goto;

GOTO
Copenhagen 2023

#GOTOcph

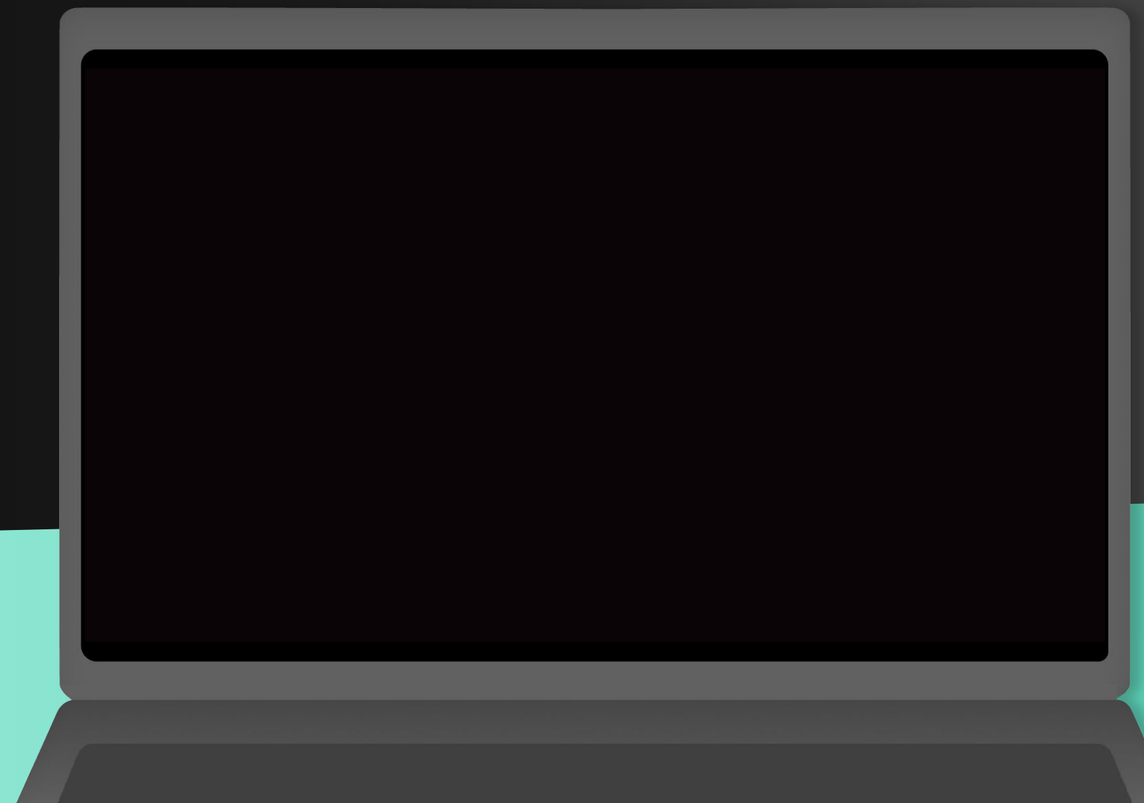
Backstage

Spotify Plugins for Backstage:

How commercial and open source software go hand-in-hand at Spotify

Joon Park

Engineering Manager, Spotify



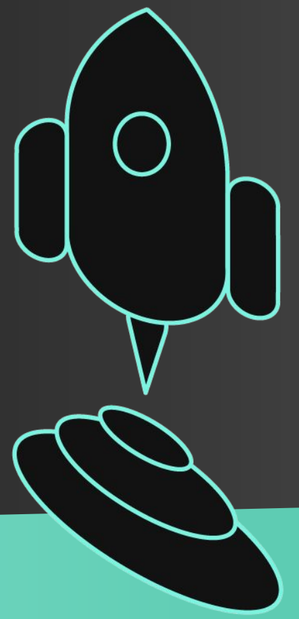
GOTO Copenhagen

2023 Oct 4

Backstage

Joon Park

JoonPark13



What is Backstage?

Plugin ecosystem

Spotify Plugins for Backstage

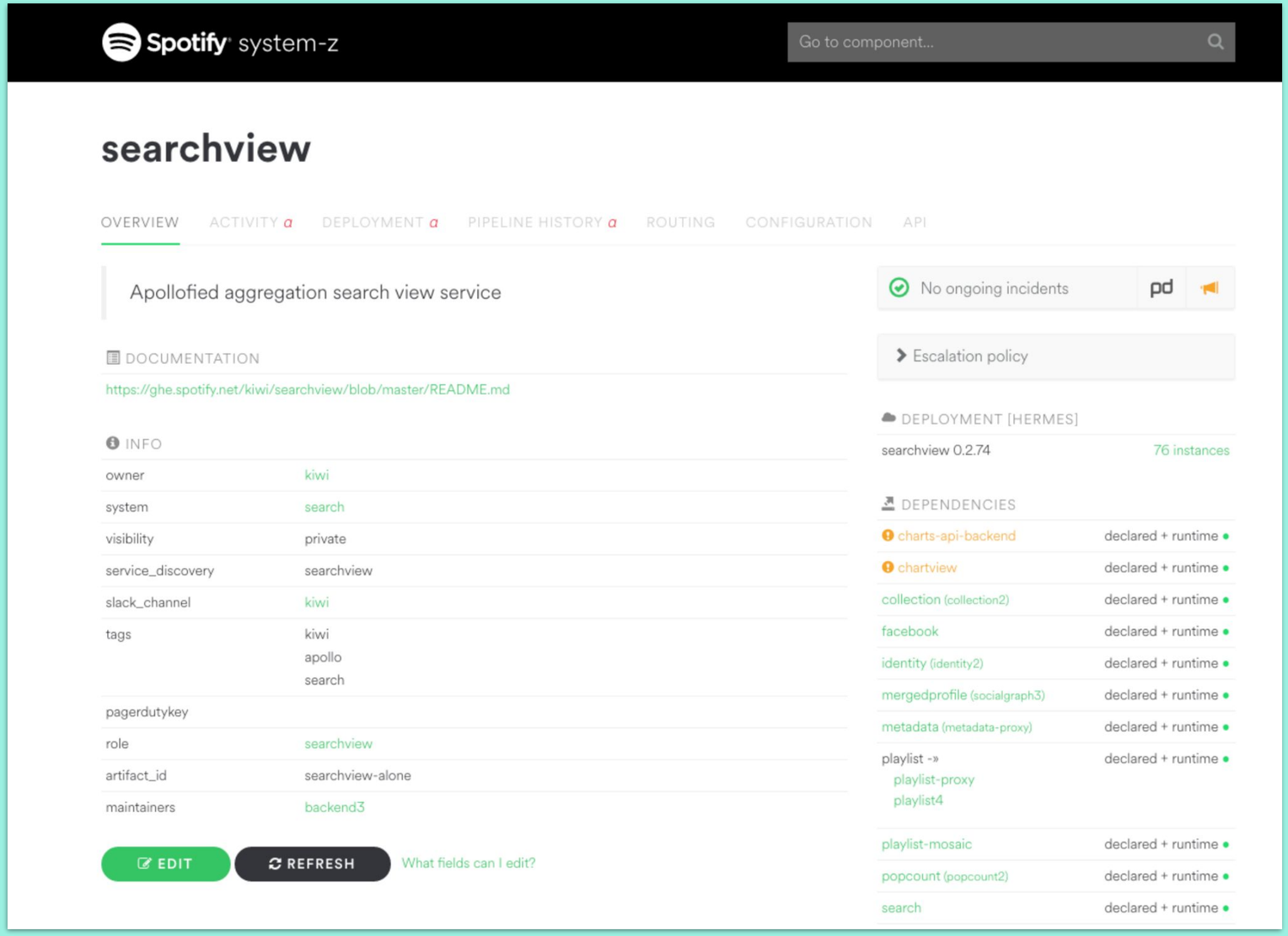
Authorization in Backstage

Setting ourselves up for success

Lessons learned

Backstage

An open platform for building developer portals


Spotify system-z
Go to component...
searchview
OVERVIEW
ACTIVITY α
DEPLOYMENT α
PIPELINE HISTORY α
ROUTING
CONFIGURATION
API
Apollofied aggregation search view service
DOCUMENTATION
https://ghe.spotify.net/kiwi/searchview/blob/master/README.md
INFO
owner kiwi
system search
visibility private
service_discovery searchview
slack_channel kiwi
tags kiwi apollo search
pagerdutykey
role searchview
artifact_id searchview-alone
maintainers backend3
EDIT
REFRESH
What fields can I edit?
No ongoing incidents
pd
Escalation policy
DEPLOYMENT [HERMES]
searchview 0.2.74
76 instances
DEPENDENCIES
charts-api-backend declared + runtime
chartview declared + runtime
collection (collection2) declared + runtime
facebook declared + runtime
identity (identity2) declared + runtime
mergedprofile (socialgraph3) declared + runtime
metadata (metadata-proxy) declared + runtime
playlist -» playlist-proxy playlist4 declared + runtime
playlist-mosaic declared + runtime
popcount (popcount2) declared + runtime
search declared + runtime

Backstage

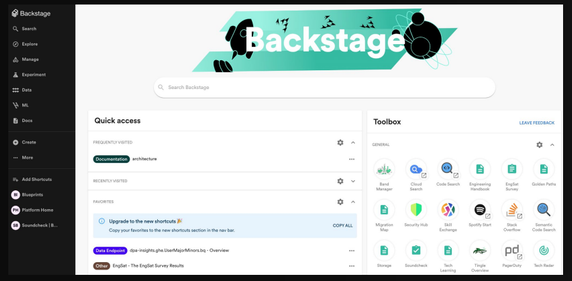
66,000,000 BC

The invention of spreadsheets

System Z is born

2018

System Z becomes **Backstage**



2019

Backstage is released into **open source**

2020

Backstage is **donated to** the CNCF

2021

Spotify Plugins for Backstage launches

2022

Backstage moves to **CNCF Incubator**

2023

The future

Backstage becomes the universal developer portal

Backstage in open source

1M+
end-users

2K+
adopters

100+
contributed plugins

9,000+
members

21K+
stars

1K+
contributors

15K+
contributions

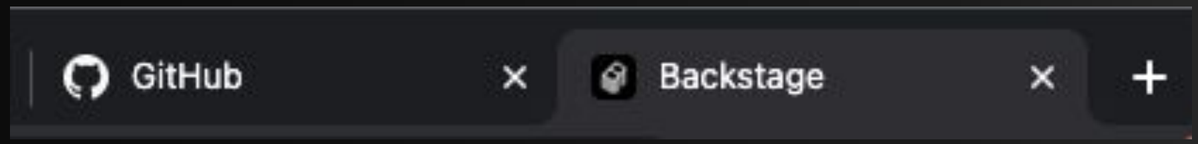
4K+
project forks

https://github.com/

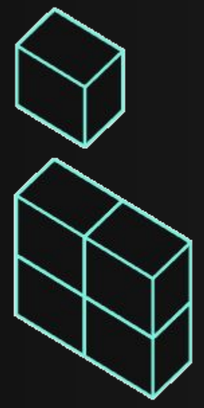
Backstage

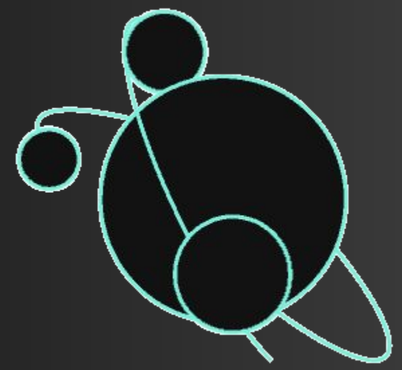
https://github.com/

Backstage



Backstage



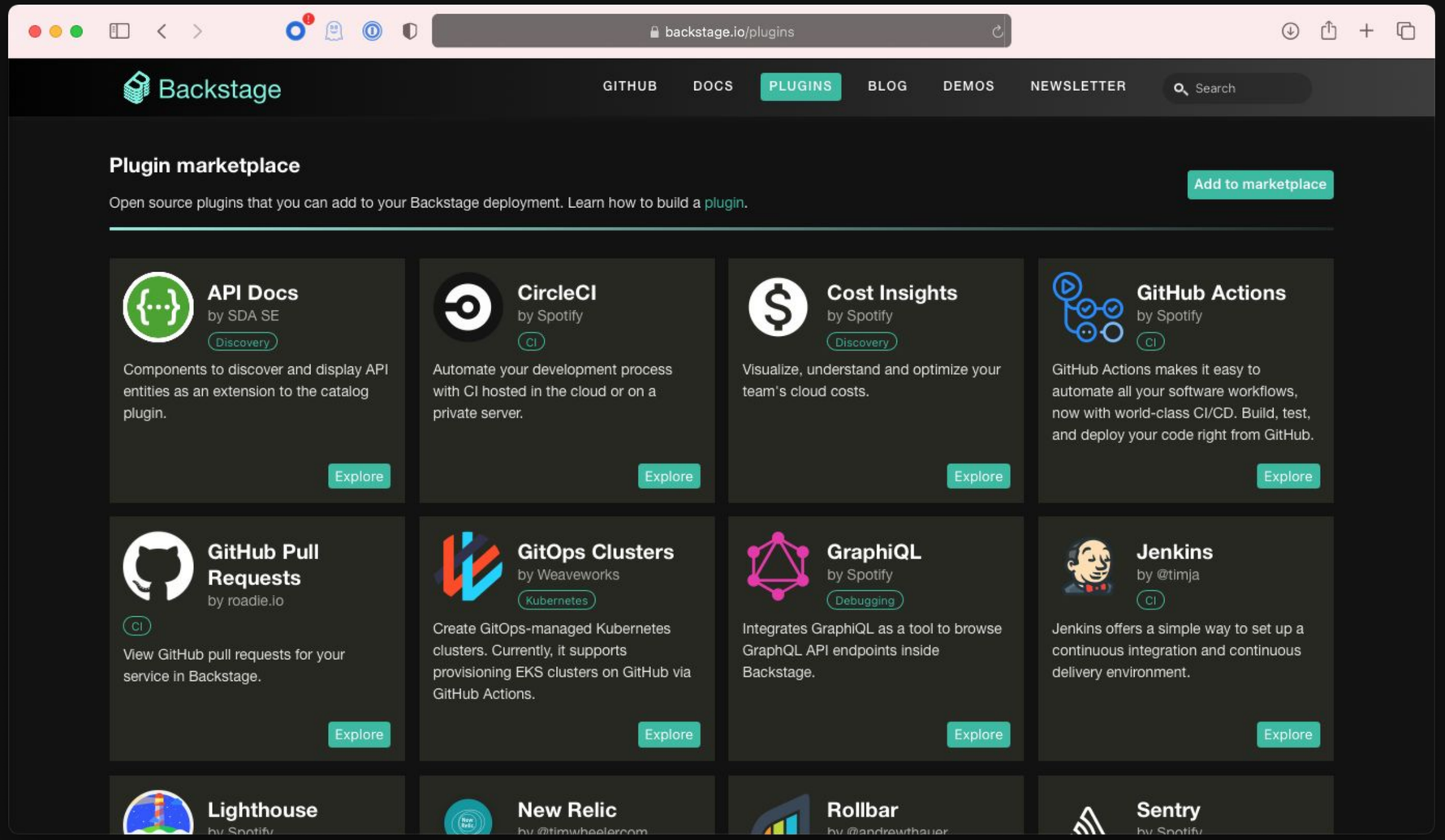


Speed

Standards

Ecosystem

Open source plugins



Core plugins

Software templates

TechDocs

Software catalog

Search

designed for…

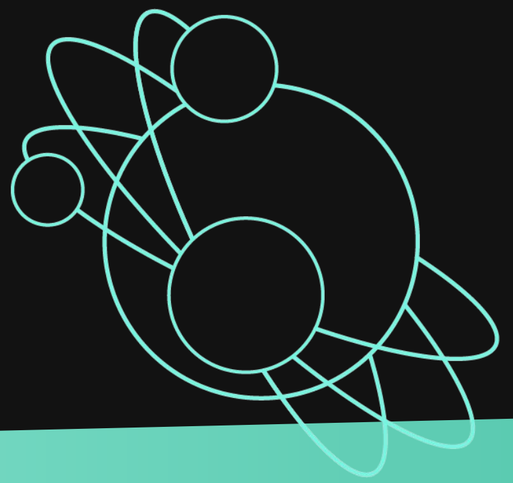
Software creation

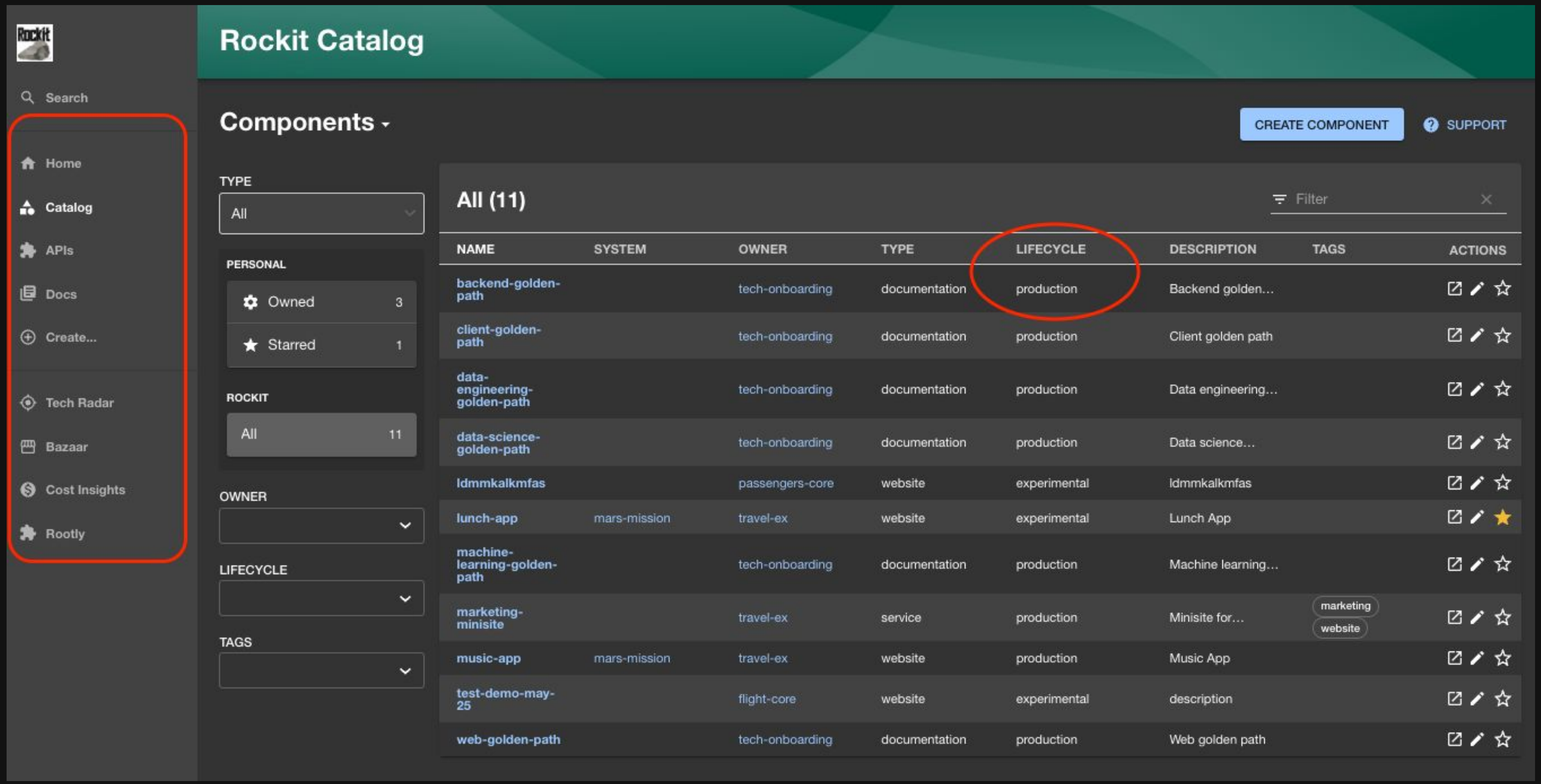
Documentation & onboarding

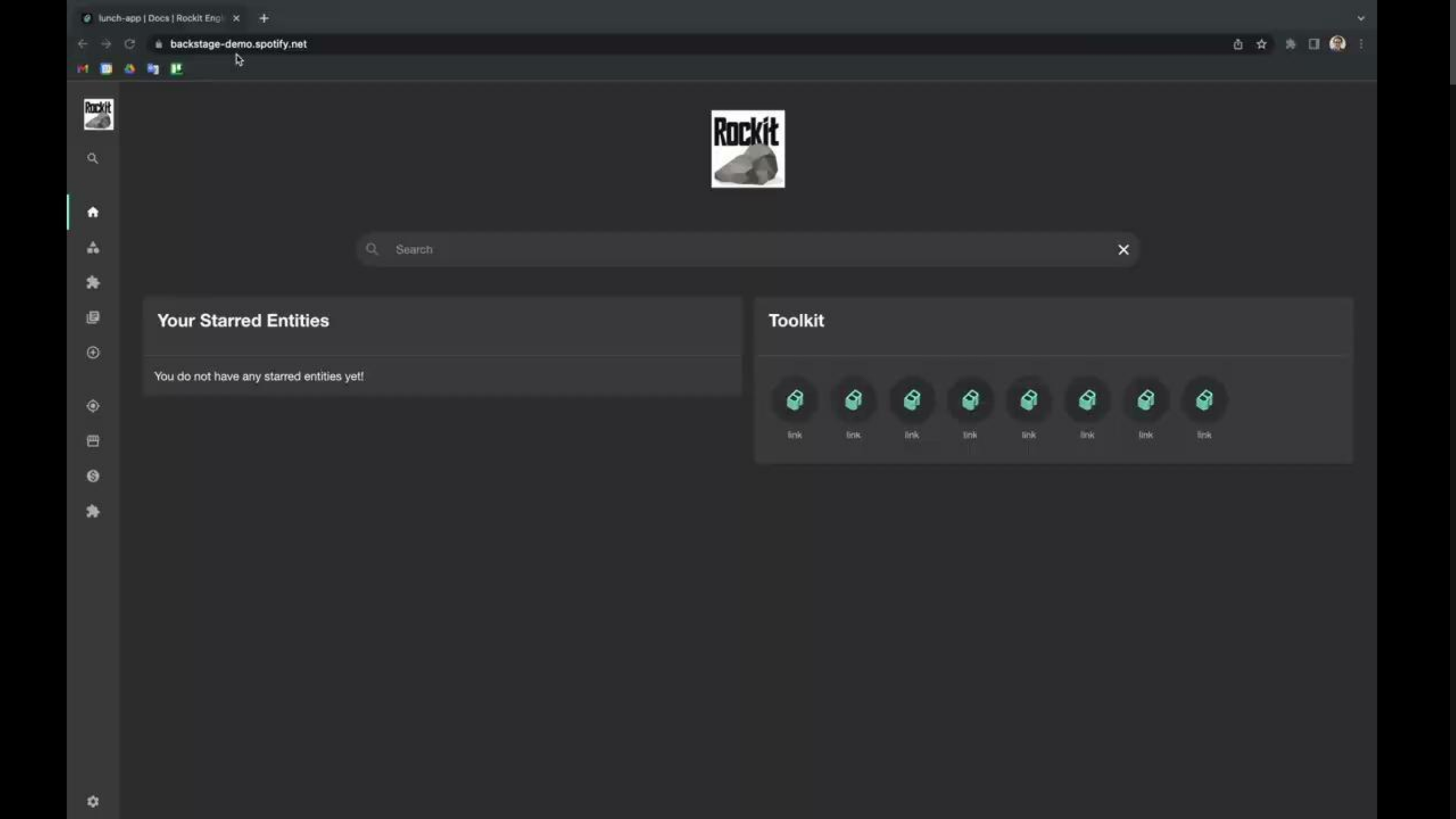
Centralization of assets

…Search 😅

Centralization of technology assets



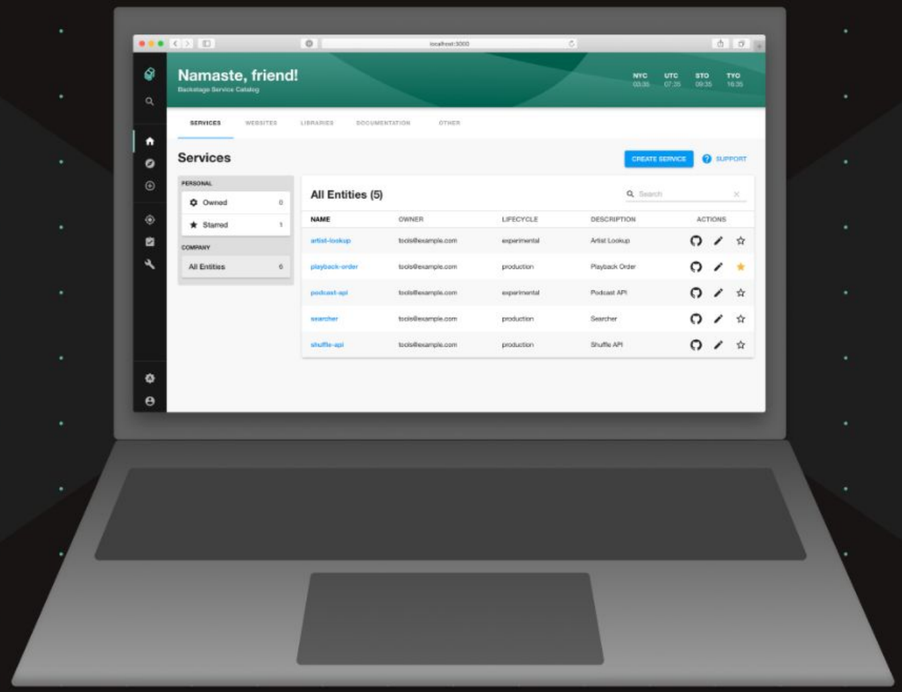

Rockit Catalog
Search
Home
Catalog
APIs
Docs
Create...
Tech Radar
Bazaar
Cost Insights
Rootly
Components
CREATE COMPONENT
SUPPORT
TYPE
All
PERSONAL
Owned 3
Starred 1
ROCKIT
All 11
OWNER
LIFECYCLE
TAGS
All (11)
Filter
NAME SYSTEM OWNER TYPE LIFECYCLE DESCRIPTION TAGS ACTIONS
backend-golden-path tech-onboarding documentation production Backend golden...
client-golden-path tech-onboarding documentation production Client golden path
data-engineering-golden-path tech-onboarding documentation production Data engineering...
data-science-golden-path tech-onboarding documentation production Data science...
ldmmkalkmfas passengers-core website experimental ldmmkalkmfas
lunch-app mars-mission travel-ex website experimental Lunch App
machine-learning-golden-path tech-onboarding documentation production Machine learning...
marketing-minisite travel-ex service production Minisite for... marketing website
music-app mars-mission travel-ex website production Music App
test-demo-may-25 flight-core website experimental description
web-golden-path tech-onboarding documentation production Web golden path


lunch-app | Docs | Rockit Eng
backstage-demo.spotify.net
Search
Your Starred Entities
You do not have any starred entities yet!
Toolkit
link
link
link
link
link
link
link
link

Backstage

```
9 lines (9 sloc)    182 Bytes

1  apiVersion: backstage.io/v1alpha1
2  kind: Component
3  metadata:
4      name: podcast-api
5      description: Podcast API
6  spec:
7      type: service
8      lifecycle: experimental
9      owner: tools@example
```



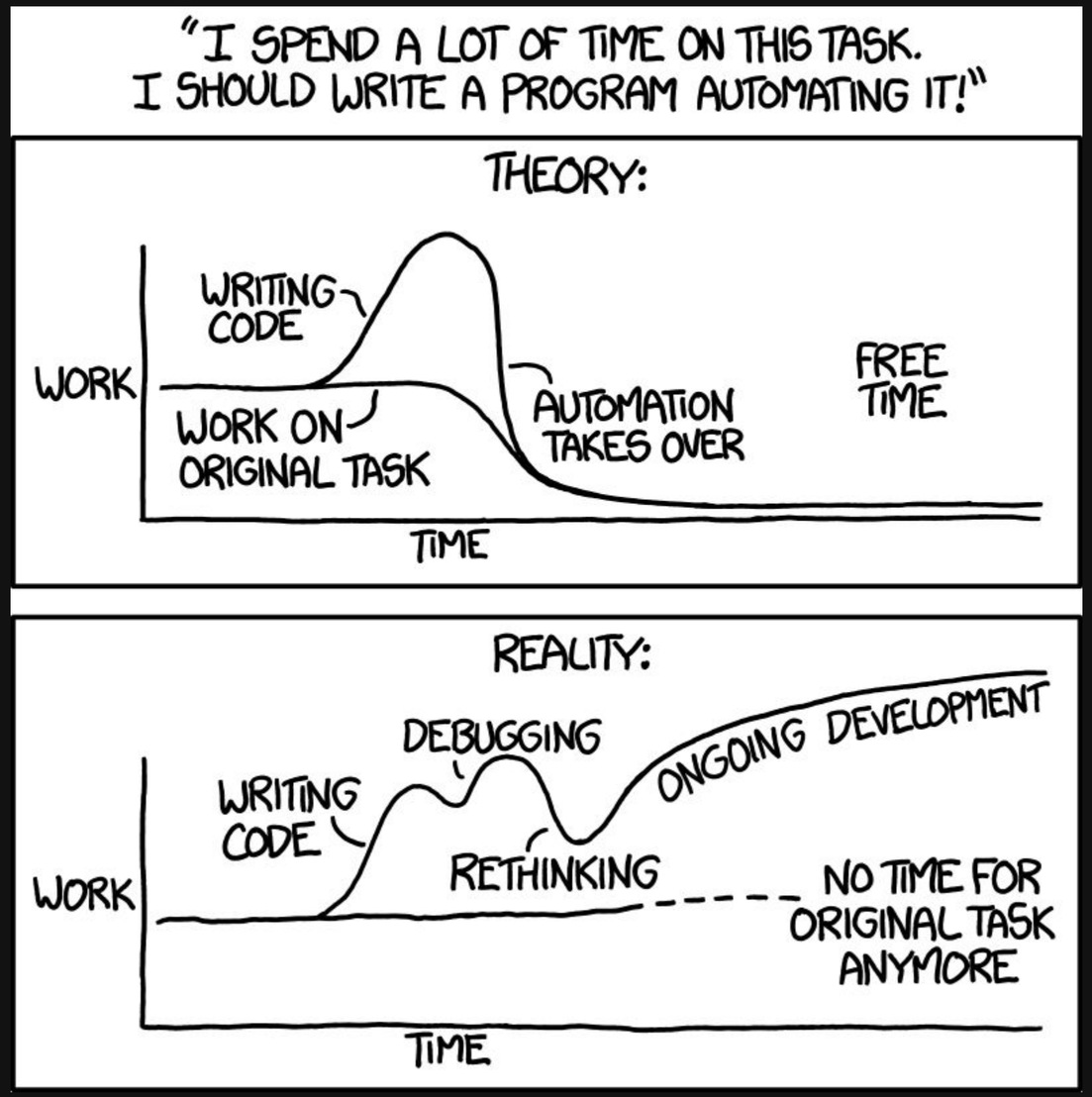
SERVICE DEFINITIONS

BACKSTAGE SERVICE CATALOG

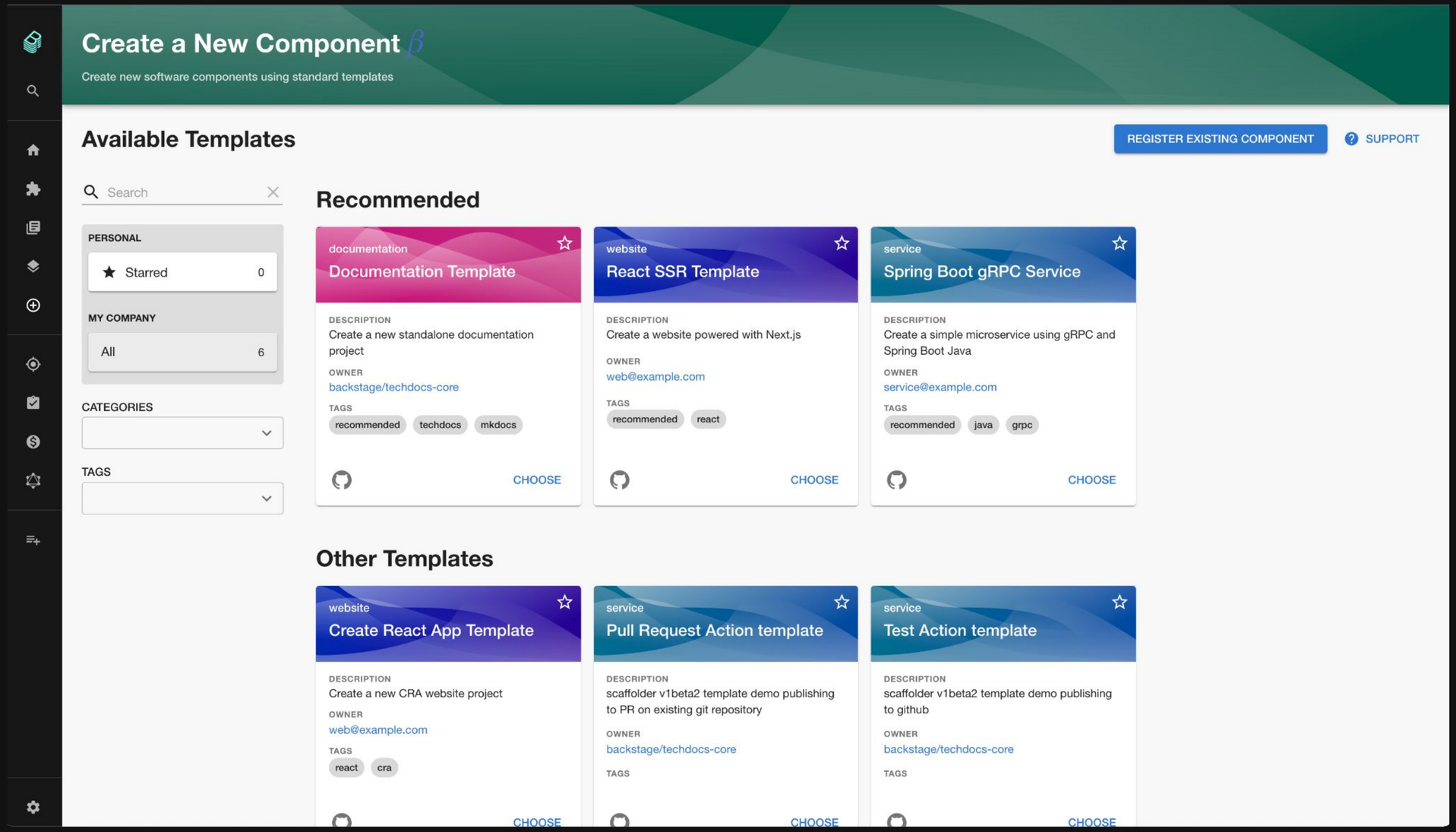
INTEGRATED TOOLING VIA PLUGINS

Example: catalog-info.yaml

```yaml
apiVersion: backstage.io/v1alpha1
kind: Component
metadata:
 name: petstore
 description: The Petstore is an example service that provides an OpenAPI spec.
 links:
   - url: https://github.com/swagger-api/swagger-petstore
     title: GitHub Repo
     icon: github
spec:
 type: service
 lifecycle: experimental
 owner: team-c
 providesApis:
   - petstore-api
```



https://xkcd.com/1319/


Create a New Component β
Create new software components using standard templates
Available Templates
REGISTER EXISTING COMPONENT
SUPPORT
Search
PERSONAL
Starred 0
MY COMPANY
All 6
CATEGORIES
TAGS
Recommended
documentation
Documentation Template
DESCRIPTION
Create a new standalone documentation project
OWNER
backstage/techdocs-core
TAGS
recommended techdocs mkdocs
CHOOSE
website
React SSR Template
DESCRIPTION
Create a website powered with Next.js
OWNER
web@example.com
TAGS
recommended react
CHOOSE
service
Spring Boot gRPC Service
DESCRIPTION
Create a simple microservice using gRPC and Spring Boot Java
OWNER
service@example.com
TAGS
recommended java grpc
CHOOSE
Other Templates
website
Create React App Template
DESCRIPTION
Create a new CRA website project
OWNER
web@example.com
TAGS
react cra
CHOOSE
service
Pull Request Action template
DESCRIPTION
scaffolder v1beta2 template demo publishing to PR on existing git repository
OWNER
backstage/techdocs-core
TAGS
CHOOSE
service
Test Action template
DESCRIPTION
scaffolder v1beta2 template demo publishing to github
OWNER
backstage/techdocs-core
TAGS
CHOOSE

Create a new component

Create new software components using standard templates in your organization

Available Templates

SUPPORT

Search

PERSONAL

Starred 0

SPOTIFY

All 54

CATEGORIES

TAGS

Golden Paths

t-swift
Backend Service Blueprint

Golden Path

Backend Service Blueprint. The best choice for consumer-facing services and APIs. Includes support for gRPC, HTTP, deployment with Kubernetes, logging, metrics and Envoy sidecar for transcoding, and optionel Edge Proxy and CNAME.

CHOOSE

flatmap-squad
Styx Scheduled Scio Pipeline

Golden Path

Main template for Scio pipelines. Please consider using Styx Scheduled Scio Pipeline (Flyte) (docs), which will eventually replace this template.

CHOOSE

t-swift
Website Blueprint

Golden Path

Next.js application skeleton for creating isomorphic web applications, includes optional Edge Proxy, Web Enrichment Platform and CNAME.

CHOOSE

Backend Service

kingpin
Continuum Section Provider

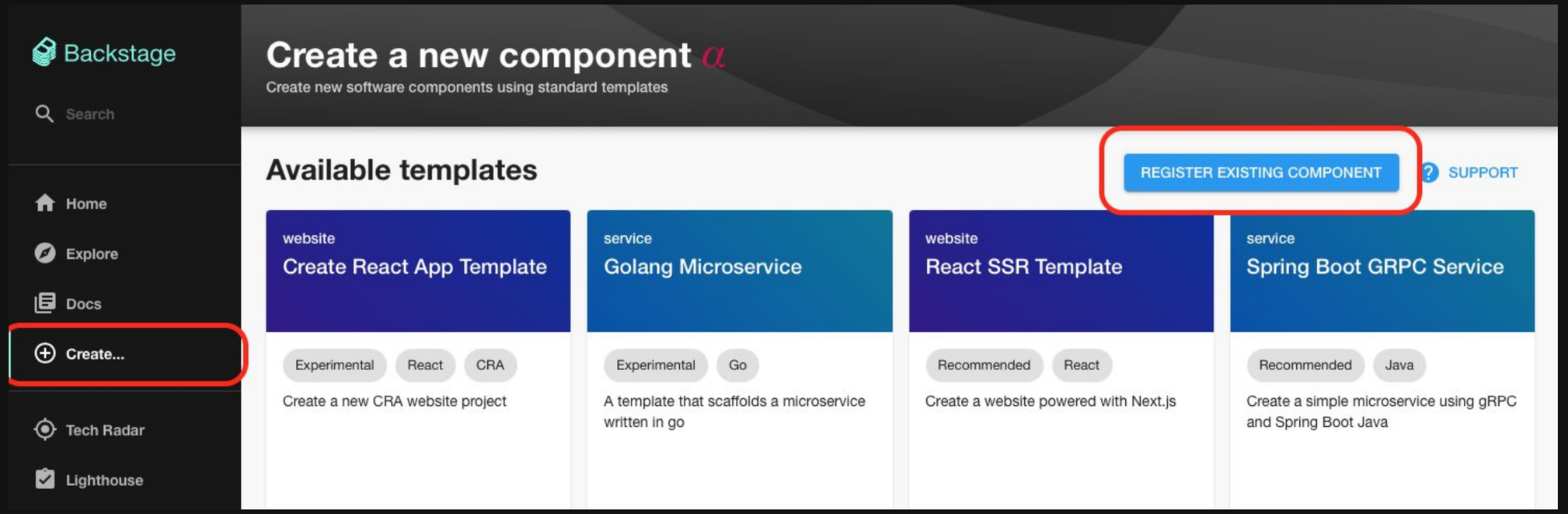
Create a backend Continuum Section Provider service deployed on Kubernetes.

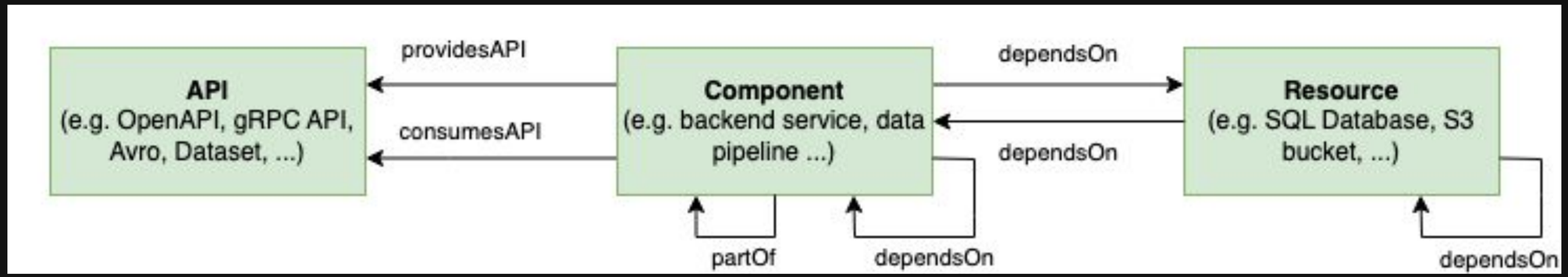
nibbler
Web Enrichment Platform Enricher

A project template for an external enricher using Apollo. For use with web-verticals that run on the Web Enrichment Platform

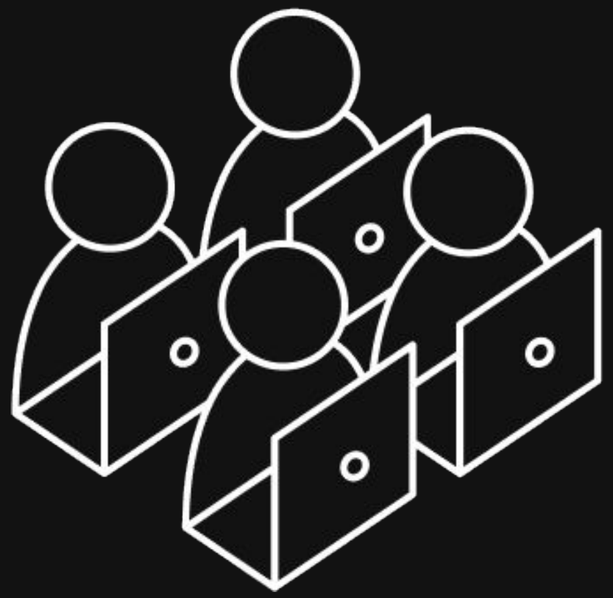
python
Python REST Fast API

Template for Python backend REST API built on FastAPI.


Backstage
Search
Home
Explore
Docs
Create...
Tech Radar
Lighthouse
Create a new component α
Create new software components using standard templates
Available templates
REGISTER EXISTING COMPONENT
SUPPORT
website
Create React App Template
Experimental
React
CRA
Create a new CRA website project
service
Golang Microservice
Experimental
Go
A template that scaffolds a microservice written in go
website
React SSR Template
Recommended
React
Create a website powered with Next.js
service
Spring Boot GRPC Service
Recommended
Java
Create a simple microservice using gRPC and Spring Boot Java

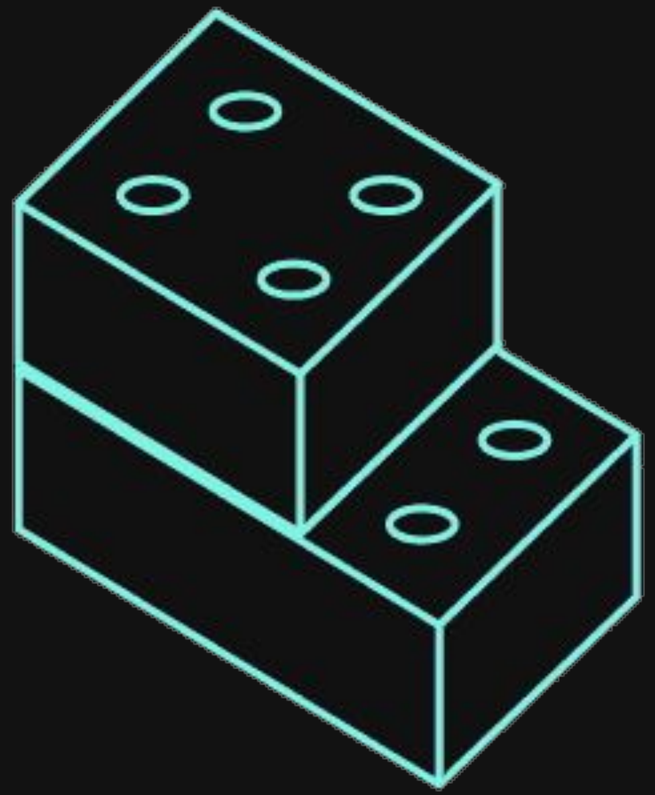

API
(e.g. OpenAPI, gRPC API, Avro, Dataset, ...)
providesAPI
consumesAPI
Component
(e.g. backend service, data pipeline ...)
dependsOn
dependsOn
Resource
(e.g. SQL Database, S3 bucket, ...)
partOf
dependsOn
dependsOn

User



Group

Backstage



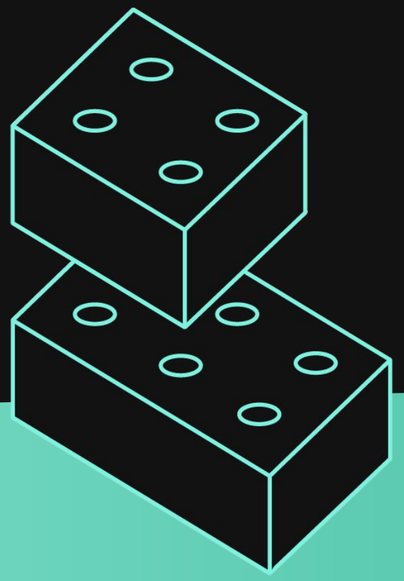
Core plugins:

- Software templates
- TechDocs
- Software catalog

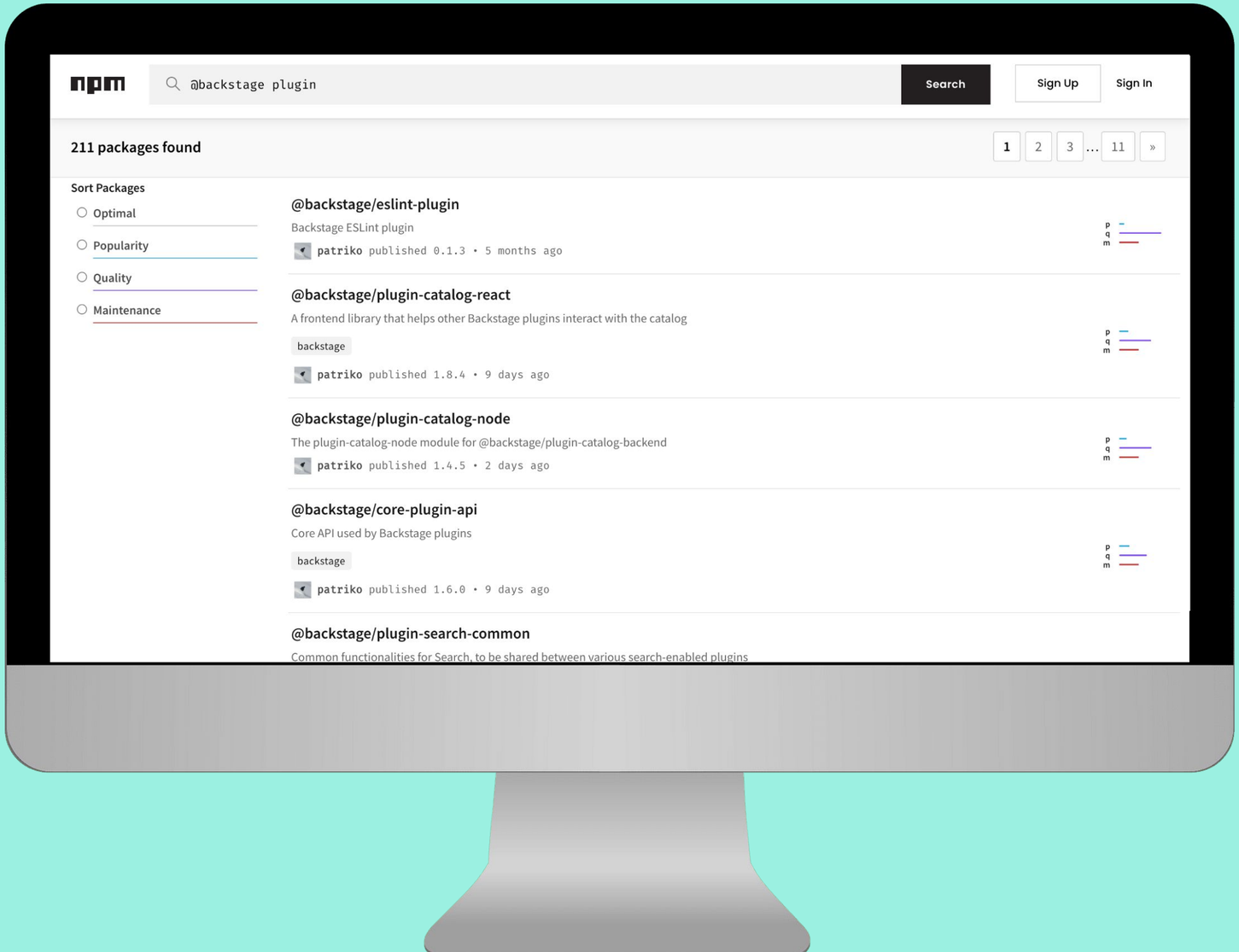
And so many others…

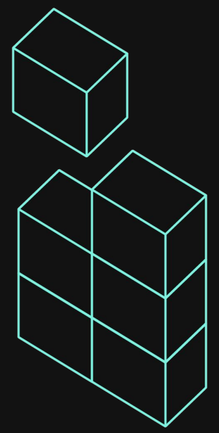
- Search
- Kubernetes
- PagerDuty
- Stack Overflow
- AWS
- Google Analytics
- CircleCI
- Jenkins
- GraphiQL
- The list goes on!

What is a plugin, really?

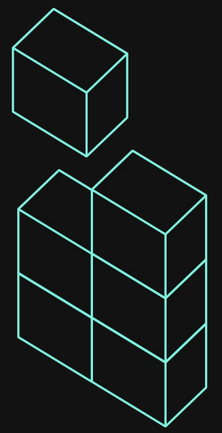


npm install @backstage/plugin-awesomeness





@backstage/plugin-lumon

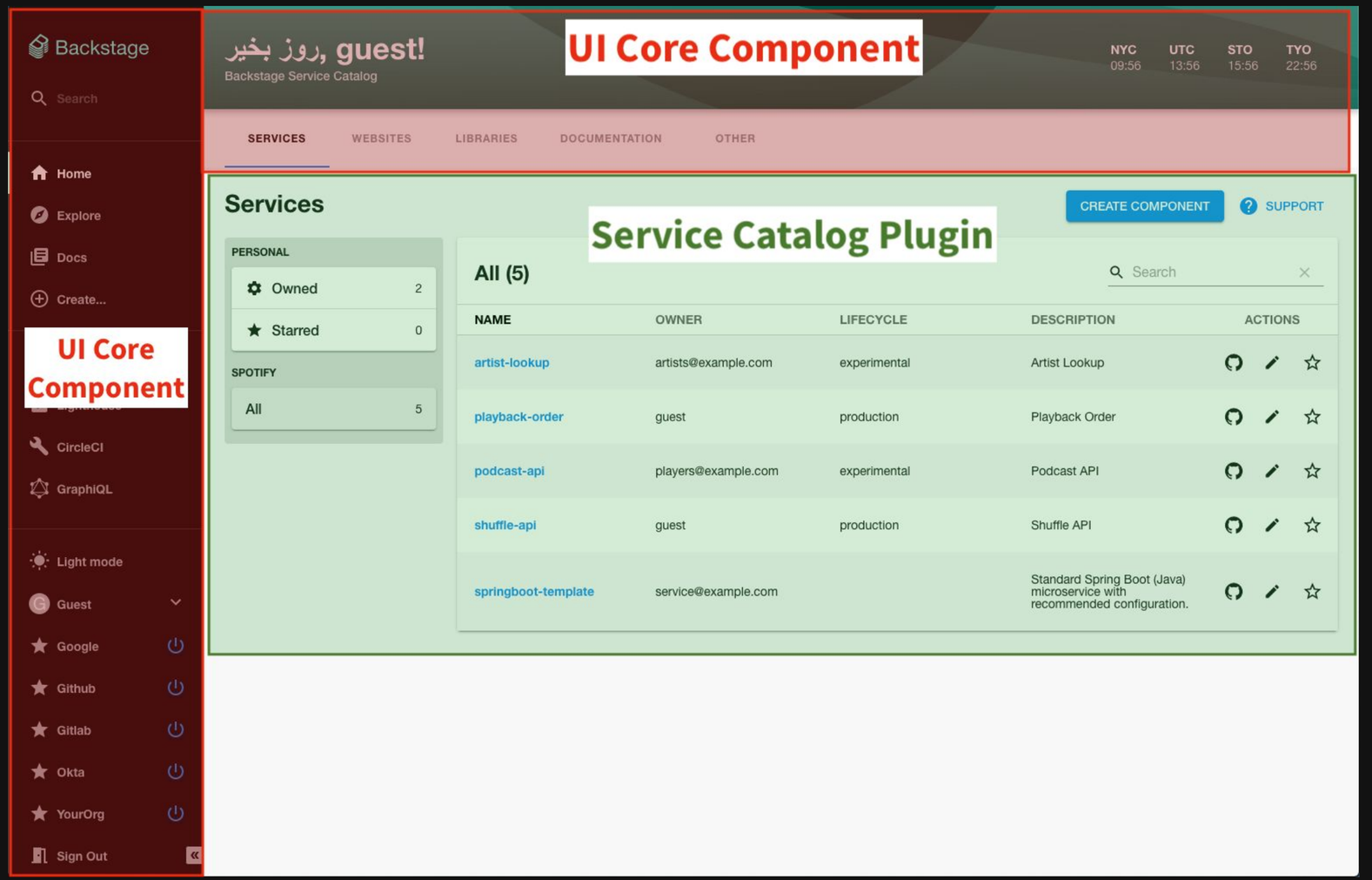


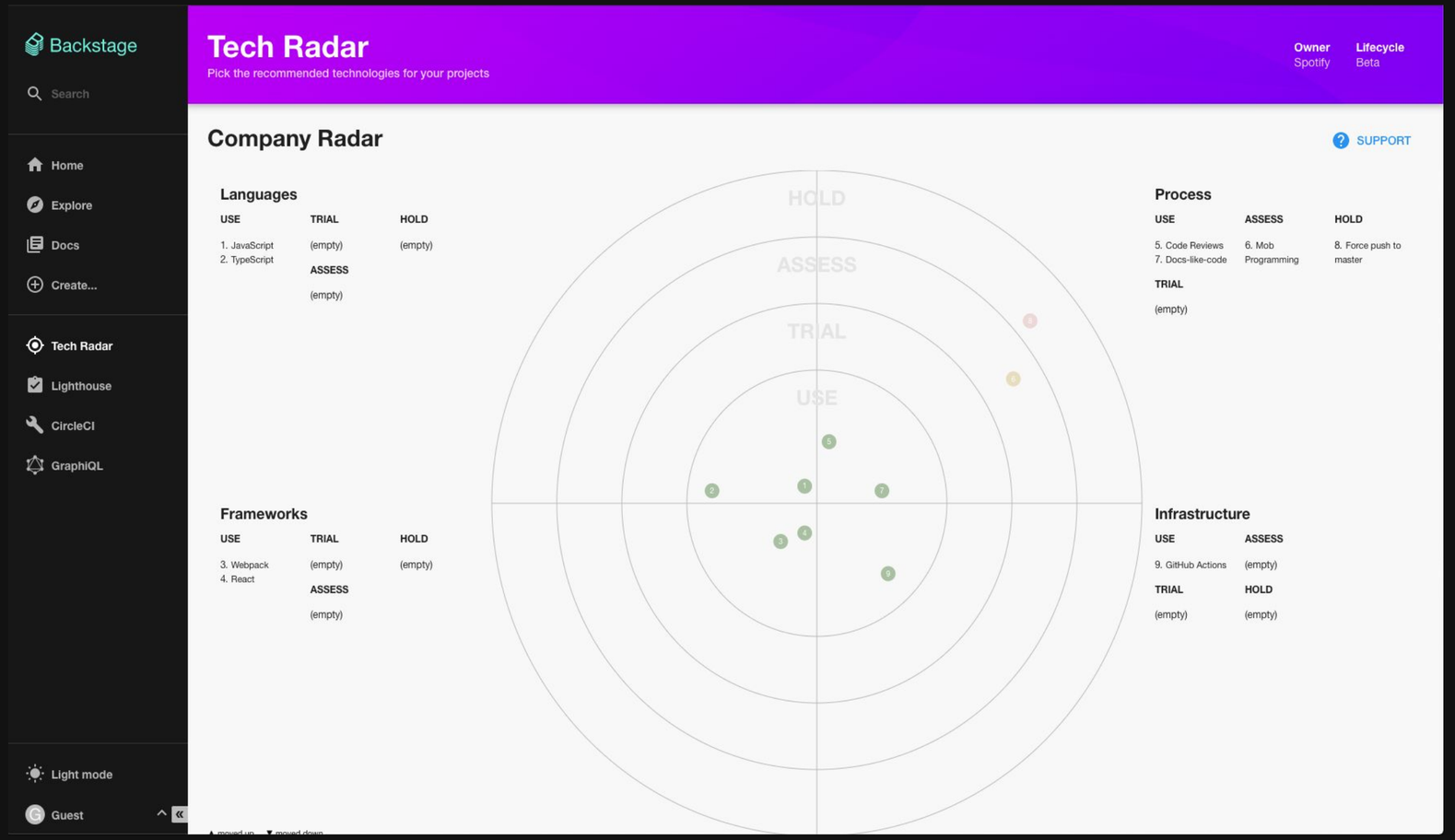
@backstage/plugin-lumon-backend

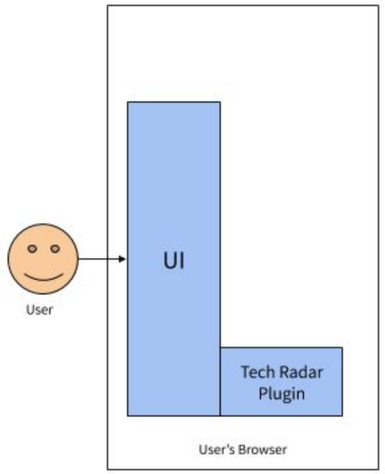
@backstage/plugin-lumon-react

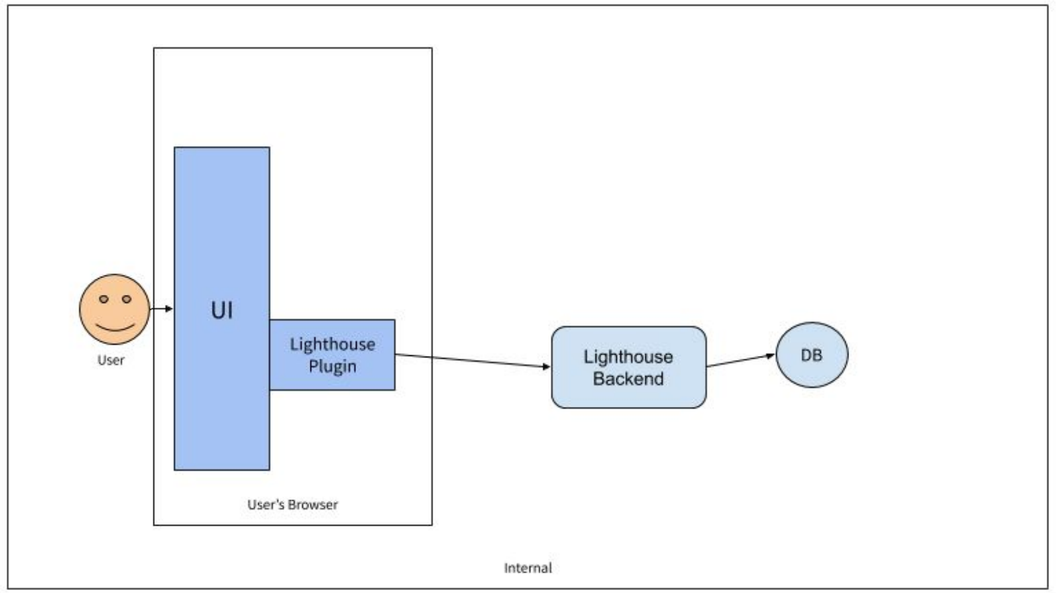
@backstage/plugin-lumon-common

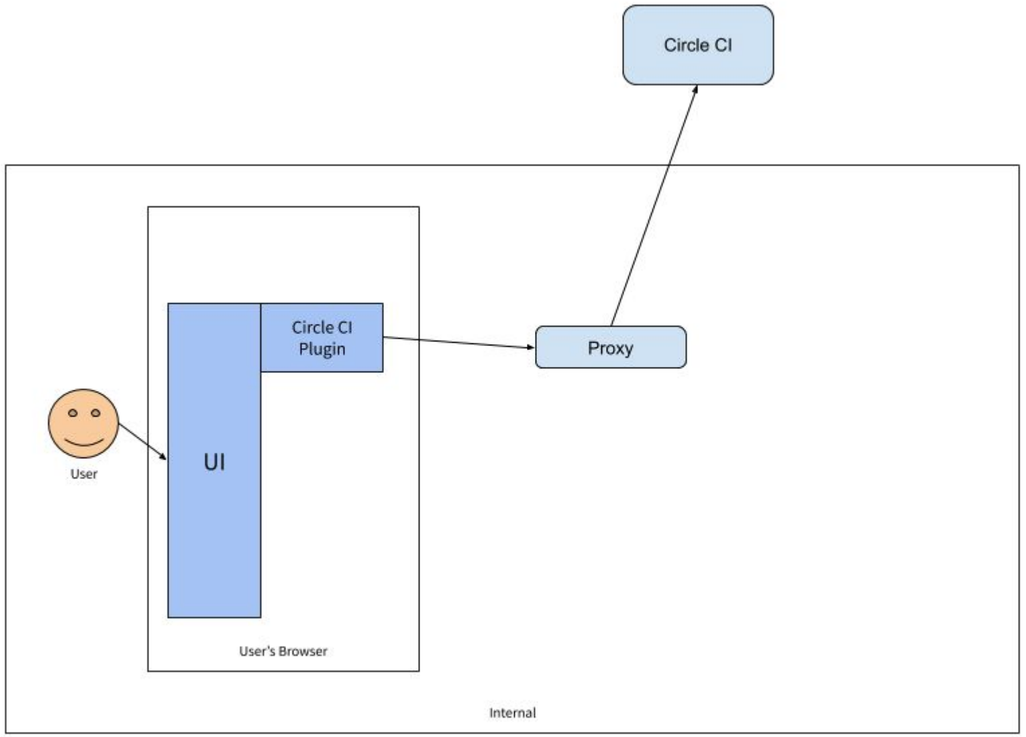
@backstage/plugin-lumon-node


Backstage
Search
Home
Explore
Docs
Create...
UI Core Component
CircleCI
GraphiQL
Light mode
Guest
Google
Github
Gitlab
Okta
YourOrg
Sign Out
روز بخیر, guest!
Backstage Service Catalog
UI Core Component
NYC 09:56
UTC 13:56
STO 15:56
TYO 22:56
SERVICES
WEBSITES
LIBRARIES
DOCUMENTATION
OTHER
Services
Service Catalog Plugin
CREATE COMPONENT
SUPPORT
PERSONAL
Owned 2
Starred 0
SPOTIFY
All 5
All (5)
Search
NAME
OWNER
LIFECYCLE
DESCRIPTION
ACTIONS
artist-lookup
artists@example.com
experimental
Artist Lookup
playback-order
guest
production
Playback Order
podcast-api
players@example.com
experimental
Podcast API
shuffle-api
guest
production
Shuffle API
springboot-template
service@example.com
Standard Spring Boot (Java) microservice with recommended configuration.


Backstage
Search
Home
Explore
Docs
Create...
Tech Radar
Lighthouse
CircleCI
GraphiQL
Light mode
Guest
Tech Radar
Pick the recommended technologies for your projects
Owner
Spotify
Lifecycle
Beta
Company Radar
SUPPORT
Languages
USE
1. JavaScript
2. TypeScript
TRIAL
(empty)
ASSESS
(empty)
HOLD
(empty)
HOLD
ASSESS
TRIAL
USE
Process
USE
5. Code Reviews
7. Docs-like-code
ASSESS
6. Mob Programming
HOLD
8. Force push to master
TRIAL
(empty)
Frameworks
USE
3. Webpack
4. React
TRIAL
(empty)
ASSESS
(empty)
HOLD
(empty)
Infrastructure
USE
9. GitHub Actions
ASSESS
(empty)
TRIAL
(empty)
HOLD
(empty)


User
UI
Tech Radar
Plugin
User's Browser


User
UI
Lighthouse Plugin
User's Browser
Lighthouse Backend
DB
Internal


Circle CI
Circle CI
Plugin
Proxy
UI
User
User's Browser
Internal

What we've covered so far…

- ✅ What is Backstage?
- ✅ Software catalog
- ✅ Plugin ecosystem

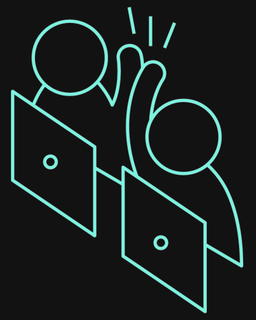
Open Source

Commercial

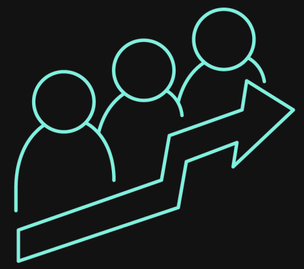
Effectiveness = *f* (Productivity, Happiness)

RBAC

Soundcheck

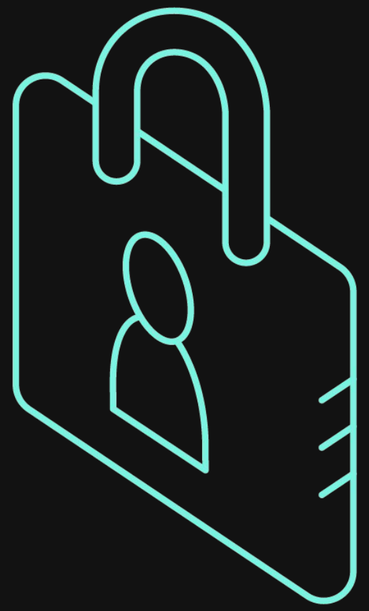


Skill Exchange



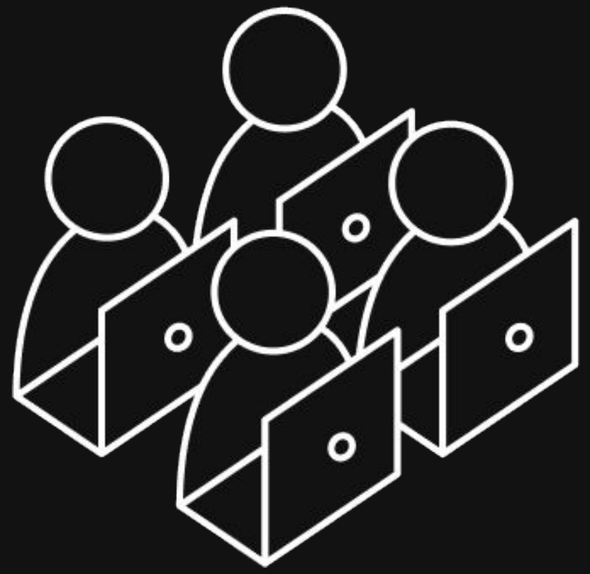
Pulse

Insights

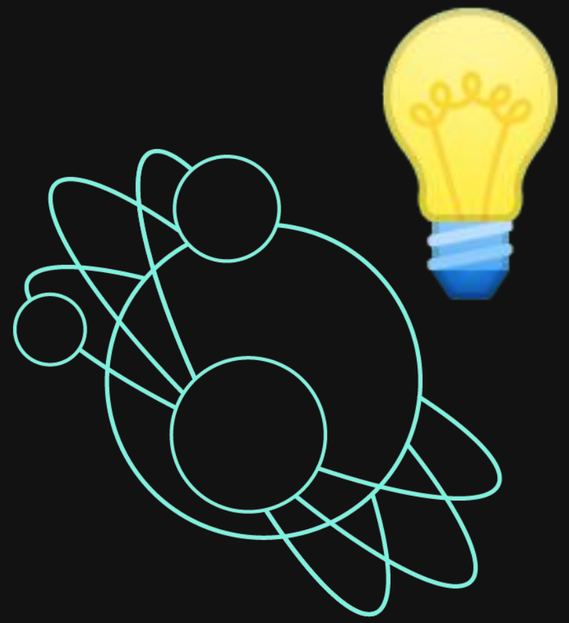


RBAC

Provide Backstage administrators a way to configure user access to entities and features within Backstage.



Open source



Software catalog

[PRFC] Backstage permissions and authorization #7761

Closed **timbonicus** wants to merge 33 commits into `master` from `mob/authorization-framework`

Conversation 10 · Commits 33 · Checks 0 · Files changed 118

timbonicus commented on Oct 25, 2021 • edited · Member

Status: Closed for comments on 2021-11-12

Need

Backstage has many different **authentication** methods, but has no built-in way to **authorize** access to specific data, APIs, or interface actions. Authorization is a way to enforce rules about what actions are allowed for a given user of a system. In Backstage today, endpoints are not protected and all actions are available to anyone.

Authorization is a common need, and can be a blocker for some organizations wishing to adopt Backstage. This capability is frequently requested by the community (#5679, #4616, #3218, #6316, to name a few).

Backstage adopters have a broad range of authorization requirements. Backstage's power is its flexibility; any authorization system should allow for many different authorization methods. This could include implementations like role-based access control (RBAC), attribute-based access control (ABAC), bespoke logic expressed in code, or integrations with external authorization providers.

Proposal

Note: the changes proposed in this PRFC are intended as a proof-of-concept, and as such aren't ready for release, and aren't intended to be merged to master in their current state.

We propose introducing an authorization *framework* in Backstage which enables:

- **Integrators** to choose the method used to authorize access to actions and data.
- **Plugin authors** to independently add permissions and authorization support in their plugins.
- **Contributors** to implement and share new authorization methods for integrators to use.

Backstage should have a way to create permissions, optionally relating to a specific resource (such as a catalog entity), and a means to verify a given user's ability to perform the associated action. This method of authorization should be easily customizable.

These permissions may include, for example:

- Permission to view, add, or unregister entities in the Backstage software catalog
- Permission to visit a certain route in the Backstage app
- Permission to execute a certain action for an entity (such as triggering a CI build)

The set of permissions must be extensible by plugins, to allow fine-grained authorization; yet integrators should be able to set sensible, broad defaults *without* needing explicit configuration for each new plugin installed. It should also be possible to provide novel authorization methods, such as integrating with external authorization providers.

Protection of backend data and operations is the primary goal, but there should also be a way to check permissions in the frontend – such as to hide a button or sidebar item that the user is not allowed to use.

For adopters: Adding authorization to a Backstage app

▶ Click to expand...

For plugin authors: Adding authorization to a Backstage plugin

▶ Click to expand...

Appendix

▶ Click to expand...

Next steps

We're excited to get feedback on this proposal, and encourage folks to grab the branch and try it out! We'd love to hear in particular about any use cases this framework doesn't support. **This PRFC will be open until Friday, November 5th**. In the meantime, we'll be working on getting the new packages proposed here ready to merge into master in separate pull requests.

```ts
/**
 * Generic type for building {@link Permission} types.
 * @public
 */
export type PermissionBase<TType extends string, TFields extends object> = {
  /**
   * The name of the permission.
   */
  name: string;
  /**
   * {@link PermissionAttributes} which describe characteristics of the permission, to help
   * policy authors make consistent decisions for similar permissions without referring to them
   * all by name.
   */
  attributes: PermissionAttributes;
} & {
  /**
   * String value indicating the type of the permission (e.g. 'basic',
   * 'resource'). The allowed authorization flows in the permission system
   * depend on the type. For example, a `resourceRef` should only be provided
   * when authorizing permissions of type 'resource'.
   */
  type: TType; // Property appears on separate object to prevent expansion of Permission types in api reports.
} & TFields;
```

```ts
/**
 * ResourcePermissions are {@link Permission}s that can be authorized based on
 * characteristics of a resource such a catalog entity.
 * @public
 */
export type ResourcePermission<TResourceType extends string = string> =
  PermissionBase<
    'resource',
    {
      /**
       * Denotes the type of the resource whose resourceRef should be passed when
       * authorizing.
       */
      resourceType: TResourceType;
    }
  >;
```

```ts
/**
 * A policy to evaluate authorization requests for any permissioned action performed in Backstage.
 *
 * @remarks
 *
 * This takes as input a permission and an optional Backstage identity, and should return ALLOW if
 * the user is permitted to execute that action; otherwise DENY. For permissions relating to
 * resources, such a catalog entities, a conditional response can also be returned. This states
 * that the action is allowed if the conditions provided hold true.
 *
 * Conditions are a rule, and parameters to evaluate against that rule. For example, the rule might
 * be `isOwner` and the parameters a collection of entityRefs; if one of the entityRefs matches
 * the `owner` field on a catalog entity, this would resolve to ALLOW.
 *
 * @public
 */
export interface PermissionPolicy {
  handle(
    request: PolicyQuery,
    user?: BackstageIdentityResponse,
  ): Promise<PolicyDecision>;
}
```

Role Based Access Control

RBAC Policies

IMPORT

Backstage policy

Active

"Version 2"

Details

Published on 29 Nov 2022, 14:19:31

Published by user:default/joonpark13

DUPLICATE

Backstage policy - Draft

Details

Updated on 29 Nov 2022, 14:20:59

Updated by user:default/joonpark13

EDIT DELETE DRAFT PUBLISH

Previous Versions

Filter

NAME	STATUS	PUBLISH DATE	PUBLISHED BY	ACTIONS
Backstage policy	Inactive	29 Nov 2022, 14:19:05	user:default/joonpark13	
Allow all	Inactive	29 Nov 2022, 14:18:43	user:default/joonpark13	
Default policy	Inactive	29 Nov 2022, 14:17:38	Default	

Role Based Access Control

Home / Version 4 / *Catalog Read-Only*

Version 4 | Catalog Read-Only

BACK TO POLICY

Members

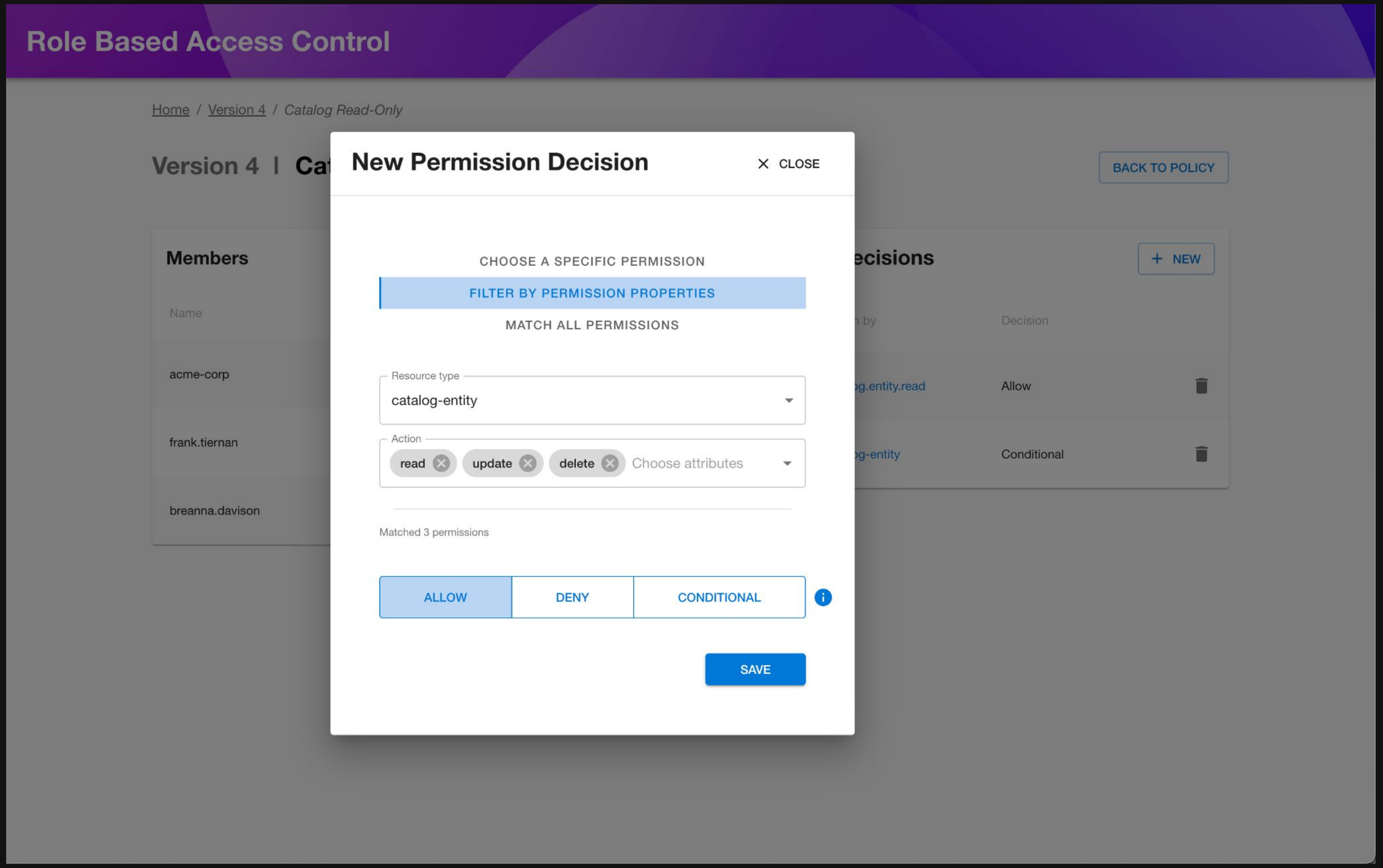
Select specific users/groups

Name	Type
acme-corp	group
frank.tiernan	user
breanna.davison	user

Permission Decisions

+ NEW

Reorder	Match by	Decision
	catalog.entity.read	Allow
	catalog-entity	Conditional


Role Based Access Control
Home / Version 4 / Catalog Read-Only
Version 4 | Cat
BACK TO POLICY
Members
Name
acme-corp
frank.tiernan
breanna.davison
ecisions
+ NEW
n by
Decision
g.entity.read
Allow
g-entity
Conditional
New Permission Decision
CLOSE
CHOOSE A SPECIFIC PERMISSION
FILTER BY PERMISSION PROPERTIES
MATCH ALL PERMISSIONS
Resource type
catalog-entity
Action
read
update
delete
Choose attributes
Matched 3 permissions
ALLOW
DENY
CONDITIONAL
SAVE

New Permission Decision

CLOSE

CHOOSE A SPECIFIC PERMISSION

FILTER BY PERMISSION PROPERTIES

MATCH ALL PERMISSIONS

Permission name: catalog.entity.read

ALLOW | DENY | CONDITIONAL

Any of

Rule: HAS_SPEC
Parameter: "lifecycle"
Parameter: "production"

Rule: IS_ENTITY_KIND
Parameter: ["user"]

ADD

SAVE

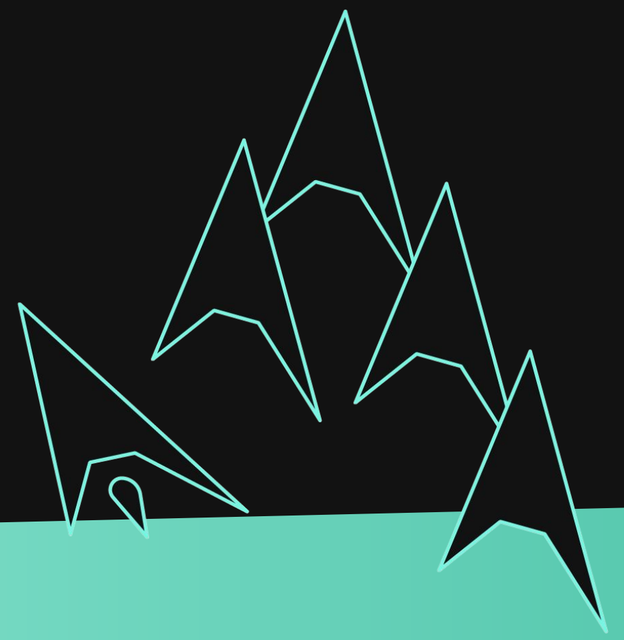
Settings

Open Source

Commercial

https://xkcd.com/1046/

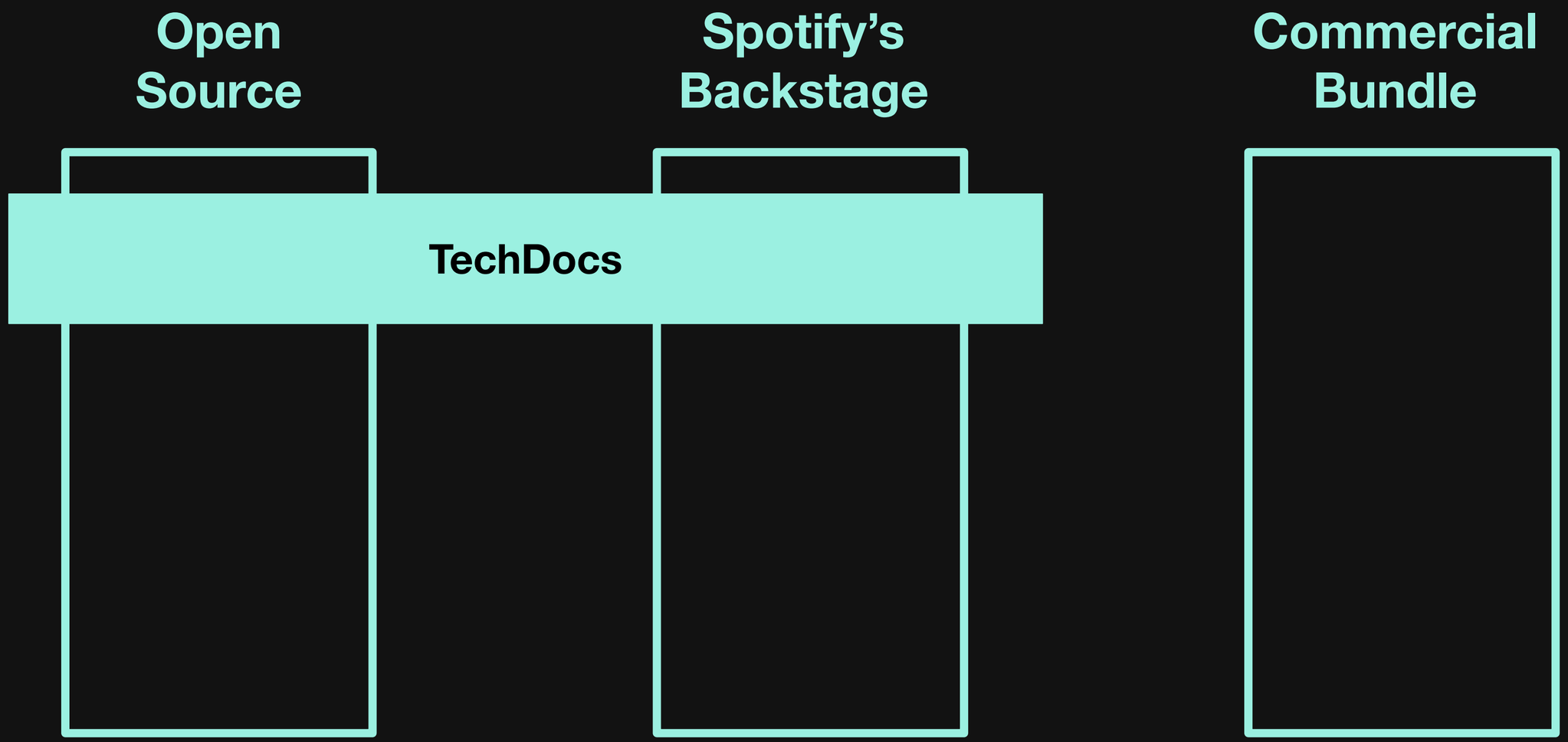
How do we enable all this?

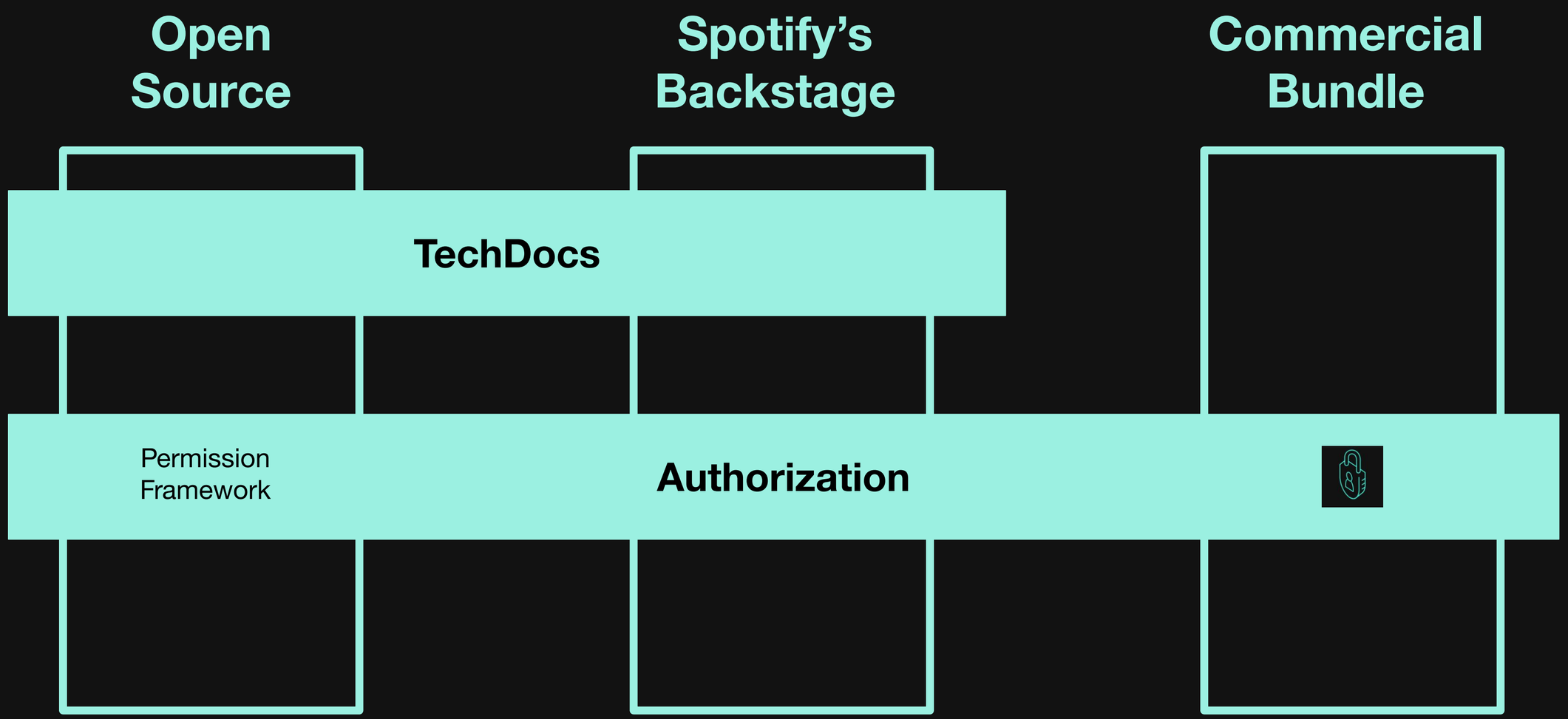


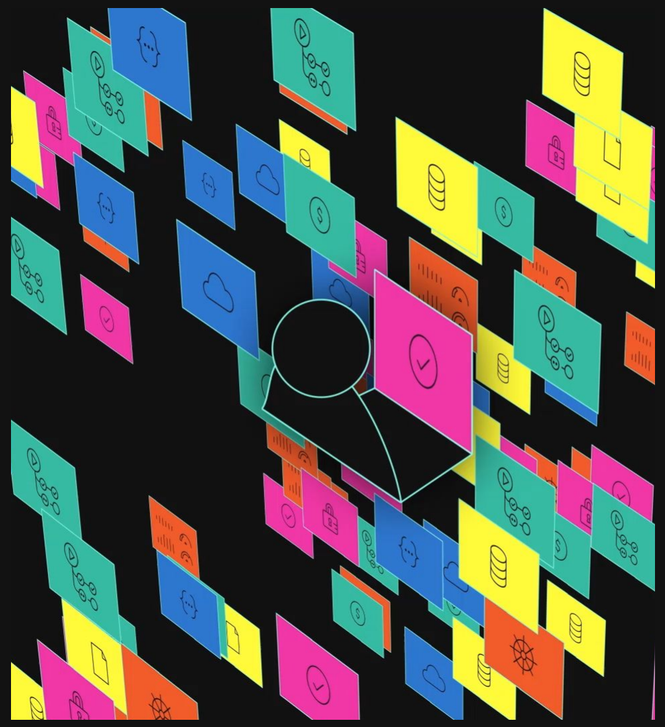
Open Source

Spotify's Backstage

Commercial Bundle


Open Source
Spotify's Backstage
Commercial Bundle
TechDocs


Open Source
Spotify's Backstage
Commercial Bundle
TechDocs
Permission Framework
Authorization



What have we learned?

Keeping up with rapid community growth is hard!

The flexibility of the plugin ecosystem is amazing! (But onboarding is a bit hard)

It takes conscious effort, but it *is* possible for teams to execute in a way that aligns incentives.

Backstage

Join the **Backstage** community

https://backstage.io

https://backstage.spotify.com

goto;

Don't forget to
rate this session
in the **GOTO Guide app**