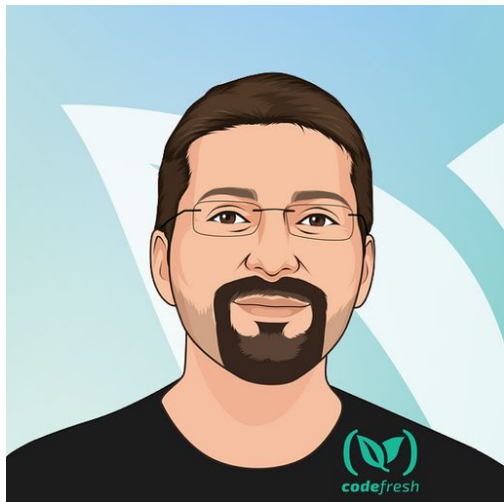




GitOps for Apps AND Infrastructure

GitOpsCon EU 2023

Your host



Kostis Kapelonis

Developer Advocate - Codefresh

Argo Maintainer

kostis@codefresh.io

twitter.com/codepipes

About Codefresh

Modern Deployment Platform

Comes with CI, CD and GitOps modules

Enterprise Ready

Code-to-cloud visibility across apps and clusters

Continuous Delivery

Progressive delivery without compromising stability powered by Argo CD and Argo Rollouts

The screenshot displays the Codefresh dashboard for the 'Loans' product. The interface is organized into three columns, each representing a different environment: 'qa (non-production)', 'staging (non-production)', and 'production (production)'. Each column shows the deployment status for 'loans-qa v1.0.1', 'loans-staging v1.0.0', and 'loans-production v1.0.0' respectively. The 'qa' environment shows a deployment of 'loans-qa' with a '1/1' status. The 'staging' environment shows a deployment of 'loans-staging' with a '2/2' status and a 'Promoted by' section indicating a successful promotion by 'post-promotion-wf-w7j4l'. The 'production' environment shows a deployment of 'loans-production' with a '2/2' status and a 'Promoted by' section indicating a successful promotion by 'post-promotion-wf-ls9mz'. The dashboard also includes search filters for 'Image Name', 'Committer', and 'Jira Ticket', and a 'Manage Apps' button in the top right corner.

Agenda

- GitOps for Apps
- Is terraform following GitOps?
- GitOps for Infrastructure
- Demo with Argo CD/Crossplane

GitOps for Applications



Argo CD Deployments

Applications / guestbook

APP DETAILS

APP DIFF

SYNC

SYNC STATUS

HISTORY AND ROLLBACK

DELETE

REFRESH

Healthy

Synced

To HEAD (6bed858)
Authored by Alex Collins <alexec...>
Updates examples to better reflec...

Sync OK

To 6bed858
Succeeded 7 days ago (Mon Aug 17 2020 19:27:35 GMT+0300)
Authored by Alex Collins <alexec@users.noreply.github.com>
Updates examples to better reflect hook usage today (#41)



guestbook



guestbook-ui



guestbook-ui



rev:1



show 2 hidden resources



guestbook-ui-65b878495d-8rcbt



pod

running

1/1

Avoid Configuration Drift with GitOps

SUMMARY PARAMETERS MANIFEST **DIFF** EVENTS

Compact diff Inline Diff

```
/Service/default/guestbook-ui
1 apiVersion: v1                2 apiVersion: v1
2 kind: Service                  2 kind: Service
3 metadata:                      3 metadata:
4 labels:                        4 labels:
5   app.kubernetes.io/instance: guestbook  5   app.kubernetes.io/instance: guestbook
6 name: guestbook-ui             6 name: guestbook-ui
7 spec:                          7 spec:
8 ports:                         8 ports:
9   - port: 80                   9   - port: 80
10  targetPort: 80               10  targetPort: 80
11  - port: 8080                 11  - port: 8080
12  targetPort: 80              12  targetPort: 80
13 selector:                    13 selector:
14 app: guestbook-ui            14 app: guestbook-ui
```

Applications / guestbook

APP DETAILS APP DIFF SYNC SYNC STATUS HISTORY AND ROLLBACK DELETE REFRESH

Healthy OutOfSync Sync OK



Applications / guestbook

APP DETAILS APP DIFF SYNC SYNC STATUS HISTORY AND ROLLBACK DELETE REFRESH

Healthy Synced Sync OK

What about

things outside Kubernetes?

- Databases
- Load Balancers
- S3 Buckets
- Queues
- Virtual Machines
- System Images
- Lambda functions
- Caches
- DNS records
- Kubernetes clusters themselves

Two distinct worlds

ArgoCD used only for
Kubernetes Applications (or
Flux)

Terraform

(or Pulumi/Ansible/Chef/Puppet)

used for Infrastructure -
including Kubernetes clusters

Two distinct worlds

- Kubernetes native tooling
- Fully GitOps
- Always know state
- Avoid Configuration drift
- Easy auditing

- Non- Kubernetes tool
- Apply infra with jobs/tasks/actions
- Suffer from configuration drift

Two distinct worlds

Kubernetes manifests (YAML)

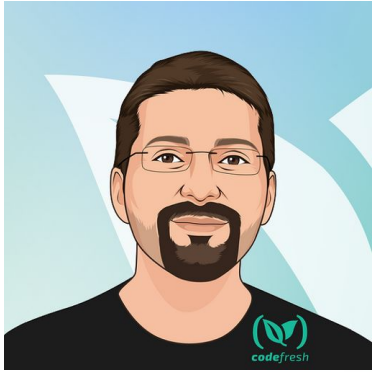
HCL (or other scripts)



Is Terraform GitOps?



Disclaimer - I LOVE Terraform



Terraform

GitOps

Principles

v1.0.0

Founding Members



GitOps Principles

v1.0.0

1 Declarative

A **system** managed by GitOps must have its desired state expressed **declaratively**.

2 Versioned and Immutable

Desired state is **stored** in a way that enforces immutability, versioning and retains a complete version history.





3 Pulled Automatically

Software agents automatically pull the desired state declarations from the source.

4 Continuously Reconciled

Software agents **continuously** observe actual system state and **attempt to apply** the desired state.

Terraform does not comply with all principles

1.  HCL is declarative
2.  Git is used for files, but Terraform has its own state
3.  Terraform CLI is not an agent
4.  Not continuously reconciled by default



Terraform state does not follow GitOps

1. ❌ Source of Truth is NOT GIT
2. ❌ State can be manipulated manually
3. ❌ State is reconciled in the next run only
4. ❌ No guarantee that what is in Git is also in the Terraform state object



Terraform state does not follow GitOps

Developer / Terraform / Terraform CLI / commands / state / rm

Command: state rm

v1.6.x (latest) ▾

The main function of [Terraform state](#) is to track the bindings between resource instance addresses in your configuration and the remote objects they represent. Normally Terraform automatically updates the state in response to actions taken when applying a plan, such as removing a binding for a remote object that has now been deleted.

You can use `terraform state rm` in the less common situation where you wish to remove a binding to an existing remote object without first destroying it, which will effectively make Terraform "forget" the object while it continues to exist in the remote system.

2 Versioned and Immutable

Desired state is **stored** in a way that enforces immutability, versioning and retains a complete version history.

Terraform state horror stories

A Month Later

- trying to prevent accidental cluster deletions by codifying them
- unknowingly modified global state during review builds

Europe
US
Asia

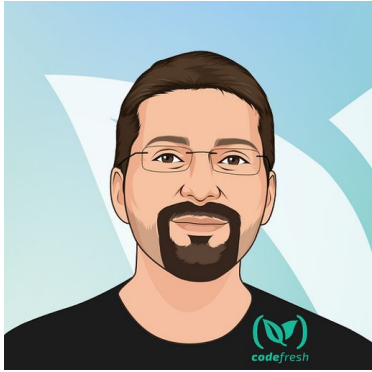
KubeCon | CloudNativeCon
Europe 2019

Play (k)

7:24 / 20:22 • A Month Later > Live video streaming brought to you by: Google Cloud

Keynote: How Spotify Accidentally Deleted All its Kube Clusters with No User Impact - David Xia

Reminder - I LOVE Terraform



Terraform

Can we do better for GitOps?



What we need

1. Native Kubernetes Resources (CRDs)
2. Native Kubernetes Controllers
3. Sync loop (Reconciliation) from agents in the cluster
4. Constant guarantee what was is in Git is also in the external Resources
5. We need Argo CD for Infrastructure and not just application resources

We want to take it to the next level



<https://unsplash.com/photos/pqHRNS8Mojc>

Same world

Kubernetes manifests (YAML)
for applications

Kubernetes manifests (YAML)
for Infrastructure

Same world

Kubernetes manifests (YAML)
for applications

Kubernetes manifests (YAML)
for Infrastructure

GitOps Everywhere

Use the Kubernetes API for
everything

- Drift detection
- Automatic reconciliation
- Native Kubernetes controllers
- Metrics and monitoring
- Common tooling
- Universal workflows



Crossplane

Define any resource in K8s

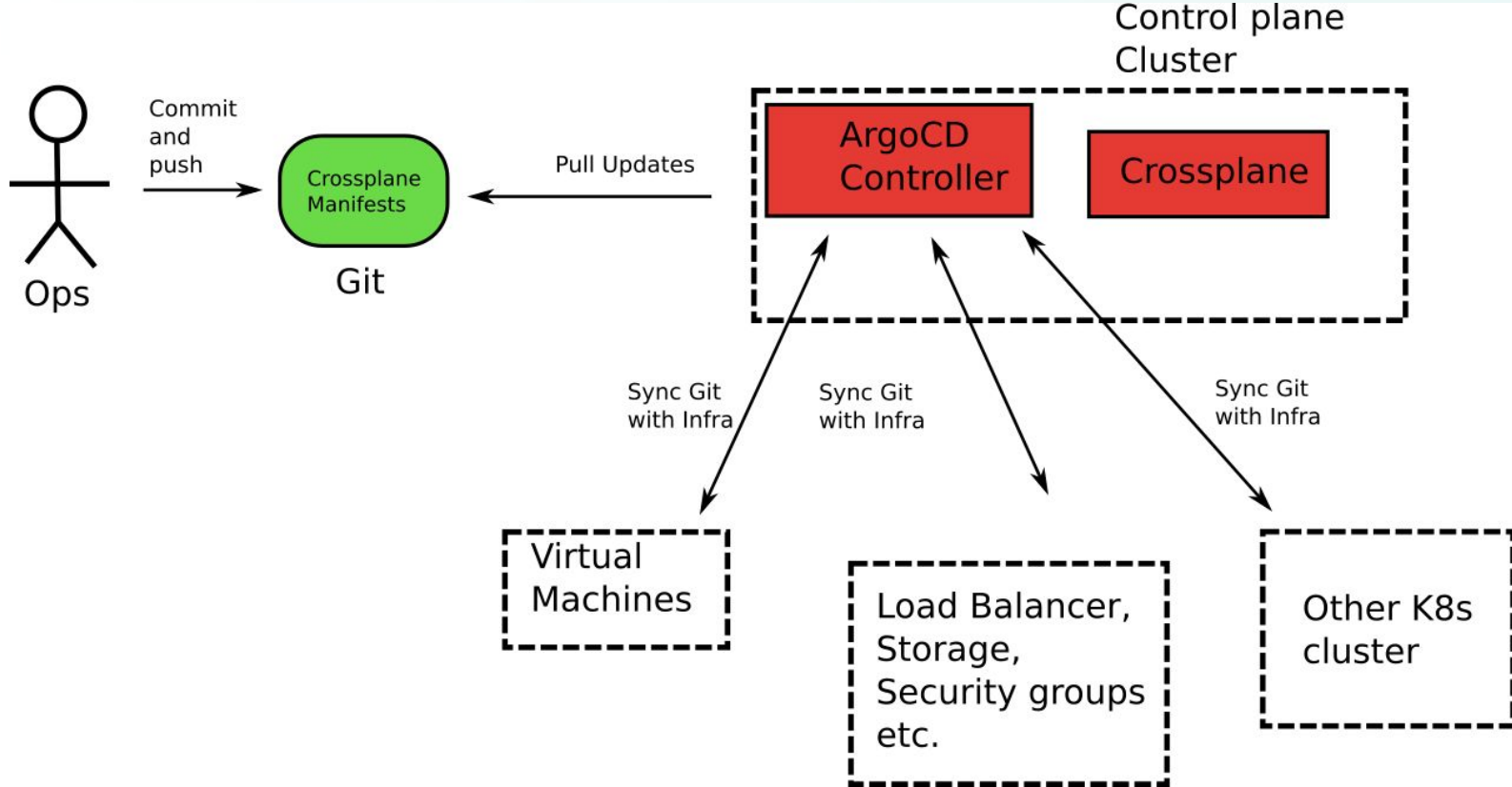
Works for Google, Azure, AWS, Alibaba. You can also write your own provider

```
apiVersion: s3.aws.upbound.io/v1beta1
kind: Bucket
metadata:
  name: my-example-bucket
spec:
  forProvider:
    region: us-east-2
  providerConfigRef:
    name: default
```

```
apiVersion: rds.aws.upbound.io/v1beta1
kind: Instance
metadata:
  annotations:
    meta.upbound.io/example-id: rds/v1beta1/instance
    uptest.upbound.io/timeout: "3600"
  labels:
    testing.upbound.io/example-name: example-dbinstance
name: example-dbinstance-kostis
spec:
  forProvider:
    allocatedStorage: 20
    autoGeneratePassword: true
    autoMinorVersionUpgrade: true
    backupRetentionPeriod: 14
    backupWindow: 09:46-10:16
    engine: postgres
    engineVersion: "13.7"
    instanceClass: db.t3.micro
```

```
apiVersion: ec2.aws.upbound.io/v1beta1
kind: Subnet
metadata:
  name: sample-subnet1
spec:
  forProvider:
    region: us-west-1
    availabilityZone: us-west-1b
    vpcIdRef:
      name: sample-vpc
    cidrBlock: 172.16.10.0/24
```

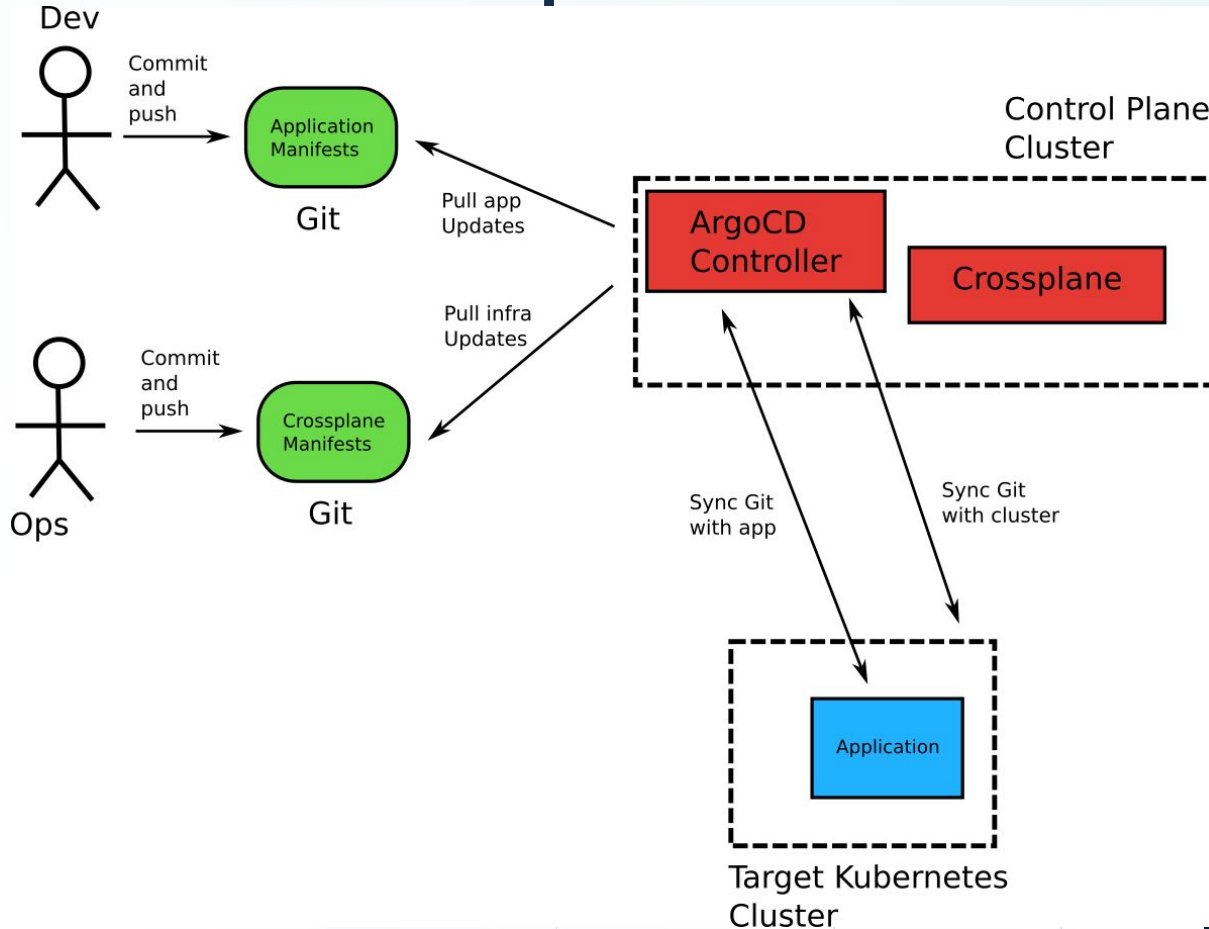
How Crossplane Works



How Crossplane Works

The screenshot displays the Crossplane Applications dashboard. At the top, there's a header with 'Applications' on the left and 'APPLICATIONS' on the right. Below the header, there are navigation buttons for '+ NEW APP' and 'SYNC APPS', a search bar 'Search applications...', and a 'Log out' button. A 'Filters' sidebar on the left shows 'SYNC STATUS' (Unknown: 0, Synced: 3, OutOfSync: 0) and 'HEALTH STATUS' (Unknown: 0, Progressing: 0, Suspended: 0, Healthy: 3, Degraded: 0, Missing: 0). Below the filters, there are input fields for 'LABELS' and 'PROJECTS', with 'crossplane' selected under projects. The main content area shows three application cards, each with a diamond icon and a title. Each card displays metadata (Project, Labels, Status, Repository, Target Rev..., Path, Destination, Namespa...) and three action buttons: SYNC, REFRESH, and DELETE. The 'container-registry' card shows Project: crossplane, Status: Healthy Synced, and Namespa: container. The 'loadbalancer' card shows Project: crossplane, Status: Healthy Synced, and Namespa: lb. The 's3-bucket' card shows Project: crossplane, Status: Healthy Synced, and Namespa: bucket. The top right corner indicates 'Items per page: 10'.

How Crossplane Works





Demo Time

Crossplane and Argo CD

Define any resource in K8s

Even Db Migrations

```
apiVersion: db.atlasgo.io/v1alpha1
kind: AtlasSchema
metadata:
  name: myapp
spec:
  # Load the URL of the target database from a Kubernetes secret.
  urlFrom:
    secretKeyRef:
      key: url
      name: mysql-credentials
  # Define the desired schema of the target database. This can be defined in either
  # plain SQL like this example or in Atlas HCL.
  schema:
    sql: |
      create table users (
        id int not null auto_increment,
        name varchar(255) not null,
        email varchar(255) unique not null,
        short_bio varchar(255) not null,
        primary key (id)
      );
```



 *codefresh*

Apps



upbound

Infra

atlas



atlascloud

DBs

GitOps Everywhere - benefits

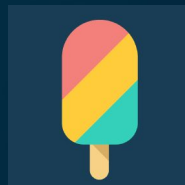
1. Kubernetes native tooling
2. Unified way of describing resources in declarative format
3. Easy auditing (via Git)
4. Avoid configuration drift (or at least detect it early)
5. Implement common workflows (e.g. Git approvals, policy controllers)
6. Gain all benefits of other tools from the ecosystem

Questions?

kostis@codefresh.io



Argo CD



Crossplane



Atlasgo