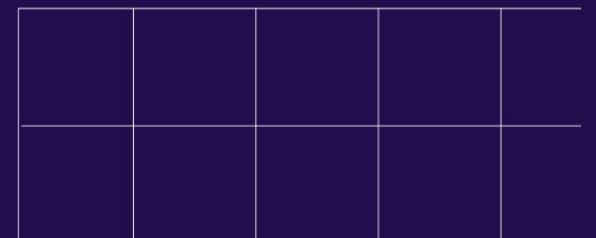


Shahar Shmaram | Ran Mansoor

How we migrated over 1000 services to Backstage using GitOps and survived to talk about it!

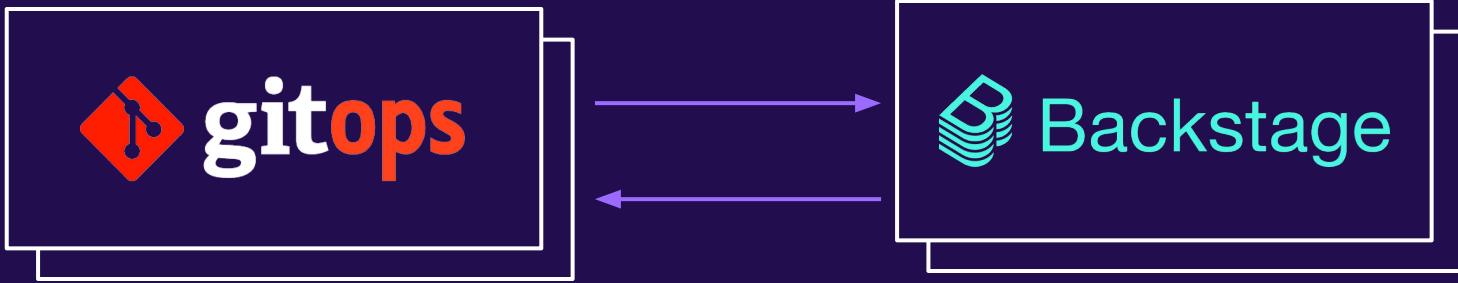




KubeCon + CloudNativeCon Europe 2023









Ran Mansoor
Platform Software Engineer



Shahar Shmaram
Platform Software Engineer

This is AppsFlyer today

\$40bn

Ad spend Measured
per year

~14K

Customers

\$300M

In funding

\$300+M

ARR

65%

Global market share*

25

Offices globally

\$9.8M

Fraud blocked
per day

10K

Integrated technology
partners

1,500+

Employees



8th EIGHT ROADS

T... CAPITAL PARTNERS

Goldman
Sachs

pitango

QUMRA
CAPITAL

GENERAL
ATLANTIC

salesforce ventures

AppsFlyer Engineering

+400

AppsFlyer engineers.
Operate in autonomous
squads.

+1000

Microservices **handling**
over 3M events per
second.

~250K

Cloud resources and
dozens of SaaS
integrations.

Platform Team

~50

Engineers

6

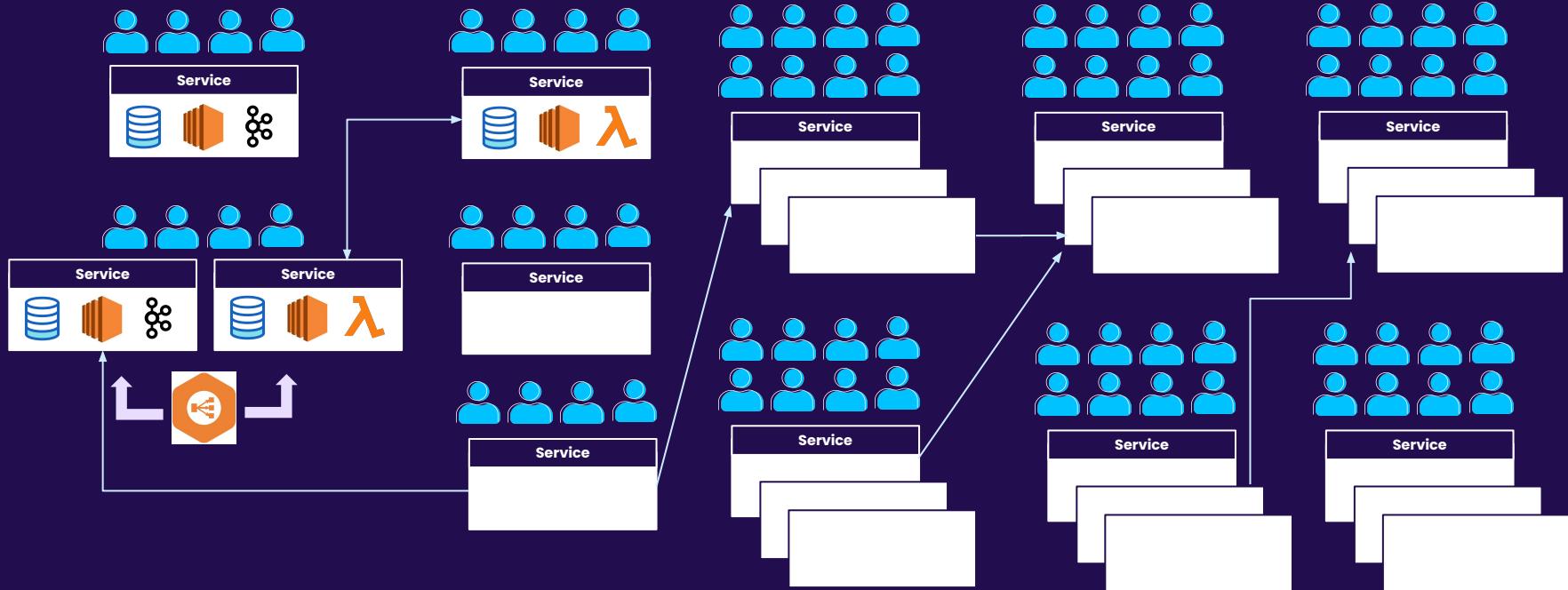
Teams

3

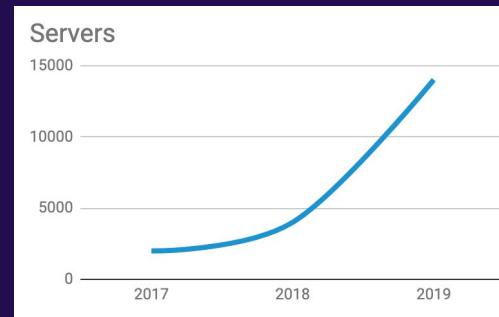
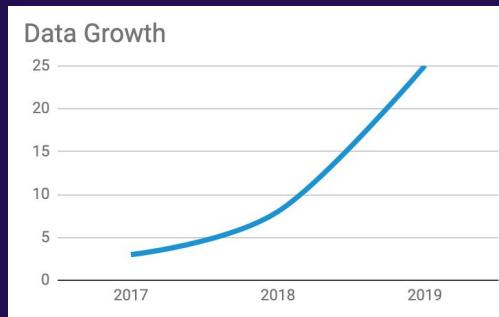
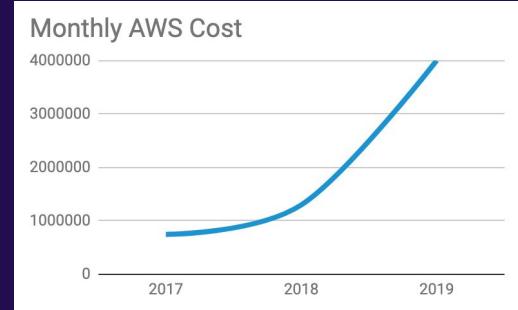
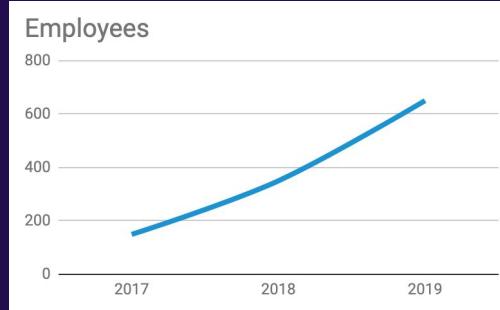
Product
Managers

+350

R&D
engineers



Hyper-Growth



We Won the market but...

- ✖ No alignment
- ✖ Manually managed resources
- ✖ Unknown resources dependencies and ownership
- ✖ Exploding budget
- ✖ Lack of technical documentation

Frustration



Solution Principals



Auditable



Declarative



Single Source of Truth



Community Driven



Self Serve



Visibility



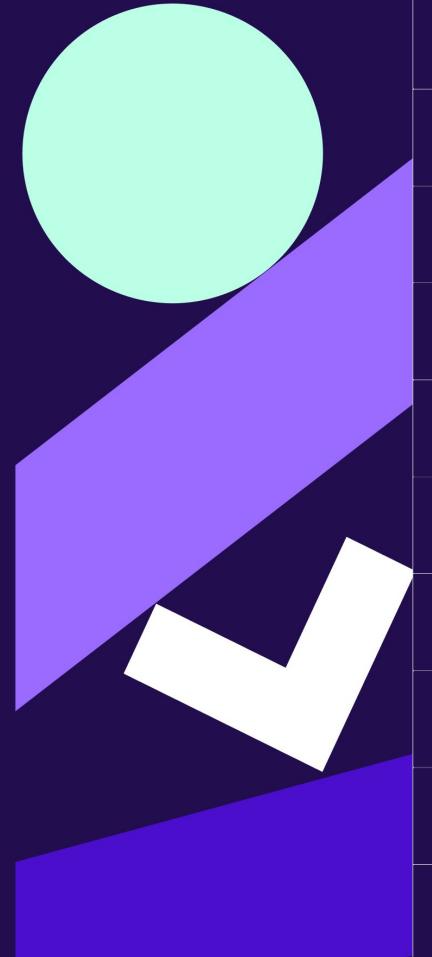
Gitops

- Leverages git as the single source of truth for managing the entire software development lifecycle.
- Emphasizes the use of declarative infrastructure-as-code (IaC) and automated deployment pipelines.



Gitops - Core Principles

- Declarative
- Versioned & Immutable
- Pulled Automatically
- Continuously Reconciled



Solution Principals



Auditable



Declarative



Single Source of Truth



Community Driven



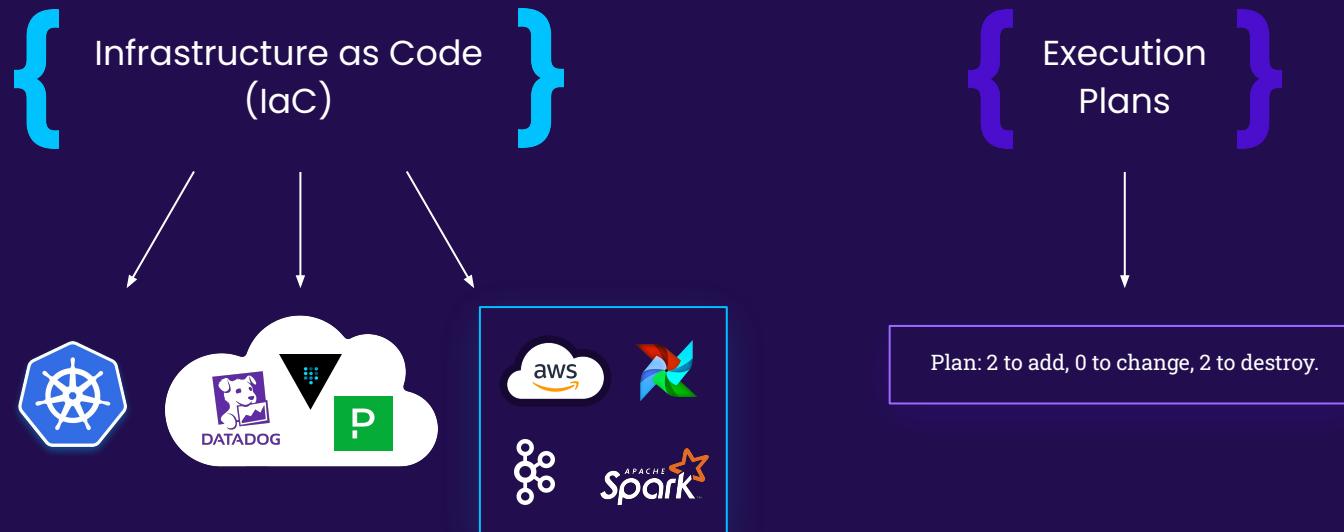
Self Serve



Visibility

Terraform

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently



Solution Principals



Auditable



Declarative



Single Source of Truth



Community Driven



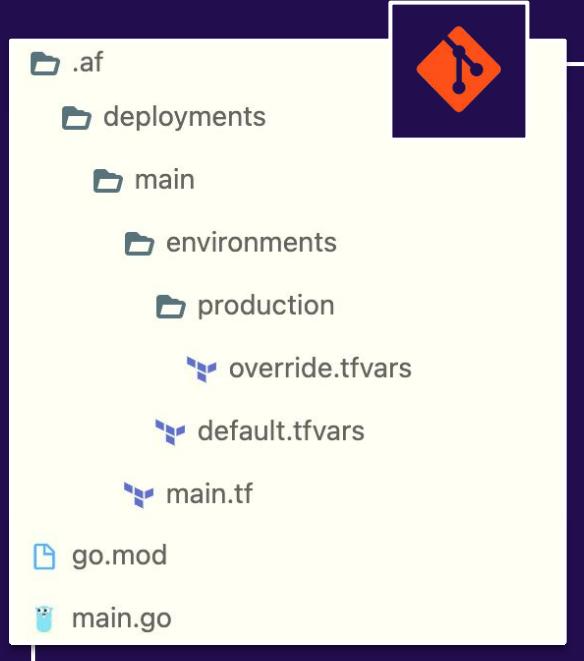
Self Serve



Visibility

Meta Folder

- Configuration close to application code
- .af folder in every git repository
- Contains deployment manifest and resources the application needs



Solution Principals



Auditable



Declarative



Single Source of Truth



Community Driven



Self Serve



Visibility

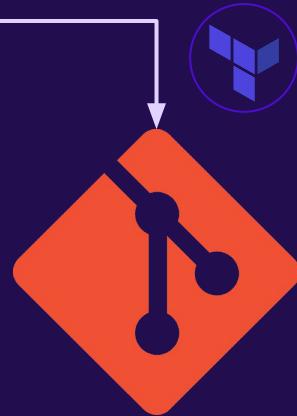


Developer A



Developer A

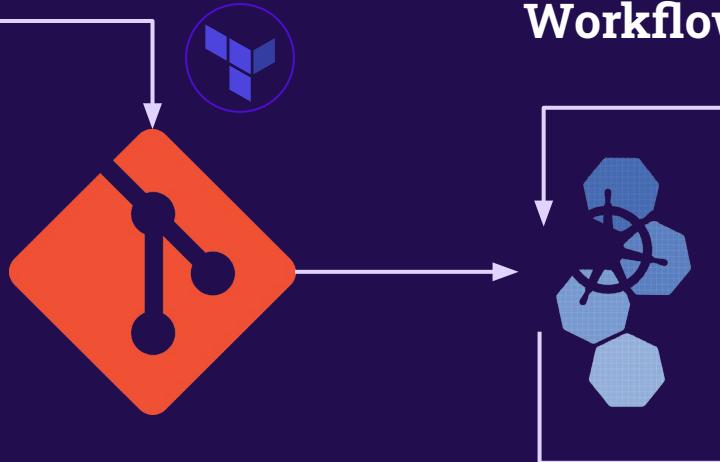
- Commit code changes
- Declare service deployment plan





- Commit code changes
- Declare service deployment plan

GitOps Workflow

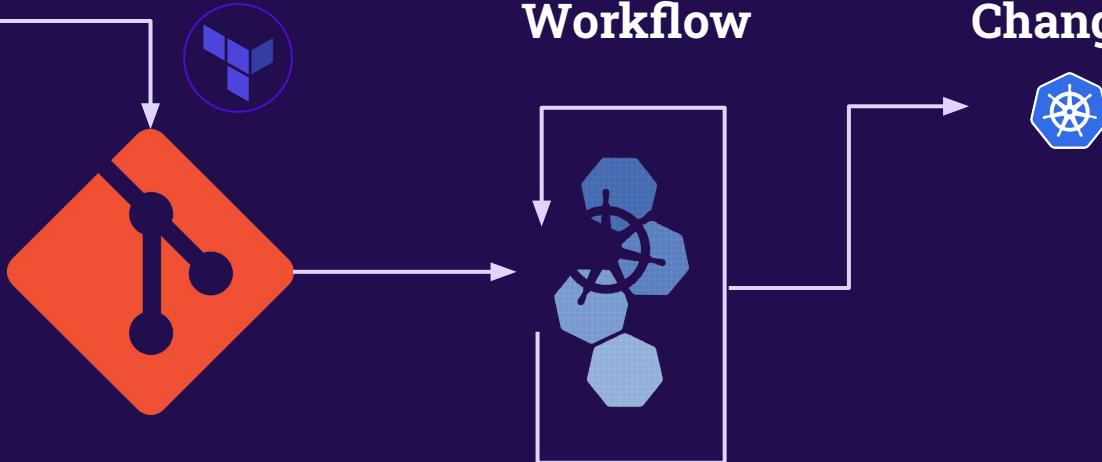


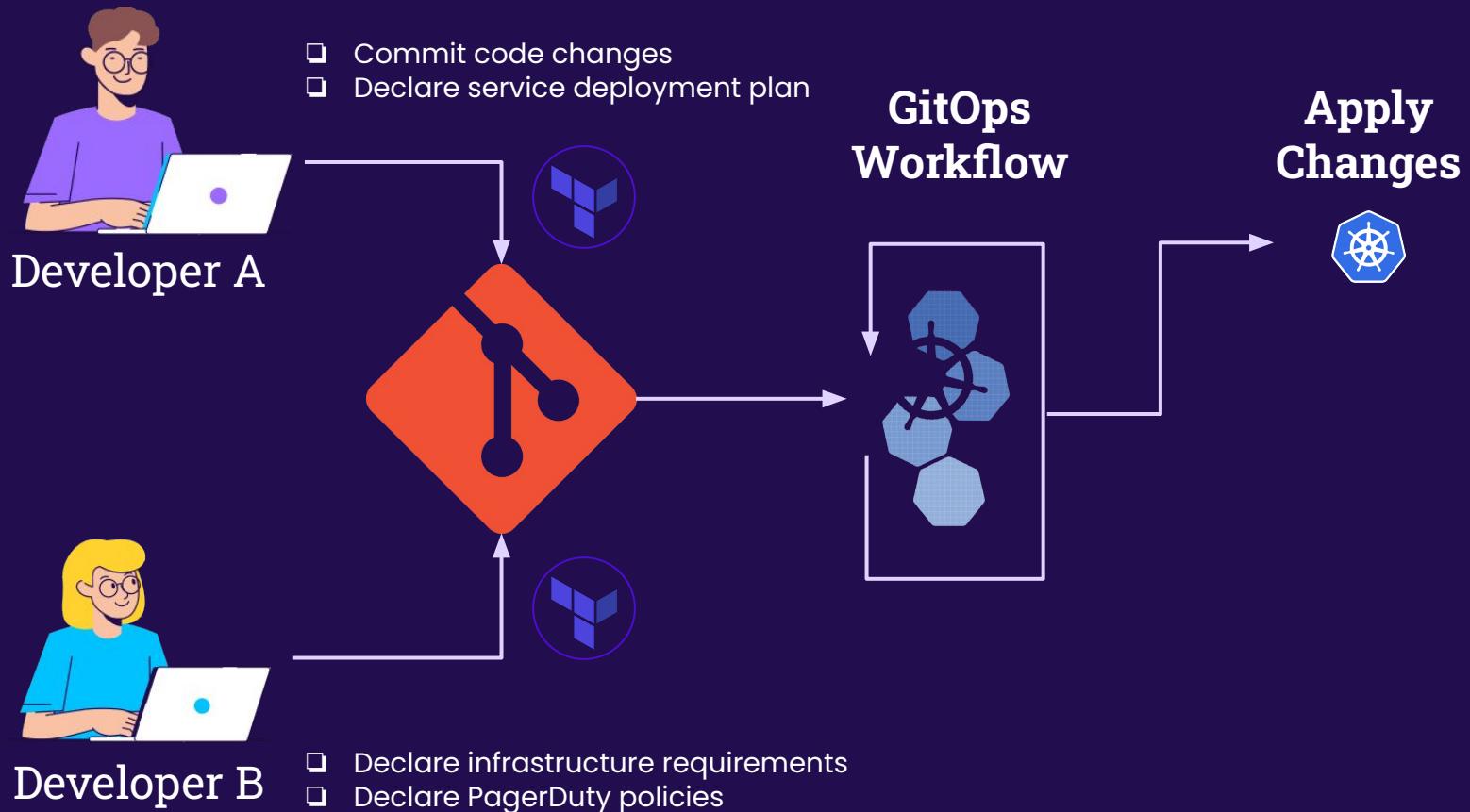


- Commit code changes
- Declare service deployment plan

GitOps Workflow

Apply Changes

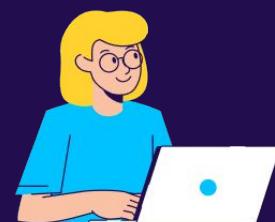






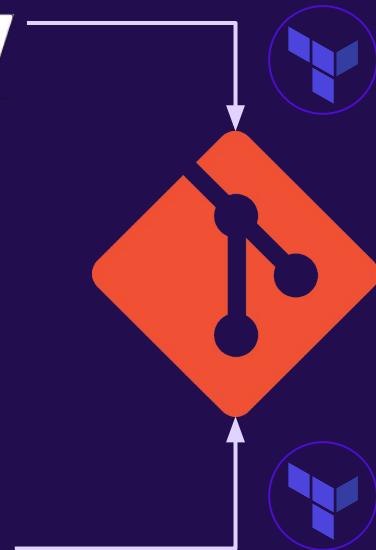
Developer A

- Commit code changes
- Declare service deployment plan

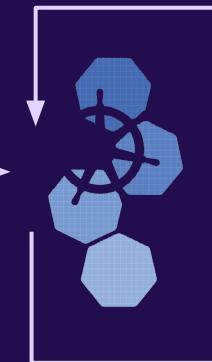


Developer B

- Declare infrastructure requirements
- Declare PagerDuty policies



GitOps Workflow



Apply Changes



Developer C

Solution Principals



Auditable



Declarative



Single Source of Truth

 Terraform -  git



Community Driven

 kubebuilder  flux



Self Serve



Visibility

- ✓ Managed resources
- ✗ Where can I find all my teams' services?
- ✗ Who is the owner of a service?
- ✗ Who is consuming from my Kafka cluster?
- ✗ Who depends on my resource?

A photograph of a person's head and shoulders, wearing a white bucket hat and binoculars, sitting in a field of tall, dry grass. The person is looking directly at the camera. The background is a dense field of grass.

No Visibility

Backstage

An open platform for building
developer portals



Core Features

Easy To Use Software Catalog

- Manage all the software in one place
- a uniform overview
- Knowledge on all resources with metadata using YAML files

Software Catalog

Bună ziua, guest!

Backstage Service Catalog

NYC 07:56 UTC 11:56 STO 13:56 TYO 20:56

SERVICES WEBSITES LIBRARIES DOCUMENTATION OTHER

CREATE COMPONENT SUPPORT

Owned (3)

NAME	OWNER	LIFECYCLE	DESCRIPTION	ACTIONS
playback-order	guest	production	Playback Order	
searcher	guest	production	Searcher	
shuffle-api	guest	production	Shuffle API	

PERSONAL

- Owned 3
- Starred 0

SPOTIFY

- All 6

SEARCH

SETTINGS

LOGOUT

Software Catalog

```
1  apiVersion: backstage.io/v1alpha1
2  kind: Component
3  metadata:
4    name: ib-test
5    description: |
6      A service for testing Backstage functionality. Configured for GitHub Actions and
7      Sentry.
8  annotations:
9    github.com/project-slug: roadiehq/sample-service
10   # The Sentry organization is stored in the app-config.yaml of the Backstage instance.
11   sentry.io/project-slug: sample-service
12   backstage.io/techdocs-ref: dir:./
13   pagerduty.com/integration-key: 1cf15c07f6f440e0a8d65b7ce80be834
14   snyk.io/org-name: dtuite
15   snyk.io/project-ids: b6ab231c-a019-4def-a148-4a10a79d6302
16   lighthouse.com/website-url: https://api.development.env.tryflux.com/lighthouse-audit
17 spec:
18   type: service
19   owner: engineering
20   lifecycle: experimental
21   providesApis:
22     - sample-service
```

Core Features

Documentation as code

- Easy to write - markdown files
- Maintainable
- Can easily be added as part of the code
- Easy to find - docs close to infra

Documentation as code

TechDocs Entity

The screenshot shows the Backstage documentation interface. At the top, there's a navigation bar with icons for home, search, and settings, followed by the text "Docs / Backstage" and the title "Backstage". Below the title, it says "Main documentation for Backstage features and platform APIs". To the right of the title, there are three status cards: "Component backstage", "Owner CNCF", and "Lifecycle experimental". A "refresh" icon is also present.

The main content area has a sidebar on the left with a tree view of documentation sections:

- Backstage
 - Overview
 - What is Backstage?
 - Backstage architecture** (selected)
 - Roadmap
 - Vision
 - The Spotify story
 - Strategies for adopting
 - Logo assets
 - Getting started
 - Features
 - Integrations
 - Plugins
 - Configuration
 - Authentication and identity

The main content area displays the selected section, "Backstage architecture". It includes a heading, a table of contents, and a detailed description with a bulleted list of core components.

Table of contents

- Terminology
- Overview
- User Interface
- Plugins and plugin backends
- Installing plugins
- Plugin architecture
- Standalone plugins
- Service backed plugins
- Third-party backed plugins
- Databases
- Containerization

Content Details

Backstage architecture

Terminology

Backstage is constructed out of three parts. We separate Backstage in this way because we see three groups of contributors that work with Backstage in three different ways.

- Core - Base functionality built by core developers in the open source project.
- App - The app is an instance of a Backstage app that is deployed and tweaked. The app ties together core functionality with additional plugins. The app is built and maintained by app developers, usually a productivity team within a company.
- Plugins - Additional functionality to make your Backstage app useful for your company. Plugins can be specific to a company or open sourced and reusable. At Spotify we have over 100 plugins built by over 50 different teams. It has been very powerful to get contributions from various infrastructure teams added into a single unified developer experience.

Previous: [What is Backstage?](#) | Overview | Next: [Roadmap](#)

Core Features

Powerful Search

- Everything can be found
- Index all entities and properties
- Customizable

Powerful Search

Search Alpha

how to

Kind

- All
- javascript
- java
- python
- git

How do I redirect to another webpage?
Author: venkatachalam
Answer(s): 58 Tag: javascript Tag: jquery Tag: redirect

How to modify existing, unpushed commit messages?
Author: Laurie Young
Answer(s): 27 Tag: git Tag: git-commit Tag: git-rewrite-history Tag: git-amend

How do I force "git pull" to overwrite local files?
Author: Jakub Troszok
Answer(s): 50 Tag: git Tag: version-control Tag: overwrite Tag: git-pull Tag: git-fetch

How do I revert a Git repository to a previous commit?
Author: Crazy Serb
Answer(s): 41 Tag: git Tag: git-checkout Tag: git-reset Tag: git-revert

How to check whether a string contains a substring in JavaScript?

Core Features

Automated Software Templates

- “get started” guide
- Easy to create using YAML file
 - translated to code
- Customizable

Software Templates

Create a New Component β

Create new software components using standard templates

Available Templates

Recommended

documentation Documentation Template ★

website React SSR Template ★

service Spring Boot gRPC Service ★

Other Templates

website Create React App Template ★

service Pull Request Action template ★

service Test Action template ★

REGISTER EXISTING COMPONENT ? SUPPORT

PERSONAL
Starred 0

MY COMPANY
All 6

CATEGORIES

TAGS

SEARCH

Software Templates

Create a New Component

Create new software components using standard templates

Create Team

- 1 Fill in the following parameters

Team name*

Unique name of the component

Description

Notification channel*

BACK

NEXT STEP

Core Features

Self contained Plugins

- Functional building blocks
- Allows to expose capabilities
- Isolated code
- Marketplace

Plugins

The screenshot shows the Spotify Backstage management interface. At the top, it displays "Arratsalde on, Stefan" and "Thanks for taking care of 51 components and 19 projects". The top right shows time zones for NYC (10:18 AM), UTC (03:18 PM), LON (03:18 PM), and STO (04:18 PM). The left sidebar has a "Overview" dropdown menu with options like Services, Websites, Libraries, Workflows, Data Endpoints, App Features, GCP Projects, Storage, and Other. The main dashboard features several cards:

- Squad Metrics**: "Test Certified" at 24% (highlighted with a green border).
- Cloud Cost**: A line graph with a small icon.
- Security Issues**: 16 issues (highlighted with a green border).
- Incidents**: 10 days since last incident.
- Cycle Time**: A line graph.

Squad Ownership summary:
24 Services → 3 Websites → 9 Data Endpoints → 7 Workflows → 5 Libraries >

Platform Upgrades & Migrations: Python 3, Scio, Gabito, Bionic, Java 11.

Plugins

Backstage

GITHUB DOCS PLUGINS BLOG DEMOS NEWSLETTER Search

Add to marketplace

Plugin marketplace

Open source plugins that you can add to your Backstage deployment. Learn how to build a [plugin](#).

 API Docs by SDA SE [Discovery](#)

Components to discover and display API entities as an extension to the catalog plugin.

[Explore](#)

 CircleCI by Spotify [CI](#)

Automate your development process with CI hosted in the cloud or on a private server.

[Explore](#)

 Cost Insights by Spotify [Discovery](#)

Visualize, understand and optimize your team's cloud costs.

[Explore](#)

 GitHub Actions by Spotify [CI](#)

GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD. Build, test, and deploy your code right from GitHub.

[Explore](#)

 GitHub Pull Requests by roadie.io [CI](#)

View GitHub pull requests for your service in Backstage.

[Explore](#)

 GitOps Clusters by Weaveworks [Kubernetes](#)

Create GitOps-managed Kubernetes clusters. Currently, it supports provisioning EKS clusters on GitHub via GitHub Actions.

[Explore](#)

 GraphQL by Spotify [Debugging](#)

Integrates GraphQL as a tool to browse GraphQL API endpoints inside Backstage.

[Explore](#)

 Jenkins by @timja [CI](#)

Jenkins offers a simple way to set up a continuous integration and continuous delivery environment.

[Explore](#)

 Lighthouse by Spotify

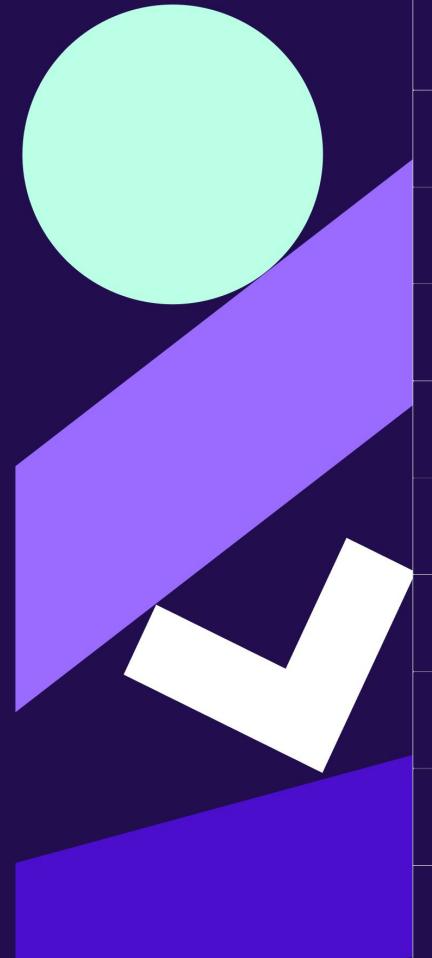
 New Relic by @timwheelerauthor

 Rollbar by @andrewthauzer

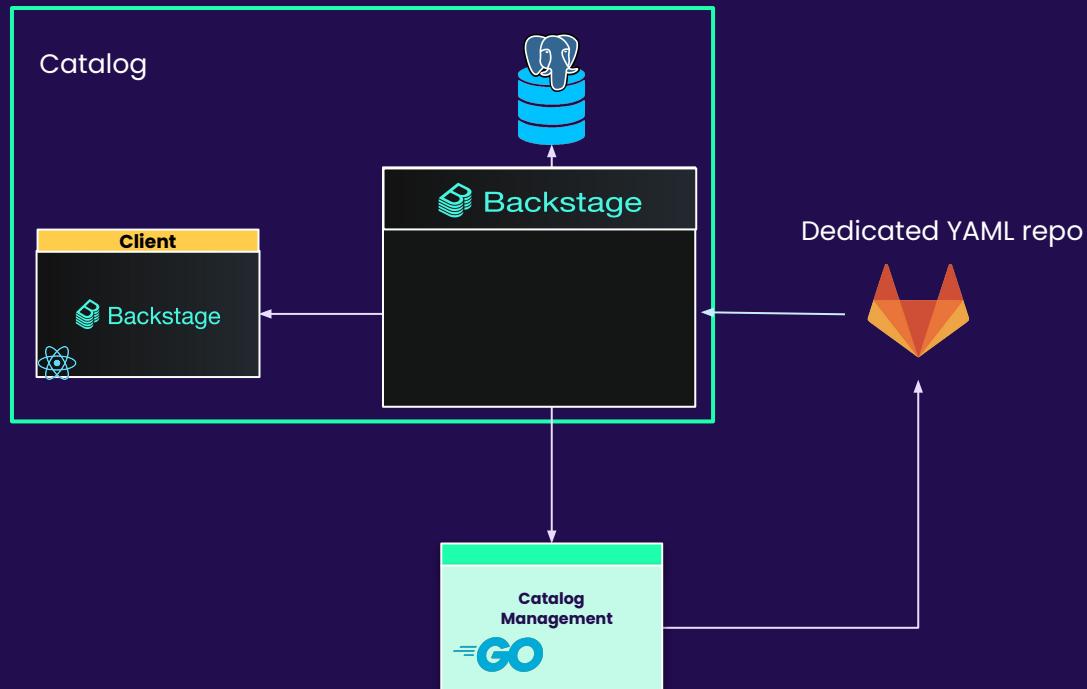
 Sentry by Spotify

One stop shop for all

- One system to externalize all platform capabilities
- Resources standardization
- Distributed plugin development done by platform team owners;
 - * Rapid development
 - * Accountability
 - * Ownership



Backstage Architecture in AppsFlyer



Solution Principals



Auditable



Declarative



Single Source of Truth

 Terraform -  git



Community Driven

 kubebuilder  flux



Self Serve



Visibility

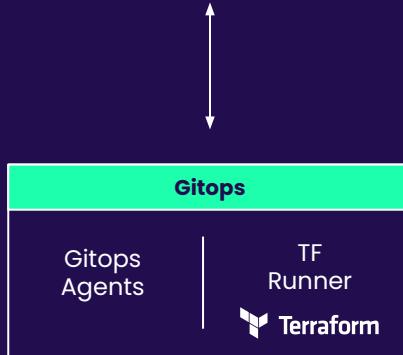
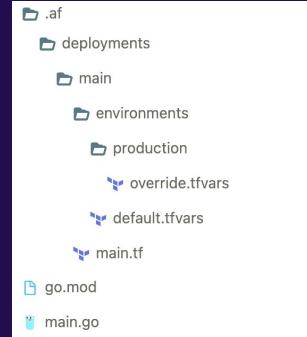


All resources needs to be
managed using TF and
appear in Backstage

Resources in TF

All resources managed by TF in .af

- Service Deployment Manifests
- Service Resources (PG Policies, DD Monitors)
- Infrastructure (RDS)





All resources should be
managed using TF and
appear in Backstage

Resources in Backstage



Service



Lambda



S3 Bucket



Pagerduty Service

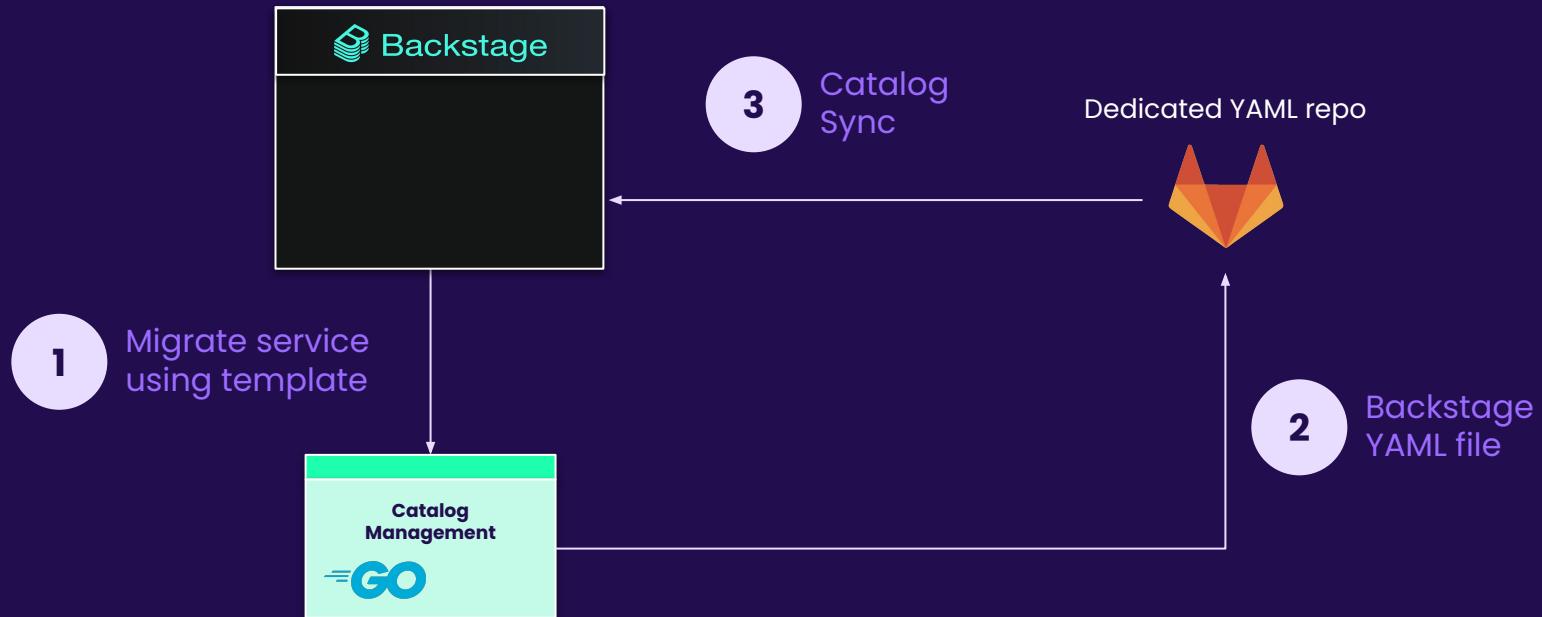


Datadog Monitor



EC2

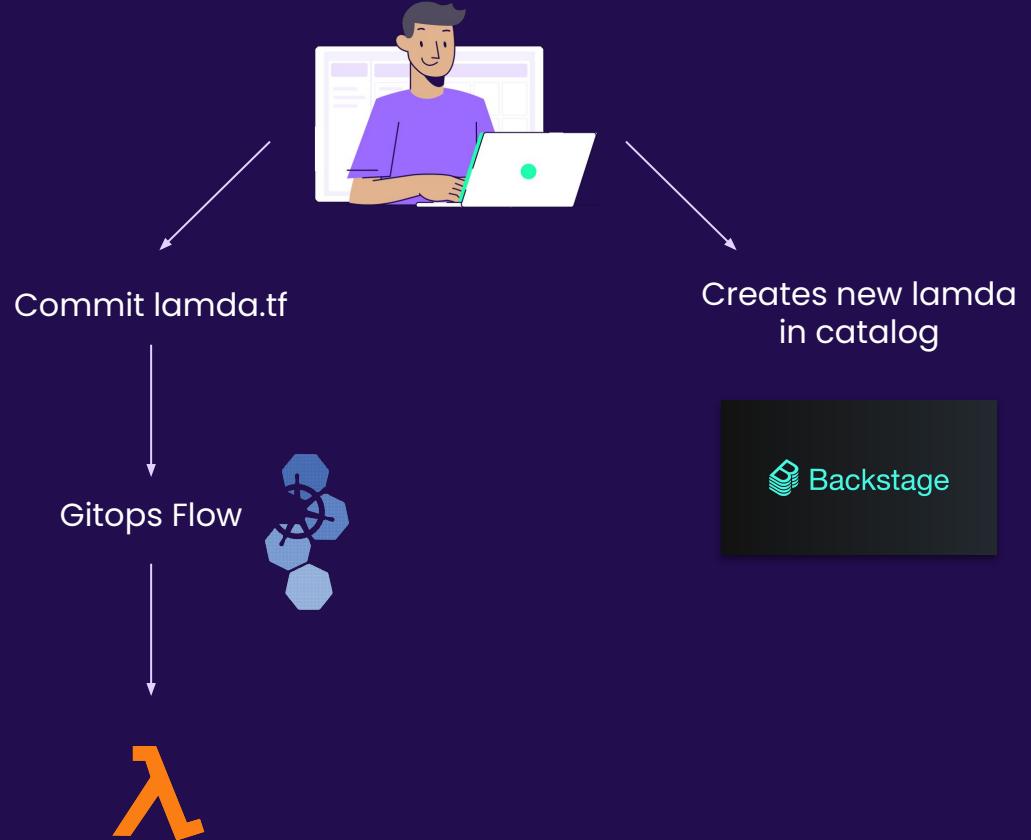
Migrating Service to Backstage



All resources should be
managed using TF and
appear in Backstage

Is it enough??

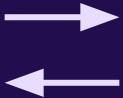
 Terraform



All resources should be
managed using TF,
appear in Backstage,
and always sync



gitops



Backstage

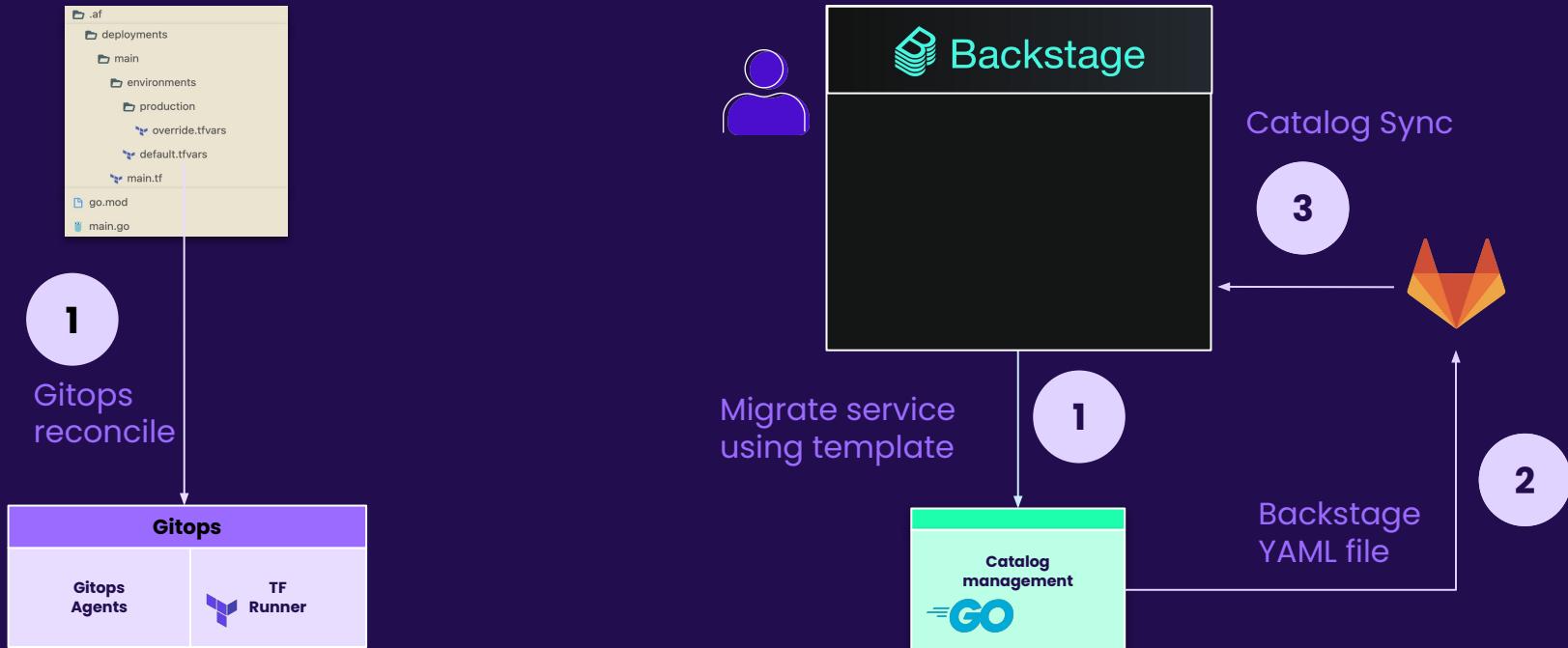


Terraform



YAML

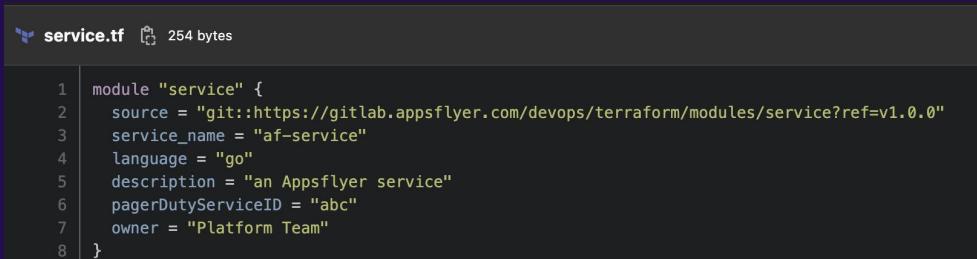
Migrate service using Backstage



What is a service in TF?

~~Service == Deployment~~

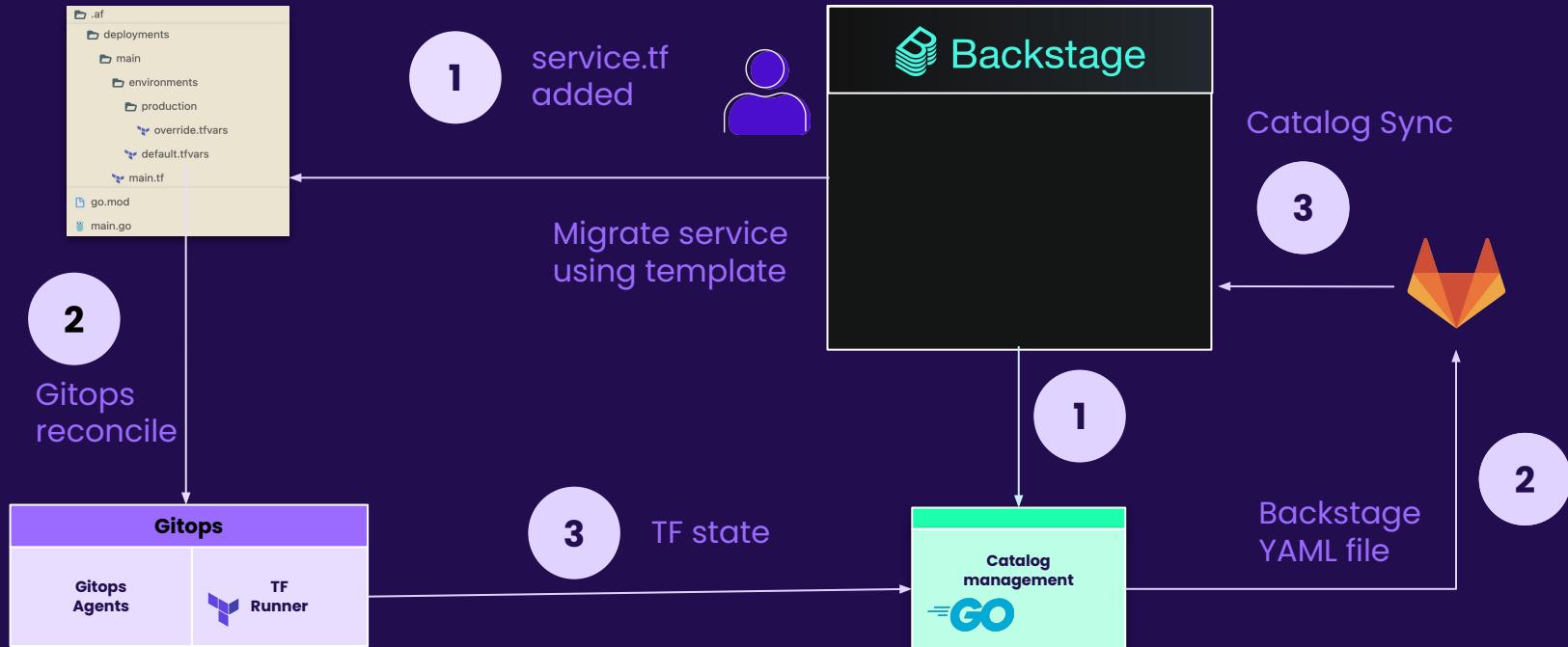
- Service can be described before deployment
- Contains metadata such as:
 - * Language
 - * Description
 - * PagerDuty Service
 - * Owner
 - * etc...



A screenshot of a code editor showing a Terraform configuration file named "service.tf". The file contains 8 lines of code defining a module named "service" with various configuration parameters. The code is as follows:

```
1 module "service" {  
2   source = "git::https://gitlab.appsflyer.com/devops/terraform/modules/service?ref=v1.0.0"  
3   service_name = "af-service"  
4   language = "go"  
5   description = "an Appsflyer service"  
6   pagerDutyServiceID = "abc"  
7   owner = "Platform Team"  
8 }
```

Migrate service using Backstage



01



Migrate service

Create the TF file in the user repository

02



Gitops validation

Uses a web socket to listen to the gitops process until it's done and creates the YAML file

03



Backstage validation

Wait until Backstage synced the new created YAML file into the catalog

The screenshot shows the 'Task Activity' interface with the following details:

- Activity for task:** fd00c98c-6403-4bac-81fb-5bc61beef013
- Steps:**
 - Migrate service (1 second)
 - GitOps validation (44 seconds)
 - Backstage validation (5 seconds)
 - Open the catalog
- Logs:** A list of 13 log entries showing the progress of the validation steps.

Log Number	Timestamp	Message
1	2022-07-27T12:28:21.940Z	Beginning step Backstage validation
2	2022-07-27T12:28:21.948Z	info: Found 5 entities to sync {"timestamp":"2022-07-27T12:28:21.947Z"}
3	2022-07-27T12:28:21.948Z	info: Component:unit-of-xray-default - starting entity sync {"timestamp":"2022-07-27T12:28:21.947Z"}
4	2022-07-27T12:28:21.948Z	info: Component:af-xray-default - starting entity sync {"timestamp":"2022-07-27T12:28:21.947Z"}
5	2022-07-27T12:28:21.950Z	info: Group:devops - starting entity sync {"timestamp":"2022-07-27T12:28:21.947Z"}
6	2022-07-27T12:28:21.951Z	info: System:cloud_platform - starting entity sync {"timestamp":"2022-07-27T12:28:21.948Z"}
7	2022-07-27T12:28:21.951Z	info: Domain:shared - starting entity sync {"timestamp":"2022-07-27T12:28:21.948Z"}
8	2022-07-27T12:28:26.966Z	info: Component:af-xray-default - synced successfully {"timestamp":"2022-07-27T12:28:26.967Z"}
9	2022-07-27T12:28:26.967Z	info: Component:unit-of-xray-default - synced successfully {"timestamp":"2022-07-27T12:28:26.968Z"}
10	2022-07-27T12:28:26.967Z	info: Group:devops - synced successfully {"timestamp":"2022-07-27T12:28:26.968Z"}
11	2022-07-27T12:28:26.968Z	info: System:cloud_platform - synced successfully {"timestamp":"2022-07-27T12:28:26.969Z"}
12	2022-07-27T12:28:26.968Z	info: Domain:shared - synced successfully {"timestamp":"2022-07-27T12:28:26.969Z"}
13	2022-07-27T12:28:26.969Z	Finished step Backstage validation

Migration Stats

+1,000

Services

+45

Teams

~5

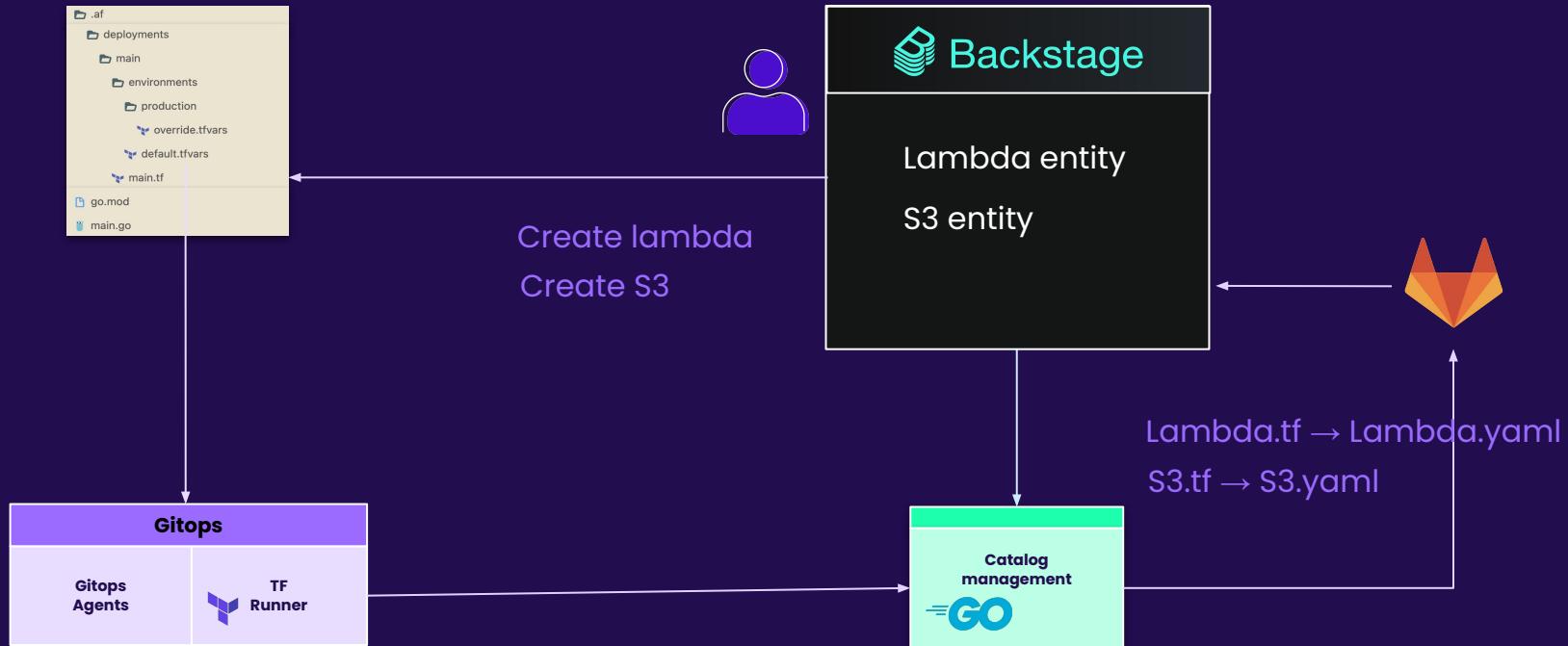
Months

**WE LIKE IT, WE REALLY
LIKE IT**

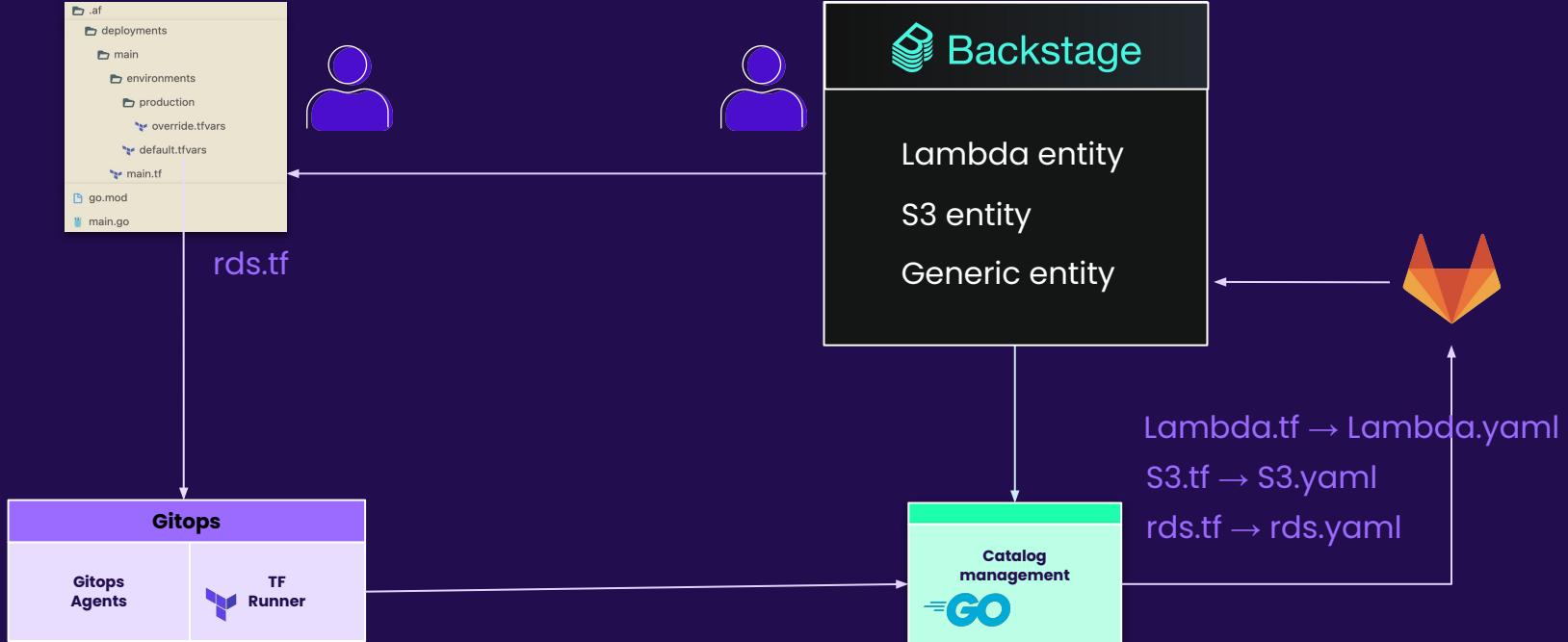


WE WANT MORE!

Support additional resources using Backstage



Support new TF resources



Non generic entity (service)

COMPONENT — SERVICE

af-service ★

Owner devops Lifecycle production

OVERVIEW CI/CD DEPLOYMENT SANTA

About

DESCRIPTION backstage api backstage example yaml

OWNER devops SYSTEM visibility

LIFECYCLE production TAGS No Tags TYPE service

PagerDuty

SERVICE DIRECTORY CREATE INCIDENT ESCALATION POLICY

Incidents Change Events

Nice! No incidents found!

ON CALL

shahar.shmaram@appsflyer.com shahar.shmaram@appsflyer.com

Relations

```
graph TD; af-service[af-service] -- "dependsOn / dependencyOf" --> resource_datadog_monitor[resource-datadog-monitor]; af-service -- "dependsOn / dependencyOf" --> resource_generic_rds[resource-generic-rds]; af-service -- "dependsOn / dependencyOf" --> resource_kafka_cluster[resource-kafka-cluster]; af-service -- "dependsOn / dependencyOf" --> resource_postgres[resource-postgres]; devops[devops] -- "ownerOf / ownership" --> af-service; visibility[visibility] -- "hasPart / partOf" --> af-service;
```

[View graph →](#)

Generic entity (RDS)

The screenshot shows a resource details page for a generic RDS entity named "generic_rds".

Header: RESOURCE – S3-BUCKET generic_rds ☆ Owner dev-team

Left Sidebar: Includes icons for Home, Search, and a three-dot menu.

Overview Tab: Active tab, showing the following details:

- About**: Stores service details.
- DESCRIPTION**: Stores service details.
- OWNER**: dev-team
- SYSTEM**: No System
- TYPE**: s3-bucket
- TAGS**: No Tags

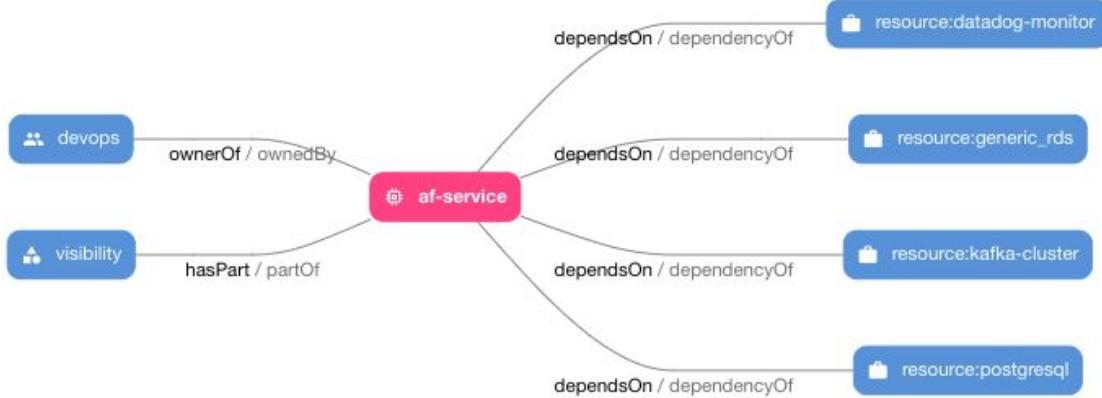
Relations Tab: Shows a dependency relationship:

- af-service dependsOn / dependencyOf resource-generic_rds

Buttons: View graph →

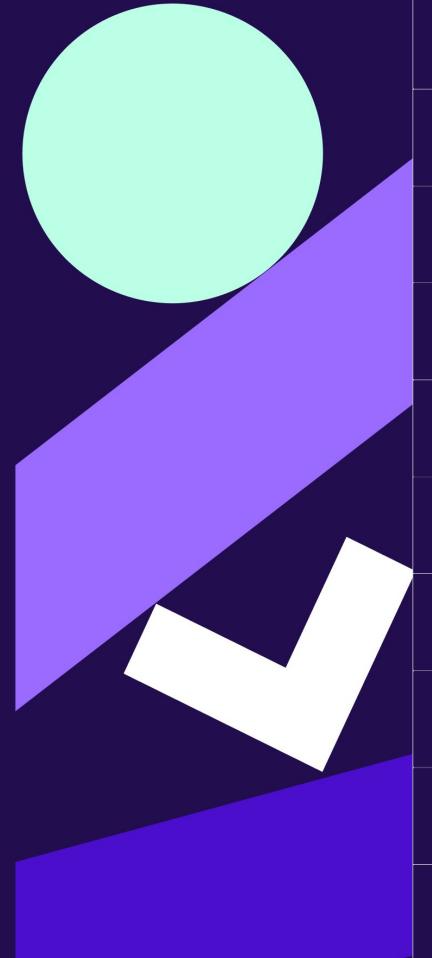
Entity representation

Relations

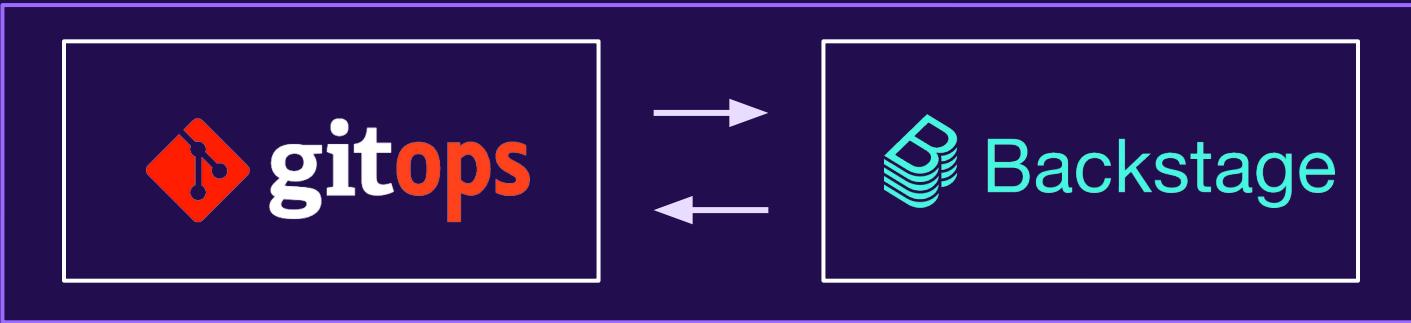


Our Main takeoffs:

- Change is hard
- Developer experience
- Always manage your resources



Total chaos



Managed resources

Thank you



Q&A

Feedback form

