



Building a self-service DBaaS for your Internal Developer Platform



Dan Mckean

Sr Product Manager, MongoDB



George Hantzaras

Engineering Director, MongoDB

Conf42: Cloud Native 2023

Agenda

- Kubernetes as the platform of platform
- Building and managing a DBaaS in your Internal Developer Platform
- Why build a DBaaS (and the risks)
- The criticality of enabling self-service in a DBaaS
- Atlas and our Atlas Operator
- Putting it all together and a demo

Kubernetes as a Platform of Platforms





Why Kubernetes?

95% of Internal Developer Platforms (IDPs) are built on top of Kubernetes.

Kubernetes is an open-source container-orchestration system for automating application deployment, scaling, and management.

<https://internaldeveloperplatform.org/>



Why Kubernetes?

It offers:

- Highly flexible networking
- Storage orchestration
- High availability
- Self-healing
- Easier migration between k8s-compliant vendors
- High degree of customization & extensibility
(Kubernetes Operators and Custom resources)



Kubernetes Operators & Custom Resources

An Operator extends the native Kubernetes control plane with custom logic that helps manage a lot of the essential tasks that are bespoke to a specific product...
(Like [MongoDB](#))


It's usually paired with Custom Resources, defined in Kubernetes using Custom Resource Definitions.

These custom resources allow for the creation of new types of Kubernetes Object, which can be monitored by the Operator, which can use them to take action...



Kubernetes Operators - External Resources

- Operators can also be used to manage resources external to Kubernetes
- Often done using custom resources in the cluster and the Operator calling the external service APIs to implement changes
- Allows the use of the same tooling, processes, permissions, and automation as deployments within Kubernetes



Building and managing a DBaaS in your IDP



TLDR; IDP

- Built to enable developer self-service of platform infrastructure
- Typically built by an Ops teams, and used by developers
- Provides a common process and method of engaging with the platform, often via templates
- This automates recurring tasks such as spinning up environments and resources and helps enforcing standards (e.g. security)
- IDPs Often abstract away the complexity of the underlying platform technologies from the developer - saving everyone from needing to be an expert
- Development teams gain autonomy by being empowered to spin up fully provisioned environments and manage them with a minimum of effort or complexity
- IDPs can be built or bought



TLDR; DBaaS

- Database as a Service is often one of the most critical IDP components
- Most applications need a database, and databases can be complex to deploy and manage
- Simplifying creation and management of databases becomes incredibly valuable - especially day 2 operations



DBaaS - Complexity and risks

- How much configuration to expose?
 - Security, sizing, performance, backup, sharding, resilience...
- Troubleshooting
- Enabling development teams to self-serve



Self-service

- Self-service is nearly always faster
- Central teams can focus on support and improvements
- Self-service empowers users
- Common methods:
 - Published assets (e.g. Helm charts)
 - GitOps
 - Portal or marketplace

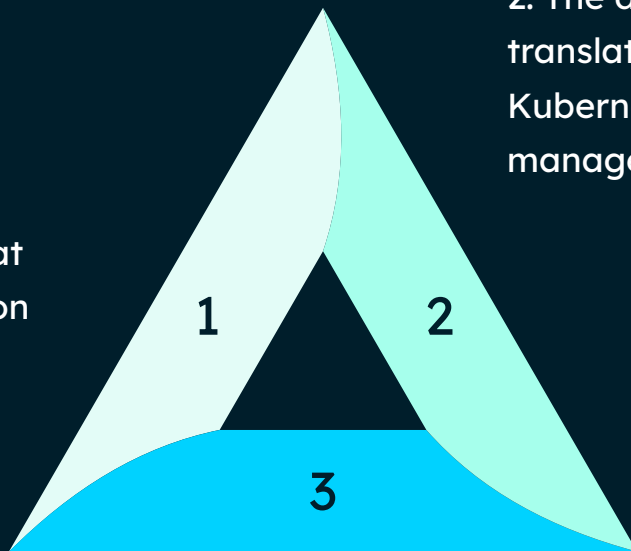
Tools for building our DBaaS





How does it work

1. It start with the developer defining what database his application needs

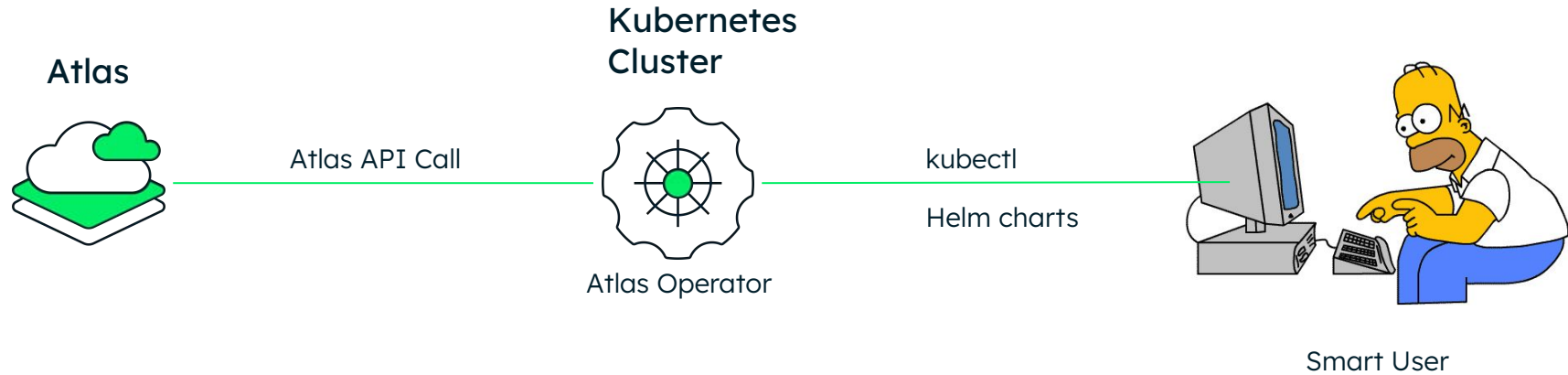


2. The database request is translated into a resource that Kubernetes can deploy and manage

3. Kubernetes creates our database deployment

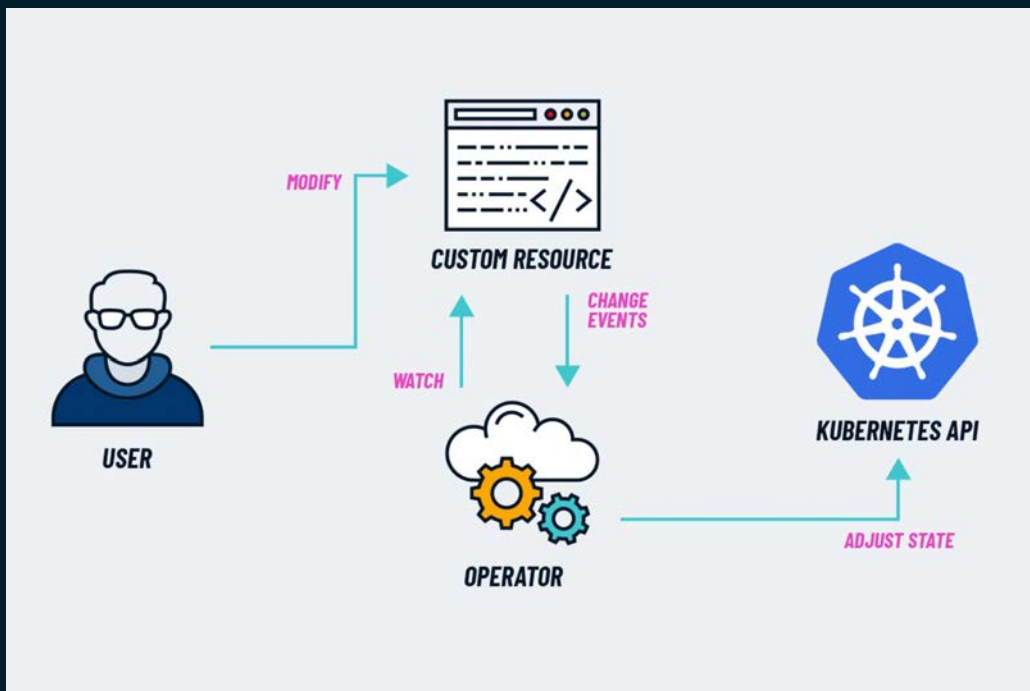


Atlas Kubernetes Operator





Under the hood





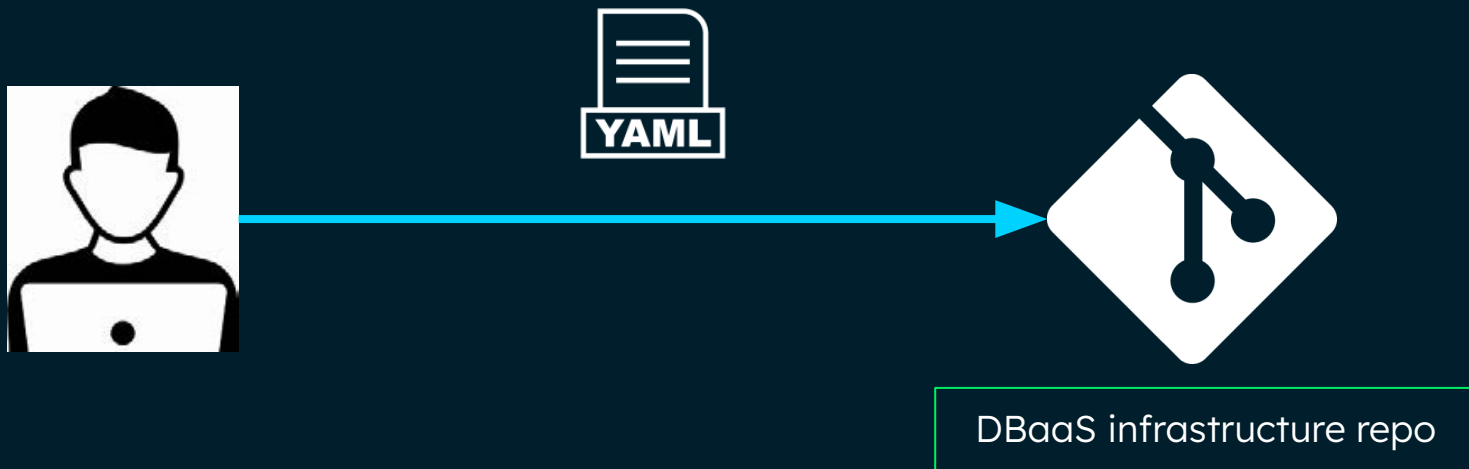
OPERATOR

Deploying a cloud database

```
apiVersion: atlas.mongodb.com/v1
kind: AtlasDeployment
metadata:
  name: my-atlas-cluster
  labels:
    app.kubernetes.io/version: 1.6.0
spec:
  projectRef:
    name: my-project
  deploymentSpec:
    name: "Test-cluster"
    providerSettings:
      instanceSizeName: M10
      providerName: AWS
      regionName: US_EAST_1
```

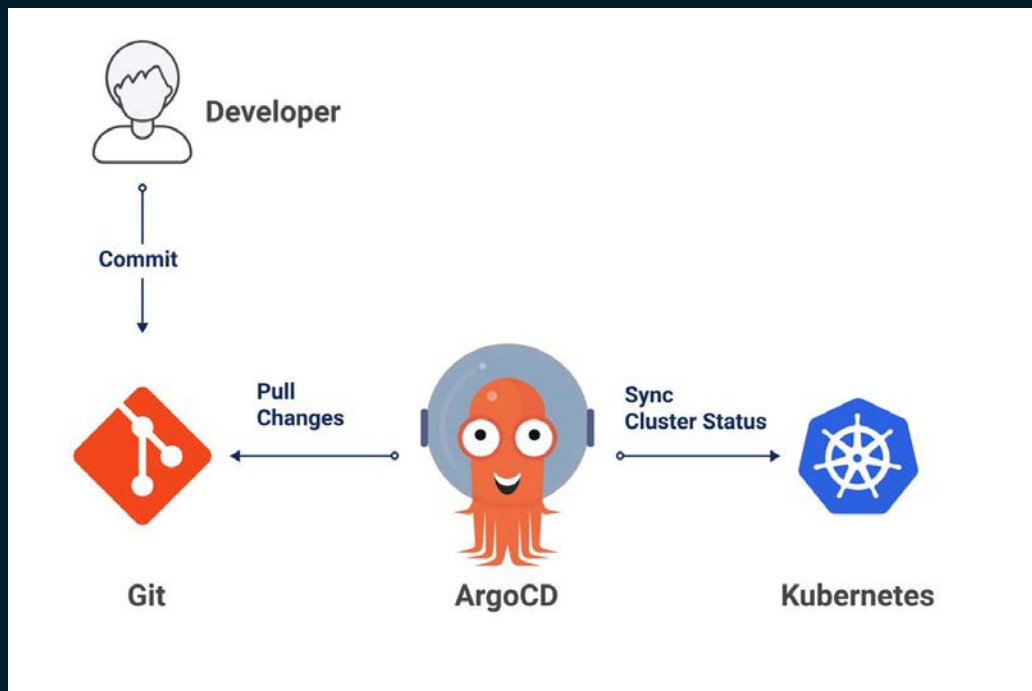


Database request is defined and pushed





ArgoCD for building a self-service experience





ArgoCD for building a self-service experience

```
Applications CREATE CANCEL  
  
+ NEW APP  
  
1  apiVersion: argoproj.io/v1alpha1  
2  kind: Application  
3  metadata:  
4    name: DBaaS  
5  spec:  
6    destination:  
7      name: ''  
8      namespace: database  
9      server: 'https://kubernetes.default.svc'  
10   source:  
11     path: /  
12     repoURL: github.com/mongodb/example-dbaas/  
13     targetRevision: HEAD  
14   sources: []  
15   project: default  
16   syncPolicy:  
17     automated:  
18       prune: true  
19       selfHeal: true  
20   syncOptions:  
21     - CreateNamespace=true  
22
```

Putting it all together





Prerequisites

Setup Atlas

- Atlas account
- Atlas API key

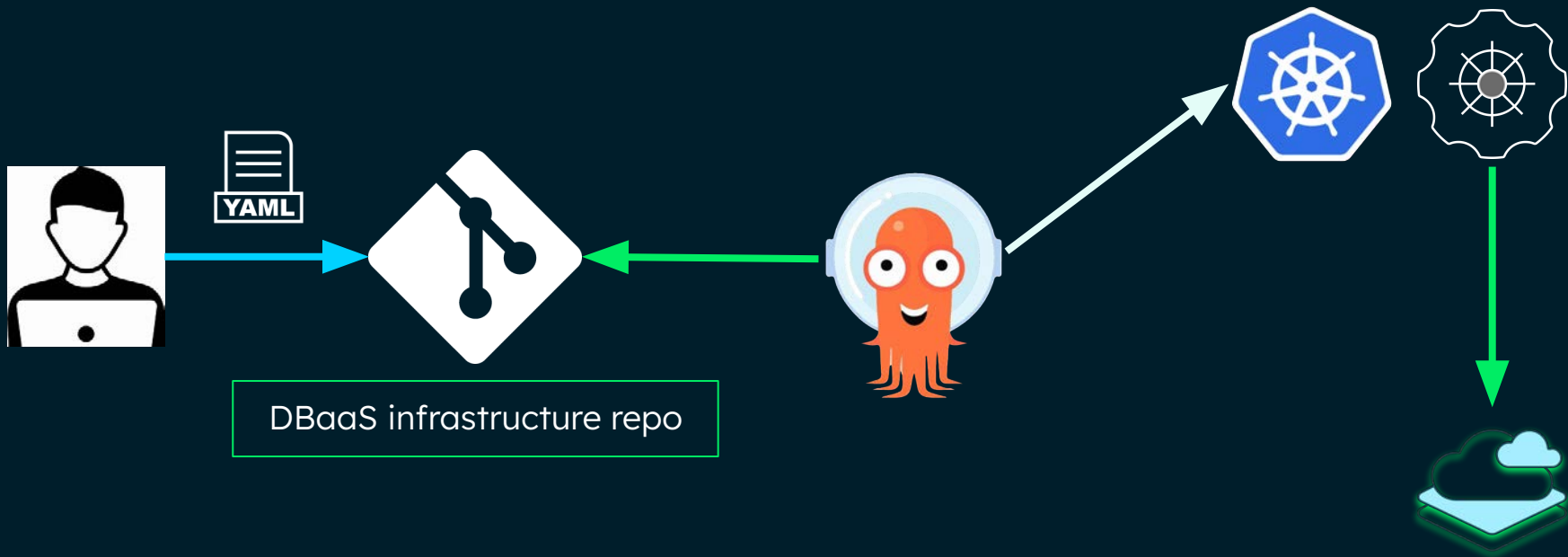
Setup Kubernetes

- Create a Kubernetes cluster
- Install the Atlas Kubernetes Operator

Setup ArgoCD

- Install ArgoCD
- Create a dedicated Infrastructure as Code repository
- Create an Argo application to watch the repository

What your DBaaS look like



Thank you for
your time.

