

Building git push workflows for Kubernetes



hasura.io

Hi!

Tanmai Gopal

Co-founder, hasura.io

@tanmaigo



What is a git-push workflow

- Write code
- `git push heroku master` to deploy

Changed the world for developers, because it used dev only tools (git). Reduced *unnecessary* abstractions.

kubectl + git. Everything can be built around these 2 systems. Build your own git-push workflows. The main goal is to simplify devops and “pipelines”.

The simplest DevOps task: Build & deploy

I have source code on my machine. I can run it locally.

I want to deploy my source code at current commit.

When git push:

- Build: Dockerfile tagged with commit
- Deploy: Apply changes to kubernetes deployment with new image tag

Before <> after

<pre>\$ docker build -t registry.com/my-image:my-tag</pre>	
<pre>\$ docker push registry.com/my-image:my-tag</pre>	<pre>\$ git push dev master</pre>
<pre>\$ kubectl set image deployment/my-deployment container=registry.com/my-image:my-tag</pre>	

Git hooks for “git push”

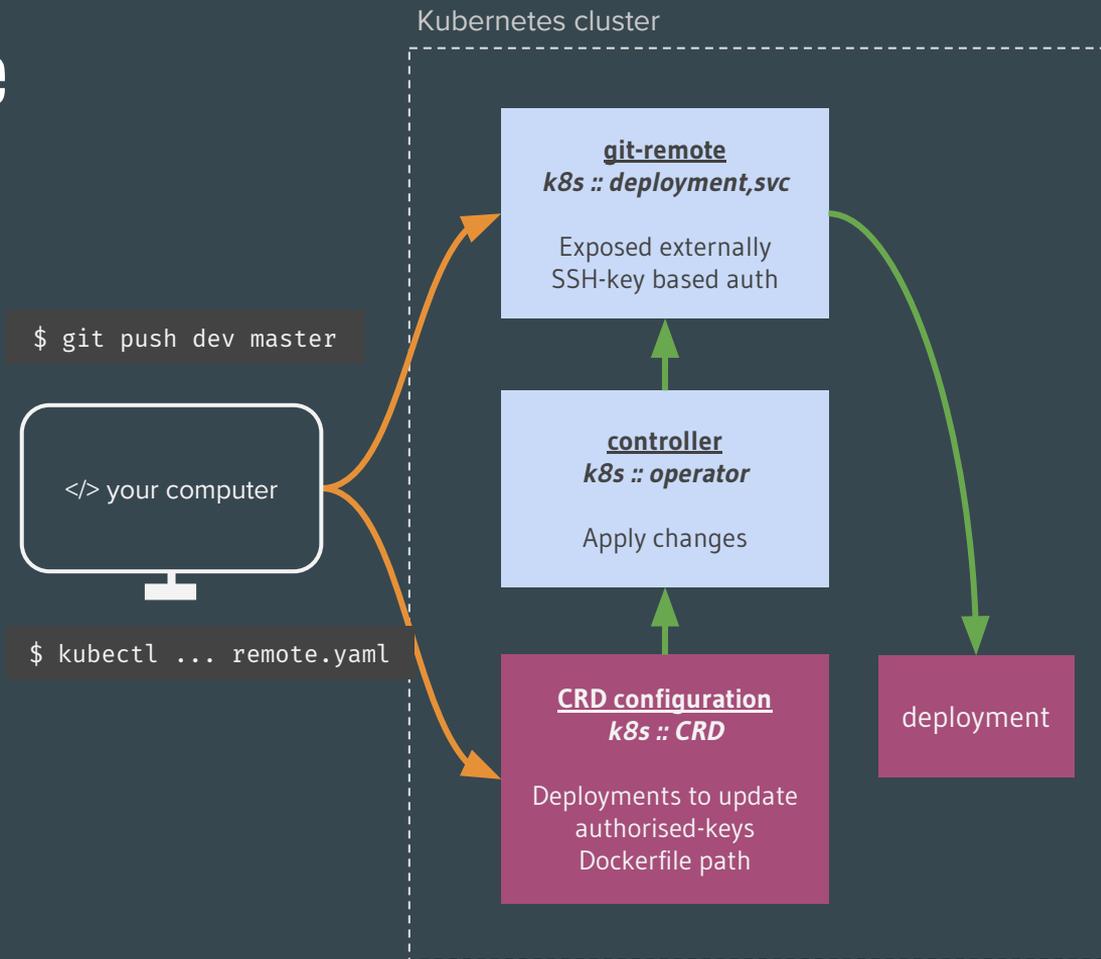
client-side	
> pre-push	Exit can abort push
server-side	
> pre-receive	Exit can abort push
> post-receive	Cannot abort push

Executable script in `.git/hooks/` named `<hook>`

`.git/hooks/pre-receive`

The obvious architecture

- Git remote agent on your cluster. Pre-receive hook:
 - Build docker image
 - Apply to k8s deployment
- Configuration:
 - SSH keys
 - Dockerfile path
- Sync configuration changes with the git-remote agent



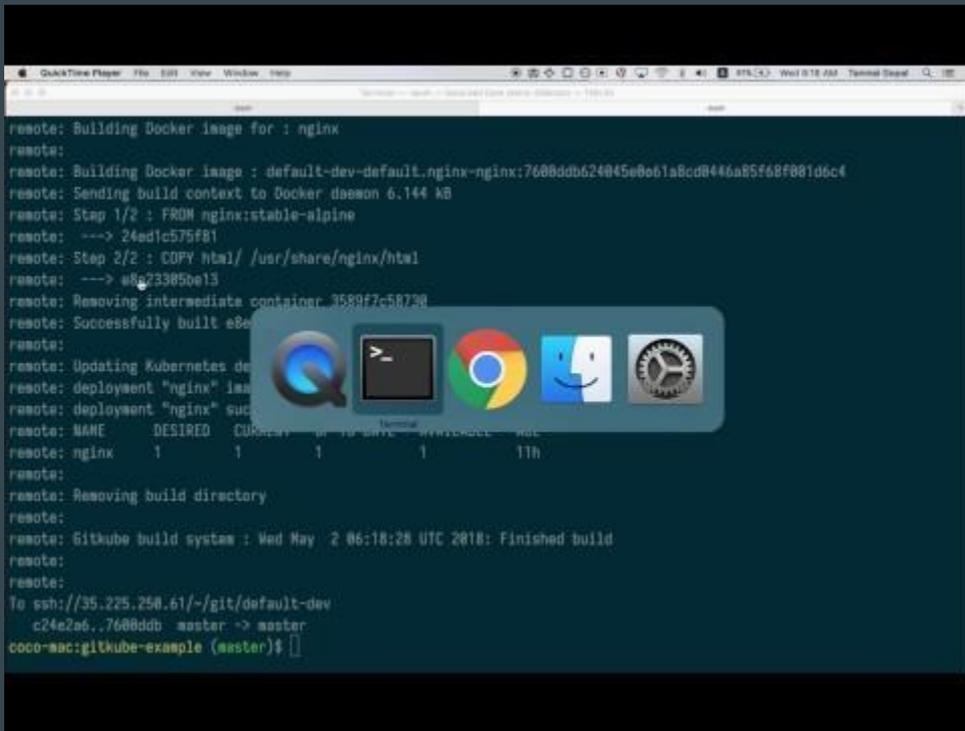
Setup

```
kubectl create -f
```

```
apiVersion: gitkube.sh/v1alpha1
kind: Remote
metadata:
  name: dev
  namespace: default
spec:
  deployments:
  - name: nginx
    containers:
    - name: nginx
      path: .
      dockerfile: Dockerfile
  authorizedKeys:
  - "ssh-rsa <key>"
```

Demo 1: git push dev master to deploy an HTML webpage

```
remote: Building Docker image for : nginx
remote:
remote: Building Docker image : default-dev-default.nginx-nginx:7608ddb624045e0e61a8cd0446a85f68f081d6c4
remote: Sending build context to Docker daemon 6.144 kB
remote: Step 1/2 : FROM nginx:stable-alpine
remote: ----> 24ed1c575f81
remote: Step 2/2 : COPY html/ /usr/share/nginx/html
remote: ----> a8a233850e13
remote: Removing intermediate container 3589f7c58730
remote: Successfully built e8e...
remote:
remote: Updating Kubernetes deployment "nginx" image
remote: deployment "nginx" successfully updated
remote: NAME          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
remote: nginx          1         1         1             1           11h
remote:
remote: Removing build directory
remote:
remote: Gitkube build system : Wed May 2 06:18:28 UTC 2018: Finished build
remote:
remote:
To ssh://35.225.250.61/~ /git/default-dev
c24e2a6..7608ddb master -> master
coco-mac:gitkube-example (master)$
```

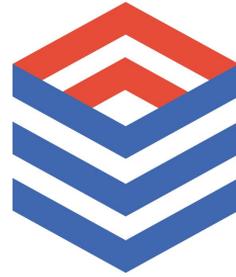


The pre-receive hook

https://github.com/hasura/gitkube/blob/master/build/gitkubed/pre_receive.sh

Run on any kubernetes cluster

Customise and extend for your own
use-case



GITKUBE

★ Star 1,673

<https://github.com/hasura/gitkube>

Benefits

- A developer only needs to have git.
- Customise the hook and just do you.
 - Your “hooks” have direct access to the cluster environment for your deployment tasks
 - Eg: Use secretKeyRefs to provide custom docker build args

We're onto something here...

Extend this idea to more devops tasks?

git push to:

1. Build and run unit-tests
2. Deploy code
3. Deploy configuration
4. Apply stateful migrations
5. Run integration tests

But all based on just one idea

`git push` to apply changes to kubernetes objects.

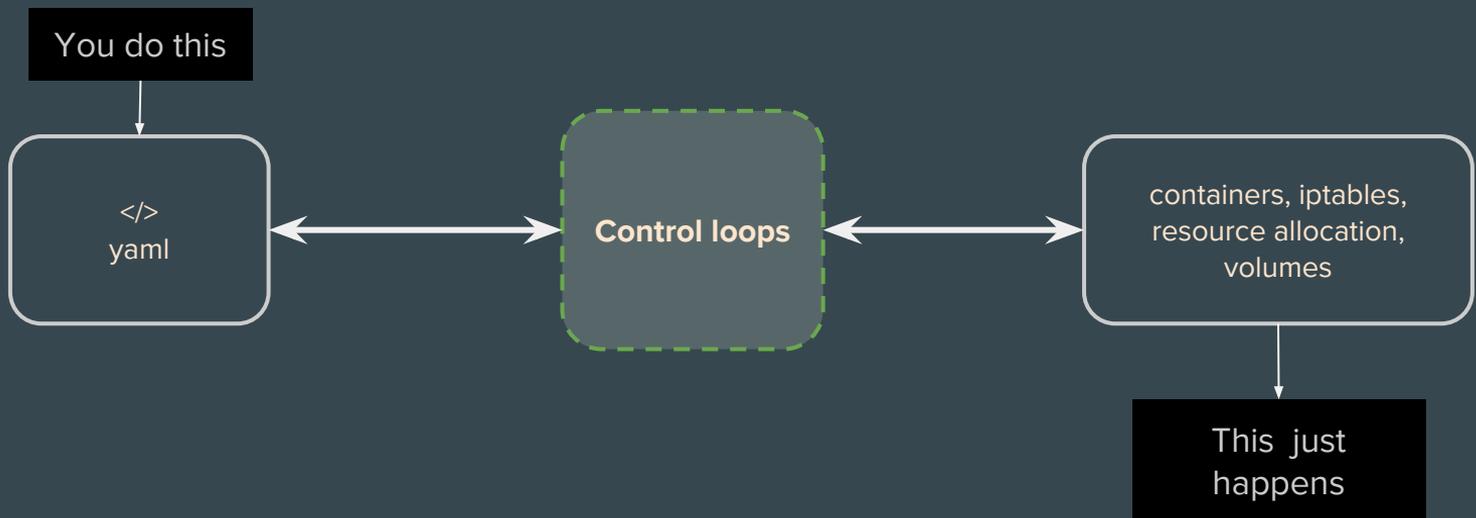
This way, git just works for AllTheThings™

```
git checkout <commit>
```

```
git push dev my-branch:master
```

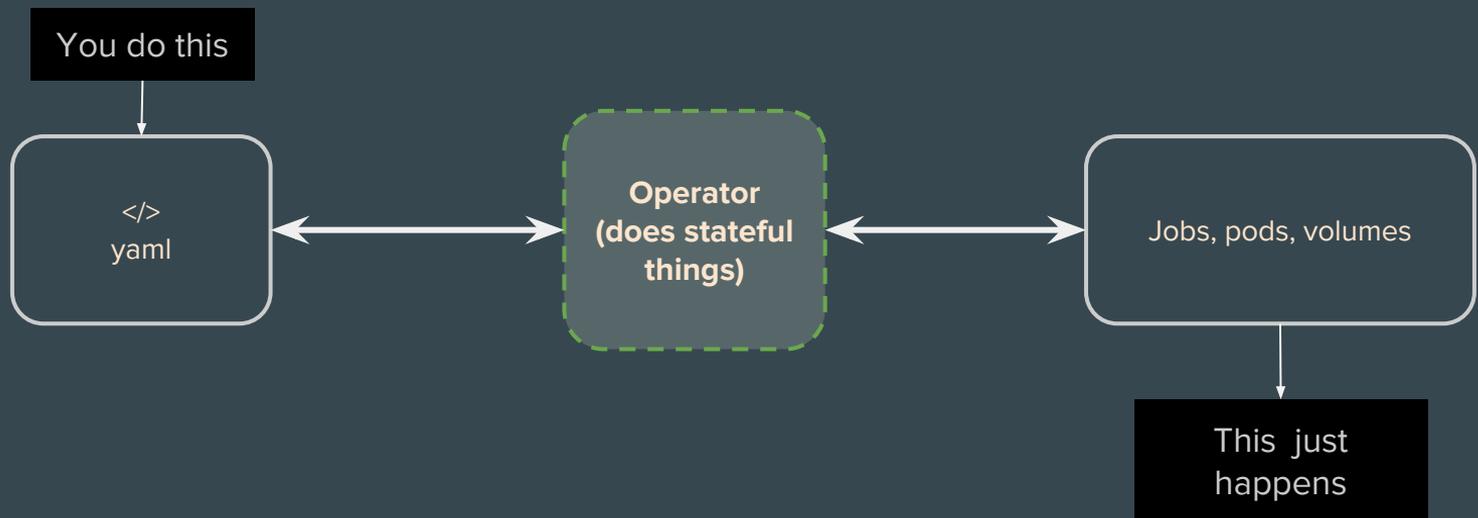
Kubernetes controller

The most awesome thing about how kubernetes works:

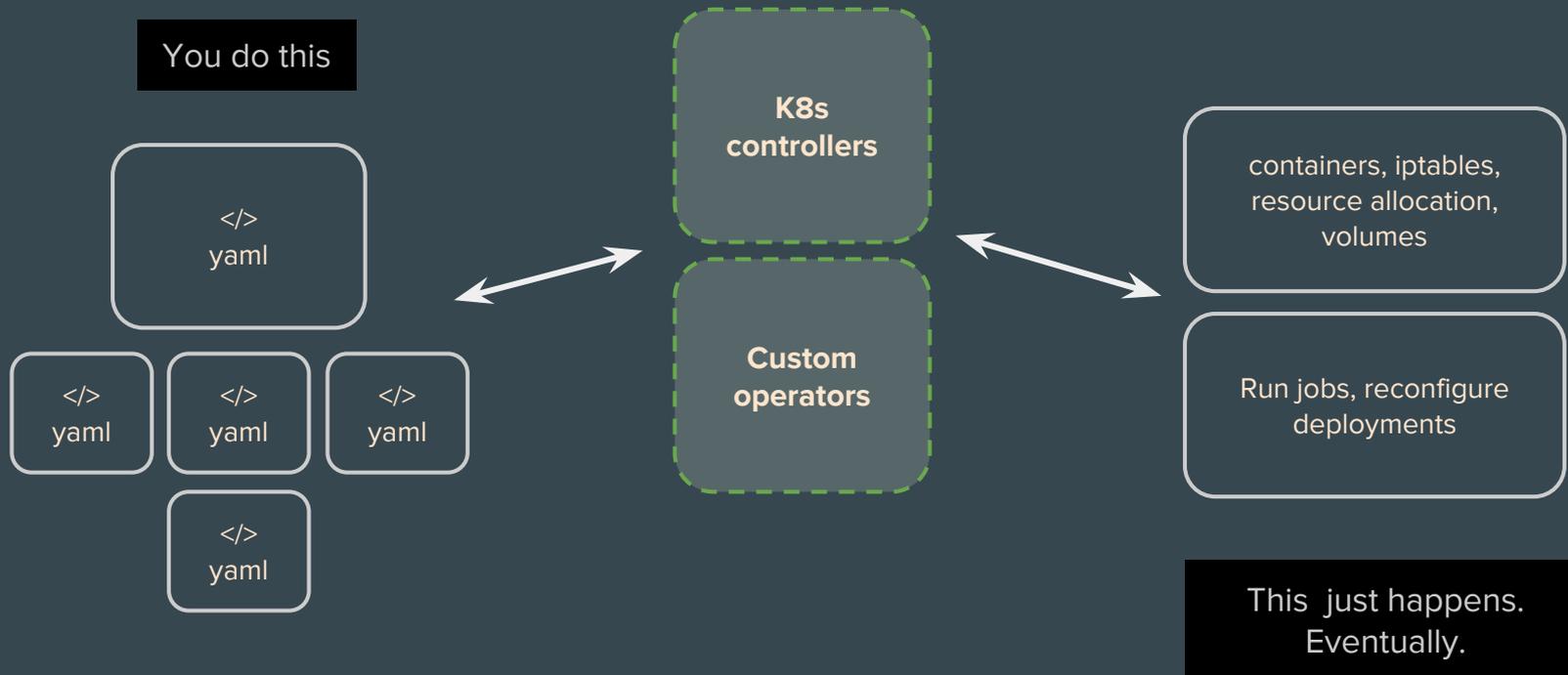


And everything is moving in this direction. Eg: The CRD + operator pattern

CRD + operator pattern



There is no notion of “sequence”



A typical DevOps pipeline needs to:

Build & test: Source code + unit tests

Deploy: Update configuration, run stateful tasks

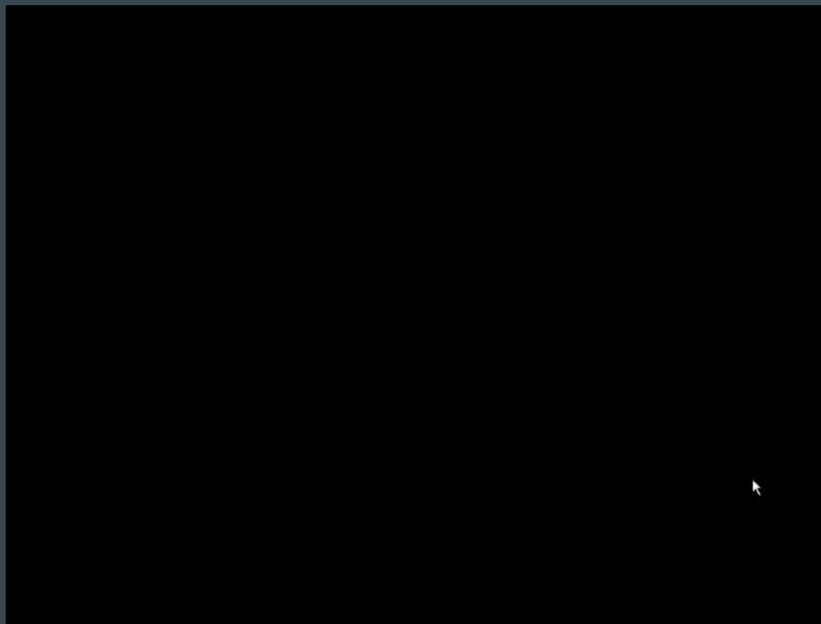
Integration tests: Test microservice1's dependency on microservice2

A typical DevOps pipeline needs to:

Build + run unit-tests	Dockerfile
Production build (artifacts)	Multi-stage dockerfile
Deploy configuration	Update kubernetes manifests
Run stateful tasks (database migrations)	Update CRs
Run integration tests	Run jobs with init-containers to check if microservices are ready

Goodbye “pipelines”?

If everything is a kubernetes manifest backed by operators, then everything is declarative. Note: this is not talking about human-in-loop governance type pipelines, just the idea of sequenced tasks. Governance pipelines are a separate concern.



Demo 4: Applying postgres migrations on git push

```
Terminal Shell Edit View Window Help
The Shell - ssh@104 - Standard Dev's ubuntu-2204-01 - 142.228
ssh@104:~/code (base) (ssh@104:~/code)

icoco-mac:hello-nginx (master) $ ls
README.md  clusters.yaml  cosf          kasura.yaml  microservices  migrations

icoco-mac:hello-nginx (master) $ cp migrations/* migrations/
icoco-mac:hello-nginx (master) $ ls migrations/
1588137194_create_table_author.up.yaml          1521796141514_Update_permission_anonymous_table_article.up.yaml
1588137249_create_table_article.up.yaml        1521796147675_Update_permission_anonymous_table_article.up.yaml
1588137279_alter_table_article_add_foreign_key.up.yaml  1521715394822_run_sql_migration.up.yaml
1588137364_add_relationship_author_table_article.up.yaml  1521715398488_add_existing_table_or_view_author_rating.up.yaml
1588137377_add_relationship_article_table_author.up.yaml  1521715442948_create_relationship_author_table_author_rating.up.yaml
1588138348_insert_sample_data.up.sql           1522192731574_Update_permission_user_table_author.up.yaml
1589889881_modify_permission_anonymous_table_article.up.yaml  1522192743061_Update_permission_user_table_article.up.yaml
1589889821_modify_permission_anonymous_table_author.up.yaml  1522196838468_Delete_permission_anonymous_table_article.up.yaml
1521691939496_Delete_permission_anonymous_table_article.up.yaml  1522196837445_Delete_permission_anonymous_table_author.up.yaml
1521796148638_Update_permission_user_table_article.up.yaml     152221886748_create_relationship_rating_table_author.up.yaml

icoco-mac:hello-nginx (master) $ ls
README.md  clusters.yaml  cosf          kasura.yaml  microservices  migrations

icoco-mac:hello-nginx (master) $ git add migrations/
icoco-mac:hello-nginx (master *) $ git commit -as 'add migrations'
```

Things that are hard with GitOps

- Secrets
 - Must be applied without committing to git
 - *pre-push hook?*
- Templating
 - Helm
 - Kubernetes native templating
- Releases, canary deployments?
 - GitOps with istio :)

Gitkube roadmap



- Easy to write custom hooks
 - Write in any language
 - Boilerplate/plugins
- UI to see past “pushes”
- `git push ≡ /vendor/webhook`

gitkube.sh

[@tanmaigo](https://twitter.com/tanmaigo)

hasura.io

[@HasuraHQ](https://twitter.com/HasuraHQ)



SU-C30