CLOUD NATIVE
COMMUNITY GROUPS

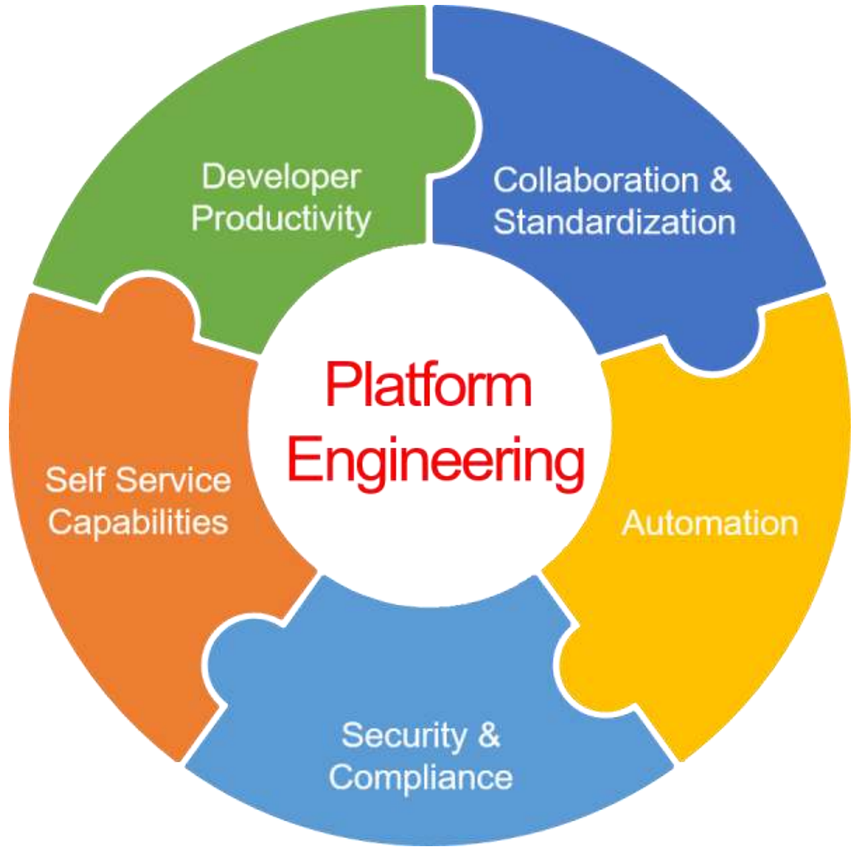
Platform Engineering : Backstage & Crossplane

Chidambaram K
Senior Engineer, Presidio

Chandra Mohan S
Associate Engineer, Presidio

Agenda

1. Understanding Roles
2. What is Platform Engineering
3. Backstage & its Terminology
4. Crossplane & its Terminology
5. Demo
6. Tools Landscape



Understanding Roles

DevOps

DevOps focuses on breaking down silos between development and operations teams, fostering collaboration, and automating workflows for faster software delivery.

SRE

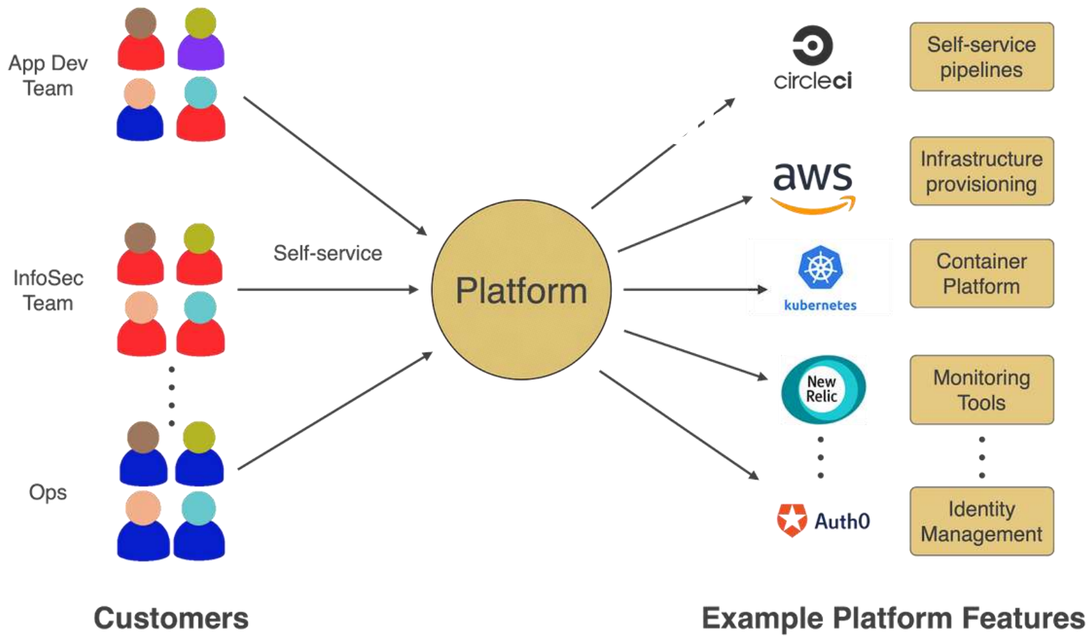
SRE builds upon DevOps principles, emphasizing reliability, incident management, and service level agreements (SLAs).

Platform

Platform engineers create and maintain internal developer platforms (IDPs) that empower developers with self-service tools and resources.

What is Platform Engineering

discipline of designing and building toolchains and workflows that enable self-service capabilities for software engineering organizations in the cloud-native era.



Backstage

Backstage is an open source framework for building developer portals. Powered by a centralized software catalog.

At its core, It’s a platform for organizing infrastructure, tools, and documentation in a single interface, giving developers full visibility into their environment. It allows teams to focus on building features instead of getting bogged down by infrastructure complexity.

Developer portal by Spotify.

Problems

Fragmented Tooling Landscape

- Teams often use diverse tools (CI/CD, monitoring, logging, etc.).
- Lack of integration between tools leads to inefficiency.
- Developers waste time switching between tools and manually managing services.

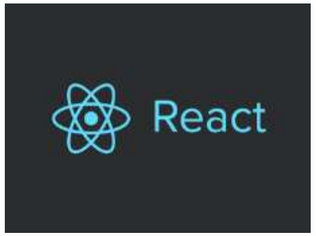
Lack of Service Ownership and Visibility

- Developers often have little insight into the lifecycle or status of their services.
- Identifying service ownership, dependencies, or documentation is difficult.

Difficulty Tracking Operational Metrics

- Monitoring and managing service health often involves using multiple systems.
- Developers struggle to track key metrics and operational health across the board.

Foundations for Functionality



Core Components

App

The app is an instance of a Backstage app that is deployed and tweaked.

Plugins

Additional functionality to make your Backstage app useful for your company.

Backstage Software Catalog

The central repository in Backstage where all software components and their metadata are stored and managed.

Entities

Items within the Catalog that represent software components, such as services, libraries, or data pipelines.

Scaffolder

A Backstage plugin that helps generate new projects or components using predefined templates and configuration.

Backstage software templates

It is used for quickly spinning up new projects and standardizing your tooling with your organization’s best practices

Backstage TechDocs

It is used for making it easy to create, maintain, find, and use technical documentation, using a "docs like code" approach

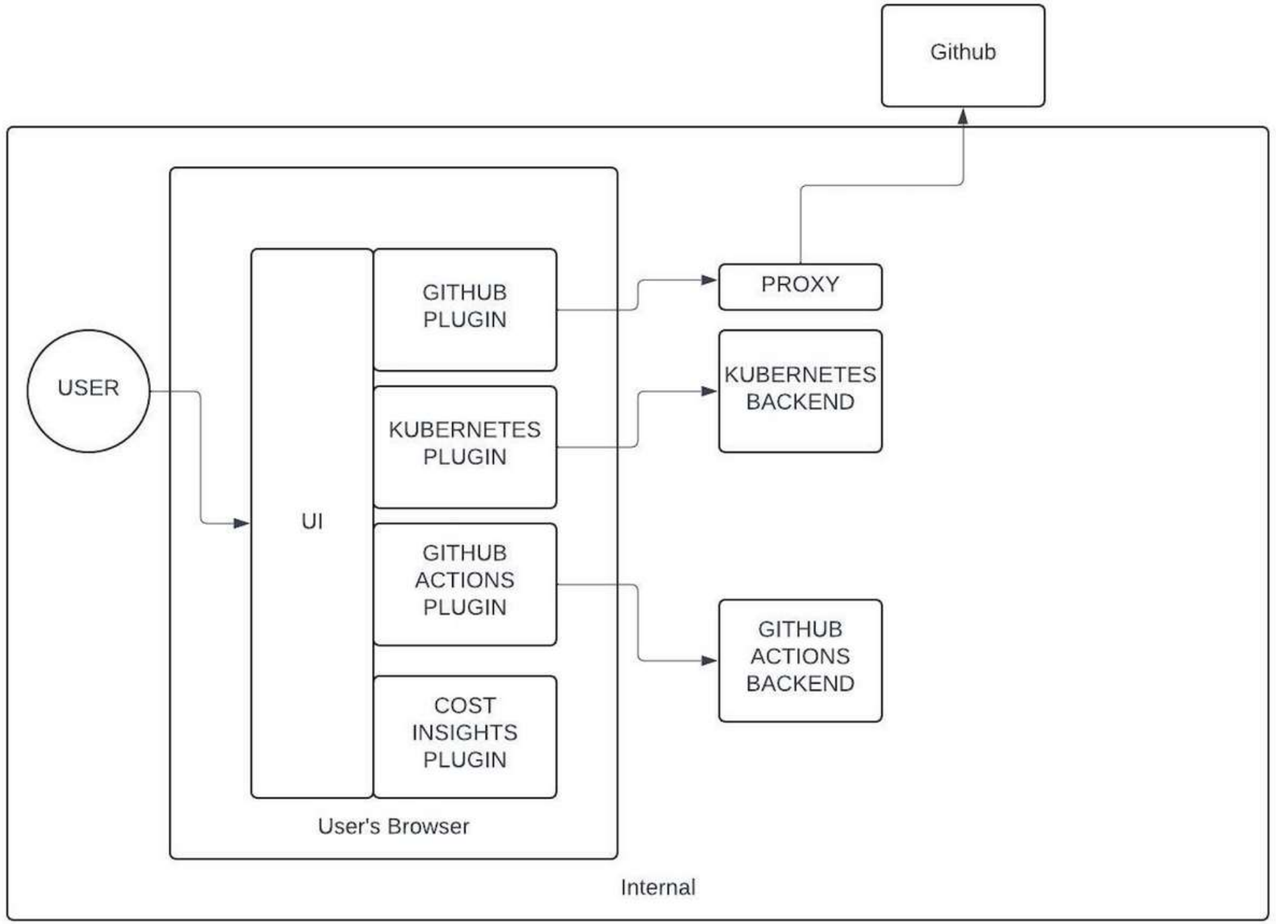
Architecture

There are 3 main components in this architecture:

1. The core Backstage UI
2. The UI plugins and their backing services
3. Databases

The Plugins can take three forms:

1. Standalone Plugins
2. Service backend
3. Third-party backend



Backstage Benefits

01 It helps maintain standards and best practices across the organization.

02 Provides a central place to manage all projects and documentation

03 Fast and simple to build software components in a standardized way.

04 It enables extensibility and scalability

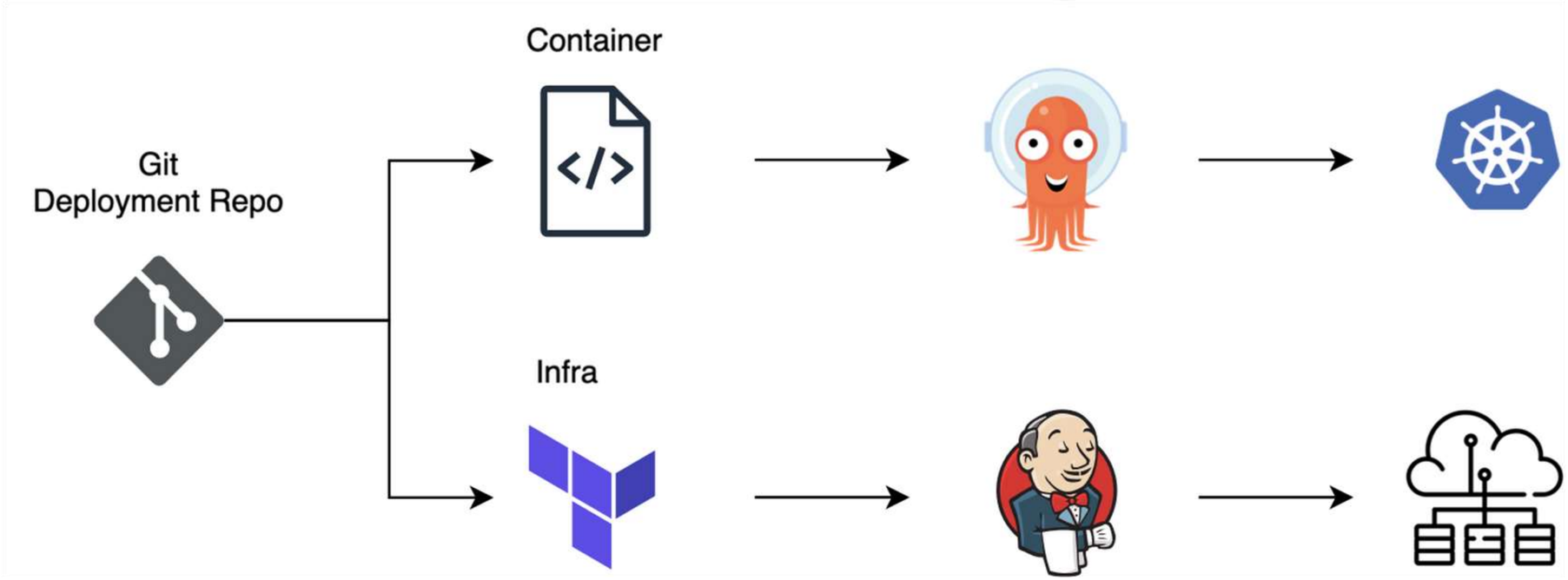
05 It’s a single, consistent experience that ties all your infrastructure tooling, resources, standards, owners, contributors, and administrators together in one place.

Crossplane

universal control plane

What Problem It Solves ?

- Manage both applications and infrastructure using Kubernetes-native APIs.
- The traditional approach involves two sets of processes for the deployment illustrated below



Traditional Pipeline

Contd. Managed Resources

```
resource "aws_s3_bucket" "example" {
  bucket = "my-tf-test-bucket"

  tags = {
    Name        = "My bucket"
    Environment = "Dev"
  }
}
```

```
cat <<EOF | kubectl create -f -
apiVersion: s3.aws.upbound.io/v1beta1
kind: Bucket
metadata:
  generateName: crossplane-bucket-
spec:
  forProvider:
    region: us-east-2
  providerConfigRef:
    name: default
EOF
```

Contd. Compositions

It's like Terraform Module Code

describe more complex deployments, combining multiple managed resources and any resource customizations

```yaml
apiVersion: apiextensions.crossplane.io/v1
kind: Composition
spec:
  resources:
    - name: StorageBucket
      base:
        apiVersion: s3.aws.upbound.io/v1beta1
        kind: Bucket
        spec:
          forProvider:
            region: "us-east-2"
    - name: VM
      base:
        apiVersion: ec2.aws.upbound.io/v1beta1
        kind: Instance
        spec:
          forProvider:
            ami: ami-0d9858aa3c6322f73
            instanceType: t2.micro
            region: "us-east-2"
```

Contd. Composite Resource Definitions (XRD)

Similar to Terraform Modules Input & Output Definition

use an OpenAPIv3 schema to further extend Kubernetes with custom API endpoints, revisions and more

```yaml
apiVersion: apiextensions.crossplane.io/v1
kind: CompositeResourceDefinition
metadata:
  name: xdatabases.custom-api.example.org
spec:
  group: custom-api.example.org
  names:
    kind: xDatabase
    plural: xdatabases
  versions:
  - name: v1alpha1
    schema:
      openAPIV3Schema:
        type: object
        properties:
          spec:
            type: object
            properties:
              region:
                type: string
              size:
                type: string
              name:
                type: string
            required:
              - region
              - size
    # Removed for brevity
```

Contd. Composite Resource (XR) & Claims (XRC)

Like Calling a Module

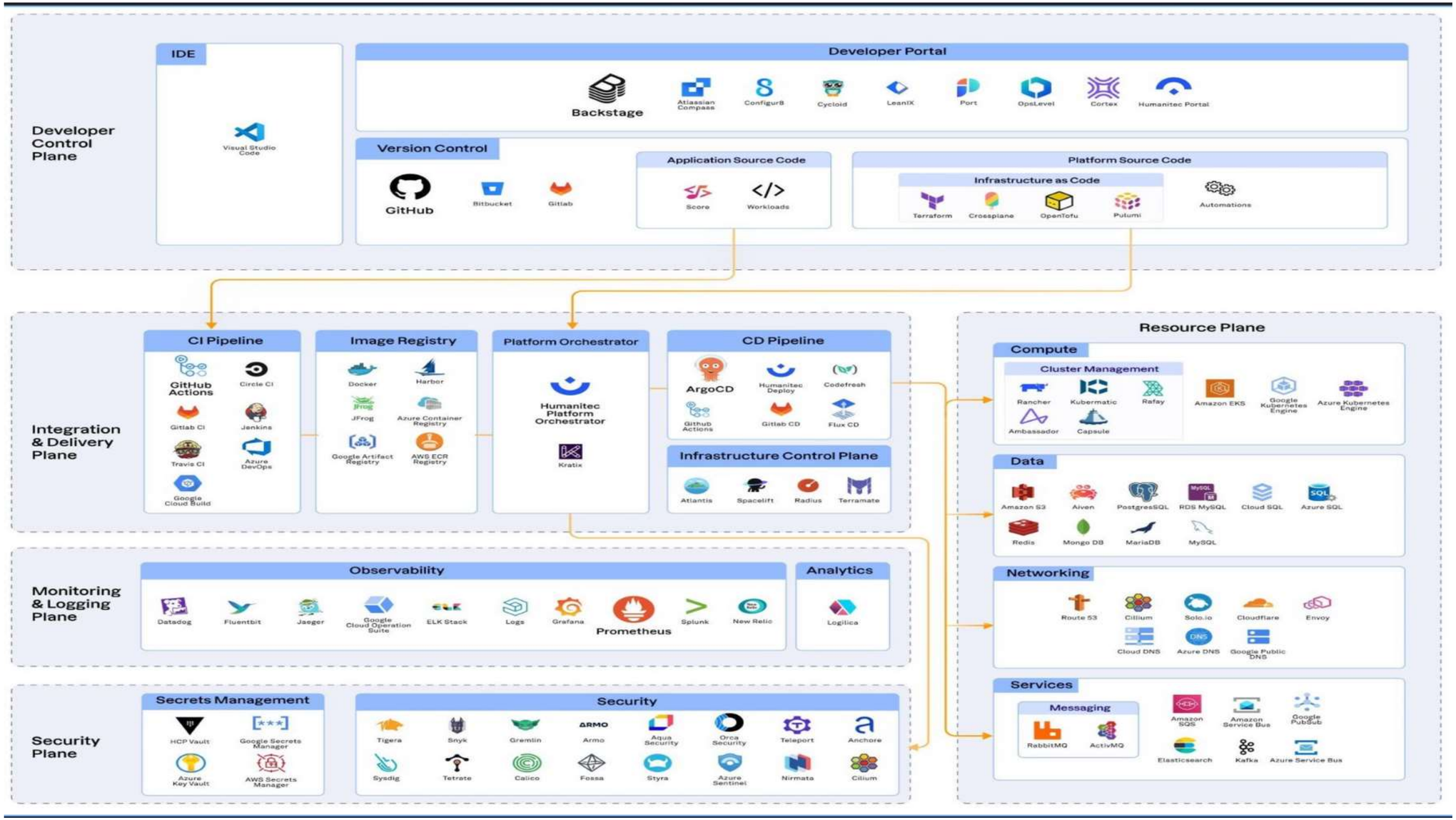
XRs use the Composition template
to
create new managed resources.

Claims (XRC) similar to XR, but with
namespace scoped

```yaml
apiVersion: example.org/v1alpha1
kind: xMyDatabase
metadata:
  name: my-composite-resource
spec:
  writeConnectionSecretToRef:
    name: my-secret
    namespace: crossplane-system
  # Removed for brevity
```

Demo

Platform Tooling Landscape



Thank you