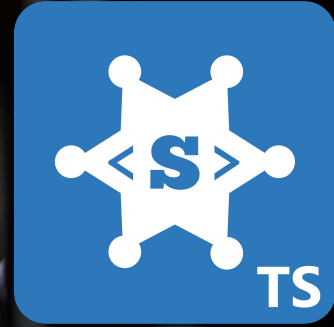


Powerduo Nx and Sheriff



Rainer Hahnekamp - 14. February 2024



About Me...



- Rainer Hahnekamp

ANGULARarchitects.io

- Developer / Trainer / Speaker



<https://www.youtube.com/@RainerHahnekamp>



<https://www.ng-news.com>



<https://github.com/softarc-consulting/sheriff>

Professional



Professional
Angular
Testing



Modern Spring
for Angular



@RainerHahnekamp

Agenda

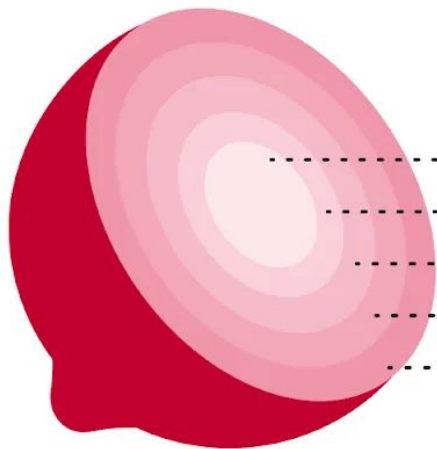
1. What is Nx?
2. Nx Quick Feature Run
3. Nx and Modules
4. Sheriff
5. Sheriff & Nx
6. Current Challenges: Barrel File / Tree Shaking / Code Splitting



Enterprise Angular Monorepo Patterns

Book v0.1 | December 1, 2018





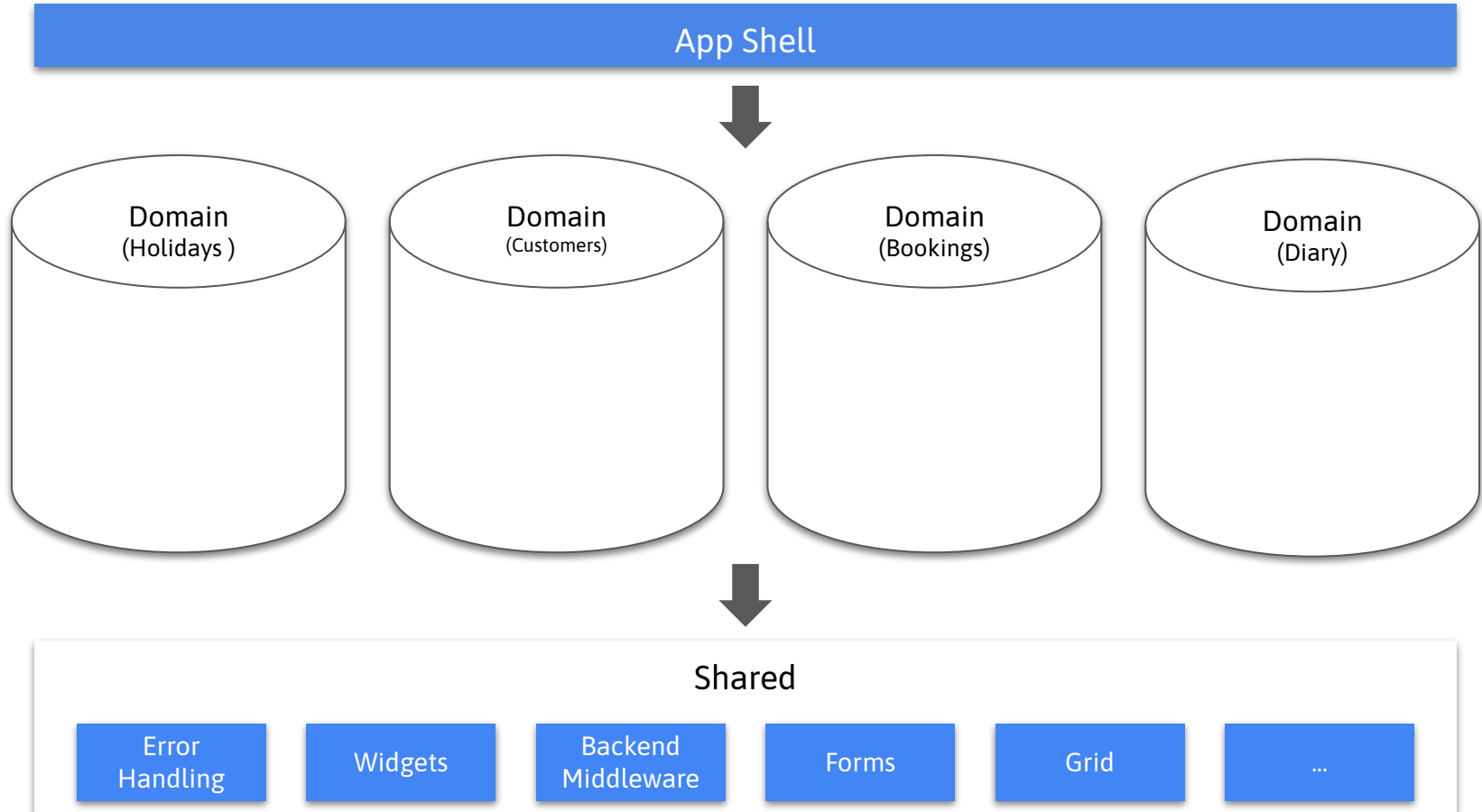
- 1 The core capabilities of the framework
- 2 Core APIs
- 3 APIs for building applications
- 4 Basic CLI capabilities
- 5 CLI Plugins



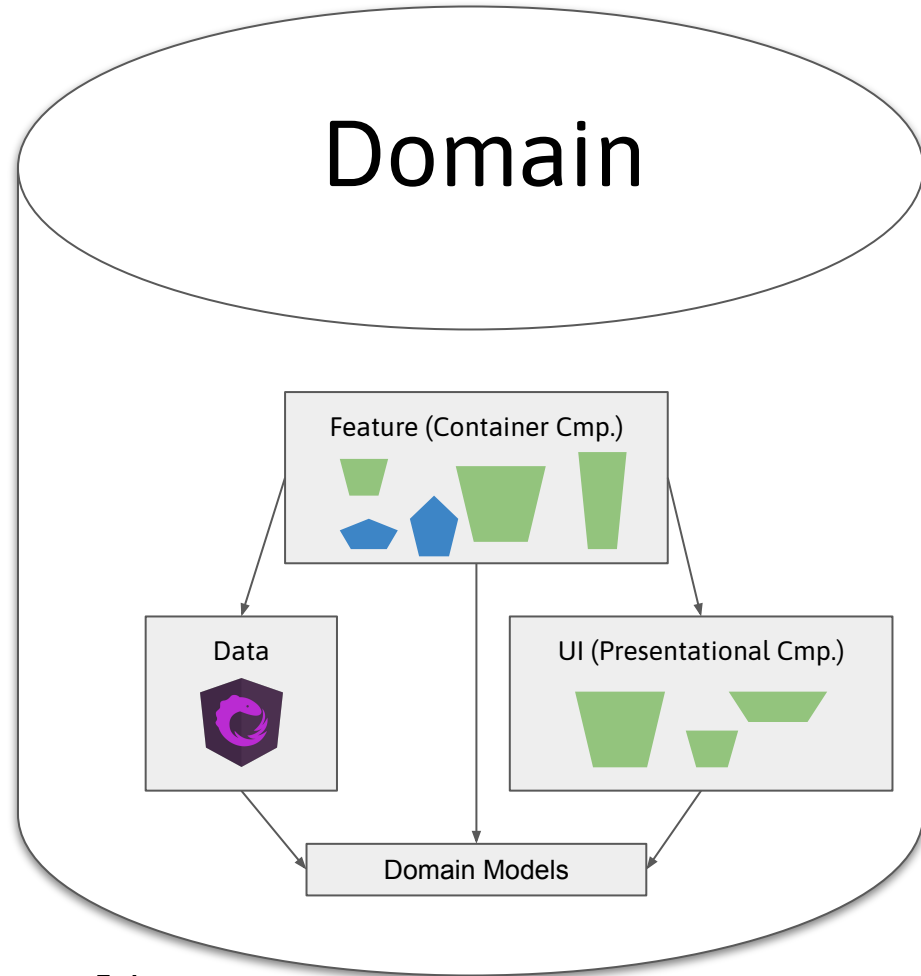
Nx Highlights

1. Tooling
2. Caching & Affected
3. Cloud
4. Visualization
5. Library Support

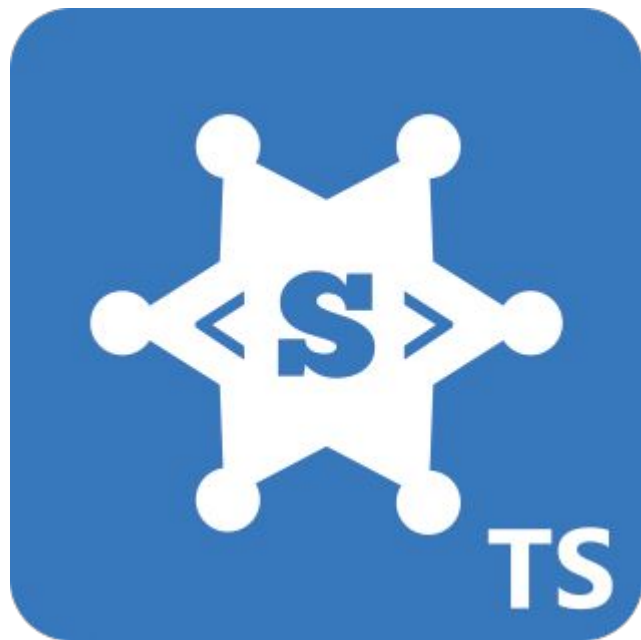
Layer 1 - Domain modules



Layer 2 Sub Modules



Sheriff: Modularity in TypeScript



- **Module Encapsulation**

Project ▾

- src
 - app
 - core
 - holidays
 - data
 - holiday-facade.service.ts
 - internal-service.ts
 - feature
 - holiday.component.ts
 - security
 - shared
 - app.component.html
 - app.component.scss
 - app.component.ts
 - app.routes.ts

```
1 import { Component, inject } from '@angular/core';
2 import { InternalService } from '../data/internal-service';
3
4 @Component({
5   selector: 'app-holiday',
6   template: ``,
7   standalone: true,
8 })
9 export class HolidayComponent {
10   data: boolean = inject(InternalService).load();
11 }
12
```

Project ▾

src

app

core

holidays

data

holiday-facade.service.ts

index.ts

internal-service.ts

feature

holiday.component.ts

security

shared


app.component.html

app.component.scss

app.component.ts

1 export { HolidayFacade } from './holiday-facade.service';

2

 RainerHahnekamp

Project ▾

src

app

core

holidays

data

holiday-facade.service.ts

index.ts

internal-service.ts

feature

holiday.component.ts

security

shared

app.component.html

app.component.scss

app.component.ts

1 import { Component, inject } from '@angular/core';

2 import { InternalService } from '../data/internal-service';

3

4

5 selector: 'app-holiday',

6 template: `

7 standalone: true,

8 })


9 export class HolidayComponent {

10 data: boolean = inject(InternalService).load();

11 }

12

ESLint: Deep import is not allowed. Use the module's index.ts or path.(@softarc/sheriff/deep-import)

 RainerHahnekamp

Project ▾

src

app

core

holidays

data

TS holiday-facade.service.ts

TS index.ts

TS internal-service.ts

feature

TS holiday.component.ts

security

shared

app.component.html

SASS app.component.scss

TS app.component.ts

1import { Component, inject } from '@angular/core';

2import { HolidayFacade } from '../data';

3

4@Component({

5 selector: 'app-holiday',

6 template: `

7 standalone: true,

8 })


9 export class HolidayComponent {

10 data: boolean = inject(HolidayFacade).data;

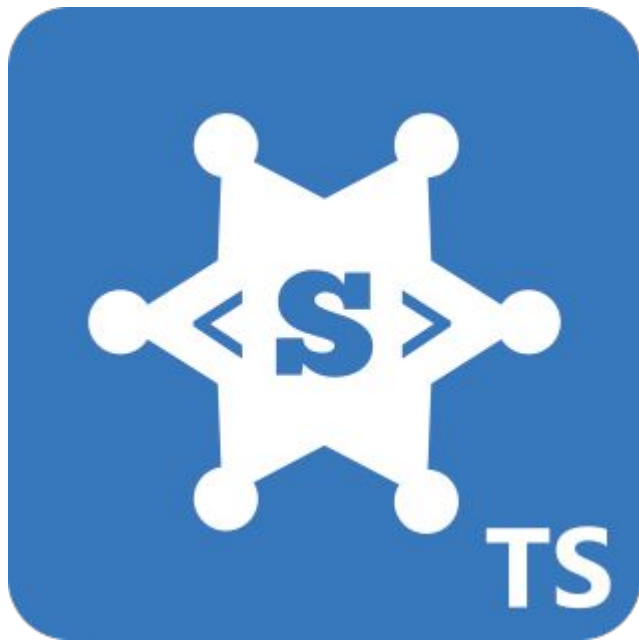
11 }

12

HolidayComponent

 RainerHahnekamp

Sheriff: Modularity in TypeScript



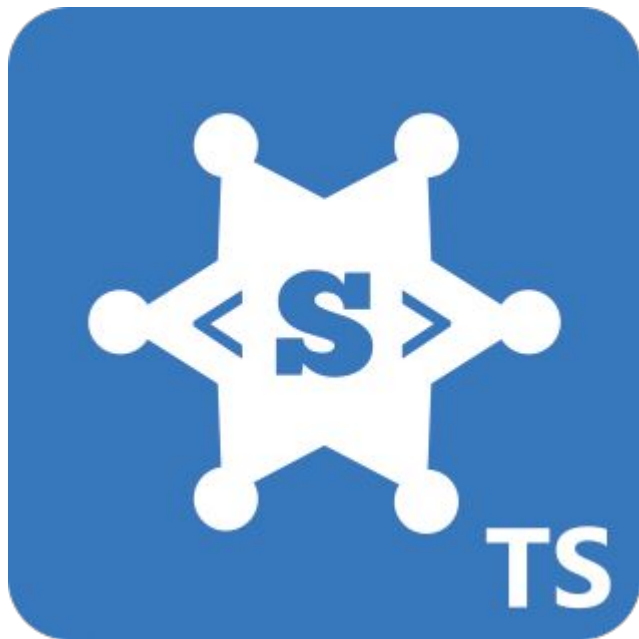
- Module Encapsulation
- **Dependency Rules**


```
1  import { SheriffConfig } from '@softarc/sheriff-core';
2
3  export const sheriffConfig: SheriffConfig = {
4    version: 1,
5    tagging: {
6      'src/app/holidays/data': ['domain:holidays', 'type:data'],
7      'src/app/holidays/feature': ['domain:holidays', 'type:feature'],
8      'src/app/holidays/ui': ['domain:holidays', 'type:ui'],
9    },
10   depRules: {},
11 };
12
```

Run

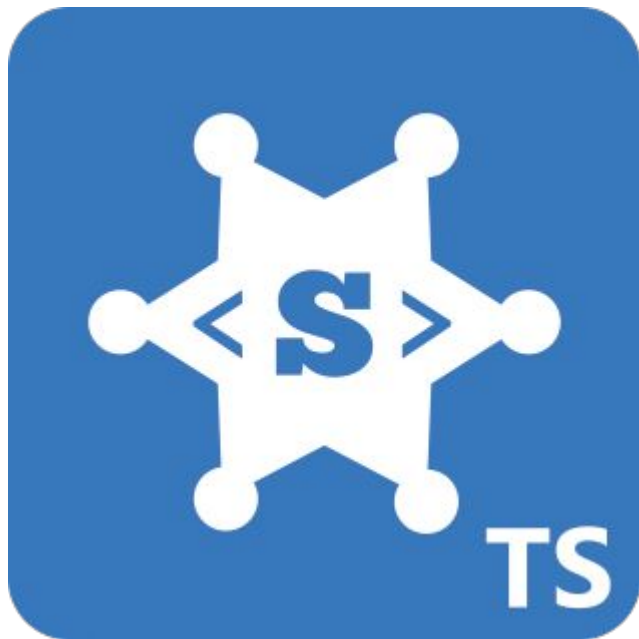
```
1 import {SheriffConfig} from "@softarc/sheriff-core";
2
3 export const sheriffConfig: SheriffConfig = {
4   version: 1,
5   tagging: {
6     'src/app/holidays/feature': ['domain:holidays', 'type:feature'],
7     'src/app/holidays/data': ['domain:holidays', 'type:data']
8   }, depRules: {
9     'domain:holidays': 'domain:holidays',
10    'type:feature': 'type:data',
11    'type:data': []
12  }
13 }
14
```

Sheriff: Modularity in TypeScript



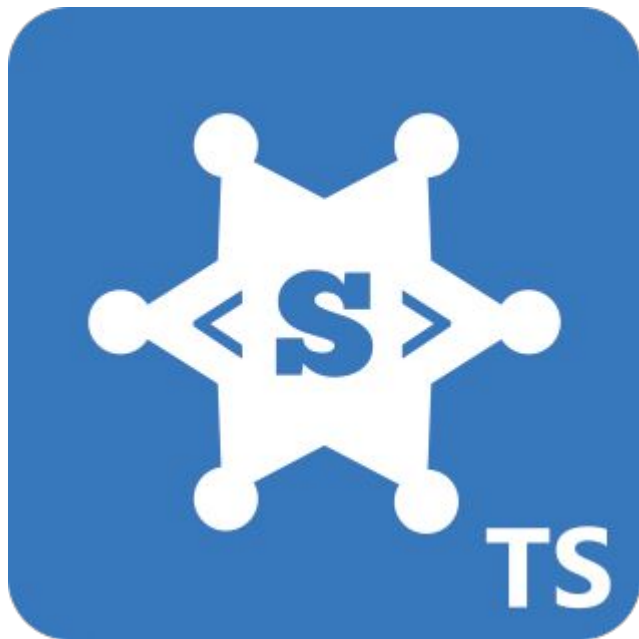
- Module Encapsulation
- Dependency Rules
- **Lightweight**

Sheriff: Modularity in TypeScript



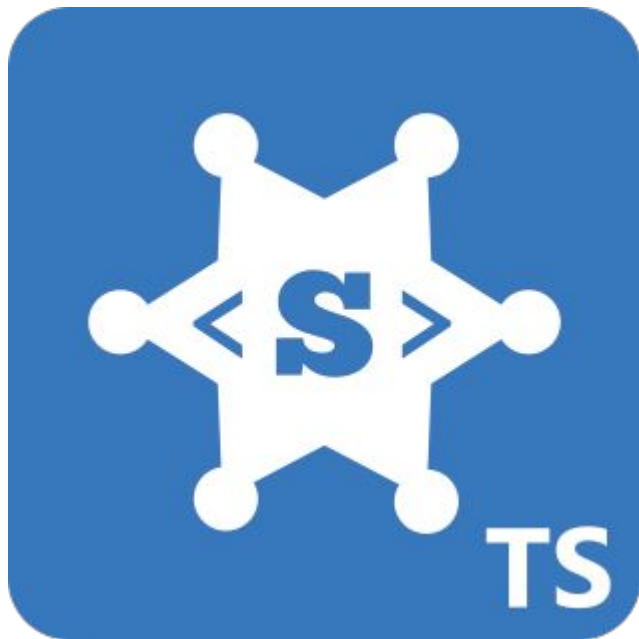
- Module Encapsulation
- Dependency Rules
- Lightweight
- **Convention over Configuration**

Sheriff: Modularity in TypeScript



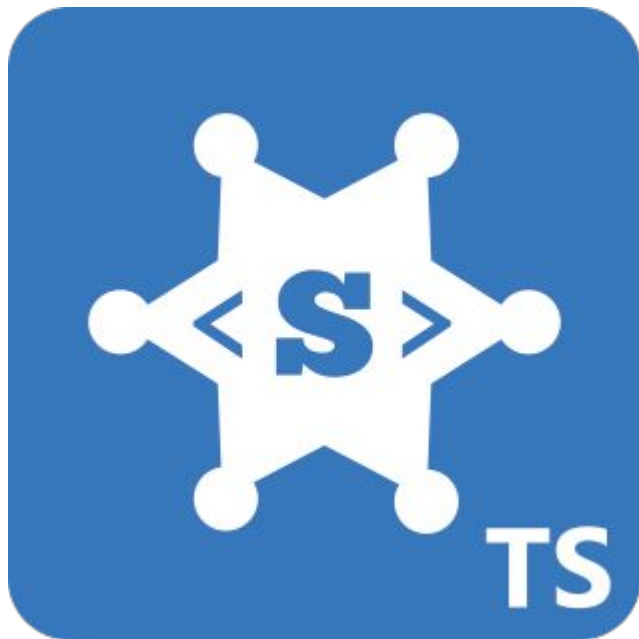
- Module Encapsulation
- Dependency Rules
- Lightweight
- Convention over Configuration
- **Zero Dependencies**

Sheriff: Modularity in TypeScript



- Module Encapsulation
- Dependency Rules
- Lightweight
- Convention over Configuration
- Zero Dependencies
- **For all TypeScript Projects**

Sheriff: Modularity in TypeScript



- Module Encapsulation
- Dependency Rules
- Lightweight
- Convention over Configuration
- Zero Dependencies
- For all TypeScript Projects
- **(Heavily) Influenced by Nx**

Nx Pros & Cons

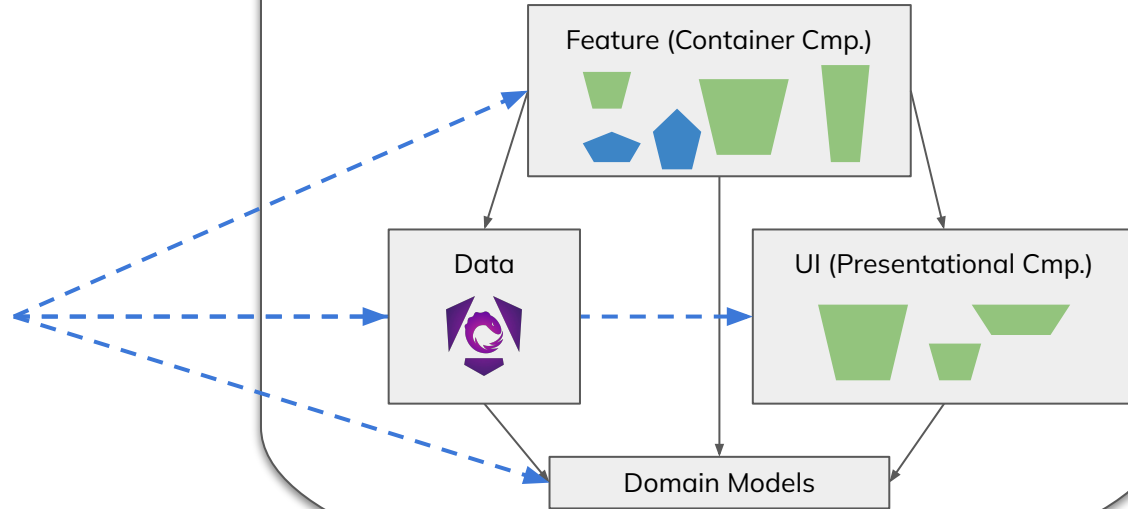
- ✓ Provides a set of **exclusive Features**
- ✓ **Must-Have** for certain application size

- ✗ Diverging from the Angular CLI
- ✗ **Small Modules unfit as Libraries**
- ✗ Additional Layer
- ✗ Not that stable as Angular CLI

Nx Lib

Domain

Sheriff
Module



Application Evolution

1. Angular CLI or NX Standalone with Sheriff
2. Nx with Domain Libs (and Sheriff)
3. Nx with build-optimized Libs (and Sheriff)

The Problem with the Barrel File...

Thanks