ТИНЬКОФФ

Микрофронтенды в Тинькофф Обслуживании

Откуда? Зачем? Как с этим жить?

Герман Панов

Герман Панов

- Около 3-х лет во фронтенде
- Полтора года разрабатывал микрофронтенды в бэк-офисе
- Сейчас разработчик Taiga UI Kit
- Нравится JavaScript (слегка)

О чем пойдет речь

- Исторический экскурс в Тинькофф Обслуживание
- Что такое микрофронтенд. Типы микрофронтендов. Жизненный цикл
- Core-часть микрофронтендов
- Процесс разработки, тестирования и публикации микрофронтендов
- Работа микрофронтендов в runtime
- Архитектура уровня приложения
- Когда пригодятся микрофронтенды
- Резюме

Экскурс

Тинькофф Обслуживание

Платформа, в которой работают операторы поддержки клиентов

2020

Монолитные приложения

- Много общего функционала и велосипедов
- Публикация библиотек из монорепозитория
- Идея платформизации
- Webpack 5

2021

Platform on Iframes

- Ядро: задания, звонки, чаты
- Микро-приложения как iframes
- area Codes и библиотека rx-post-message
- Начали смотреть в сторону виджетов

ТИНЬКОФФ

Микрофронтенды

«An architectural style where independently deliverable frontend applications are composed into a greater whole»

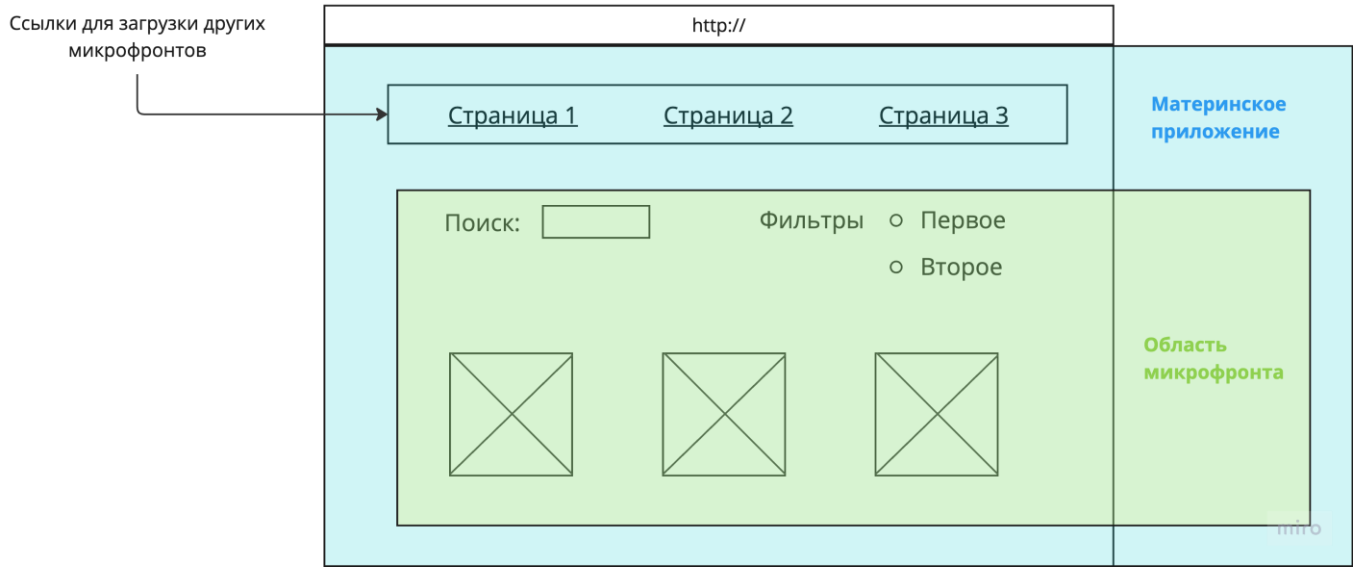
Cam Jackson - Micro Frontends

Подходы

Стандартный подход

Навигация по микрофронтам при помощи URL

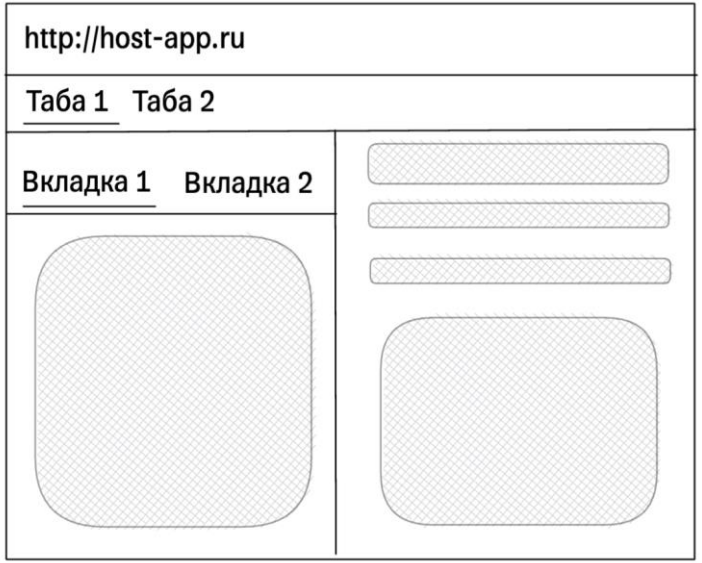
Чаще всего на одной странице один микрофронт



Микровиджетный подход

Микрофронтенды не привязаны к URL

На одной странице может быть N микрофронтов



2022-2023

Platform on MFE

- Микровиджеты на Module Federation
- Интерфейс – конструктор
- Динамичный layout
- Монорепозиторий микрофронтендов

2024

Platfrom on MFE

- ~ 150 микровиджетов
- Каталог микровиджетов
- ~12 репозиториев
- 1-20 микровиджетов на релиз
- Частота релизов ограничена здравым смыслом

Как работать с микрофронтендами?

Жизненный цикл

1	Разработать	Собрать микрофронт в js файл
2	Опубликовать	Выложить этот файл по http/https
3	Встроить	Научить приложение его загружать и встраивать в нужном месте
4	Profit	

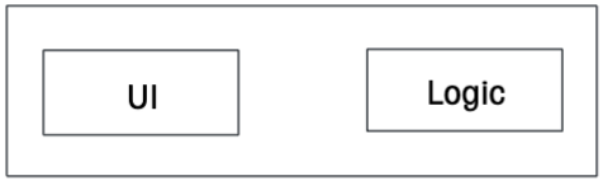
Типы микрофронтендов

По принципу работы

По архитектурному подходу

По принципу работы

Виджет



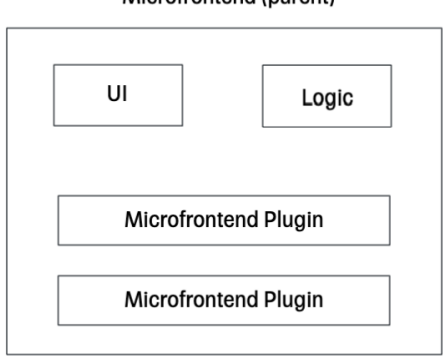
Дестроится при переключении внешнего UI

Сервис

Живет в фоне

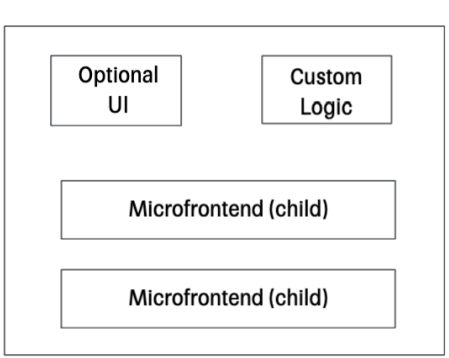
По архитектурному подходу

Плагин



Встраивается в МФ

Адаптер



Оборачивает МФ

Типы микрофронтендов

Виджет

Плагин

Сервис

Адаптер

ТИНЬКОФФ

Core-часть микрофронтендов

Схема работы

01

Загружаем контейнер
(ES module) с микрофронтом

02

Инициализируем
контейнер

03

Получаем нужный модуль,
инстанцируем компонент с
микрофронтендом, используя
Angular API

04

Рендерим компонент
в предназначенном месте

Как запускать?

Песочница (Angular)

app.module.ts

```ts
export const moduleMetadata: NgModule = {
  declarations: [AppComponent, MicrofrontendsHostComponent],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    AppRoutingModule,
    HttpClientModule,
    MicrofrontendModule.forRoot(),
    MicrofrontendSandboxModule.forRoot({
      microfrontendUrl: () => MICROFRONTEND_URL || '',
      microfrontendSandboxKnobsUrl:
        'https://host.ru/microfrontend-sandbox-knobs.js',
    }),
  ],
  providers: [latestVersionProviders(LatestVersion.Unstable)],
}

@NgModule({
  declarations: moduleMetadata.declarations,
  imports: moduleMetadata.imports,
  providers: moduleMetadata.providers,
  exports: [AppComponent],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

Песочница (Angular)

app.module.ts

```ts
export const moduleMetadata: NgModule = {
  declarations: [AppComponent, MicrofrontendsHostComponent],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    AppRoutingModule,
    HttpClientModule,
    MicrofrontendModule.forRoot(),
    MicrofrontendSandboxModule.forRoot({
      microfrontendUrl: () => MICROFRONTEND_URL || '',
      microfrontendSandboxKnobsUrl:
        'https://host.ru/microfrontend-sandbox-knobs.js',
    }),
  ],
  providers: [latestVersionProviders(LatestVersion.Unstable)],
}

@NgModule({
  declarations: moduleMetadata.declarations,
  imports: moduleMetadata.imports,
  providers: moduleMetadata.providers,
  exports: [AppComponent],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

Песочница (React)

App.ts

```
const microfrontendConfig = {
  microfrontendUrl: MICROFRONTEND_URL || '',
  microfrontendSandboxKnobsUrl:
    'https://host.ru/microfrontend-sandbox-knobs.js',
}

const App = () => {
  return (
    <MicrofrontendContextProvider config={microfrontendConfig}>
      <Router>
        <AppComponent />
        <MicrofrontendsHostComponent />
      </Router>
    </MicrofrontendContextProvider>
  )
}

const container = document.getElementById('root')
const root = createRoot(container)
root.render(<App />)
```

Exposed-модуль

mf-some.module.html

```
1  const EXPORTS = [MfSomeComponent];
2
3  @NgModule({
4      imports: [CommonModule, SomeModule],
5      declarations: [...EXPORTS],
6      exports: [...EXPORTS],
7  })
8  export class MfSomeModule implements MfModule<MfSomeComponent> {
9      // Prevents from components being tree-shaked in production
10     // https://github.com/module-federation/module-federation-examples/pull/1229/files
11     public getEntryPoint(): Type<MfSomeComponent> {
12         return MfSomeComponent;
13     }
14 }
```

Exposed-модуль

mf-some.module.html

```
const EXPORTS = [MfSomeComponent];

@NgModule({
    imports: [CommonModule, SomeModule],
    declarations: [...EXPORTS],
    exports: [...EXPORTS],
})
export class MfSomeModule implements MfModule<MfSomeComponent> {
    // Prevents from components being tree-shaked in production
    // https://github.com/module-federation/module-federation-examples/pull/1229/files
    public getEntryPoint(): Type<MfSomeComponent> {
        return MfSomeComponent;
    }
}
```

Exposed-компонент

mf-some.component.html

```
<some
    *ngIf="validatedMfInput$ | async as viewModel"
    [config]="mfInitialConfig"
    [someEntity]="viewModel"
    (someEvent)="onOutputEvent($event, eventNames.someEvent)"
    (otherEvent)="onOutputEvent($event, eventNames.otherEvent)"
></some>
```

Deep dive

Webpack

Microfrontend webpack config

default-mf-config-builder.ts

```ts
import { Configuration, WebpackOptionsNormalized, container } from 'webpack';

type Shared = ConstructorParameters<typeof container.ModuleFederationPlugin>[0]['shared'];

export const defaultMfConfigBuilder = (options: {
    remoteName: string,
    uniqueName: string,
    modulePath: string,
    shared: Shared,
    distFolder?: string,
    isNg14?: boolean,
}): Configuration => { ... }
```

Microfrontend webpack config

default-mf-config-builder.ts

```
13  // Will be used to determine window.__mf_containers__[remoteName] and exposed module name
14  const filename = 'index.js'
15
16  const plugins = [
17    new container.ModuleFederationPlugin({
18      name: options.remoteName,
19      library: options.isNg14
20        ? { type: 'module' }
21        : {
22            type: 'jsonp',
23            // Container is being passed to function REGISTER_MF_CONTAINER(loaded container)
24            name: REGISTER_MF_CONTAINER,
25          },
26      filename,
27      exposes: {
28        main: options.modulePath,
29      },
30      shared: options.shared,
31    }),
32  ];
33
34  ...
```

Microfrontend webpack config

default-mf-config-builder.ts

```
13  // Will be used to determine window.__mf_containers__[remoteName] and exposed module name
14  const filename = 'index.js'
15
16  const plugins = [
17    new container.ModuleFederationPlugin({
18      name: options.remoteName,
19      library: options.isNg14
20        ? { type: 'module' }
21        : {
22            type: 'jsonp',
23            // Container is being passed to function REGISTER_MF_CONTAINER(loaded container)
24            name: REGISTER_MF_CONTAINER,
25          },
26      filename,
27      exposes: {
28        main: options.modulePath,
29      },
30      shared: options.shared,
31    }),
32  ];
33
34  ...
```

Microfrontend webpack config

default-mf-config-builder.ts

```
13  // Will be used to determine window.__mf_containers__[remoteName] and exposed module name
14  const filename = 'index.js'
15
16  const plugins = [
17    new container.ModuleFederationPlugin({
18      name: options.remoteName,
19      library: options.isNg14
20        ? { type: 'module' }
21        : {
22            type: 'jsonp',
23            // Container is being passed to function REGISTER_MF_CONTAINER(loaded container)
24            name: REGISTER_MF_CONTAINER,
25          },
26      filename,
27      exposes: {
28        main: options.modulePath,
29      },
30      shared: options.shared,
31    }),
32  ];
33
34  ...
```

Microfrontend webpack config

default-mf-config-builder.ts

```
13  // Will be used to determine window.__mf_containers__[remoteName] and exposed module name
14  const filename = 'index.js'
15
16  const plugins = [
17    new container.ModuleFederationPlugin({
18      name: options.remoteName,
19      library: options.isNg14
20        ? { type: 'module' }
21        : {
22            type: 'jsonp',
23            // Container is being passed to function REGISTER_MF_CONTAINER(loaded container)
24            name: REGISTER_MF_CONTAINER,
25          },
26      filename,
27      exposes: {
28        main: options.modulePath,
29      },
30      shared: options.shared,
31    }),
32  ];
33
34  ...
```

Microfrontend webpack config

default-mf-config-builder.ts

```
13  // Will be used to determine window.__mf_containers__[remoteName] and exposed module name
14  const filename = 'index.js'
15
16  const plugins = [
17    new container.ModuleFederationPlugin({
18      name: options.remoteName,
19      library: options.isNg14
20        ? { type: 'module' }
21        : {
22            type: 'jsonp',
23            // Container is being passed to function REGISTER_MF_CONTAINER(loaded container)
24            name: REGISTER_MF_CONTAINER,
25          },
26      filename,
27      exposes: {
28        main: options.modulePath,
29      },
30      shared: options.shared,
31    }),
32  ];
33
34  ...
```

Microfrontend webpack config

default-mf-config-builder.ts

```
13  // Will be used to determine window.__mf_containers__[remoteName] and exposed module name
14  const filename = 'index.js'
15
16  const plugins = [
17    new container.ModuleFederationPlugin({
18      name: options.remoteName,
19      library: options.isNg14
20        ? { type: 'module' }
21        : {
22            type: 'jsonp',
23            // Container is being passed to function REGISTER_MF_CONTAINER(loaded container)
24            name: REGISTER_MF_CONTAINER,
25          },
26      filename,
27      exposes: {
28        main: options.modulePath,
29      },
30      shared: options.shared,
31    }),
32  ];
33
34  ...
```

Microfrontend webpack config

default-mf-config-builder.ts

```
65   ...
66
67   const config: Configuration = {
68     output: {
69       publicPath: 'auto',
70       uniqueName: options.uniqueName,
71       // Explicitly setting unique value for loaded chunks global namespace
72       chunkLoadingGlobal: options.uniqueName,
73     },
74     optimization: {
75       runtimeChunk: false,
76     },
77     plugins,
78   }
79
80   return config;
81 }
```

Microfrontend webpack config

default-mf-config-builder.ts

```
65   ...
66
67   const config: Configuration = {
68     output: {
69       publicPath: 'auto',
70       uniqueName: options.uniqueName,
71       // Explicitly setting unique value for loaded chunks global namespace
72       chunkLoadingGlobal: options.uniqueName,
73     },
74     optimization: {
75       runtimeChunk: false,
76     },
77     plugins,
78   }
79
80   return config;
81 }
```

При подключении

webpack.config.js

```
1 const { defaultMfConfigBuilder } = require('@twork-mf/webpack');
2
3 module.exports = defaultMfConfigBuilder({
4     remoteName: moduleName,
5     uniqueName: superUniqGenerator(),
6     modulePath: `apps/best-mfe/src/exposed/best-mfe/best-mfe.module.ts`,
7     shared: configureSharedDeps(),
8     distFolder: 'dist/microfrontends/best-mfe/0.0.0',
9 });
```

Core

Интерфейс Microfrontend

microfrontend.ts

```ts
export interface Microfrontend<T = undefined> {
    /**
     * URL of the script containing federated modules
     */
    remoteEntry: RemoteEntry;

    /**
     * Custom initial configuration
     */
    initialConfig?: T;

    /**
     * Parent injector of the microfrontend
     */
    parentInjector?: Injector;

    /**
     * Plugins for the microfrontend
     */
    plugins?: MfPlugin[];
}
```

Microfrontend component

microfrontend.component.html

```
1 <ng-template #federatedComponentContainer></ng-template>
```

Microfrontend component

microfrontend.component.ts

```ts
@Component({
    selector: 'microfrontend',
    templateUrl: './microfrontend.component.html',
    changeDetection: ChangeDetectionStrategy.OnPush,
})
export class MicrofrontendComponent<I, O, C = undefined> implements OnDestroy {
    @Input()
    set mfConfig(microfrontend: Microfrontend<C>) {
        this._mfConfig = microfrontend;
        this.onMicrofrontend(microfrontend);
    }

    @Input()
    set mfInput(input: I) {
        this.input$.next(input);
    }

    @Output() mfOutput = new EventEmitter<O>();
    @Output() mfError = new EventEmitter<Error>();

    @ViewChild('federatedComponentContainer', { read: ViewContainerRef })
    public federatedComponentContainerRef!: ViewContainerRef;
```

Microfrontend component

microfrontend.component.ts

```
58 @Component({
59     selector: 'microfrontend',
60     templateUrl: './microfrontend.component.html',
61     changeDetection: ChangeDetectionStrategy.OnPush,
62 })
63 export class MicrofrontendComponent<I, O, C = undefined> implements OnDestroy {
64     @Input()
65     set mfConfig(microfrontend: Microfrontend<C>) {
66         this._mfConfig = microfrontend;
67         this.onMicrofrontend(microfrontend);
68     }
69
70     @Input()
71     set mfInput(input: I) {
72         this.input$.next(input);
73     }
74
75     @Output() mfOutput = new EventEmitter<O>();
76     @Output() mfError = new EventEmitter<Error>();
77
78     @ViewChild('federatedComponentContainer', { read: ViewContainerRef })
79     public federatedComponentContainerRef!: ViewContainerRef;
```

Microfrontend component

microfrontend.component.ts

```
106     private onMicrofrontend(microfrontend: Microfrontend<C>): void {
107         this.destroyMicrofrontend();
108
109         if (!microfrontend) {
110             return;
111         }
112
113         const {
114             remoteEntry,
115             parentInjector,
116             initialConfig,
117             plugins,
118         } = microfrontend;
119
120         const pluginsLoad$ = plugins
121             ? this.getPluginResolver$(remoteEntry)
122                 .pipe(
123                     mergeMap(resolvedRemoteEntry => resolvedRemoteEntry
124                         ? this.pluginsService.handlePlugins$(resolvedRemoteEntry.url, plugins)
125                         : of(void 0)
126                     )
127                 )
128             : of(void 0);
129
130         const loadStream$ = this.getMicrofrontendStream$(remoteEntry).pipe(
131             concatMap(exposedModule => pluginsLoad$.pipe(
132                 map(plugins => ({ exposedModule, plugins }))
133             )),
134             shareReplay({ refCount: true, bufferSize: 1 })
135         );
```

Microfrontend component

microfrontend.component.ts

```
106     private onMicrofrontend(microfrontend: Microfrontend<C>): void {
107         this.destroyMicrofrontend();
108
109         if (!microfrontend) {
110             return;
111         }
112
113         const {
114             remoteEntry,
115             parentInjector,
116             initialConfig,
117             plugins,
118         } = microfrontend;
119
120         const pluginsLoad$ = plugins
121             ? this.getPluginResolver$(remoteEntry)
122                 .pipe(
123                     mergeMap(resolvedRemoteEntry => resolvedRemoteEntry
124                         ? this.pluginsService.handlePlugins$(resolvedRemoteEntry.url, plugins)
125                         : of(void 0)
126                     )
127                 )
128             : of(void 0);
129
130         const loadStream$ = this.getMicrofrontendStream$(remoteEntry).pipe(
131             concatMap(exposedModule => pluginsLoad$.pipe(
132                 map(plugins => ({ exposedModule, plugins }))
133             )),
134             shareReplay({ refCount: true, bufferSize: 1 })
135         );
```

Microfrontend component

microfrontend.component.ts

```ts
106    private onMicrofrontend(microfrontend: Microfrontend<C>): void {
107        this.destroyMicrofrontend();
108
109        if (!microfrontend) {
110            return;
111        }
112
113        const {
114            remoteEntry,
115            parentInjector,
116            initialConfig,
117            plugins,
118        } = microfrontend;
119
120        const pluginsLoad$ = plugins
121            ? this.getPluginResolver$(remoteEntry)
122                .pipe(
123                    mergeMap(resolvedRemoteEntry => resolvedRemoteEntry
124                        ? this.pluginsService.handlePlugins$(resolvedRemoteEntry.url, plugins)
125                        : of(void 0)
126                    )
127                )
128            : of(void 0);
129
130        const loadStream$ = this.getMicrofrontendStream$(remoteEntry).pipe(
131            concatMap(exposedModule => pluginsLoad$.pipe(
132                map(plugins => ({ exposedModule, plugins }))
133            )),
134            shareReplay({ refCount: true, bufferSize: 1 })
135        );
```

Microfrontend component

microfrontend.component.ts

```
106     private onMicrofrontend(microfrontend: Microfrontend<C>): void {
107         this.destroyMicrofrontend();
108
109         if (!microfrontend) {
110             return;
111         }
112
113         const {
114             remoteEntry,
115             parentInjector,
116             initialConfig,
117             plugins,
118         } = microfrontend;
119
120         const pluginsLoad$ = plugins
121             ? this.getPluginResolver$(remoteEntry)
122                 .pipe(
123                     mergeMap(resolvedRemoteEntry => resolvedRemoteEntry
124                         ? this.pluginsService.handlePlugins$(resolvedRemoteEntry.url, plugins)
125                         : of(void 0)
126                     )
127                 )
128             : of(void 0);
129
130         const loadStream$ = this.getMicrofrontendStream$(remoteEntry).pipe(
131             concatMap(exposedModule => pluginsLoad$.pipe(
132                 map(plugins => ({ exposedModule, plugins }))
133             )),
134             shareReplay({ refCount: true, bufferSize: 1 })
135         );
```

Microfrontend component

microfrontend.component.ts

```
137         loadStream$.subscribe();
138
139         loadStream$.pipe(
140             mergeMap(({ exposedModule, plugins }) => {
141                 const lifeCycleMapEvents = getMfeLifeCycleEventMap(remoteEntry);
142
143                 if (isLazyEntry(remoteEntry)) {
144                     return this.initComponentWithLazyModule$({
145                         mfModule: exposedModule as Type<MfModule<MfComponent<I, 0, C>>>,
146                         parentInjector,
147                         remoteEntry: '',
148                         initialConfig,
149                         plugins,
150                         lifeCycleMapEvents
151                     });
152                 }
153
154                 return this.initComponentWithUrl$({
155                     exposedModule: exposedModule as Record<string, Type<MfModule<MfComponent<I, 0, C>>>>,
156                     parentInjector,
157                     remoteEntry,
158                     initialConfig,
159                     plugins,
160                     lifeCycleMapEvents
161                 });
162             }),
163             takeUntil(this.componentDestroy$)
164         )
165             .subscribe();
166     }
```

Microfrontend component

microfrontend.component.ts

```
137         loadStream$.subscribe();
138
139         loadStream$.pipe(
140             mergeMap(({ exposedModule, plugins }) => {
141                 const lifeCycleMapEvents = getMfeLifeCycleEventMap(remoteEntry);
142
143                 if (isLazyEntry(remoteEntry)) {
144                     return this.initComponentWithLazyModule$({
145                         mfModule: exposedModule as Type<MfModule<MfComponent<I, O, C>>>,
146                         parentInjector,
147                         remoteEntry: '',
148                         initialConfig,
149                         plugins,
150                         lifeCycleMapEvents
151                     });
152                 }
153
154                 return this.initComponentWithUrl$({
155                     exposedModule: exposedModule as Record<string, Type<MfModule<MfComponent<I, O, C>>>>,
156                     parentInjector,
157                     remoteEntry,
158                     initialConfig,
159                     plugins,
160                     lifeCycleMapEvents
161                 });
162             }),
163             takeUntil(this.componentDestroy$)
164         )
165             .subscribe();
166     }
```

Microfrontend component

microfrontend.component.ts

```
137         loadStream$.subscribe();
138
139         loadStream$.pipe(
140             mergeMap(({ exposedModule, plugins }) => {
141                 const lifeCycleMapEvents = getMfeLifeCycleEventMap(remoteEntry);
142
143                 if (isLazyEntry(remoteEntry)) {
144                     return this.initComponentWithLazyModule$({
145                         mfModule: exposedModule as Type<MfModule<MfComponent<I, O, C>>>,
146                         parentInjector,
147                         remoteEntry: '',
148                         initialConfig,
149                         plugins,
150                         lifeCycleMapEvents
151                     });
152                 }
153
154                 return this.initComponentWithUrl$({
155                     exposedModule: exposedModule as Record<string, Type<MfModule<MfComponent<I, O, C>>>>,
156                     parentInjector,
157                     remoteEntry,
158                     initialConfig,
159                     plugins,
160                     lifeCycleMapEvents
161                 });
162             }),
163             takeUntil(this.componentDestroy$)
164         )
165             .subscribe();
166     }
```

Microfrontend component

microfrontend.component.ts

```
226     private initComponent$({
227         mfModule,
228         parentInjector,
229         remoteEntry,
230         initialConfig,
231         plugins,
232         lifeCycleMapEvents,
233     }: InitComponentArgs<I, O, C>): Observable<void> {
234         if (this._mfConfig) {
235             this.microfrontendEventsService.pushEvent(lifeCycleMapEvents.start(this._mfConfig));
236         }
237
238         const modifiedModule = this.modifyMfModuleWithPluginProvider(mfModule, plugins);
239
240         return from(this.compiler.compileModuleAsync<MfModule<MfComponent<I, O, C>>>(modifiedModule)).pipe(
241             tap(moduleFactory => {
242                 const {
243                     url: remoteEntryResolvedUrl,
244                     // Empty string set when using lazyModule remoteEntry
245                 } = remoteEntry ? this.remoteModuleService.getResolvedRemoteEntry(remoteEntry)! : { url: '' };
246
247                 this.storedR3Injector = Injector.create({
248                     providers: plugins
249                         ? [{ provide: MICROFRONTEND_PLUGINS, useValue: plugins }]
250                         : [],
251                     parent: parentInjector ?? this.componentInjector,
252                 }) as NgR3Injector
```

Microfrontend component

microfrontend.component.ts

```
254                 this.storedModuleRef = moduleFactory.create(this.storedR3Injector);
255
256                 let componentType: Type<MfComponent<I, O, C>>;
257
258                 if (typeof this.storedModuleRef.instance.getEntryPoint === 'function') {
259                     componentType = this.storedModuleRef.instance.getEntryPoint();
260                 } else {
261                     // @deprecated, will be removed alongside with ng12 support.
262                     componentType = (modifiedModule as ɵNgModule<I, O, C>).exports[0];
263                     this.microfrontendEventsService.pushEvent(
264                         new MfeExposedNgModuleDeprecatedEntryPoint(remoteEntry, remoteEntryResolvedUrl),
265                     );
266                 }
267
268                 const componentFactory = this.cfr.resolveComponentFactory(
269                     componentType,
270                 );
271
272                 if (!componentType) {
273                     this.microfrontendEventsService.pushEvent(new MfeExposedNgComponentNotFound(
274                         remoteEntry,
275                         remoteEntryResolvedUrl,
276                     ));
277                     this.destroyInjectors();
278
279                     return;
280                 }
```

Microfrontend component

microfrontend.component.ts

```
282                 const { instance } = this.federatedComponentContainerRef.createComponent<MfComponent<I, O, C>>(
283                     componentFactory,
284                     undefined,
285                     // Parent injector is not passed here but provided through containing module injector.
286                     // Doing so to skip NodeInjector chain so child lazy modules could store their own
    providers.
287                     undefined,
288                     [],
289                     this.storedModuleRef,
290                 );
291
292                 if (initialConfig) {
293                     instance.mfInitialConfig = initialConfig;
294                 }
295
296                 this.connectStreams(instance);
297
298                 this.microfrontendEventsService.pushEvent(lifeCycleMapEvents.end(remoteEntryResolvedUrl));
299
300                 this.cdr.detectChanges();
301             }),
302             map(() => void 0)
303         )
304     }
```

Microfrontend component

microfrontend.component.ts

```
306     private connectStreams(instance: MfComponent<I, O, C>): void {
307         if (typeof instance.mfInput$?.next === 'function') {
308             this.input$
309                 .pipe(takeUntil(this.microfrontendDestroy$))
310                 .subscribe(input => {
311                     instance.mfInput$?.next(input);
312                     this.cdr.detectChanges();
313                 });
314         }
315
316         if (typeof instance.mfOutput$?.subscribe === 'function') {
317             instance.mfOutput$
318                 .pipe(takeUntil(this.microfrontendDestroy$))
319                 .subscribe(output => this.mfOutput.emit(output));
320         }
321
322         if (typeof instance.mfError$?.subscribe === 'function') {
323             instance.mfError$
324                 .pipe(takeUntil(this.microfrontendDestroy$))
325                 .subscribe(error => this.mfError.emit(error));
326         }
327
328         if (typeof instance.mfOnInit === 'function') {
329             instance.mfOnInit();
330         }
331     }
```

Как подключается в приложении?

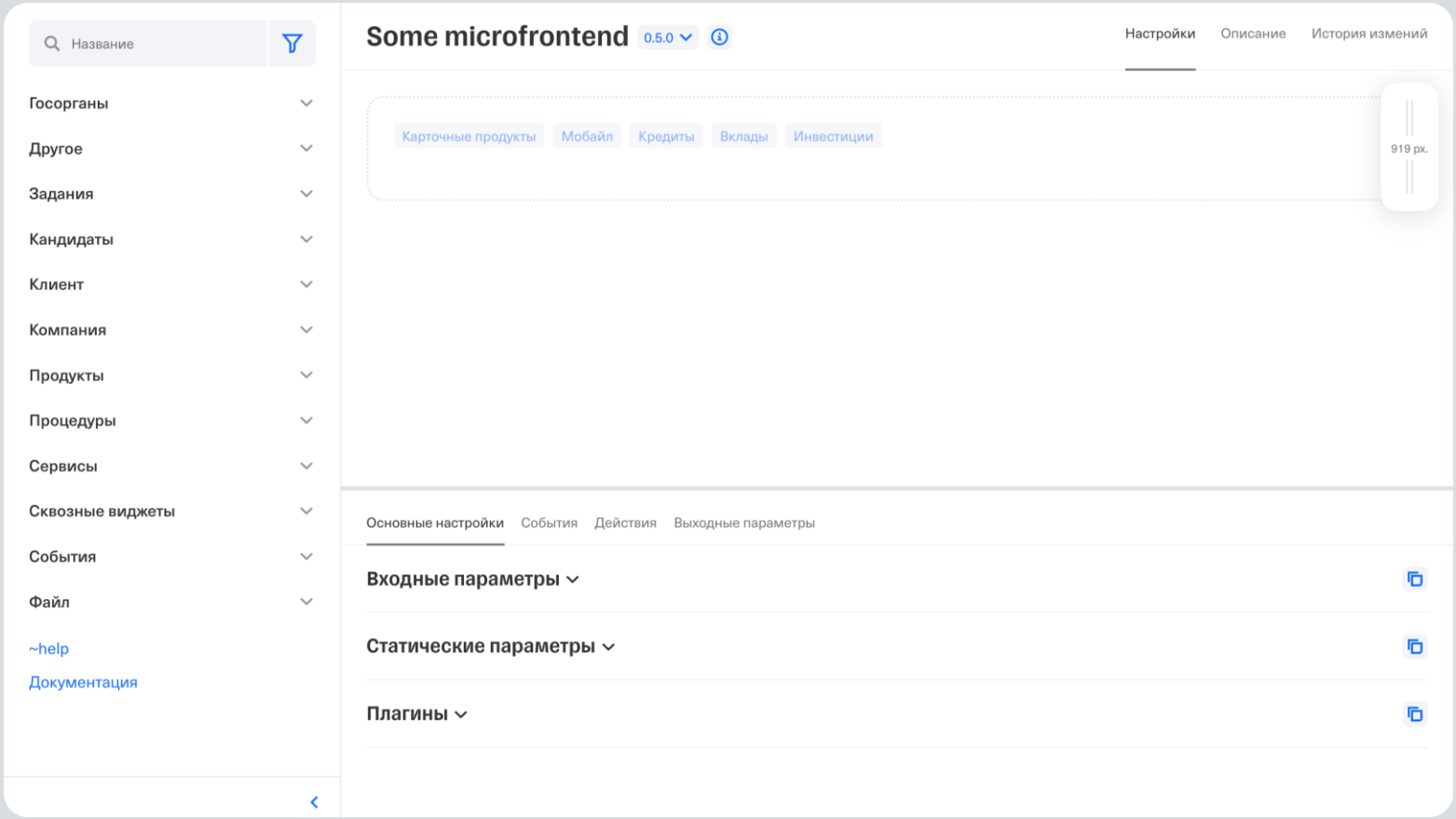
Подключение в приложении

microfrontend-wrapper.component.html

```
<microfrontend
    *ngIf="mfConfig$ | async as config"
    [mfConfig]="config"
    [mfInput]="input"
    (mfOutput)="handleOutput($event)"
    (mfError)="handleError($event)">
</microfrontend>
```

Давайте создадим микрофронт?

Может есть на витрине?



Витрина микрофронтендов

Прозрачно

Все доступные микрофронтенды в одном месте

Функционально

Можно тестировать сценарии работы микрофронтендов. Есть версионирование и описание

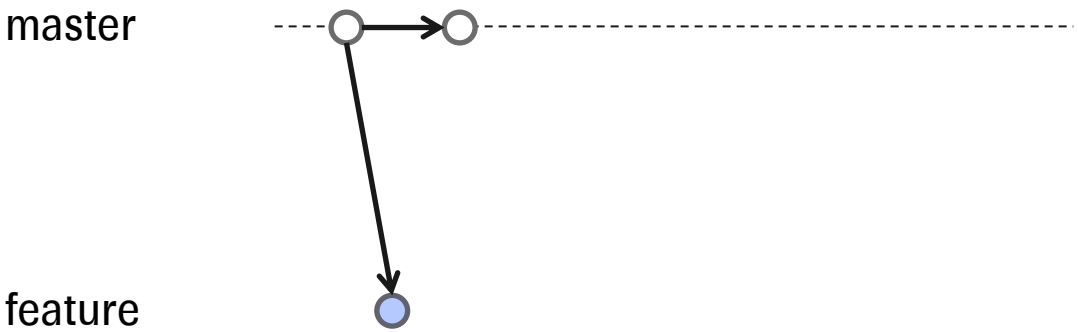
Конфигурируемо

Декларативная YML конфигурация позволяет легко добавлять новые микрофронтенды

Нет нужного микрофронтенда?

Разработка

Разработка



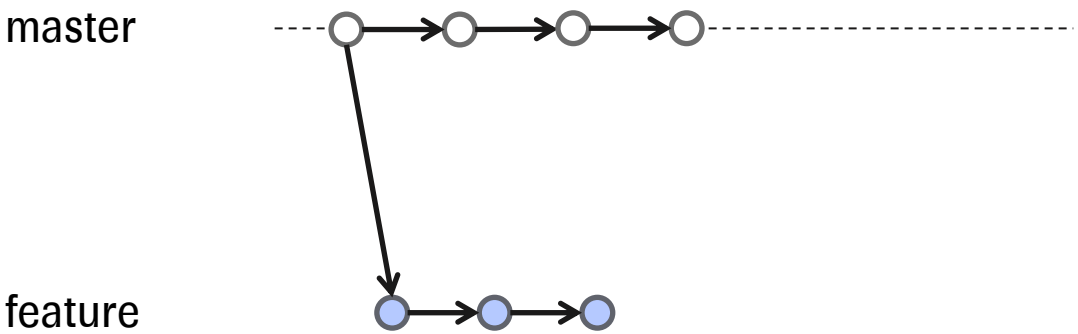
Кодогенерация

```
✗ nx g @twork-mf/schematics:package test-mf    // обычный микрофронтенд

✔ Should add authorization to sandbox? (y/N) · true
✔ Should add fake ngrx component store to sandbox? (y/N) · true

CREATE apps/test-mf/.browserslistrc
CREATE apps/test-mf/.eslintrc.json
CREATE apps/test-mf/README.md
CREATE apps/test-mf/configure-shared-deps.js
CREATE apps/test-mf/jest.config.js
CREATE apps/test-mf/package.json
CREATE apps/test-mf/project.json
CREATE apps/test-mf/tsconfig.app.json
CREATE apps/test-mf/tsconfig.editor.json
CREATE apps/test-mf/tsconfig.json
CREATE apps/test-mf/tsconfig.spec.json
CREATE apps/test-mf/webpack.config.js
CREATE apps/test-mf/metadata/actions.metadata.ts
...
```

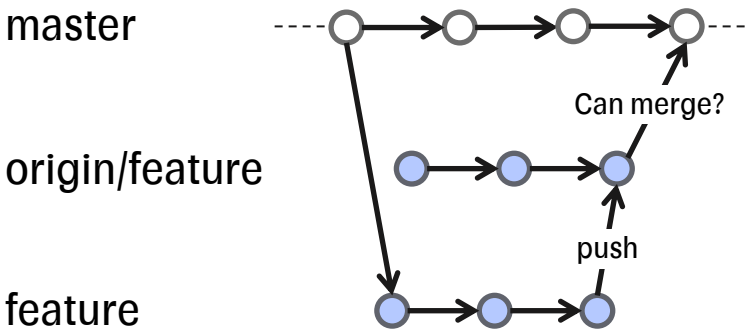
Кодогенерация

```
✗ nx g @twork-mf/schematics:add-plugin    // микрофронтенд-плагин

✔ Which type of plugin you choose? · component
✔ Name of new plugin: · test-plugin
✔ The name of the microfrontend package for which it will be created: · test-mf

CREATE libs/test-plugin/.browserslistrc
CREATE libs/test-plugin/.eslintrc.json
CREATE libs/test-plugin/README.md
CREATE libs/test-plugin/configure-shared-deps.js
CREATE libs/test-plugin/jest.config.js
CREATE libs/test-plugin/package.json
CREATE libs/test-plugin/project.json
CREATE libs/test-plugin/tsconfig.app.json
CREATE libs/test-plugin/tsconfig.editor.json
CREATE libs/test-plugin/tsconfig.json
CREATE libs/test-plugin/tsconfig.spec.json
CREATE libs/test-plugin/src/main.ts
...
```

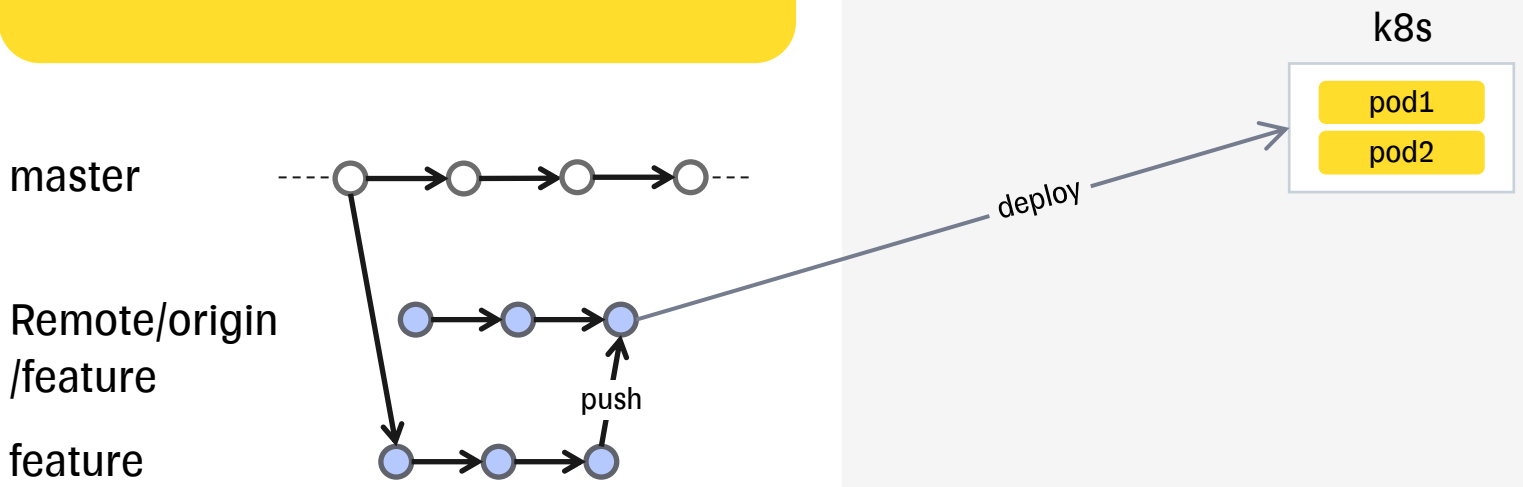
Разработка



Разработка

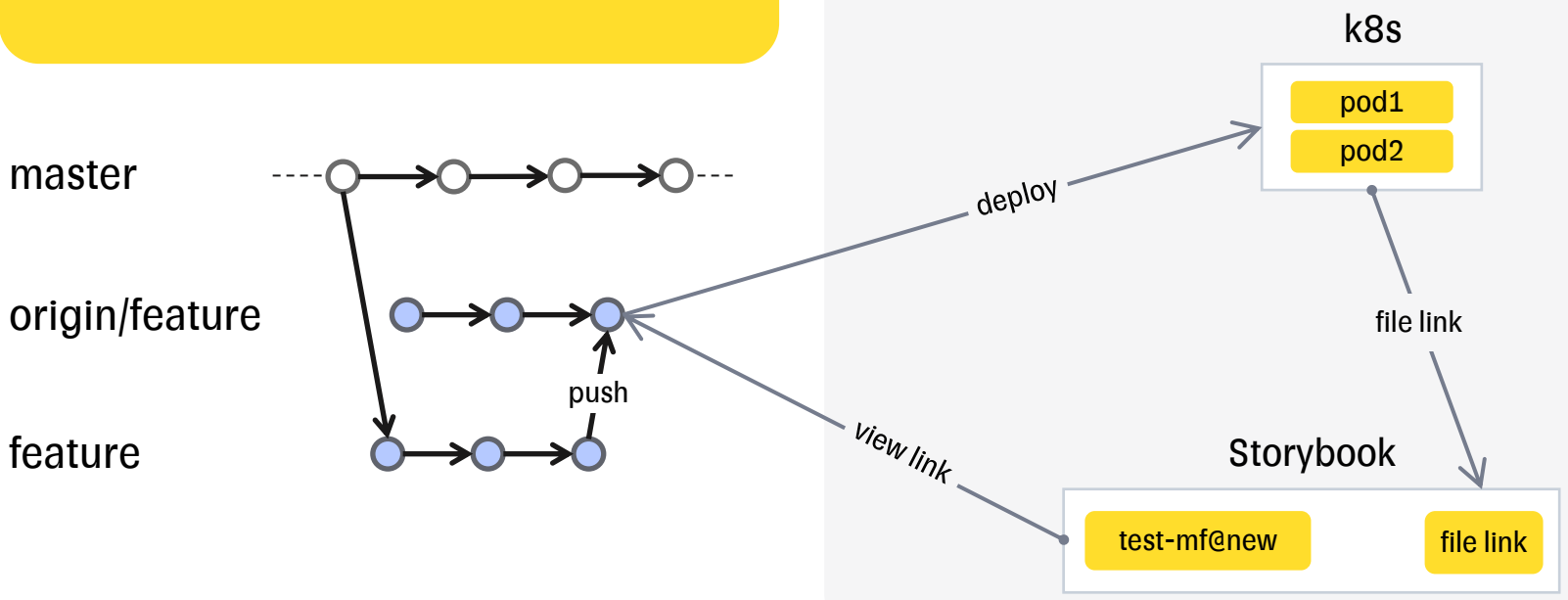


Разработка



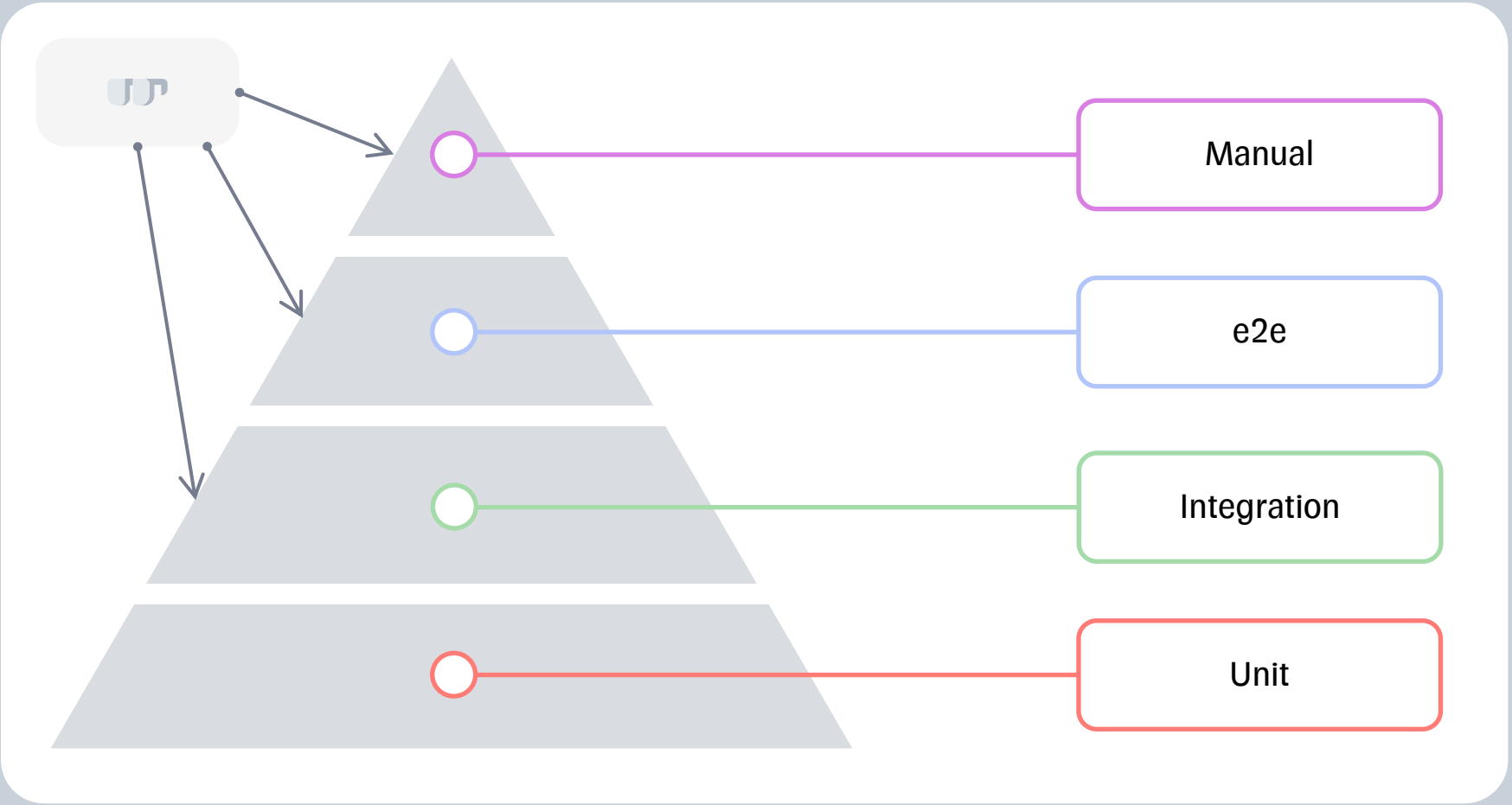
Контроль качества

Разработка


master
origin/feature
feature
push
Контроль качества
k8s
pod1
pod2
deploy
file link
view link
Storybook
test-mf@new
file link

ТИНЬКОФФ

Контроль качества


Manual
e2e
Integration
Unit

Если затронули один микрофронт

Тестирование

Standalone sandbox

Тестируется 1 микрофронт

Не нужно ждать пайплайн

Локально поднимается за 20 секунд

А если нужно проверить взаимодействие?

ТИНЬКОФФ

Интеграция

Адаптер + Виджет

```
[
        {
            "url": "https://<company-cdn-host>/test-mf-adapter/0.1.0/index.js",
            "inputs": {
                "someInput": "<json-path-variable>.someValue"
            },
            "configs": {
                "childMfUrl": "https://<company-cdn-host>/test-mfr/0.1.0/index.js",
                "restEndpoints": {
                    "apiUrl": "https://api-url.company.ru"
                }
            }
        }
]
```

Адаптер + Виджет + Плагины

```
[
    {
        "url": "https://<company-cdn-host>/test-mf-adapter/0.1.0/index.js",
        ...
        "plugins": [
            {
                "id": "plugin-id",
                "type": "component",
                "name": "test-plugin",
                "version": "0.1.0"
            }
        ]
    }
]
```

Локально тоже можно!

```
[
        {
            "url": "http://localhost:4201/index.js",
            ...
        }
]
```

Тестирование

Host sandbox

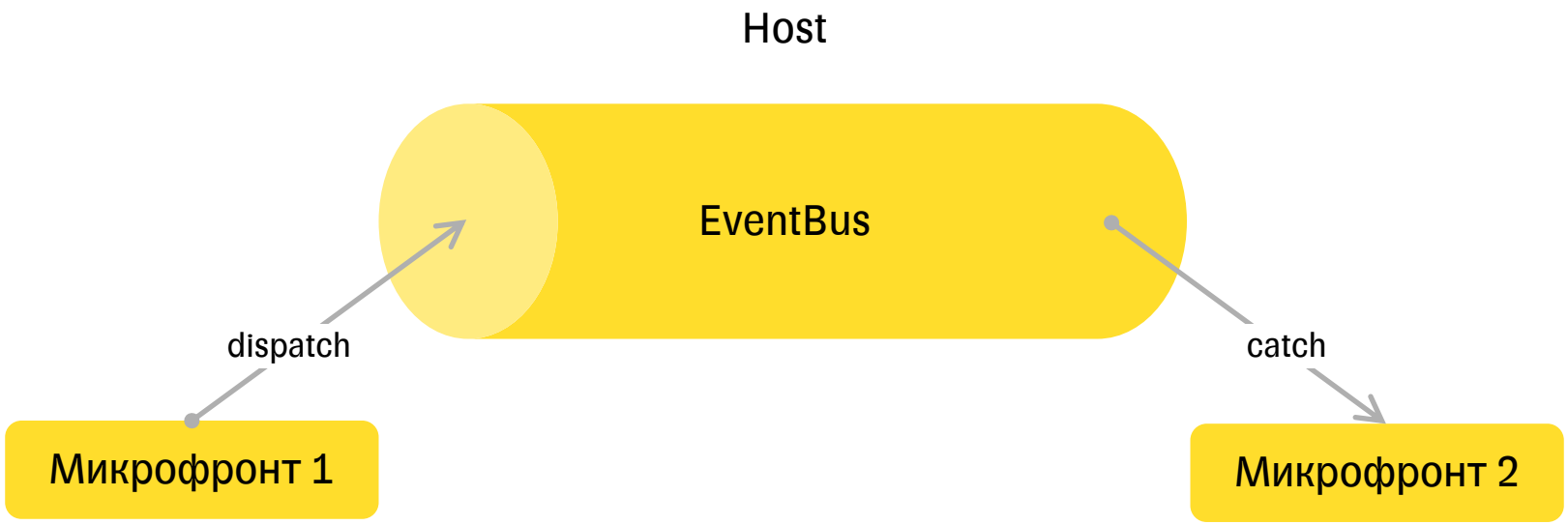
Тестируется несколько конкретных микрофронтов

Конфигурация заполняется вручную

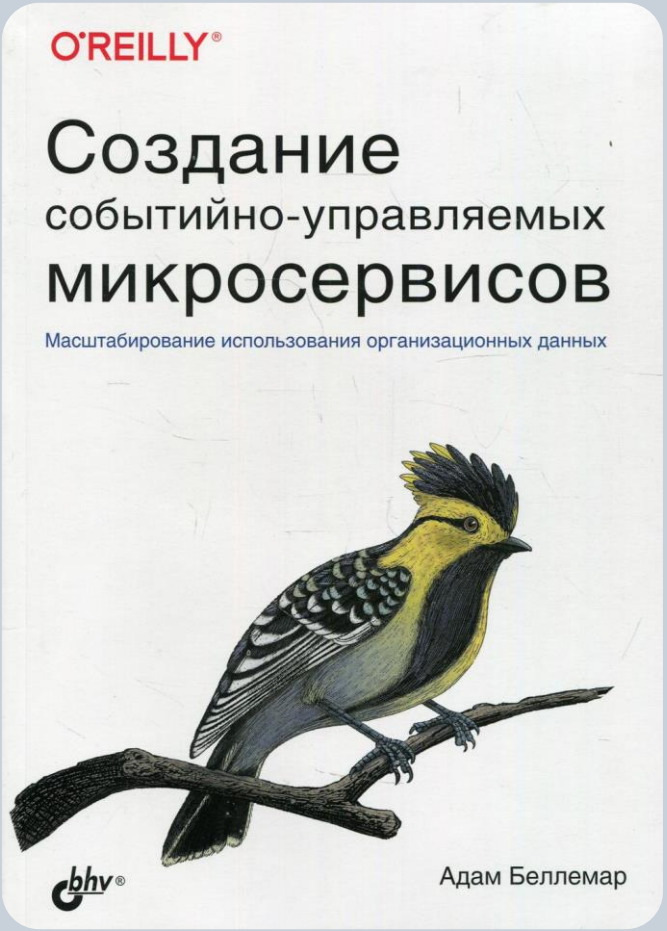
В window есть команды для настройки

А как настроить взаимодействие?

Событийно-ориентированная модель



Подробнее


O'REILLY®
Создание
событийно-управляемых
микросервисов
Масштабирование использования организационных данных
bhv®
Адам Беллемар

Как настраивать события?

В коде

callback

```
eventBusService.addActionHandler(
  'best-component',
  'closeMe',
  ({ bestPayload }: { bestPayload: boolean }) => {
    console.log(bestPayload);

    return of(void 0).pipe(delay(10000));
  },
);
```

В конфиге

event-action

```
"eventName": "likeJavaScript",
"source": "javascript-liker",
"actions": [
    {
        "name": "someName",
        "target": "someTarget",
        "payload": {}
    }
]
```

А если нужно протестировать систему целиком?

Тестирование

Host

Наиболее приближенно к реальности

Целевое решение для тестирования N микрофронтов

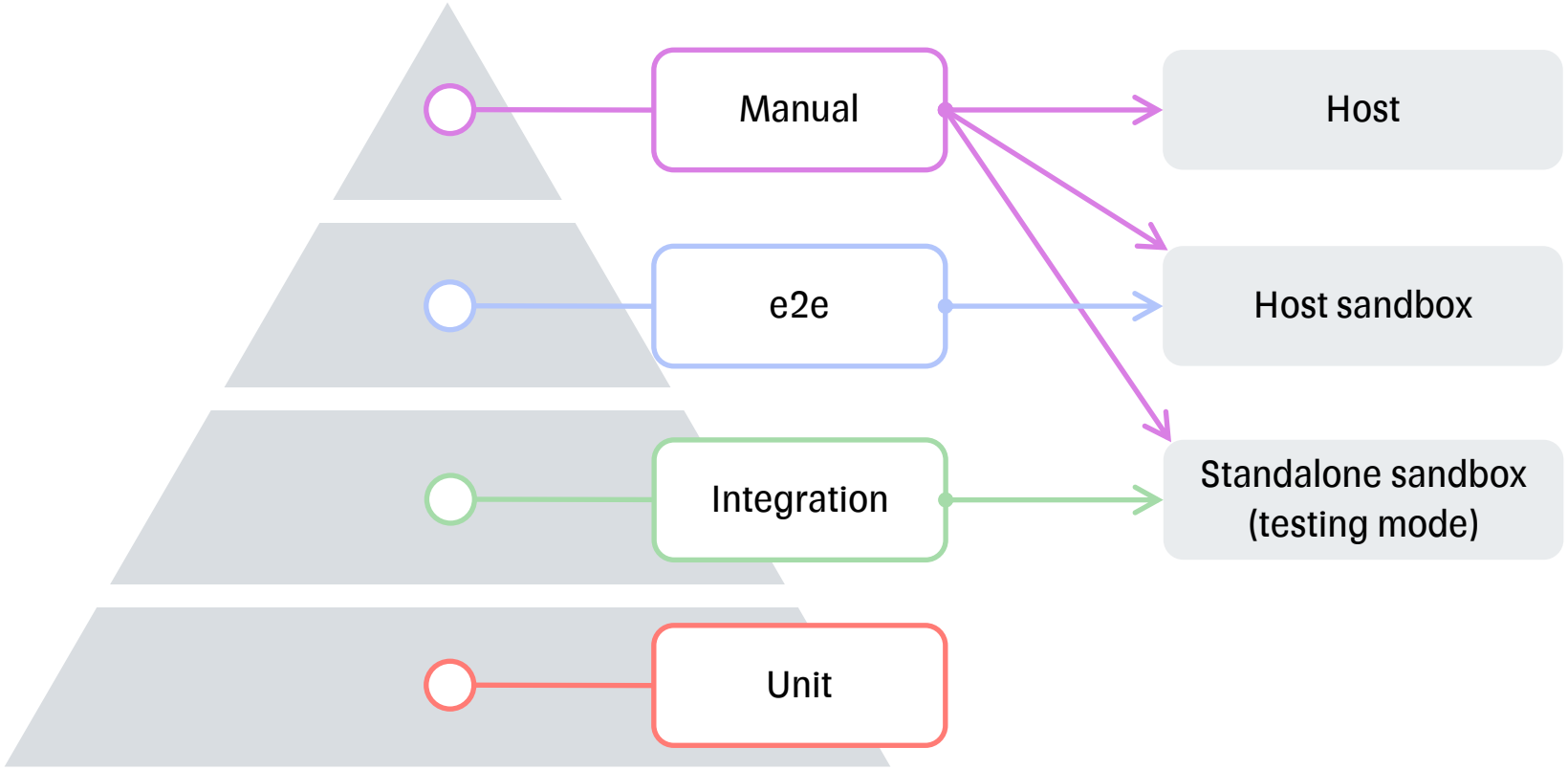
Конфигурация заполняется вручную

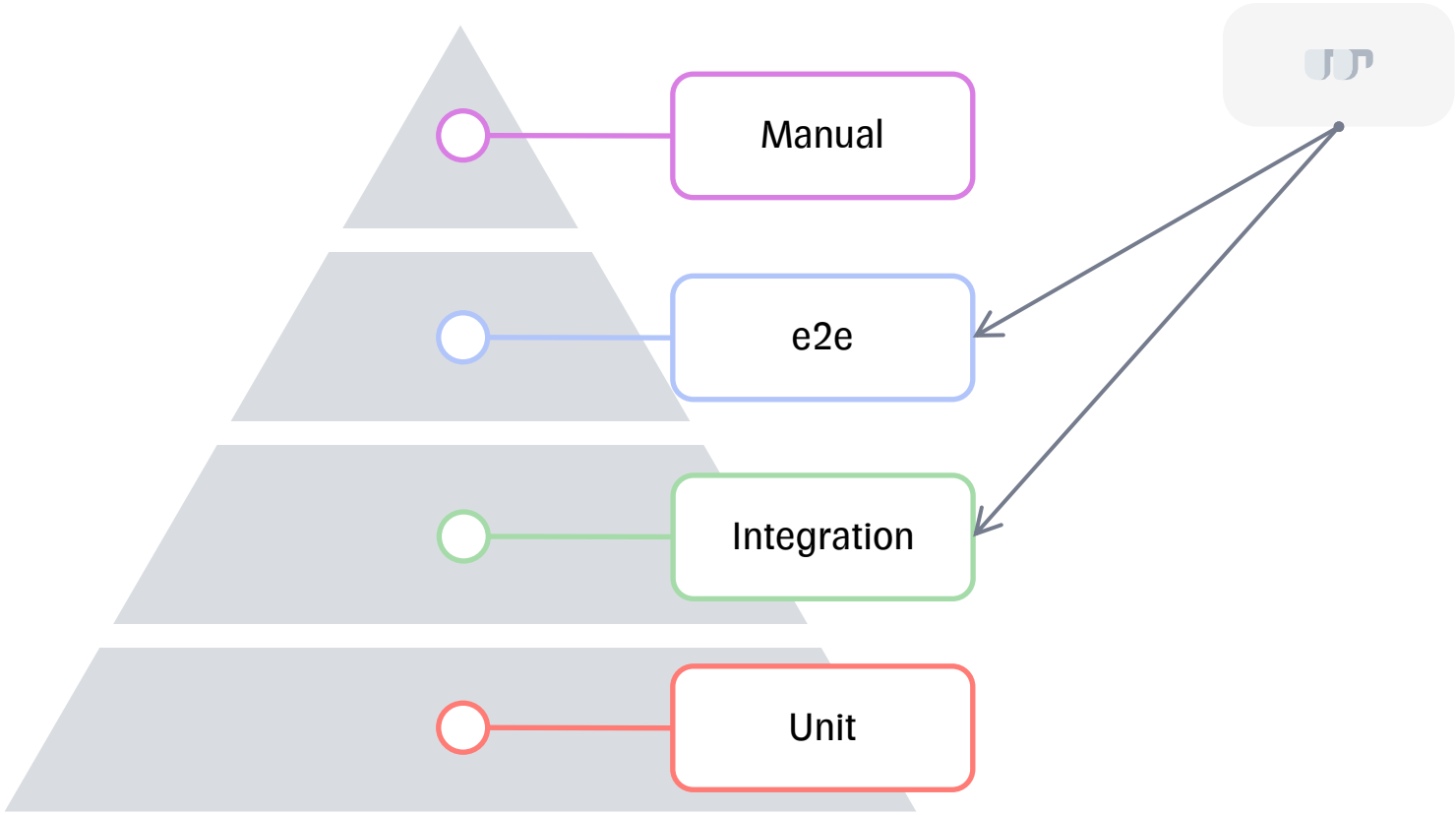
ТИНЬКОФФ

Способы тестирования микрофронтендов

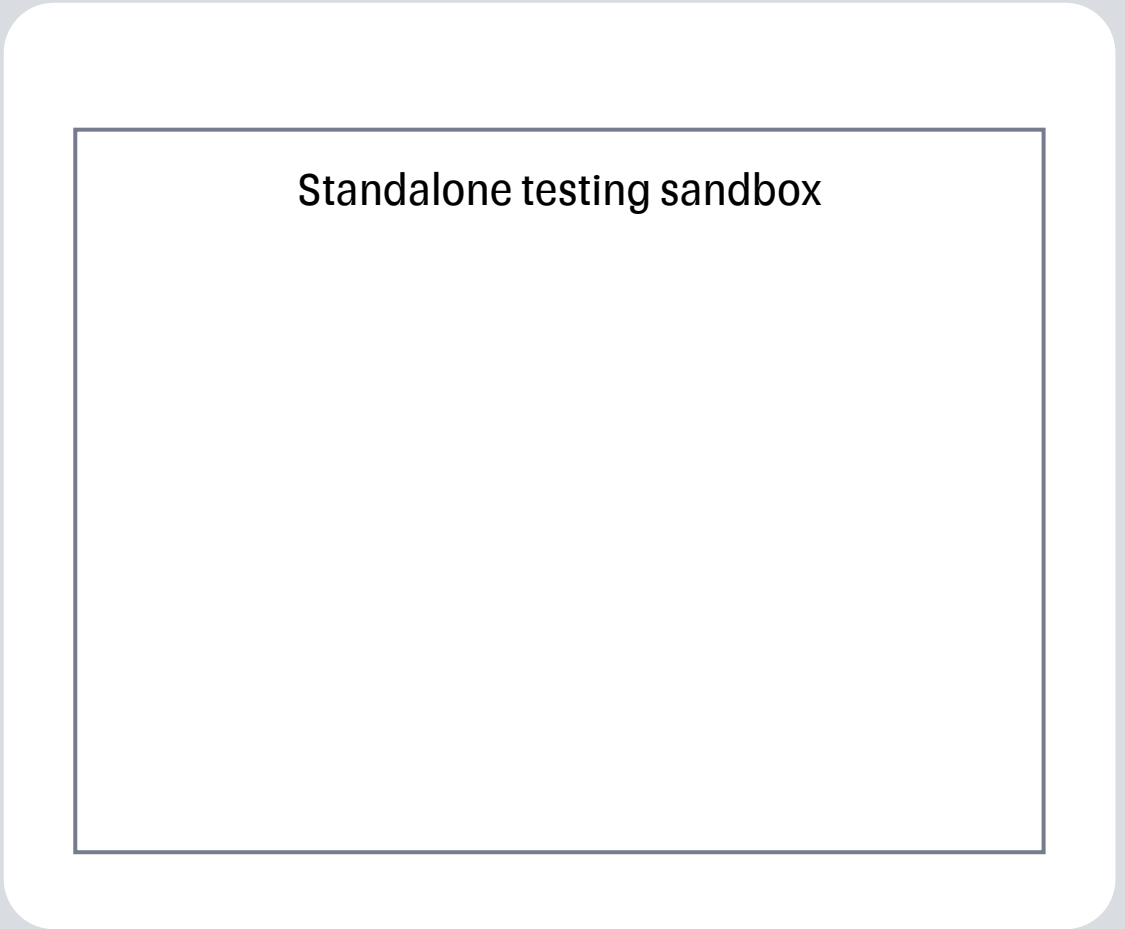
Тестирование

Standalone sandbox	Host sandbox	Host
Тестируется 1 микрофронт	Тестируется несколько конкретных микрофронтов	Наиболее приближенно к реальности
Не нужно ждать пайплайн	Конфигурация заполняется вручную	Целевое решение для тестирования N микрофронтов
Локально поднимается за 15 секунд	В window есть команды для настройки конфигурации	Конфигурация заполняется вручную

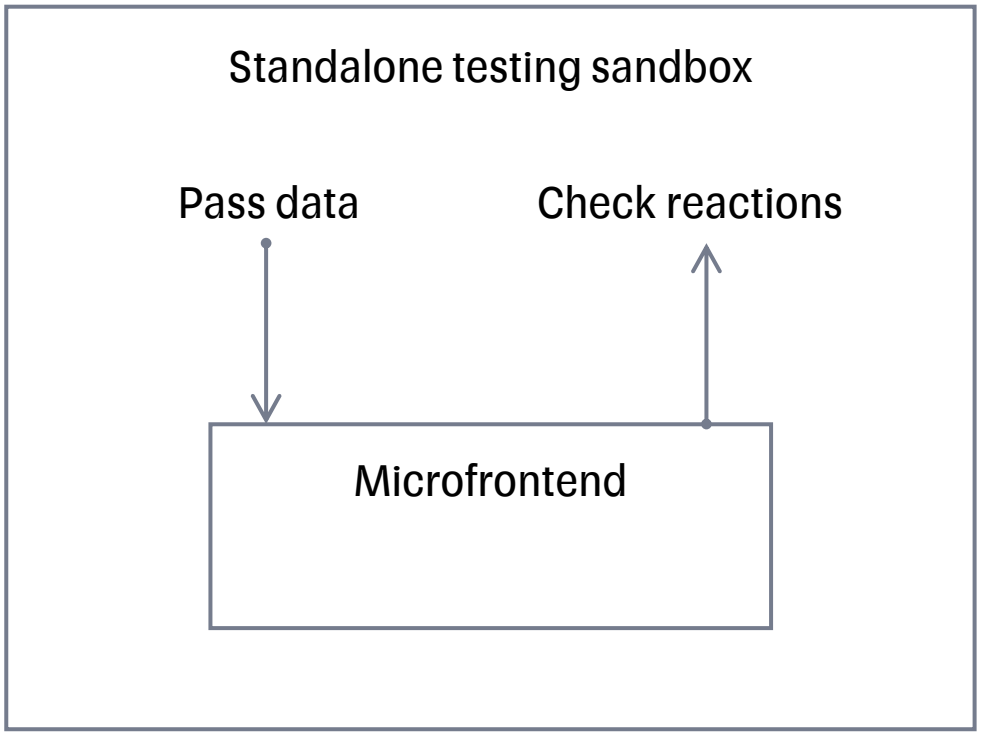

Manual
e2e
Integration
Unit
Host
Host sandbox
Standalone sandbox
(testing mode)


Manual
e2e
Integration
Unit

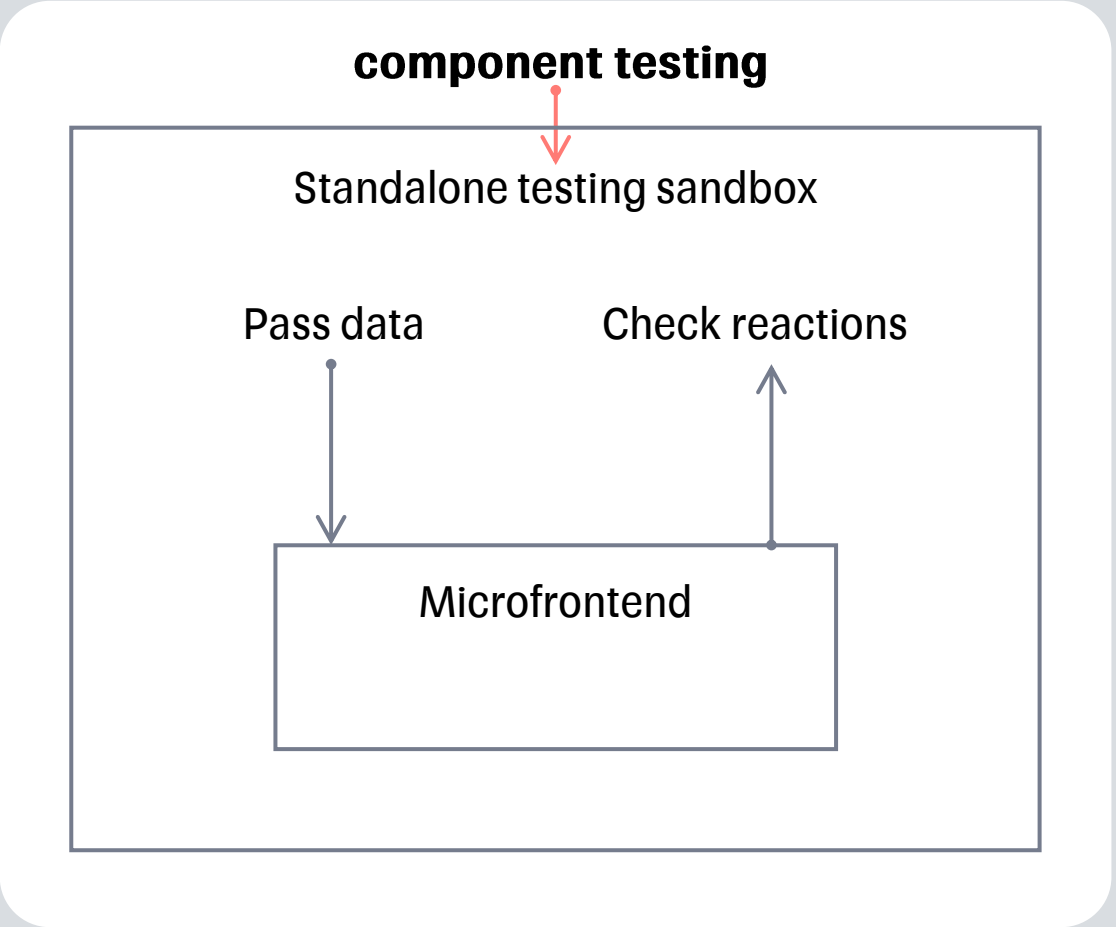
Integration



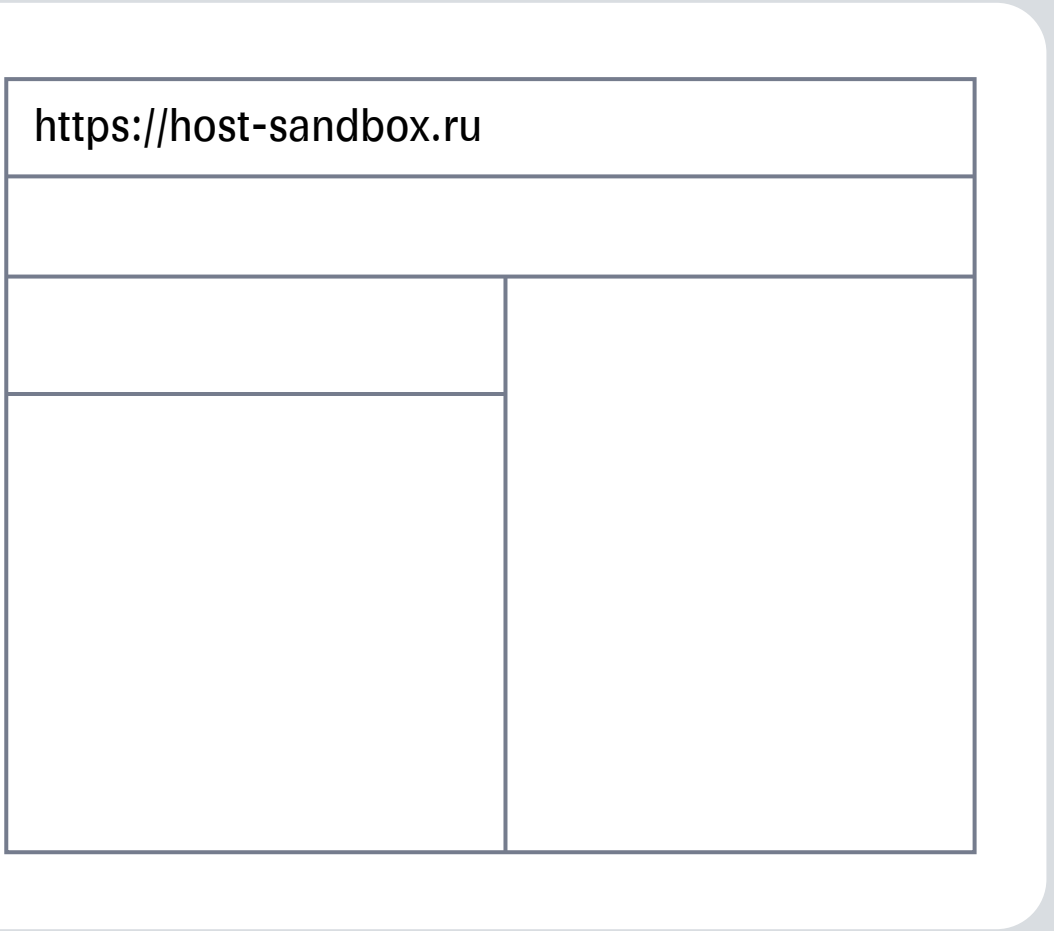
Integration



Integration



e2e

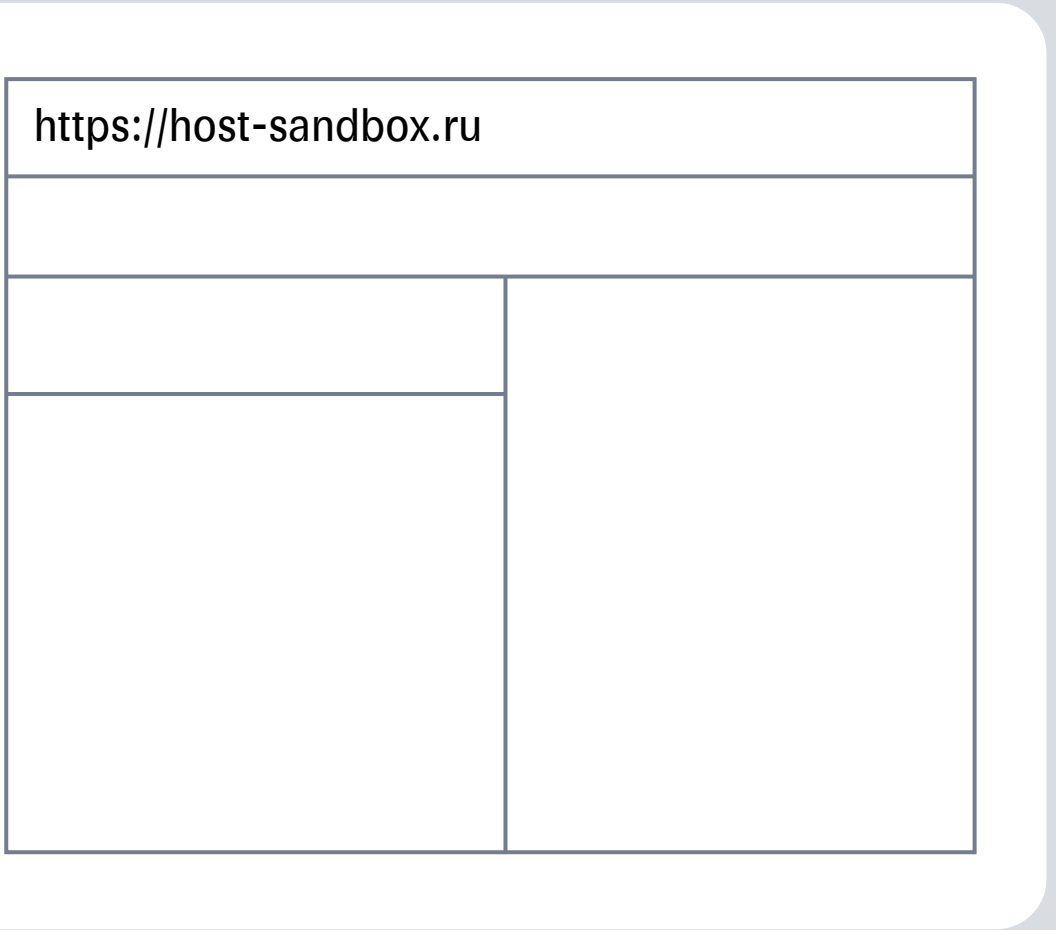


e2e

window.getConfig

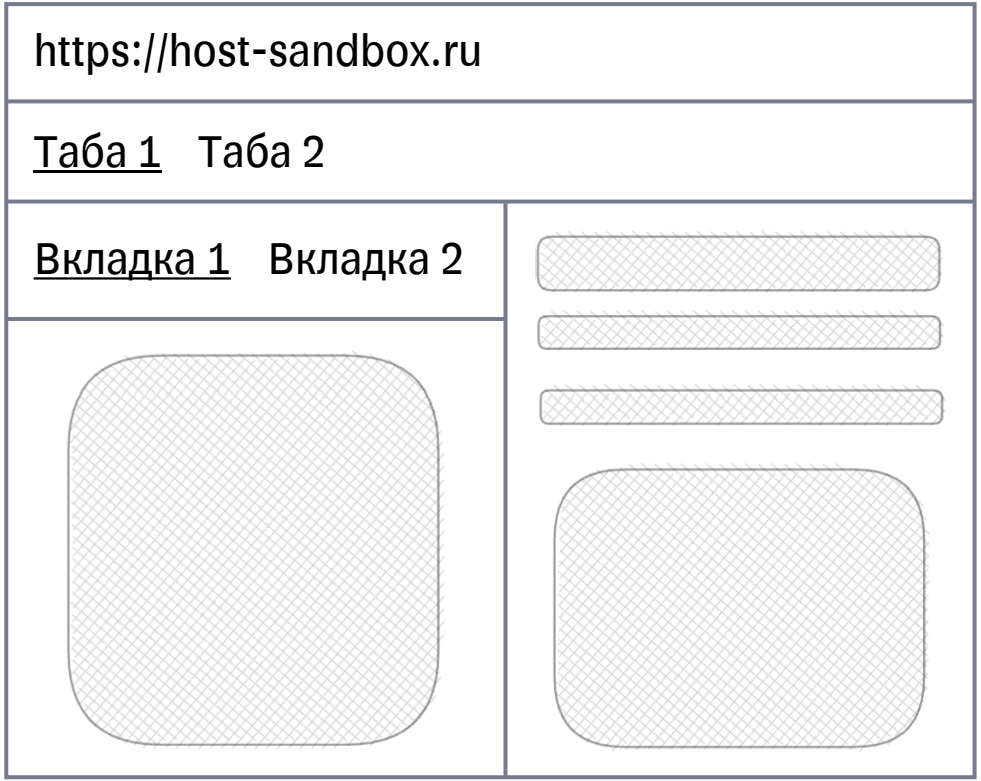
window.setConfig

window.dispatchEvent



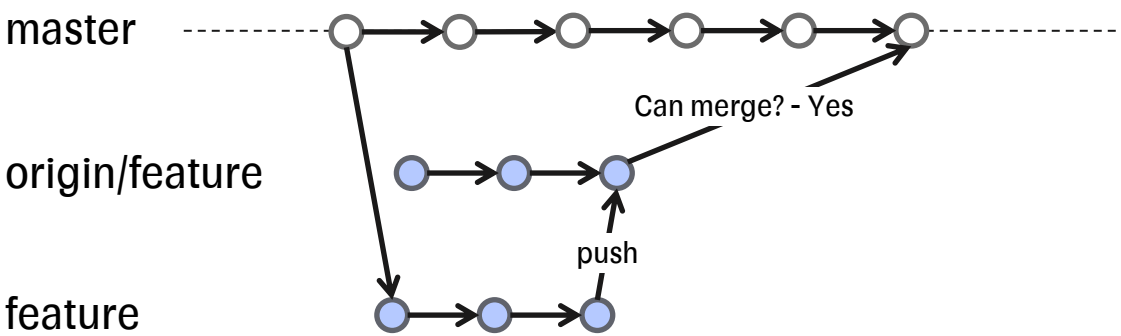
e2e

Start testing!

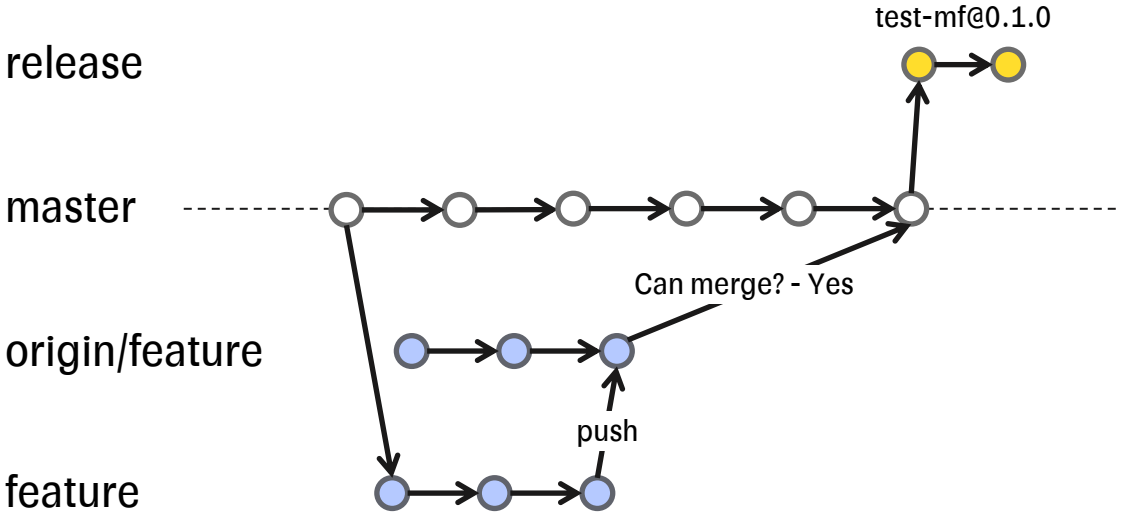


Работает как надо?

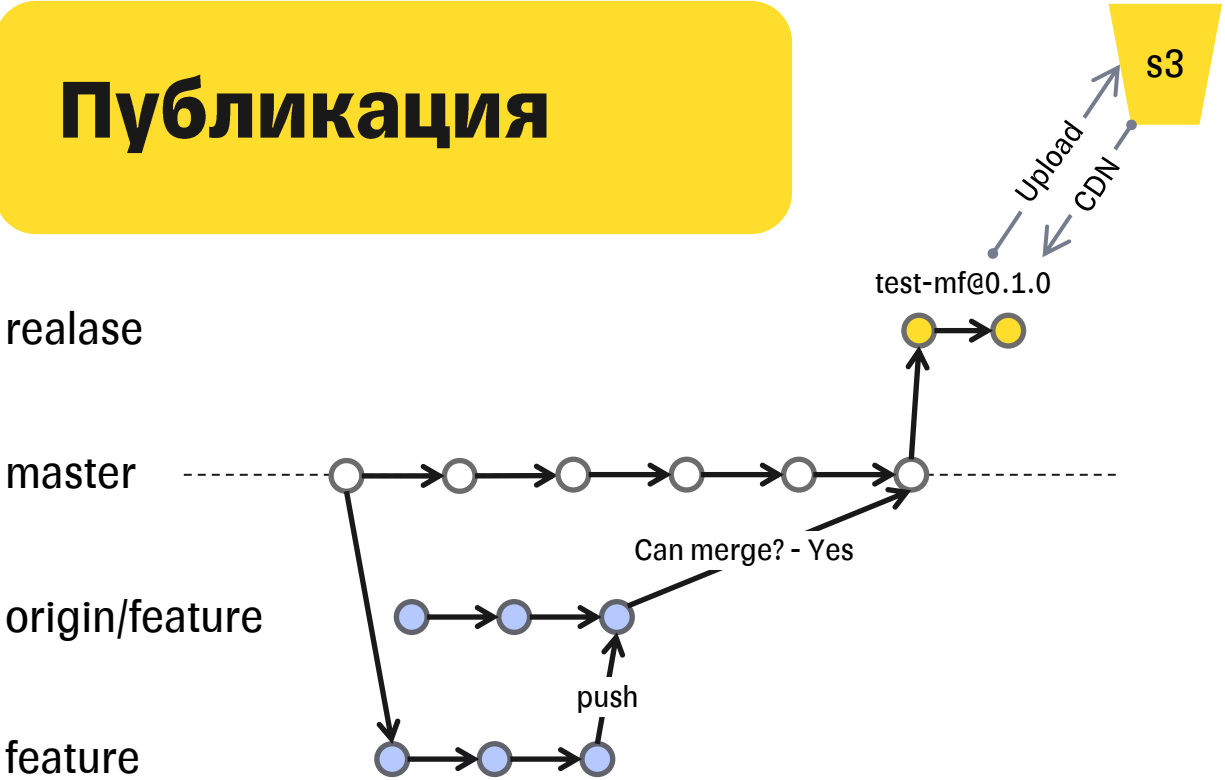
Публикация



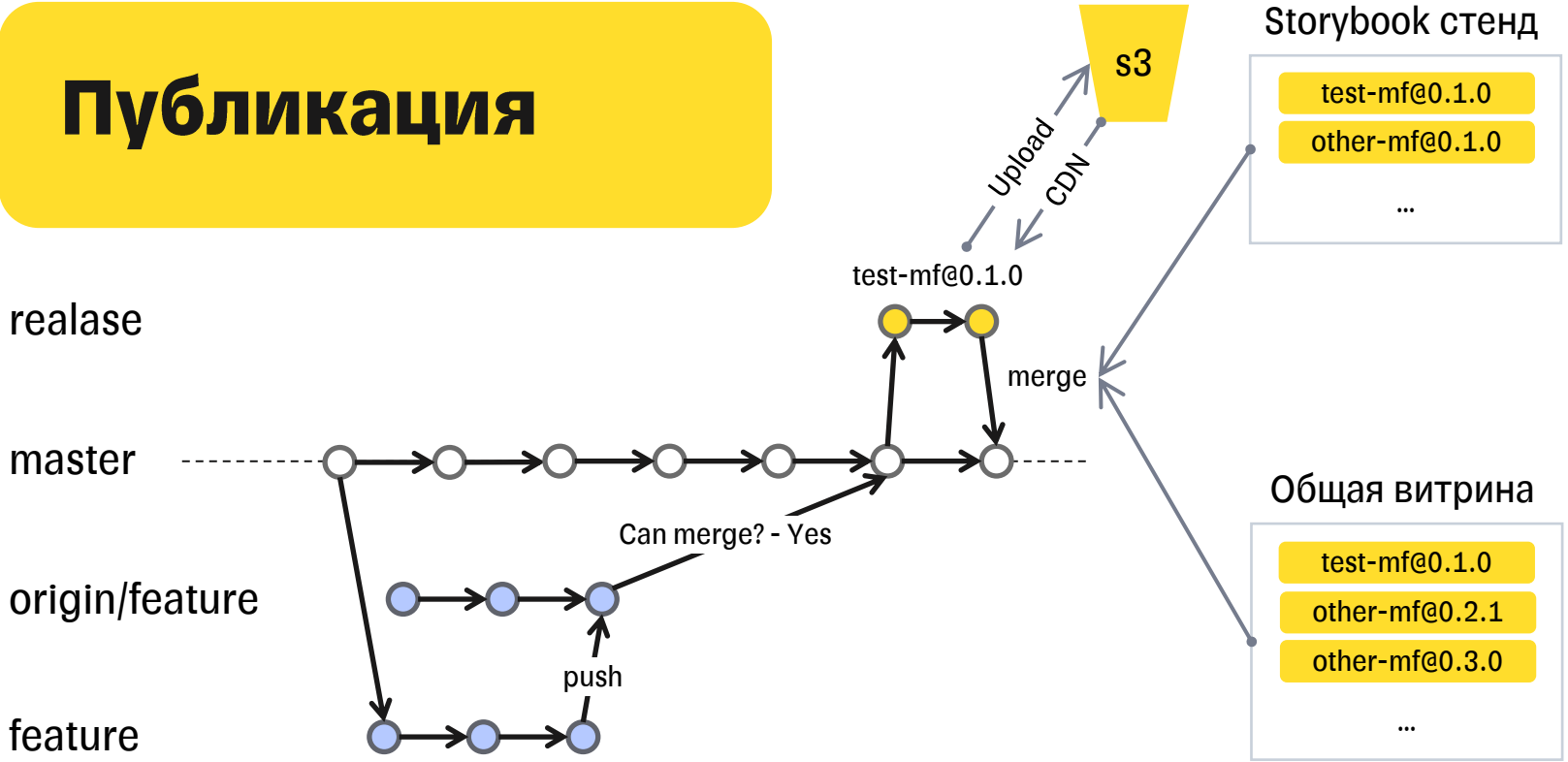
Публикация


test-mf@0.1.0
release
master
Can merge? - Yes
origin/feature
push
feature

Публикация


s3
Upload
CDN
test-mf@0.1.0
realase
master
Can merge? - Yes
origin/feature
push
feature

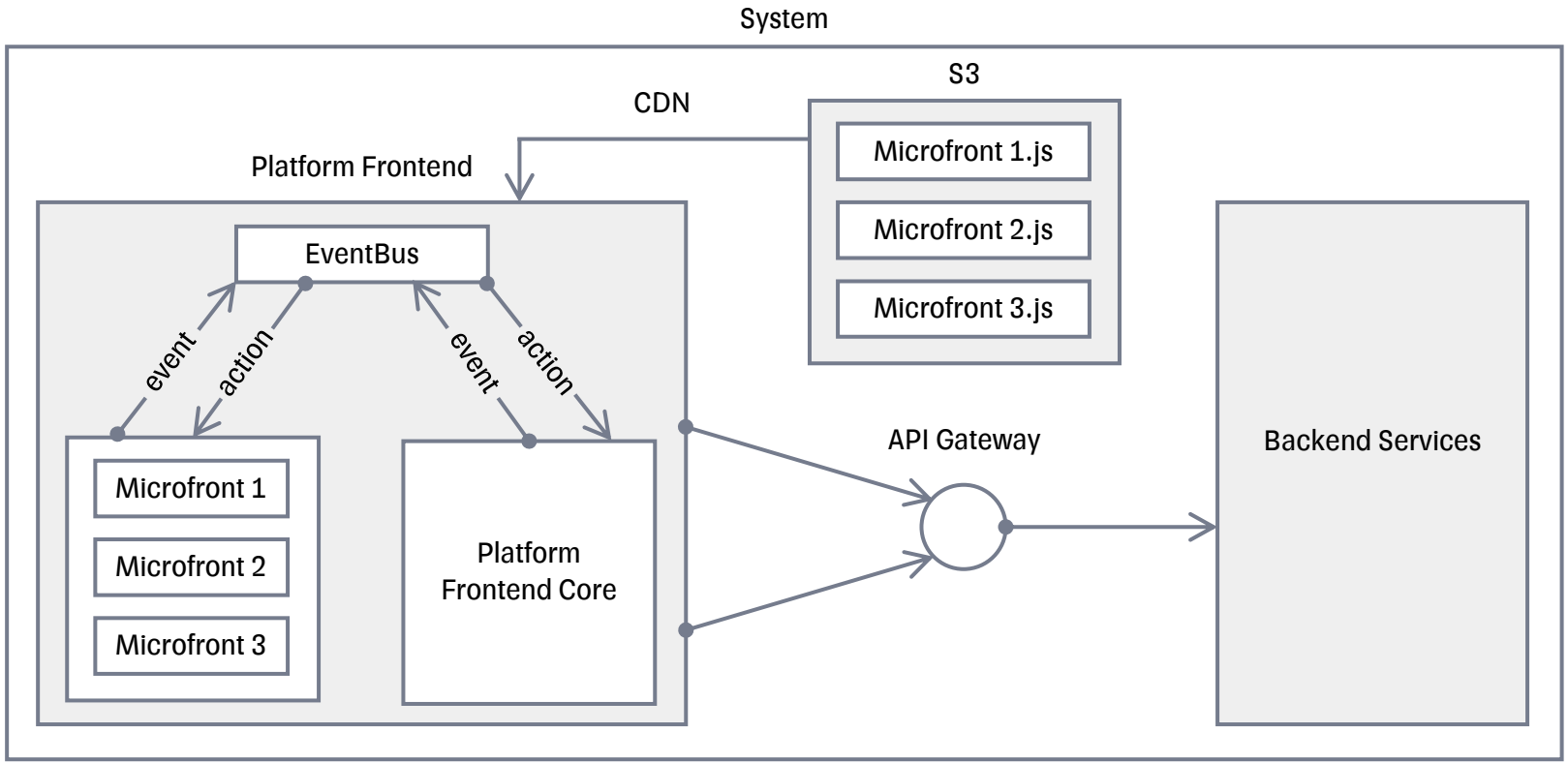
Публикация


s3
Upload
CDN
test-mf@0.1.0
realase
master
origin/feature
feature
merge
Can merge? - Yes
push
Storybook стенд
test-mf@0.1.0
other-mf@0.1.0
...
Общая витрина
test-mf@0.1.0
other-mf@0.2.1
other-mf@0.3.0
...

Можно использовать в приложении!

ТИНЬКОФФ

Архитектура уровня приложения


System
S3
CDN
Microfront 1.js
Microfront 2.js
Microfront 3.js
Platform Frontend
EventBus
event
action
event
action
Microfront 1
Microfront 2
Microfront 3
Platform
Frontend Core
API Gateway
Backend Services

Когда пригодятся микрофронтенды

- Сложный продукт, требующий разделения на домены
- Нужна вариативность конфигурируемых решений в рамках одной системы
- Необходимо часто выпускать точечные изменения, независимо от релизного цикла остального приложения
- Команд много, в каждой есть несколько разработчиков

Резюме

Бенефиты

- Быстрые релизы
- Независимая параллельная разработка
- Разделение кода
- Ограниченная ответственность

Дополнительно

- Коробочная разработка
- Гибкость конфигурации
- Независимое кеширование чанков

Расходы

- Усложнение инфраструктуры
- Усложнение отладки и тестирования

Дополнительно

- Когнитивная сложность
- Стагнация версий
- Неизведанные проблемы

Герман Панов

@mdlufy

Сергей Алемасов

@infernal_apel

ТИНЬКОФФ