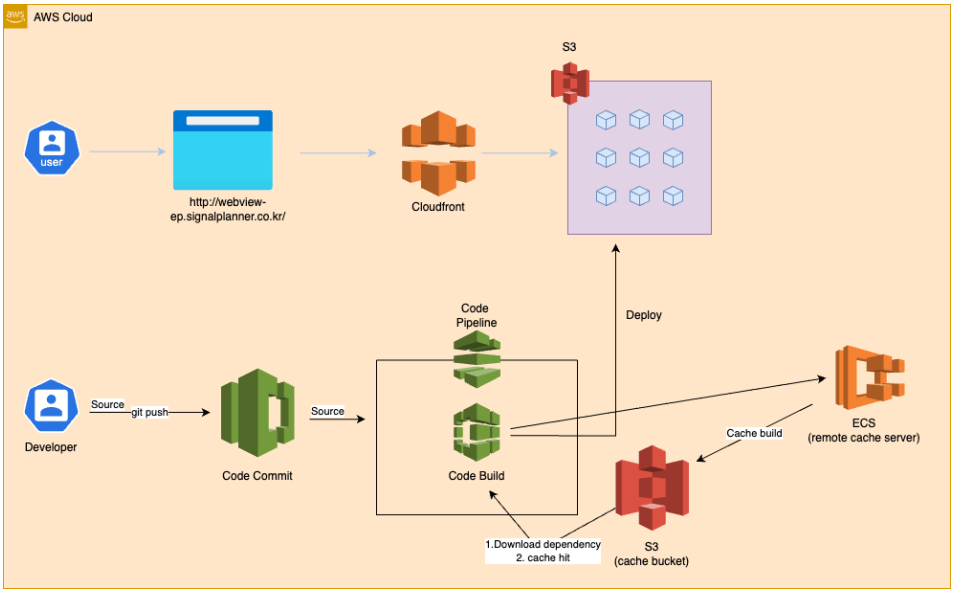
planner-webview-ep

AWS CI/CD 설계도



1. Code Build 단계에서 S3(cache bucket)을 체크해서 기존 dependency들이 저장되어 있으면 install 하지 않고 캐시된 종속성을 재사용함.
   - install 해서 종속성을 가져올 경우

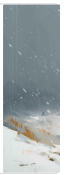
```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Progress: resolved 473, reused 0, downloaded 136, added 137
Progress: resolved 473, reused 0, downloaded 296, added 298
Progress: resolved 473, reused 0, downloaded 380, added 374
Progress: resolved 473, reused 0, downloaded 469, added 471
Progress: resolved 473, reused 0, downloaded 471, added 473, done
```

   - cache된 종속성을 가져올 경우

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Progress: resolved 476, reused 474, downloaded 0, added 464
Progress: resolved 476, reused 474, downloaded 0, added 476, done
```

2. turberepo remote cache server (ECS)
   - `turbo build` 를 사용함으로써 변경된 작업 폴더만 빌드하기 위해 오픈소스 참고해서 ECS 서버 구축

> GitHub - ducktors/turborepo-remote-cache: Open source implementation of the Turborepo custom remote cache server.
> Open source implementation of the Turborepo custom remote cache server. - ducktors/turborepo-remote-cache
> https://github.com/ducktors/turborepo-remote-cache



     - `cache hit` : 빌드 결과물이 있으면 빌드하지 않고 기존 cache된 것을 재사용함.
     - `cache miss` : cache된 결과물이 없으면 새로 빌드함.

- Code Build 단계에서 `turbo build` 실행시 ECS(remote cache server)에서 S3(cache bucket)에 cache된 결과물이 있으면 `cache hit` 이기 때문에 빌드하지 않고 결과물이 없으면 `cache miss` 이기 때문에 빌드함.

```
mydata:build: cache hit, replaying logs f10a68a58f368757
mydata:build:
mydata:build: > mydata@0.0.0 build /codebuild/output/src3248333012/src/apps/mydata
mydata:build: > tsc -b && vite build
mydata:build:
mydata:build: vite v6.0.7 building for production...
mydata:build: transforming...
mydata:build: ✓ 41 modules transformed.
mydata:build: rendering chunks...
mydata:build: computing gzip size...
mydata:build: dist/index.html                                              0.46 kB | gzip:  0.30 kB
mydata:build: dist/assets/style-DLyQebqu.css                               1.12 kB | gzip:  0.57 kB
mydata:build: dist/assets/_commonjsHelpers-B85MJLTf.js                     0.18 kB | gzip:  0.16 kB
mydata:build: dist/assets/__federation_shared_react-CC5G17d-.js            0.27 kB | gzip:  0.20 kB
mydata:build: dist/assets/__federation_shared_react-dom-BQiyK-_7.js        0.28 kB | gzip:  0.20 kB
mydata:build: dist/assets/index-C2x1Zgh_.js                                0.88 kB | gzip:  0.42 kB
mydata:build: dist/assets/remoteEntry.js                                   2.29 kB | gzip:  0.94 kB
mydata:build: dist/assets/__federation_expose_App-D5jPMpms.js              2.41 kB | gzip:  0.97 kB
mydata:build: dist/assets/index-fT0rZ_0F.js                                7.99 kB | gzip:  2.88 kB
mydata:build: dist/assets/__federation_fn_import-BShSNGDN.js              14.15 kB | gzip:  3.12 kB
mydata:build: dist/assets/index-CQoTOKjm.js                               139.66 kB | gzip: 44.46 kB
mydata:build: dist/assets/__federation_shared_react-router-dom-D42XUaPI.js  349.02 kB | gzip: 77.26 kB
mydata:build: ✓ built in 3.91s
insurance:build: cache hit, replaying logs 61058dd3d645b771
insurance:build:
insurance:build: > insurance@0.0.0 build /codebuild/output/src3248333012/src/apps/insurance
insurance:build: > tsc -b && vite build
insurance:build:
insurance:build: vite v6.0.7 building for production...
insurance:build: transforming...
insurance:build: ✓ 41 modules transformed.
insurance:build: rendering chunks...
insurance:build: computing gzip size...
insurance:build: dist/index.html                                              0.46 kB | gzip:  0.30 kB
insurance:build: dist/assets/style-DLyQebqu.css                               1.12 kB | gzip:  0.57 kB
insurance:build: dist/assets/_commonjsHelpers-B85MJLTf.js                     0.18 kB | gzip:  0.16 kB
insurance:build: dist/assets/__federation_shared_react-CC5G17d-.js            0.27 kB | gzip:  0.20 kB
insurance:build: dist/assets/__federation_shared_react-dom-BQiyK-_7.js        0.28 kB | gzip:  0.20 kB
insurance:build: dist/assets/index-DaQxlUS0.js                                0.88 kB | gzip:  0.42 kB
insurance:build: dist/assets/remoteEntry.js                                   2.29 kB | gzip:  0.94 kB
insurance:build: dist/assets/__federation_expose_App-D3JQ5k5o.js              2.41 kB | gzip:  0.98 kB
insurance:build: dist/assets/index-fT0rZ_0F.js                                7.99 kB | gzip:  2.88 kB
insurance:build: dist/assets/__federation_fn_import-BShSNGDN.js              14.15 kB | gzip:  3.12 kB
insurance:build: dist/assets/index-CQoTOKjm.js                               139.66 kB | gzip: 44.46 kB
insurance:build: dist/assets/__federation_shared_react-router-dom-D42XUaPI.js  349.02 kB | gzip: 77.26 kB
insurance:build: ✓ built in 4.12s
shell:build: cache miss, executing 2eacf6e06f5fef7b
health:build: cache miss, executing b6be17e5e496806d
shell:build:
shell:build: > shell@0.0.0 build /codebuild/output/src1887706428/src/apps/_shell
```

3. build 단계가 완료되면 위 build log에서 `cache miss` 된 디렉토리만 추출함.

```
MISS_DIRS=$(echo "$BUILD_LOG" | grep -a "cache miss" | awk -F':' '{print $1}' | uniq)
```

```
Cache missed directories: shell
health
```

4. 추출된 디렉토리만 S3 배포 및 cloudfront invalidation (캐시 무효화) 실행 (deploy.sh 참고)
   - apps 폴더에 새로운 서비스를 구축할 시 deploy.sh에 case 구문 추가해야함.

```
for dir in $MISS_DIRS; do
  case "$dir" in
    "shell")
      sync_to_s3_and_invalidate "apps/_shell" "" "/"
      ;;
    "alarm")
      sync_to_s3_and_invalidate "apps/alarm" "alarm" "/alarm/assets/remoteEntry.js"
      ;;
    "counsel")
      sync_to_s3_and_invalidate "apps/counsel" "counsel" "/counsel/assets/remoteEntry.j
s"
      ;;
    *)
      echo "Unknown directory: $dir"
```

```
      ;;
  esac
done
```

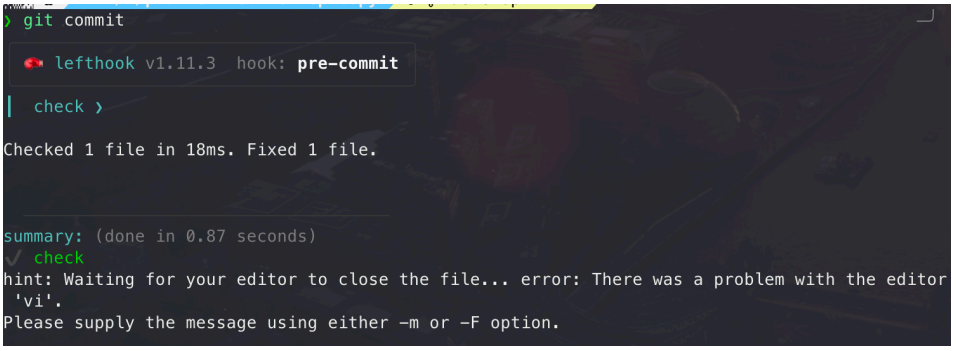
마이크로 app 프로젝트 생성 및 작업에 필요한 명령어들

```
$ pnpm plop
? Enter your app name! example # 생성해야할 app name을 적어줌.
? Enter your port! # 사용할 port, 단 사용중인 포트 확인.

$ pnpm install
$ pnpm --filter [App Name] dev:alone # 각 서비스 작업시 사용 HOC 사용 (O)
$ pnpm dev # module federation을 통한 빌드 결과물을 shell을 통해 확인 HOC 사용 (X)
$ pnpm build:dev # dev 빌드
$ pnpm build:stg # stage 빌드
$ pnpm build # live 빌드

$ pnpm --filter [App Name] add [dependency] # dependency 설치
$ pnpm --filter [App Name] add -D [dependency] # devDependency 설치
$ pnpm --filter [App Name] remove [dependency]
$ pnpm add -W [dependency] # root에 dependency 설치시 -W flag 추가
$ pnpm clean # remove all node_modules
```

Git commit 시 Biome lint 및 formatting fix 자동화 기능



프로젝트 Scaffolding

1. 코드 작성시 각 서비스 단일 책임 원칙을 기본으로 함.
2. 사실 packages는 private registry에서 관리해서 nexux 서버에 배포해서 관리하는게 정석이지만 리소스 부족 및 관리 포인트가 많아지는 문제로 인해 터보레포 내부에서 internal packages로 사용.

▼ apps

▼ _shell

1. 중앙에서 서비스 라우팅 관리
2. 우선 중앙에서 로직 관리는 하지 않음.

alarm → 각 서비스

counsel → 각 서비스

▼ packages

▼ ui

공통 UI 관리 (버튼, layout 등)

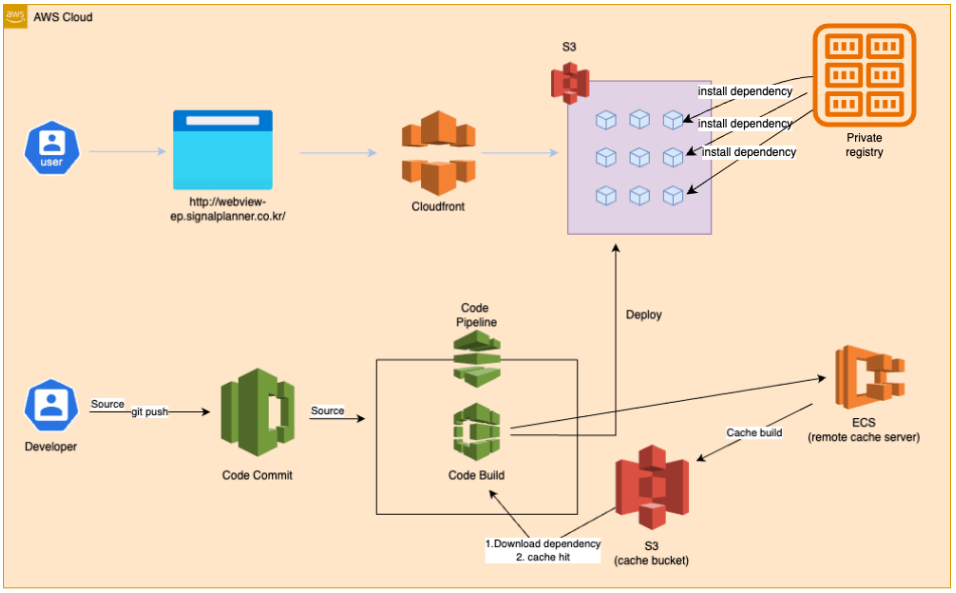
▼ utils

공통 utilize 함수 관리 (axios instance, log event, sentry 초기화, device 체크 등)

▼ api

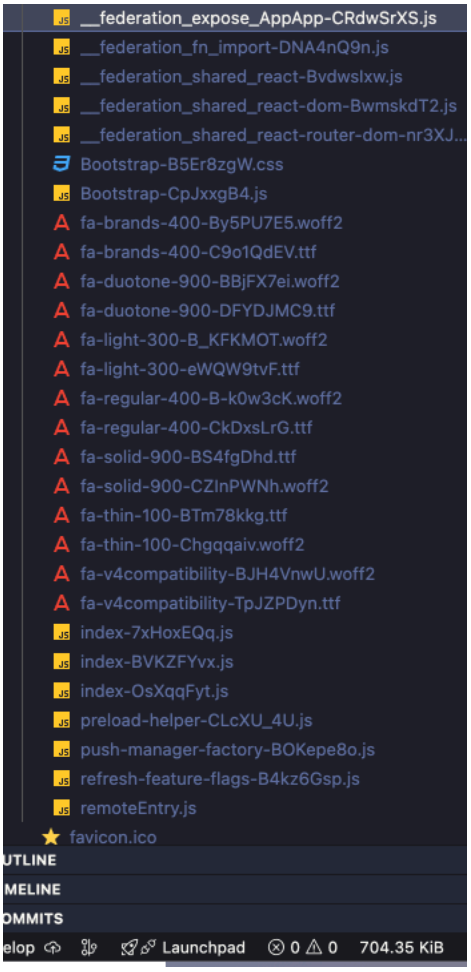
공통 api 관리

가장 이상적인 Microfrontend 청사진



개선해야할 부분

1. module federation expose 시 App 전체를 expose 하기 때문에 일부 빌드 결과물이 500kb 이상 나옴.



👉 해결방안 : 코드 스플리팅 하기 위해 dynamic import 사용해보거나 rollup option 해보기