ariga

codefresh

Updating Databases the GitOps way

ArgoCon US 2023

Your Hosts

Rotem Tamir

CTO Ariga

rotem@ariga.io

Kostis Kapelonis

DevRel Codefresh

kostis@codefresh.io

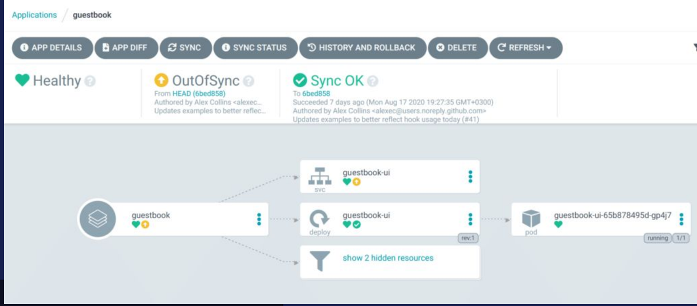
Agenda

- Database migrations and Argo CD
- Existing approaches (init container, jobs, sync hooks)
- What to avoid and best practices
- The Atlas Kubernetes Operator
- Demo

Database upgrades

Past, present, future

A short evolutionary history of app deployment


Applications / guestbook

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

Healthy

OutOfSync
From 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)

GitOps

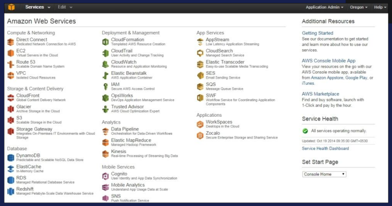
```
# kubectl apply -f stuff.yaml
```

Declarative

```
#!/bin/bash

# Do Stuff
```

Imperative



Amazon Web Services

Compute & Networking
Direct Connect — Dedicated Network Connection to AWS
EC2 — Virtual Servers in the Cloud
Route 53 — Scalable Domain Name System
VPC — Isolated Cloud Resources

Storage & Content Delivery
CloudFront — Global Content Delivery Network
Glacier — Archive Storage in the Cloud
S3 — Scalable Storage in the Cloud
Storage Gateway — Integrates On-Premises IT Environments with Cloud Storage

Database
DynamoDB — Predictable and Scalable NoSQL Data Store
ElastiCache — In-Memory Cache
RDS — Managed Relational Database Service
Redshift — Managed Petabyte-Scale Data Warehouse Service

Deployment & Management
CloudFormation — Templated AWS Resource Creation
CloudTrail — User Activity and Change Tracking
CloudWatch — Resource and Application Monitoring
Elastic Beanstalk — AWS Application Container
IAM — Secure AWS Access Control
OpsWorks — DevOps Application Management Service
Trusted Advisor — AWS Cloud Optimization Expert

Analytics
Data Pipeline — Orchestration for Data-Driven Workflows
Elastic MapReduce — Managed Hadoop Framework
Kinesis — Real-time Processing of Streaming Big Data

Mobile Services
Cognito — User Identity and App Data Synchronization
Mobile Analytics — Understand App Usage Data at Scale
SNS — Push Notification Service

App Services
AppStream — Low Latency Application Streaming
CloudSearch — Managed Search Service
Elastic Transcoder — Easy-to-use Scalable Media Transcoding
SES — Email Sending Service
SQS — Message Queue Service
SWF — Workflow Service for Coordinating Application Components

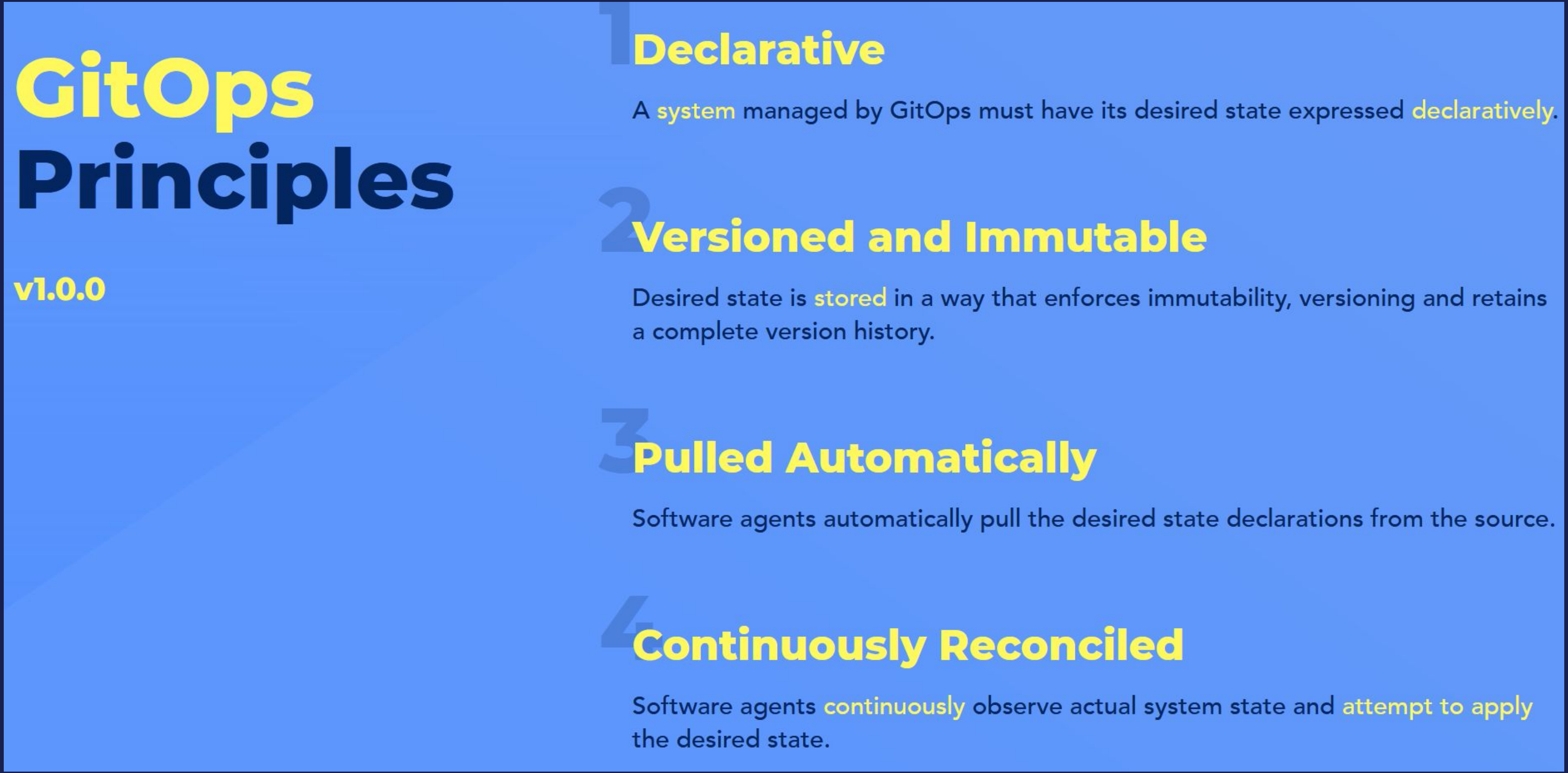
Applications
WorkSpaces — Desktops in the Cloud
Zocalo — Secure Enterprise Storage and Sharing Service

Additional Resources
Getting Started
See our documentation to get started and learn more about how to use our services.
AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.
AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.

Service Health
All services operating normally.
Updated: Oct 19 2014 09:35:00 GMT+0530
Service Health Dashboard

Set Start Page
Console Home

Manual
(ClickOps)



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

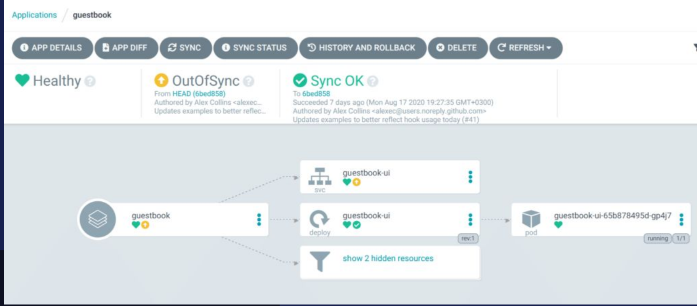
Software agents automatically pull the desired state declarations from the source.

4 Continuously Reconciled

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

opengitops.dev

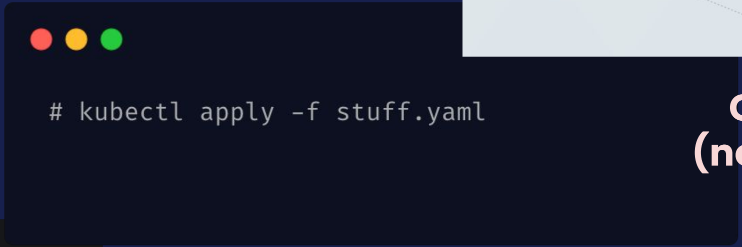
A short evolutionary history of db migrations



Healthy

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

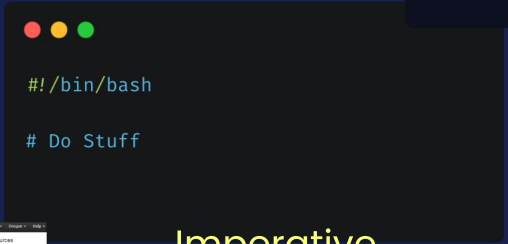
Sync OK
To 6bed858
Succeeded 7 days ago (Mon Aug 17 2020 19:27:35 GMT+0300)
Authored by Alex Collins <alexec@users.noreply.github.com>
Updates examples to better reflect hook usage today (#41)



```
# kubectl apply -f stuff.yaml
```

GitOps (nobody is here)

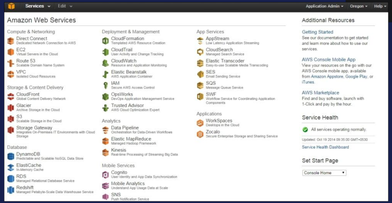
Declarative (some are here)



```
#!/bin/bash

# Do Stuff
```

Imperative (most are here)



Amazon Web Services

Compute & Networking
Direct Connect — Dedicated Network Connection to AWS
EC2 — Virtual Servers in the Cloud
Route 53 — Scalable Domain Name System
VPC — Isolated Cloud Resources

Storage & Content Delivery
CloudFront — Global Content Delivery Network
Glacier — Archive Storage in the Cloud
S3 — Scalable Storage in the Cloud
Storage Gateway — Integrates On-Premises IT Environments with Cloud Storage

Database
DynamoDB — Predictable and Scalable NoSQL Data Store
ElastiCache — In-Memory Cache
RDS — Managed Relational Database Service
Redshift — Managed Petabyte-Scale Data Warehouse Service

Deployment & Management
CloudFormation — Templated AWS Resource Creation
CloudTrail — User Activity and Change Tracking
CloudWatch — Resource and Application Monitoring
Elastic Beanstalk — AWS Application Container
IAM — Secure AWS Access Control
OpsWorks — DevOps Application Management Service
Trusted Advisor — AWS Cloud Optimization Expert

Analytics
Data Pipeline — Orchestration for Data-Driven Workflows
Elastic MapReduce — Managed Hadoop Framework
Kinesis — Real-time Processing of Streaming Big Data

Mobile Services
Cognito — User Identity and App Data Synchronization
Mobile Analytics — Understand App Usage Data at Scale
SNS — Push Notification Service

App Services
AppStream — Low Latency Application Streaming
CloudSearch — Managed Search Service
Elastic Transcoder — Easy-to-use Scalable Media Transcoding
SES — Email Sending Service
SQS — Message Queue Service
SWF — Workflow Service for Coordinating Application Components

Applications
WorkSpaces — Desktops in the Cloud
Zocalo — Secure Enterprise Storage and Sharing Service

Additional Resources

Getting Started
See our documentation to get started and learn more about how to use our services.

AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.

AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.

Service Health
All services operating normally.
Updated: Oct 19 2014 09:35:00 GMT+0530
Service Health Dashboard

Set Start Page
Console Home

Manual (a lot of organizations)

My app deployments

My db upgrades

Anti-patterns

What NOT to do

Anti-pattern I

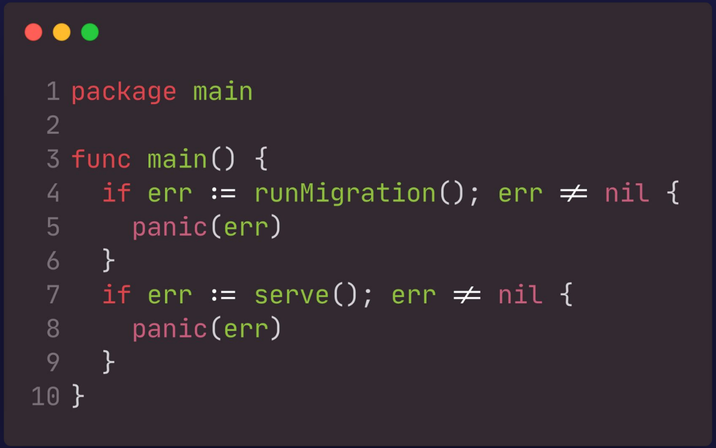
Running migrations manually

Manual DB migrations - avoid

1. **Error prone**
2. **Slowest link in the chain**
3. **Not repeatable, not auditable**
4. **Stressful**

Anti-pattern II

Migrations during startup

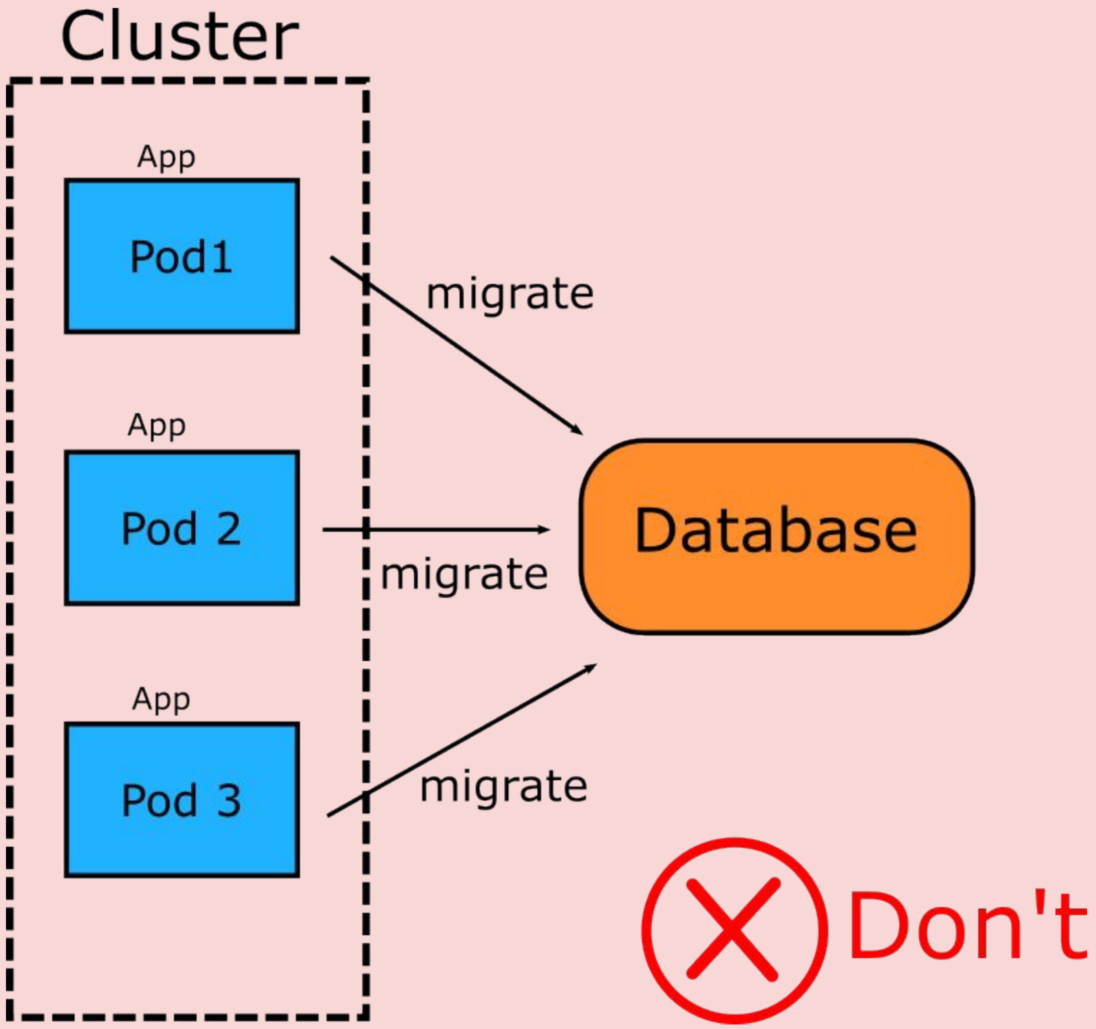


```go
package main

func main() {
  if err := runMigration(); err != nil {
    panic(err)
  }
  if err := serve(); err != nil {
    panic(err)
  }
}
```

Run migrations on app init - Avoid

1. **Attack surface** - do not bundle an extra tool and its dependencies in your app container, use separate credentials for DDL and runtime.
2. **When migrations fail** - reduced capacity, crash-looping
3. **Migrations must be synchronized** - this means that effectively only one replica can init at any given time.


Cluster
App
Pod1
App
Pod 2
App
Pod 3
migrate
migrate
migrate
Database
Don't

What to DO

Automate DB upgrades

Automate DB upgrades

1. **Treat schema versions as artifacts**
2. **Handle DB upgrades like infra (or app) changes**
3. **Have full control over DB upgrades (and auditing)**
4. **DB migrations are a discrete step**
5. **Give db upgrades the same respect as app/infrastructure**

Let’s discuss

Database migrations with Kubernetes

Options for Kubernetes/Argo CD

1. During application startup (avoid)
2. Use Init containers (meh)
3. Use Kubernetes Jobs (meh)
4. Use Helm hooks or Pre-sync Argo CD hooks (meh)
5. Use a GitOps Operator for DBs (recommended)

Init-containers

❌ Packaging a CLI tool (not K8s native)

❌ Decoupled from application startup

❌ Failed migrations leave app in unknown state

❌ No visibility/No auditing

Kubernetes Jobs

❌ Packaging a CLI tool (not K8s native)

✅ Decoupled from application startup

❌ Hard to associate/correlate with apps

❌ No visibility on what happened

✅ Auditing

Helm/Argo CD Hooks

❌ Packaging a CLI tool (not K8s native)

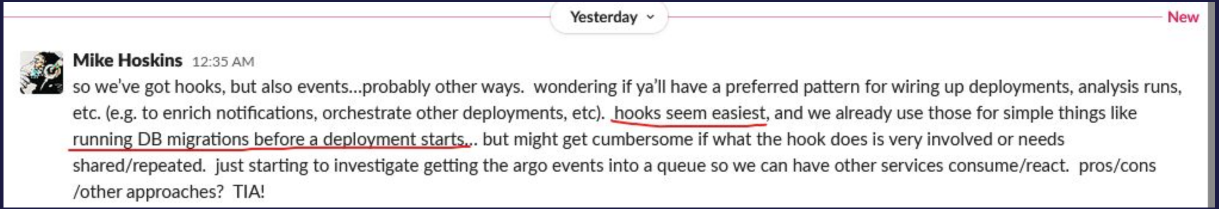
✅ Decoupled from application startup

❌ Issues with re-syncs

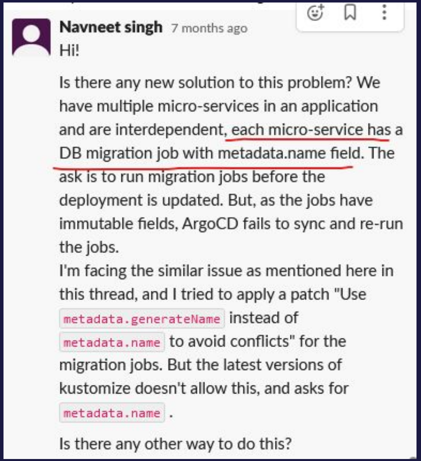
❌ No visibility on what happened

❌ May not be stored in Git

People are looking for a K8s native solution

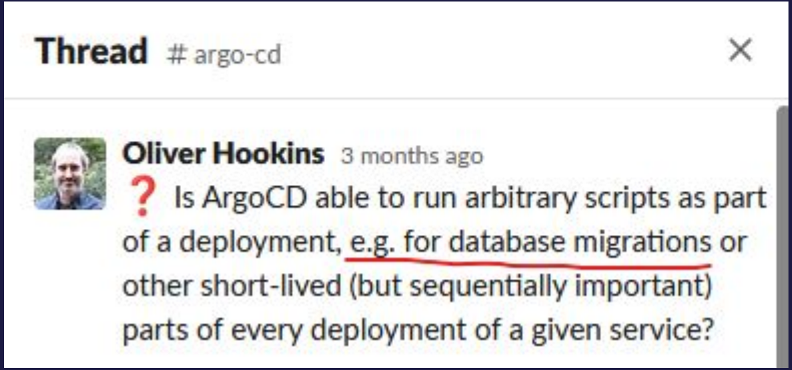

Yesterday New

Mike Hoskins 12:35 AM
so we've got hooks, but also events...probably other ways. wondering if ya'll have a preferred pattern for wiring up deployments, analysis runs, etc. (e.g. to enrich notifications, orchestrate other deployments, etc). hooks seem easiest, and we already use those for simple things like running DB migrations before a deployment starts... but might get cumbersome if what the hook does is very involved or needs shared/repeated. just starting to investigate getting the argo events into a queue so we can have other services consume/react. pros/cons /other approaches? TIA!

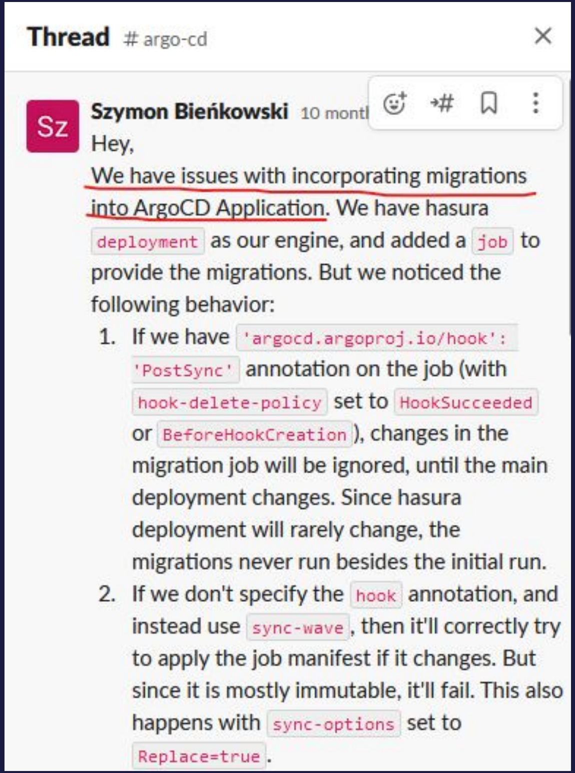

Navneet singh 7 months ago
Hi!

Is there any new solution to this problem? We have multiple micro-services in an application and are interdependent, each micro-service has a DB migration job with metadata.name field. The ask is to run migration jobs before the deployment is updated. But, as the jobs have immutable fields, ArgoCD fails to sync and re-run the jobs.
I'm facing the similar issue as mentioned here in this thread, and I tried to apply a patch "Use `metadata.generateName` instead of `metadata.name` to avoid conflicts" for the migration jobs. But the latest versions of kustomize doesn't allow this, and asks for `metadata.name` .

Is there any other way to do this?


Thread # argo-cd

Oliver Hookins 3 months ago
❓ Is ArgoCD able to run arbitrary scripts as part of a deployment, e.g. for database migrations or other short-lived (but sequentially important) parts of every deployment of a given service?


Thread # argo-cd

Szymon Bieńkowski 10 mont
Hey,
We have issues with incorporating migrations into ArgoCD Application. We have hasura `deployment` as our engine, and added a `job` to provide the migrations. But we noticed the following behavior:

1. If we have `'argocd.argoproj.io/hook': 'PostSync'` annotation on the job (with `hook-delete-policy` set to `HookSucceeded` or `BeforeHookCreation`), changes in the migration job will be ignored, until the main deployment changes. Since hasura deployment will rarely change, the migrations never run besides the initial run.
2. If we don't specify the `hook` annotation, and instead use `sync-wave`, then it'll correctly try to apply the job manifest if it changes. But since it is mostly immutable, it'll fail. This also happens with `sync-options` set to `Replace=true`.

GitOps for databases

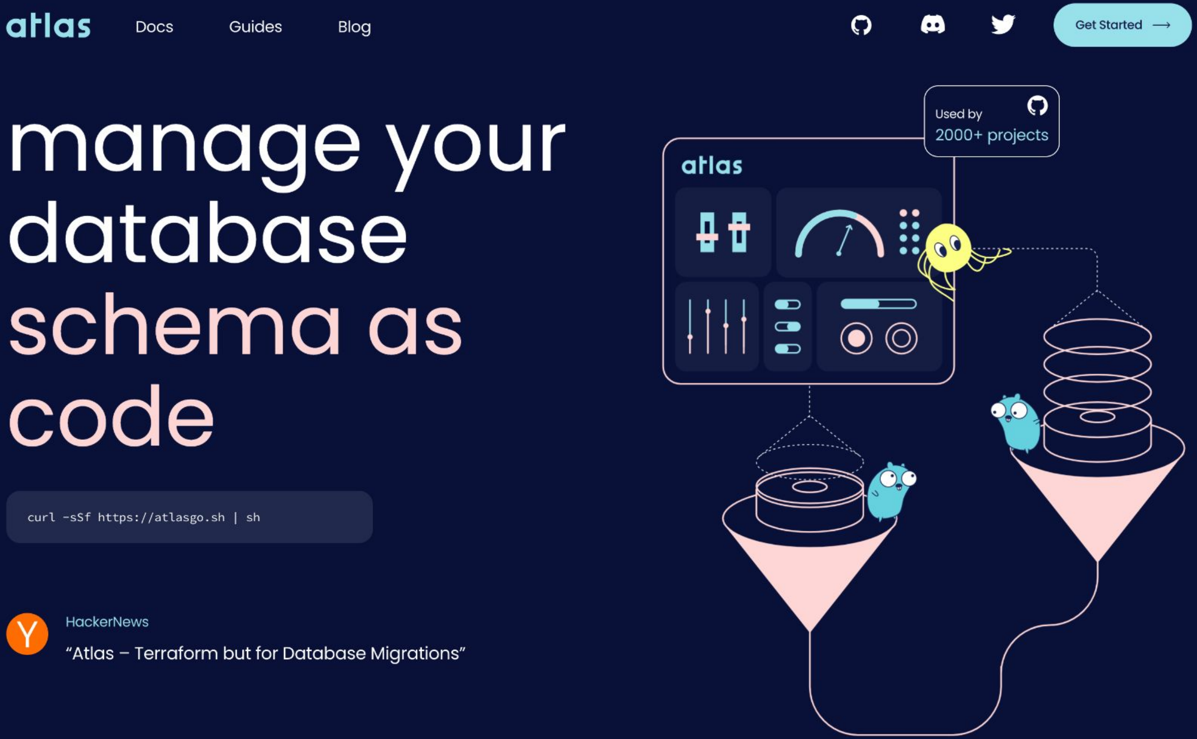
Meet Atlas

open-sourced
2021

stargazers
4k

contributors
+65

Projects using on GitHub
2.4k

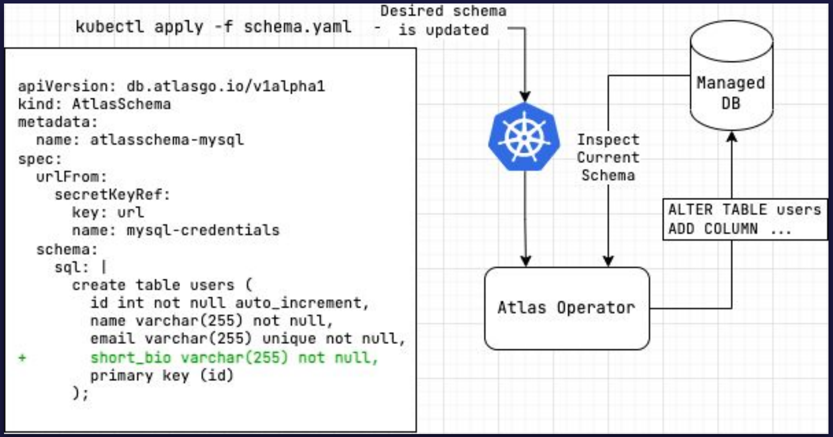


manage your database schema as code

```
curl -sSf https://atlasgo.sh | sh
```

HackerNews
"Atlas – Terraform but for Database Migrations"

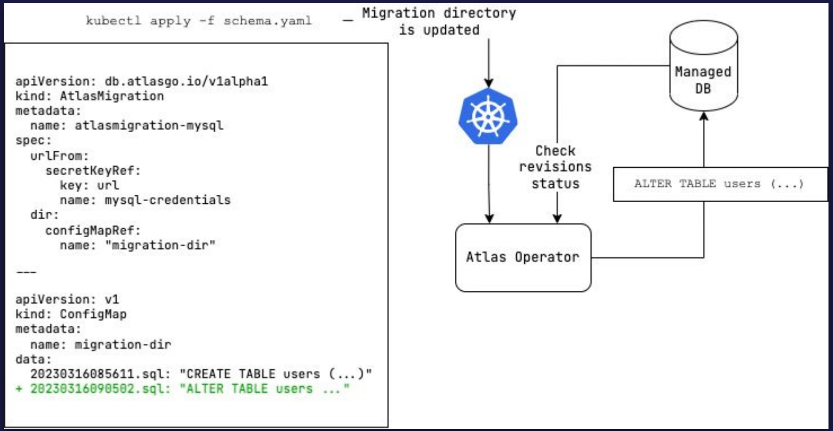
Introducing the Atlas Operator



```
kubectl apply -f schema.yaml
```

```
apiVersion: db.atlasgo.io/v1alpha1
kind: AtlasSchema
metadata:
  name: atlasschema-mysql
spec:
  urlFrom:
    secretKeyRef:
      key: url
      name: mysql-credentials
  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)
      );
```

```
ALTER TABLE users
ADD COLUMN ...
```



```
kubectl apply -f schema.yaml
```

```
apiVersion: db.atlasgo.io/v1alpha1
kind: AtlasMigration
metadata:
  name: atlasmigration-mysql
spec:
  urlFrom:
    secretKeyRef:
      key: url
      name: mysql-credentials
  dir:
    configMapRef:
      name: "migration-dir"
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: migration-dir
data:
  20230316085611.sql: "CREATE TABLE users (...)"
+ 20230316090502.sql: "ALTER TABLE users ..."
```

```
ALTER TABLE users (...)
```

Features

- ✅ Native Operator
- ✅ Versioned Migrations (`AtlasMigration`)
- ✅ Declarative Migrations (`AtlasSchema`)
- ✅ MySQL, Postgres, SQLite, SQLServer...
- ✅ Safety + Policies

Operators for GitOps: Why?

1. **Resilience.** A reconciliation loop is more resilient than retrying a script.
2. **Semantics.** A CRD extends the Kubernetes API. It's *.spec* can be validated and manipulated, it's *.status* can be observed and consumed.
3. **Operations.** Codifying domain expertise and multi step decision trees.

"We can wrap existing schema management solutions into containers, and run them in Kubernetes as Jobs.

But that is SILLY. That is not how we work in Kubernetes."

-Viktor Farcic, DevOps ToolKit

Atlas Operator Demo

Running Migrations with an Operator

✅ The Kubernetes-native way!

✅ Decouples migrations from your app as a discrete step

✅ Supports safety features to prevent bad changes

✅ Exposes a clear migration status/health check

✅ 100% GitOps Automation for your DB schema

The Trinity

atlas

codefresh

upbound

atlascloud

Apps

Infra

DBs

Wrapping up

- The Atlas Operators is a Kubernetes native solution for DB upgrades
- It defines dedicated K8s Resources for migrations
- It's open source! github.com/ariga/atlas-operator
- Can use either (imperative) or (declarative)
- Treat your DBs as infrastructure

Questions?

rotem@ariga.io

kostis@codefresh.io

atlasgo.io

argoproj.github.io

crossplane.io

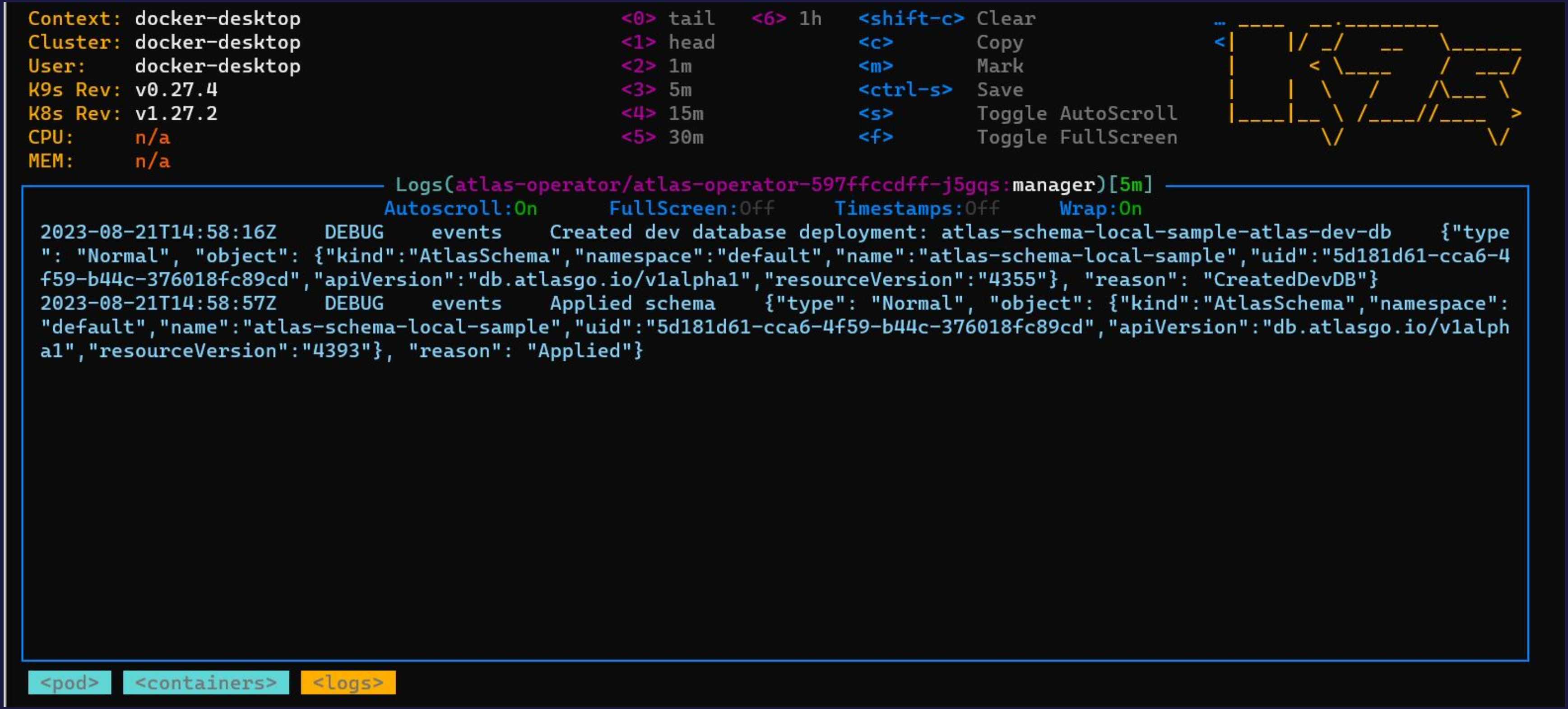
Do you want more?

Backup slides

Rollbacks are B.S

- Most tools advocate for pre-planning “down” migrations.
- But practically, NO ONE uses them in production. Why?
- Dealing with partial failures? Really drop?
- The answer: declarative roll-forward
- The future: integration into the operator model

Preview/Validate changes in “dev” DB



```
Context:  docker-desktop                        <0> tail    <6> 1h    <shift-c> Clear
Cluster:  docker-desktop                        <1> head              <c>       Copy
User:     docker-desktop                        <2> 1m                <m>       Mark
K9s Rev:  v0.27.4                               <3> 5m                <ctrl-s>  Save
K8s Rev:  v1.27.2                               <4> 15m               <s>       Toggle AutoScroll
CPU:      n/a                                   <5> 30m               <f>       Toggle FullScreen
MEM:      n/a
                      Logs(atlas-operator/atlas-operator-597ffccdff-j5gqs:manager)[5m]
                     Autoscroll:On     FullScreen:Off     Timestamps:Off     Wrap:On
2023-08-21T14:58:16Z    DEBUG    events    Created dev database deployment: atlas-schema-local-sample-atlas-dev-db    {"type
": "Normal", "object": {"kind":"AtlasSchema","namespace":"default","name":"atlas-schema-local-sample","uid":"5d181d61-cca6-4
f59-b44c-376018fc89cd","apiVersion":"db.atlasgo.io/v1alpha1","resourceVersion":"4355"}, "reason": "CreatedDevDB"}
2023-08-21T14:58:57Z    DEBUG    events    Applied schema    {"type": "Normal", "object": {"kind":"AtlasSchema","namespace":
"default","name":"atlas-schema-local-sample","uid":"5d181d61-cca6-4f59-b44c-376018fc89cd","apiVersion":"db.atlasgo.io/v1alph
a1","resourceVersion":"4393"}, "reason": "Applied"}

<pod>  <containers>  <logs>
```