

SEPTEMBER 2022

# Self-Service as a Design Principle

How two product people approached a Backstage implementation at Anaplan.

# We'll cut to the chase. **What's in it for you?**



## **Learn our lessons**

- How we embodied self-service across the full value stream
- How we designed for consistency at scale



## **Reduce your cognitive load**

- Skip the learning curve with a scalable Information Architecture
- Triangulate the Backstage docs, repo, Storybook, and Figma



## **Take our design assets**

- We developed these with a product mindset and self-service at their core
- Improve consistency across teams that are developing for Backstage

# Hello!

---



**Rob Kingscote (he/him)**  
Product Management, Anaplan



**Erin Ahmed (she/her)**  
Experience Design, Slalom

# The problem space



## 38 Engineering teams

38 teams looking after 102 services, which have all been delivered slightly differently over the years.



## 6 Platform teams

6 platform teams (Developer Experience, CI, CD, Runtime, etc.) looking after 48 enabling services.



## 1 Product manager

One PM for 6 teams with no permanent dedicated design support.

# Guiding policies

---



**Self serve where possible.**



**Minimize cognitive load.**



**Create the thinnest viable platform.**



**Deliver only what is needed.**

# Goals



**01**

## **Reduce cognitive load**

A single consistent interface for all infrastructure tooling, services and documentation.

Developers can discover, engage and operate tooling built to enable increased software delivery productivity.

**02**

## **Improve discoverability**

An easy-to-navigate outline of Anaplan's estate taxonomy, tooling, APIs, and docs.

An easy way to find out what we have, what it does, and who to speak to learn more.

**03**

## **Simplify onboarding**

Help developers hit the ground running and build products faster, both for new joiners and the creation of new services and APIs.

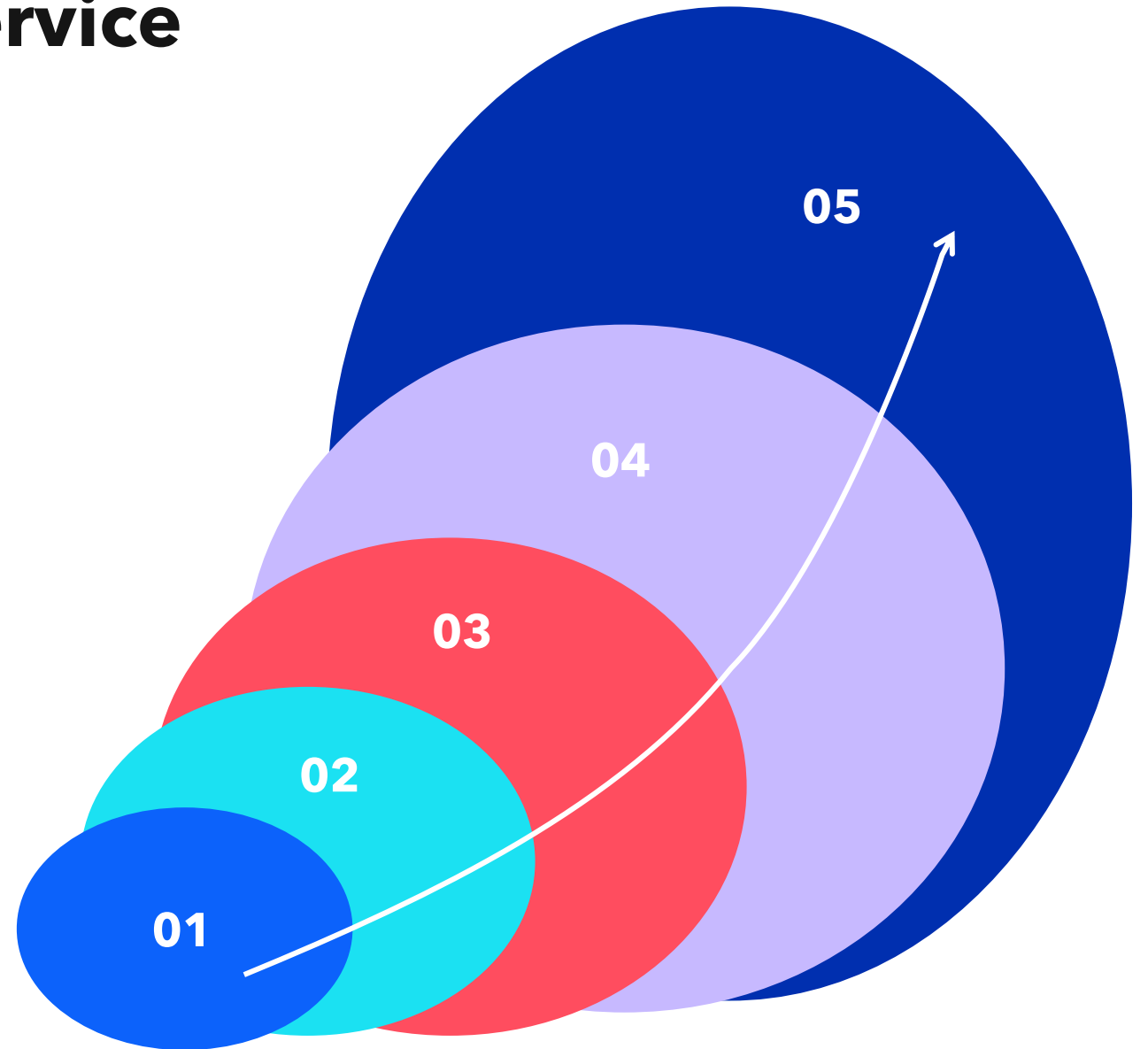
**04**

## **Accelerate time to value**

Enable the developer enablement group with capabilities to surface tooling to users in a consistent way, decreasing the time it takes to get value in front of users.

# Design centered in self-service

- 01 Product & design
- 02 Developer Experience Team
- 03 Developer Infrastructure Team
- 04 Software engineers ('c'ustomers)
- 05 The business



# Our approach



Product & design

Dev Experience Team

Dev Infra Group

Software engineers

The business

**Compliance Manager**  
I occasionally internal system the company compliance regulations

**Application or Service Developer**  
I work on a feat Anaplan, build and backend customers.

**Infrastructure Developer**  
I work in the Developer Infrastructure Group at Anaplan, building frontend and backend services for Anaplan developers.

show external exist and are

"Service teams should own their own code. It's my job to make that as easy as possible for them."

**Goals**

- Build and deploy quality code efficiently
- Solve common challenges Anaplan developers face going from PPR to prod
- Enable application and service team developers to own the lifecycle of their services with minimal operational overhead
- Provide robust, scalable, on-demand resources to own tools, envs, etc.) to enable service teams to deliver value to customers

**Frustrations**

- Significant operational inefficiencies: no scaffolding for new services and many teams solve the same problems differently
- Service teams are not enabled nor expected to own their services in production
- Current pipeline is overcomplicated and inhibiting "customer" success
- Pipeline flow is constrained due to bottlenecks, e.g. config

**LAYOUT**

HomePage

SearchPage

EntityPage

CatalogPage

ExplorePage

**PAGE TYPES**

EntityPage

CapabilitySubPage

**COMPONENTS**

**SURFACES**

InfoCard

TabbedCard

ItemCard

**DATA DISPL**

Table

Software Catalog / ... / Deploy

users-service Category 1 Descriptor 1

OVERVIEW BUILD DEPLOY MANAGE TEST RELEASE MONITOR

**CD Pipeline**  
Description of this tab

**This capability hasn't been set up yet**

To manage your deployments from Backstage, follow the steps to meet the pre-requisites and click 'Enable' to get started.

ENABLE GO TO DOCS

Personas & Use Cases



Information Architecture



Component Library



Prototypes





# Spotlight on **Personas**



**Application or  
Service Developer**



**Infrastructure  
Developer**



**SRE / DevOps  
Engineer**



Quality  
Assurance



Compliance  
Manager



InfoSec  
Analyst



Engineering  
Manager



Engineering  
Executive

# Spotlight on Information Architecture

### LAYOUT

- SignInPage
- HomePage
- SearchPage
- CatalogPage
- ExplorePage
- TechDocsCustomHome
- AuditPage
- ADDITIONAL PLUGINS

### PAGE TYPES

- EntityPage
- CapabilitySubPage

### COMPONENTS

- SURFACES:
  - InitCard
  - TabbedCard
  - ItemCard
  - AboutCard
  - OwnershipCard
- DATA DISPLAY:
  - Table
  - StructureMetadataTable
  - EntityLinkCard
  - UserProfileCard
  - GroupProfileCard
- DATA VISUALIZATION:
  - ProgressBars
  - DependencyGraph
- USER ACTION:
  - SearchBar
  - SimpleStepper

## EntityPage

<https://github.com/microsoft/backstage/tree/master/packages/app/components/entity>

### Domain

OVERVIEW

**About**

VIEW TECHNOLOGIES VIEW SOURCE

DESCRIPTION  
This line is the description for the data endpoint. Validated aggregated stream activity fact table, used for metrics.

OWNER  
user-francis.doherty

TAGS  
tag1

**Links**

- Link with icon
- Link with icon
- Link with icon
- Link with icon

### HasSystemsCard

NAME	OWNER	DESCRIPTION
infrastructure	team-a	Short system description
shared-services	team-a	Short system description
vendors	team-b	Short system description

1 TO MANY

### Component

OVERVIEW BUILD DEPLOY TEST MANAGE RELEASE MONITOR

**About**

VIEW TECHNOLOGIES VIEW SOURCE CONTACT OWNER

DESCRIPTION  
This line is the description for the data endpoint. Validated aggregated stream activity fact table, used for metrics.

OWNER  
team-a

TYPE  
Library

LIFECYCLE  
Production

TAGS  
tag1, tag2

**Relations**

```

graph TD
    A[entity] --> B[module]
    A --> C[module]
    B --> D[module]
    C --> E[module]
    
```

GO TO FULL DIAGRAM

### System

OVERVIEW

**AboutCard**

VIEW TECHNOLOGIES VIEW SOURCE

DESCRIPTION  
This line is the description for the stream activity fact table, used for metrics.

OWNER  
team-a

DOMAIN  
platform

TAGS  
tag1

**EntityLinkCard**

- Link with icon
- Link with icon
- Link with icon

### API

OVERVIEW BUILD DEPLOY

**About**

VIEW TECHNOLOGIES VIEW SOURCE

DESCRIPTION  
This line is the description for the stream activity fact table, used for metrics.

OWNER  
team-a

TYPE  
Library

TAGS  
tag1, tag2

# Spotlight on Component Library

The image displays a structured overview of a component library, organized into several key sections:

- LAYOUT**: A vertical list of page types including **SignInPage**, **HomePage**, **SearchPage**, **CatalogPage**, **ExplorePage**, **TechDocsCustomHome**, and **AuditPage**. A dashed box at the bottom indicates **ADDITIONAL PLUGINS**.
- PAGE TYPES**: A grid of various page layouts categorized into **EntityPage** and **CapabilitySubPage**.
- COMPONENTS**: A central hub for individual UI elements, divided into:
  - SURFACES**: A sub-section with a blue border highlighting **InfoCard**, **TabbedCard**, and **ItemCard**.
  - DATA DISPLAY**: Includes a **Table** and **StructureMetadataTable**.
  - DATA VISUALIZATION**: Features **ProgressBars** and a **DependencyGraph**.
  - USER ACTIONS**: Includes a **SearchBar** and a **SimpleStepper**.
- InfoCard**: A detailed view showing three variations of an information card with a "GO TO XYZ LOCATION" button.
- TabbedCard**: A detailed view showing two variations of a tabbed card with four tabs labeled "Label".
- ItemCard**: A detailed view showing an item card with a blue header, subtitle, and two "Chip" components.



# Thank you

Personas, Information Architecture, and Component Library can be found in the **Backstage Discord #visual-design channel**.