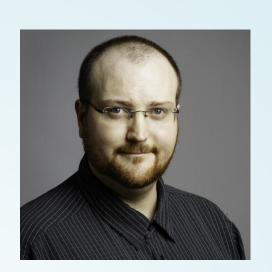
Montrer le cache sur la console

MERCI POUR VOTRE TEMPS!

Présenter par







Anthony QUÉRÉ Jules SPICHT Sylvain LAVAZAIS



Sources:

