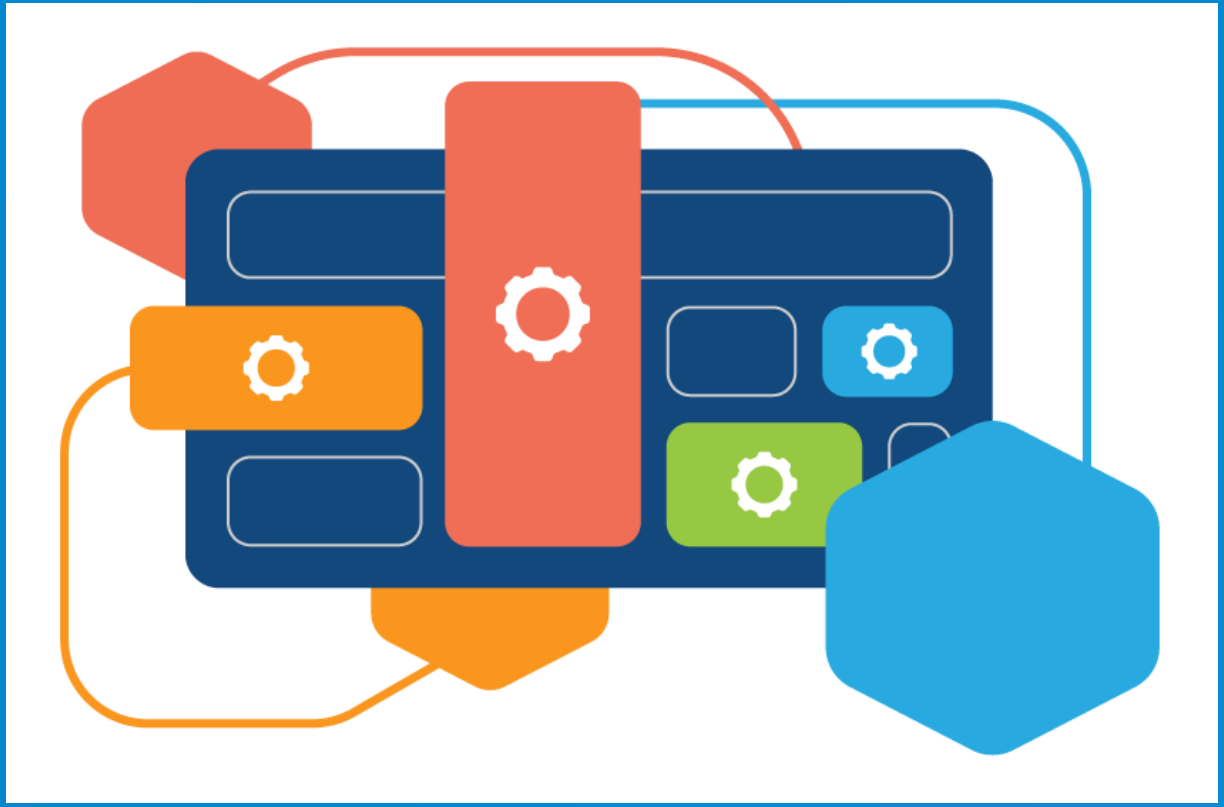
Création d'un Microfrontend avec Module Federation



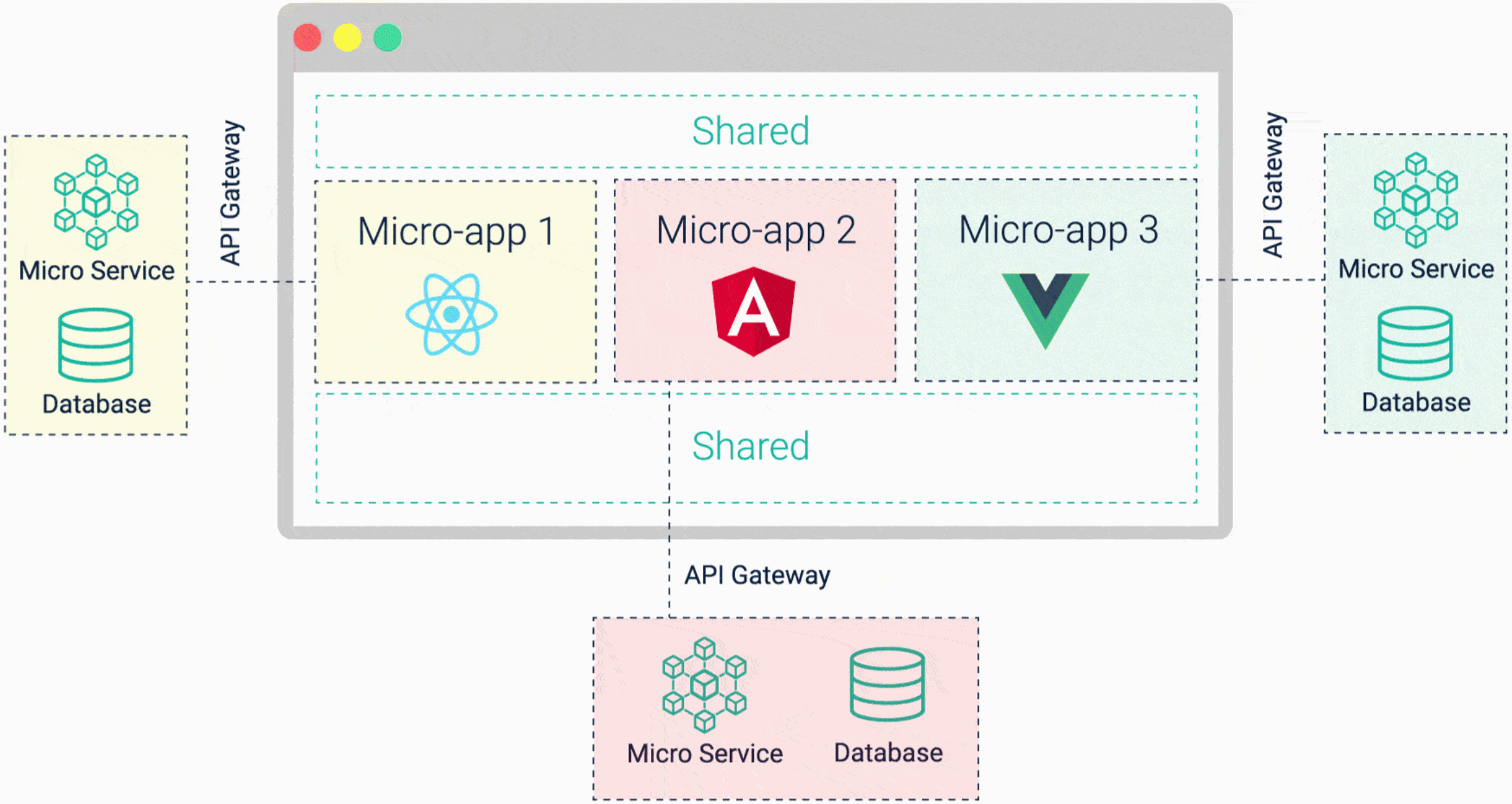
https://ddecrulle.github.io/workshop-module-federation

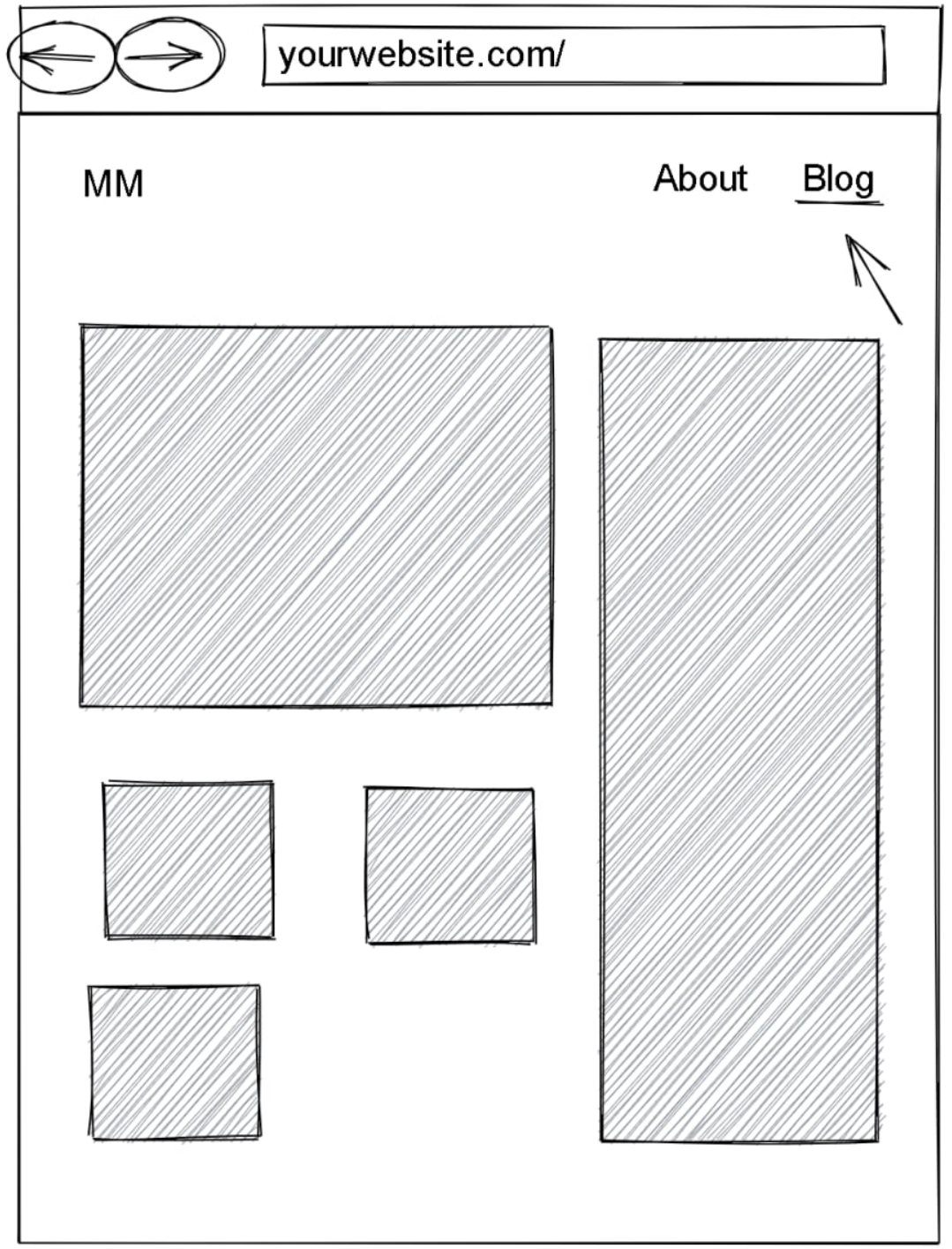
Microfrontend ?

➡️ Microservices

Définition :

> An architectural style where independently deliverable frontend applications are composed into a greater whole.
>
> *Martin Fowler*

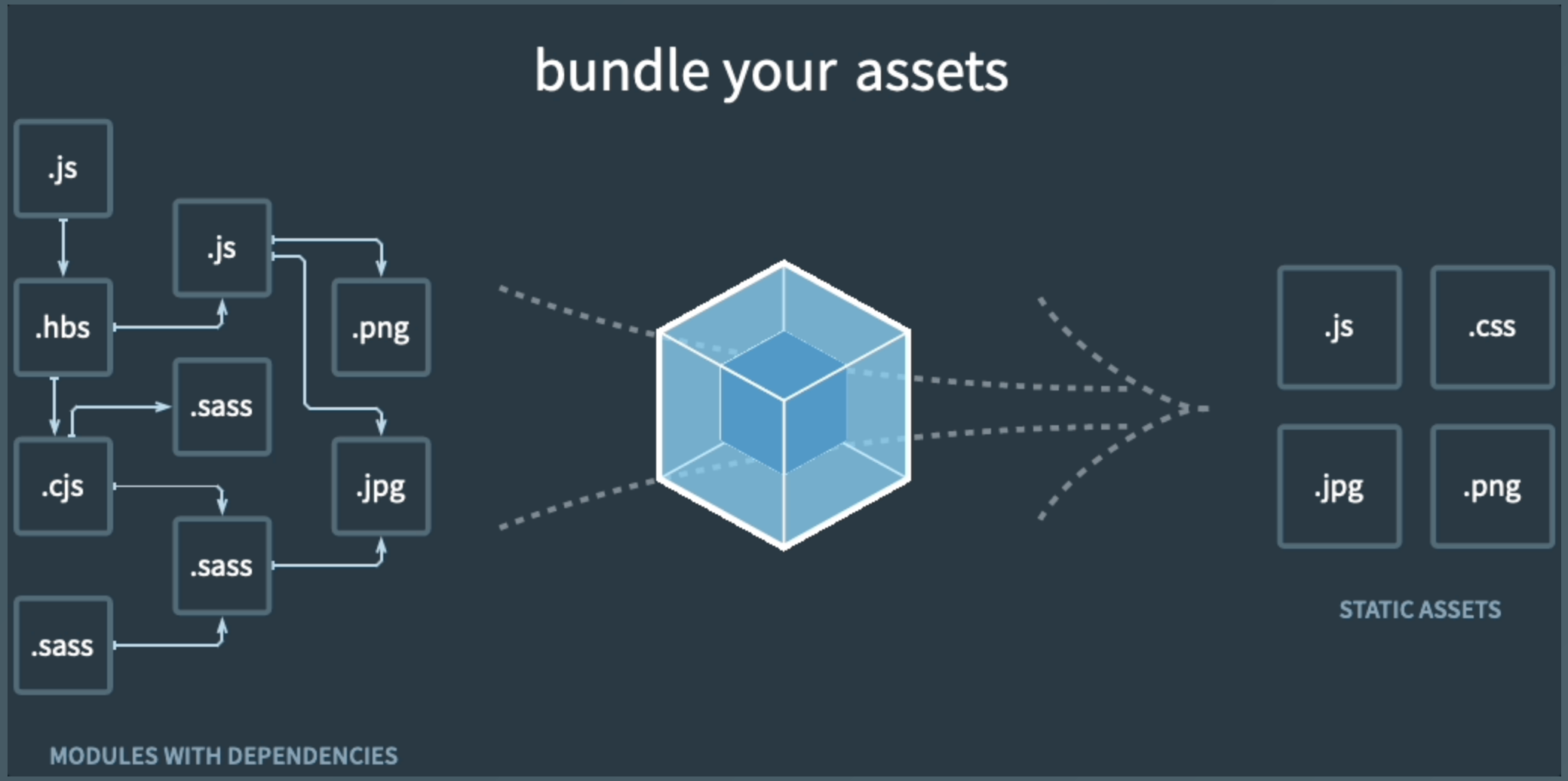

Micro Service
Database
API Gateway
Shared
Micro-app 1
Micro-app 2
Micro-app 3
Shared
API Gateway
API Gateway
Micro Service
Database
Micro Service
Database


yourwebsite.com/
MM
About
Blog

Comment faire ?

- Avec Module Federation
- Avec des Web Components
- Avec des IFrames

Webpack


bundle your assets
.js
.hbs
.cjs
.sass
.sass
.js
.sass
.png
.jpg
MODULES WITH DEPENDENCIES
.js
.css
.jpg
.png
STATIC ASSETS

Module Federation

- Plugin Webpack (Webpack 5)
- Chargement asynchrone de modules distants (pas dans le code de l'application).
  - Le code est chargé dynamiquement à l'exécution, avec les **dépendances** si nécessaires.
- Plus large que le Microfrontend, peut aussi être utilisé côté backend

Comment faire avec React ?

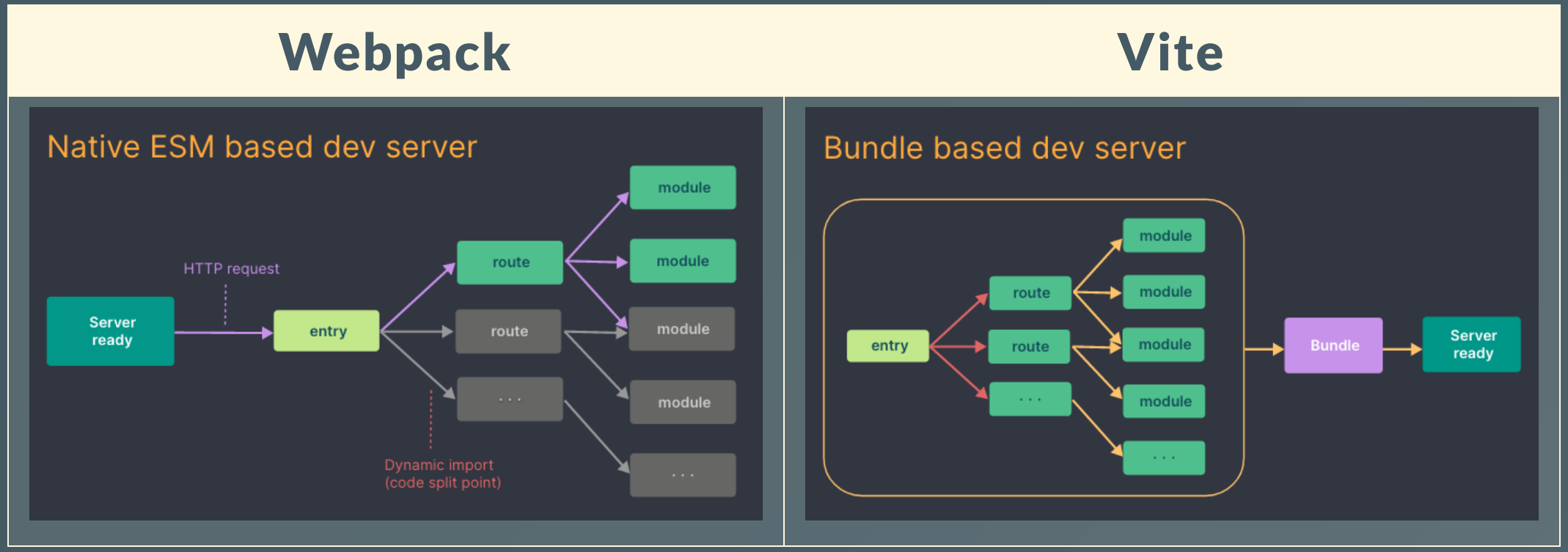
Tout dépend de la façon dont on souhaite construire notre application !

- En utilisant Webpack directement
- Create React App
- Create React App Rewired
- Vite
- Next

La documentation officielle fournie des exemples

Vite

Outil front-end JS pour améliorer la rapidité de développement avec une compilation optimisée pour la production.



On commence ?

Toutes les ressources sont disponibles sur le dépôt github ddecrulle/workshop-module-federation.

Vous pouvez commencer par forker le dépôt.

Le starter est très simple et contient essentiellement de la CI et des outils pour faire du monorepo.

Création des projets

Créons 2 projets vite

```
yarn create vite
  # Project Name : host
  # Framework React/Typescript
```

```
yarn create vite
  # Project Name : remote
  # Framework React/Typescript
```

Lancement des applications

```
npx lerna boostrap #Télécharge les dépendances
yarn dev #Lance les 2 applications
yarn build #Build les 2 applications
```

Fixons les ports de lancement, host : 5000, remote 5001.

```
"scripts": {
   "dev": "vite --port 5000 --strictPort",
    "build": "tsc && vite build",
    "lint": "eslint src --ext ts,tsx --report-unused-disable-directives --max-warnings 0",
    "preview": "vite preview --port 5000 --strictPort"
}
```

Modification du Remote

Je vous propose de customiser le bouton du remote.

Pour ce faire créons un composant Button dans `components/Button.tsx` et importons le dans notre `App.tsx`.

Button.tsx

```
import "./Button.css";

import { useState } from "react";

export const Button = () => {
  const [state, setState] = useState(0);
  return (
    <div>
      <button
        id="click-btn"
        className="shared-btn"
        onClick={() => setState((s) => s + 1)}
      >
        Click me: {state}
      </button>
    </div>
  );
};
```

Button.css

```css
.shared-btn {
  background-color: skyblue;
  border: 1px solid white;
  color: white;
  padding: 15px 30px;
  text-align: center;
  text-decoration: none;
  font-size: 18px;
}
```

Dans App.tsx remplaçons le bouton par celui que nous venons de créer

```
import { Button } from "./components/Button"

...
{/* remplacer */}

<button onClick={() => setCount((count) => count + 1)}>
          count is {count}
</button>
{/* par */}

<Button />
```

Le code jusqu'à cette étape.

Configuration de Module Federation

Le plugin Vite : @originjs/vite-plugin-federation

```
yarn add -D @originjs/vite-plugin-federation -W
```

Ressources

- Micro Frontends par Martin Fowler
- Webpack 5 Module Federation : A game-changer in JavaScript architecture par Zack Jackson (le créateur de Module Federation)
- The History of Microfrontends
- Module Federation
- Vite plugin Federation