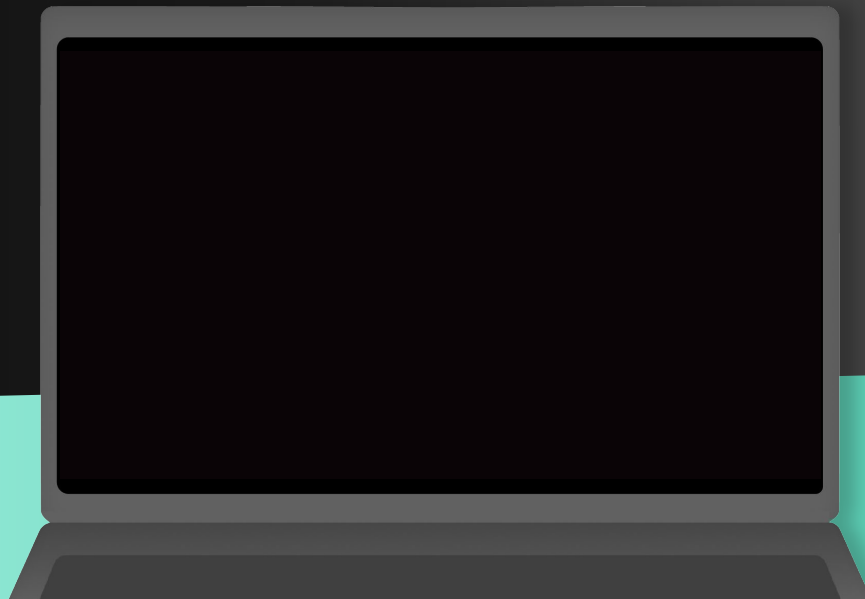




How to Expand Your IDP


The New Building Blocks of Backstage

Ben Lambert & Patrik Oldsberg






Ben Lambert

 benjdlambert

Senior Engineer @ Spotify
Core Maintainer | Backstage



Patrik Oldsberg

 Rugvip

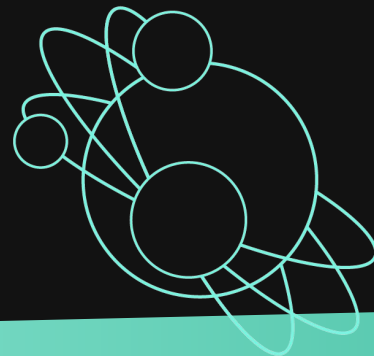
Senior Engineer @ Spotify
Core Maintainer | Backstage

Agenda

- Project updates
- Quick-fire round
- Builder experience



Project Updates



New Project Areas

Documentation

- André Wanlin – Spotify
- Aramis Sennyey - DoorDash
- Peter Macdonald – VodafoneZiggo

Meeting on Fridays

Framework Special Interest Group (SIG)

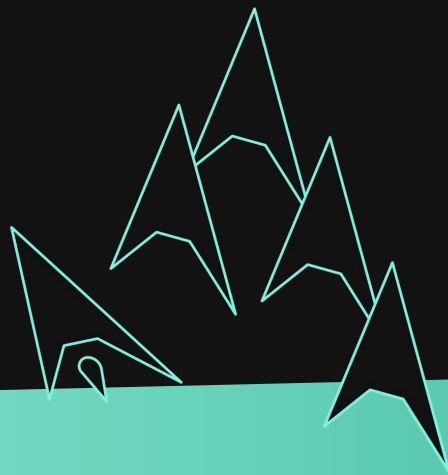
- Catalog SIG + Adoption SIG ➡ Framework SIG
- Design discussions
- Reviews - Pull Requests & Backstage Enhancement Proposals

Community Plugins

- 134 plugin packages across 77 plugin workspaces
- Updated governance to simplify path to plugin maintainer

Quick-fire Round

Hold on to your hats 🚀



`npx @backstage/create-app` 

- Yarn v4
- New Backend System

Backend System 1.0 🎉

- Released in 1.31
- Including the new auth system
- Old system is deprecated
 - Loosely scheduled to remove support by end of 2024
 - Report or comment to highlight any blocking issues

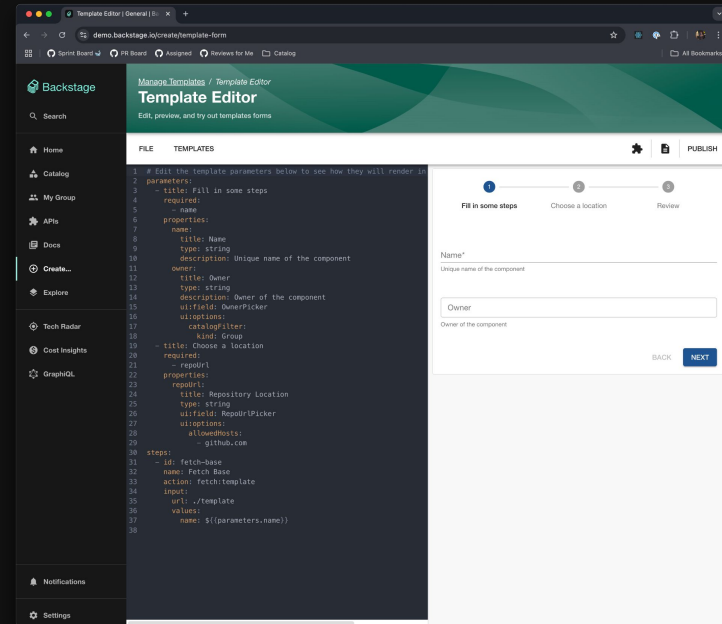
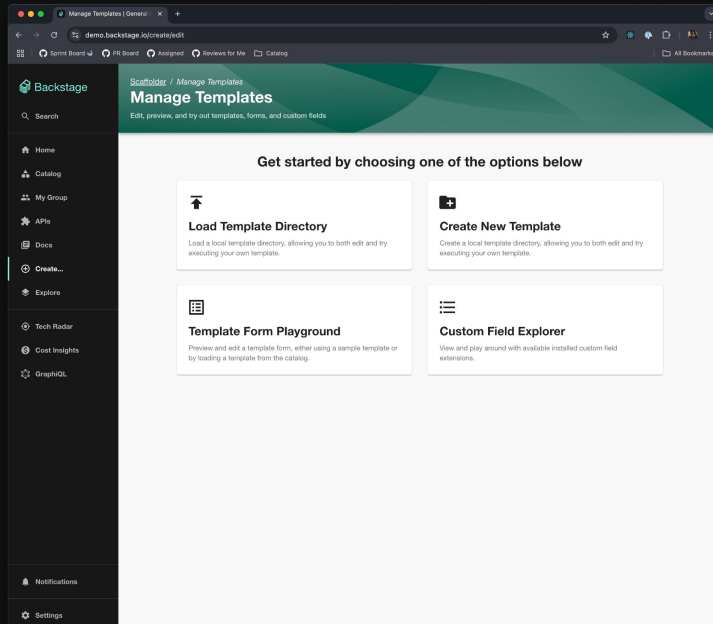
2024 Security Audit

- First audit in 2022
 - 12 findings – 2 critical, 2 high, 5 medium, 3 low
 - Introduced the Backstage Threat Model
- Second audit
 - 4 findings – 3 high, 1 medium
 - All fully remedied in the 1.31 release
 - Covered the new authentication system

Built-in Event Bus

- Old solution
 - Forwarded events in-process only
 - Worked for forwarding webhooks, but not much else
- New solution
 - Event bus built into the events plugin backend
 - At-most-once delivery, graceful shutdown

Scaffolder UI Updates



Scaffolder Form Decorators

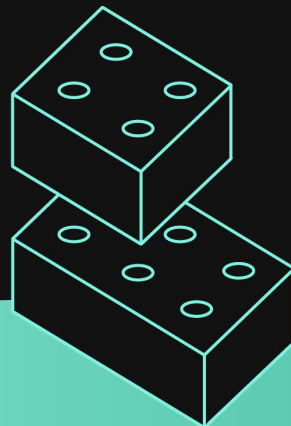
- `secrets` are no longer coupled to `FieldExtensions`
 - No more hidden `FieldExtensions` to collect `secrets`
- Allows mutating both `secrets` and `formData` just before submission
 - Less risk that `secrets` have expired by the time the jobs run

Scaffolder Retries

- Jobs that get stuck due to rollouts can be resumed
- Jobs that fail can be retried
- Checkpoints to save state between retries for idempotency
- Big shoutout to Bogdan for the work here! 🎉

Demo

Plugins & Builder Experience

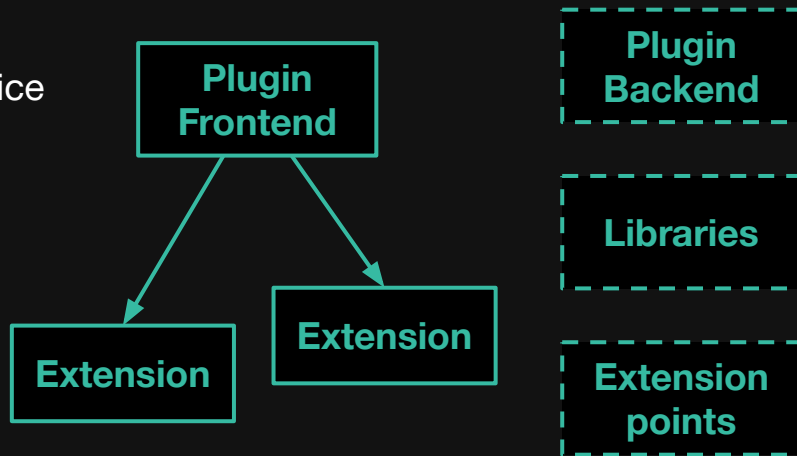


Why a plugin system?

- Vision: Power thousands of highly successful Internal Developer Portals
- Characteristics of a typical IDP
 - Large number of features and integrations
 - Rapid iteration, creation, and removal of features
 - Distributed ownership across many teams
 - Moderate connectivity across features

What is a plugin in Backstage?

- Isolated, and owned by a single team
- Standalone feature or integration with external service
- A collection of frontend extensions
- Optional extras
 - Plugin backend
 - Libraries for other plugins
 - Extension points



Where do plugins come from?

- Core features of Backstage
 - Catalog, Scaffolder, Auth, Search, etc.
- 3rd-party plugins
 - Open source community
 - Service providers
 - Commercial
- Your own plugins
 - Tailored to your own organization
 - Built by platform teams, and others

Why build your own plugins

- Centralize existing internal tools
- Tailor features to your own organization
- Drive adoption of Backstage

Tips for building your own plugins

- Aim for distributed ownership
- Solve real problems
- Guidelines for integrations with external services
 - Provide at-a-glance view of most important data
 - Expose 1-3 key actions
 - Deep link to the service for everything else

Plugin builder experience goals

- Help build plugins that...
 - provide a consistent and cohesive experience
 - are maintainable
 - can be customized and extended
- Help plugin builders by providing...
 - great tooling
 - clear and intuitive APIs
 - solutions for common use-cases
 - a strong community

Builder Experience

Framework – Frontend System



Frontend System – Scope

- Deep customizability / theming
- Extensibility
- Clear communication patterns
- Maintainability and isolation

Frontend System – New System

- Current system is a good first iteration
 - But we can do better
- Remove the need for JSX app tree
- Push more responsibility to plugins

Blueprints

```
export const scaffolderPage = PageBlueprint.make({  
  params: {  
    routeRef: rootRouteRef,  
    defaultPath: '/create',  
    loader: () => import('../components/Router')  
  },  
})
```

Overrides

```
export default catalogPlugin.withOverrides({
  extensions: [
    catalogPlugin.getExtension('page:catalog').override({
      params: {
        defaultPath: '/',
      },
    }),
  ],
})
```

Demo

Frontend System – Roadmap

- Fully implemented base app
 - Catalog entity pages
 - Sidebar
- Broader adoption in the plugin ecosystem
- Validate at scale – Adopt internally at Spotify

Builder Experience

Framework – Backend System



Backend System – Scope

- Not required for your own plugins!
- Lightweight microservices framework
- Modular design that allows for extensibility
- Emphasis on standardized service interfaces

Backend System – Services

Core Services

- Auth
- Cache
- Config
- Database
- Discovery
- Health
- Http Auth
- Http Router
- Lifecycle
- Logger
- Permissions
- Plugin Metadata
- Scheduler
- Url Reader
- User Info

Plugin Services

- Catalog API
- Events
- Notifications
- Signals

Backend System – Testing

- Well defined interfaces – standardized mocks
- Mocks for each core service
- Integration tests via `startTestBackend` with build-in mocks

Demo

Backend System – Roadmap

- New core services
- Extensible auth service – bring your own service auth
- Improved handling of database migrations – rollbacks

Other Areas – Roadmap

- More flexible CLI – good place to contribute!
- Better Software Catalog backend performance
- Improved design system

Thank you!

backstage.io

Feedback

