Kubernetes volume plugins Flocker

@agonzalezro

richd77@flickr

Who am I.

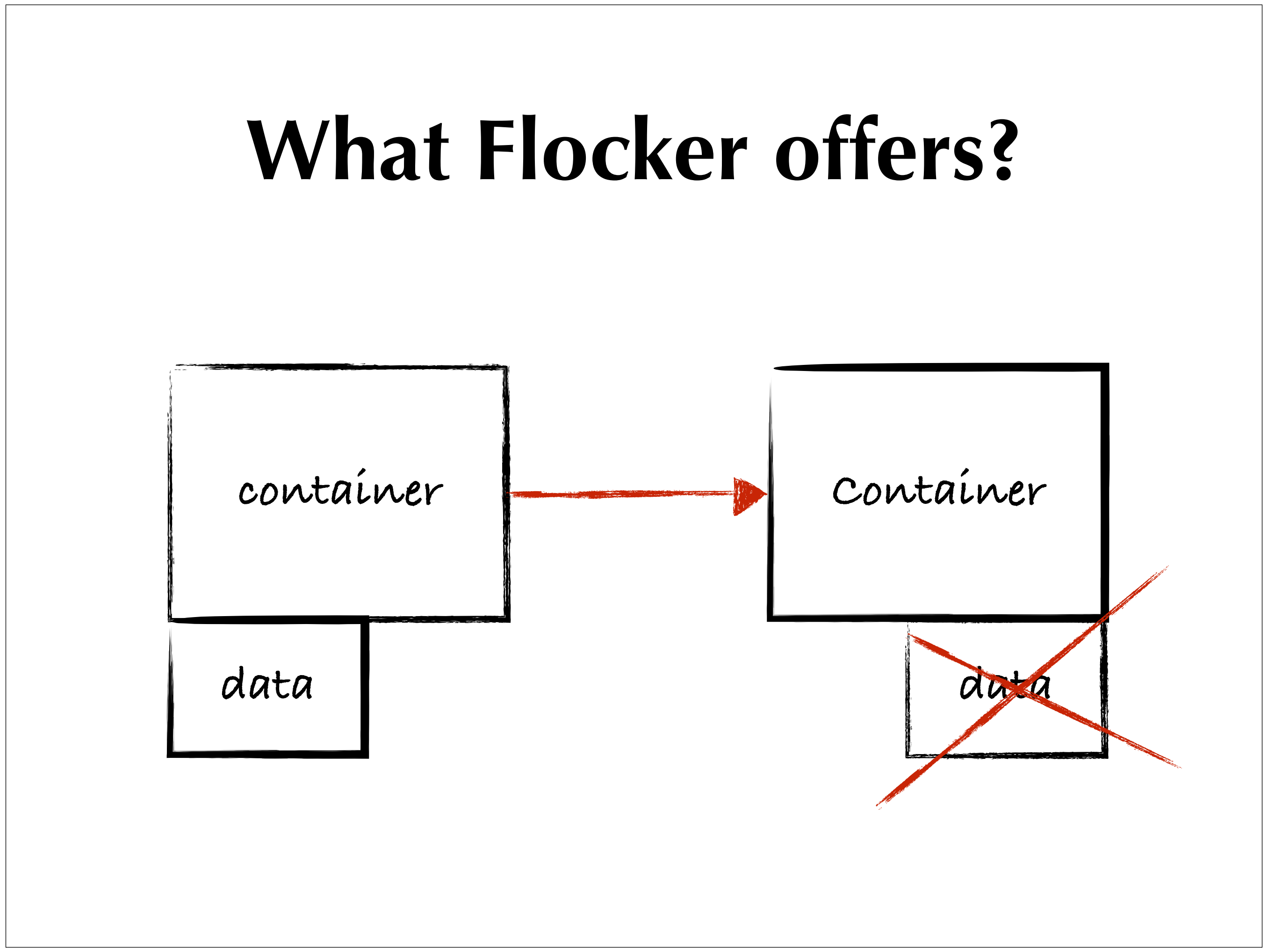
Flocker™ by ClusterHQ™

JETSTACK©

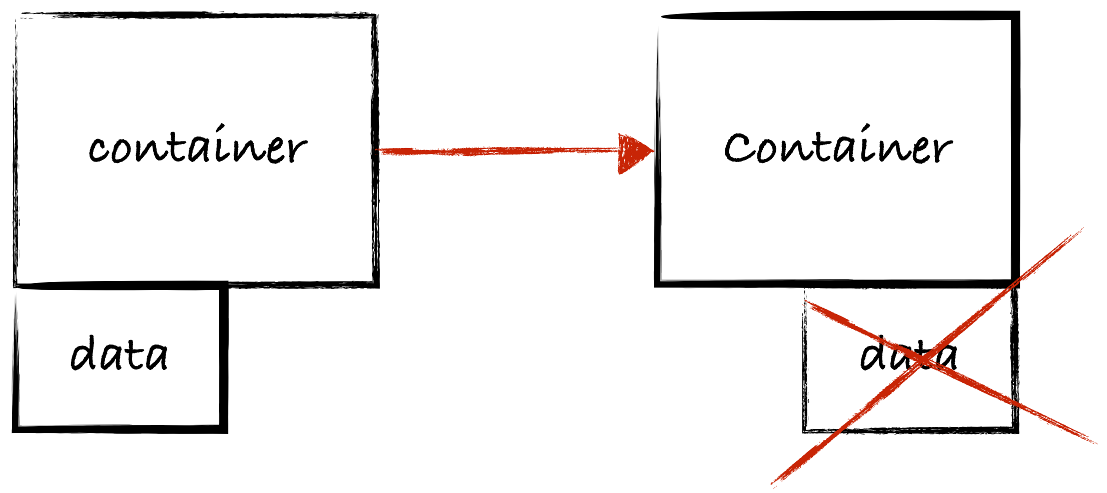
Who I’ve been working with.

TOC

I suspect that if you are in a k8s meetup you already know probably more than me about k8s.
Summary: things that went well, things that didn’t & the future.



What Flocker offers?



How many people has used it?
Portable containers even with state
Horizontal scaling made easy, you can move the containers and the data goes with them
This is good by itself, but there is more…

What else does it offer?

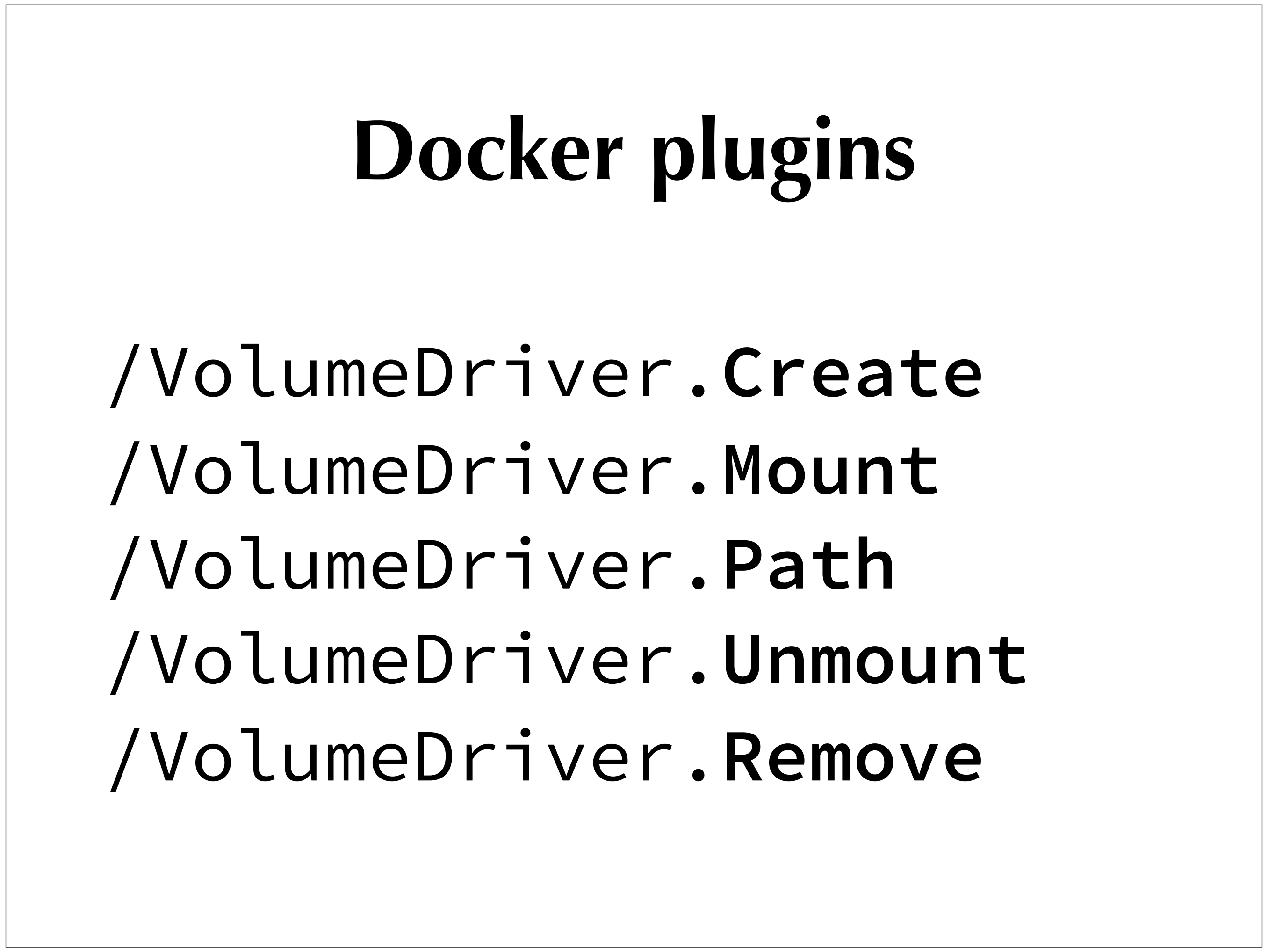
- AWS EBSRackspace Cloud Block Storage
- Anything that supports the OpenStack Cinder API
- EMC ScaleIO
- EMC XtremIO
- Local storage using our ZFS driver (currently Experimental)

All this backed storages

Flocker Docker plugin

HTTP Rest API with twisted

Why is it different from k8s plugin?
Twisted: async framework for Python



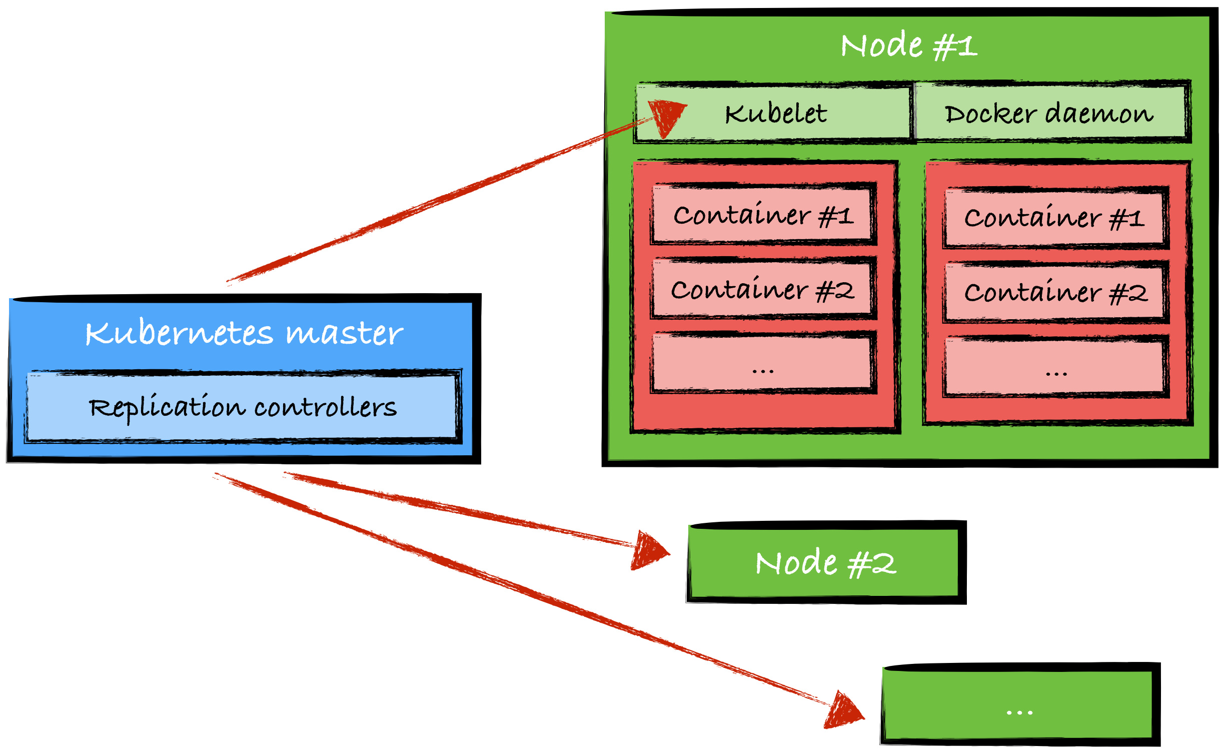
Docker plugins

```
/VolumeDriver.Create
/VolumeDriver.Mount
/VolumeDriver.Path
/VolumeDriver.Unmount
/VolumeDriver.Remove
```

We are going to see later that there are similarities

Ok, how do we fit things together using Kubernetes?
Pretty difficult to keep containers up & running

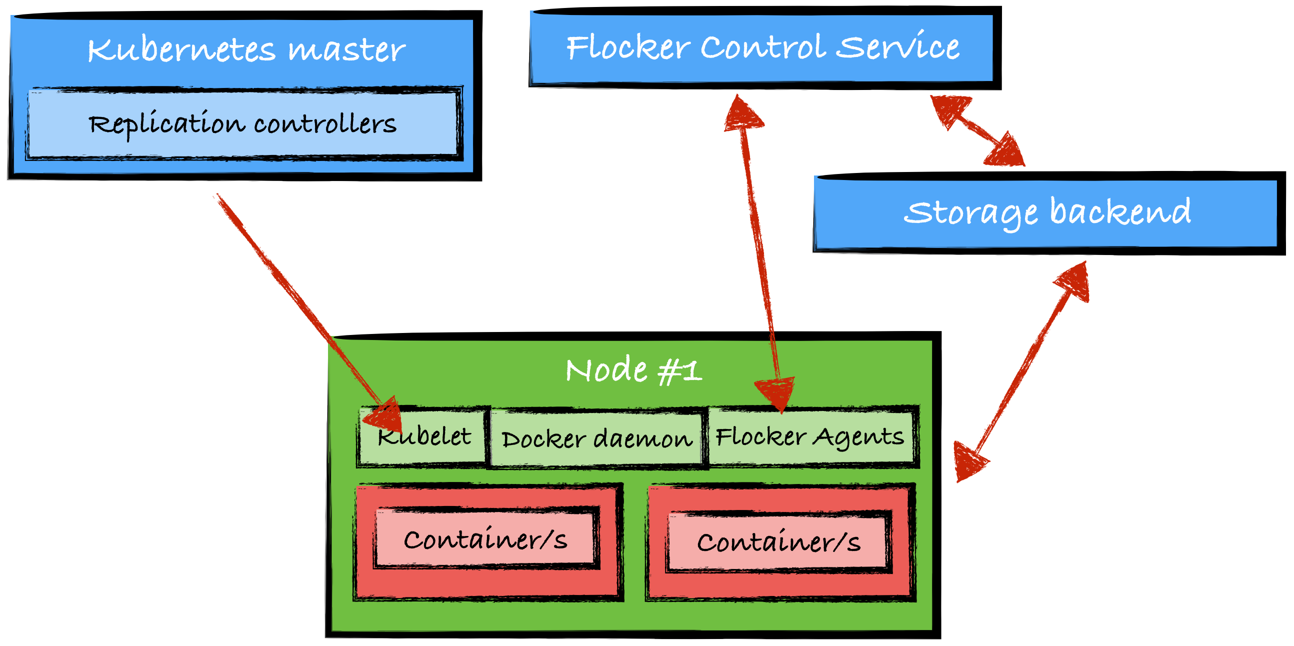
k8s architecture



Simplified architecture of k8s.

The kubelet is the primary "node agent": takes pod specs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy.
Docker daemon -> could be rocket, that's one of the ideas behind k8s.

How does Flocker fit in?



Flocker Control Service: provides a REST over HTTP API to modify the desired configuration of the cluster.
Flocker Dataset Agent: responsible for (de)attachment and (un)mounting. Checking current state and desired state from Control Service.
Flocker Container Agent: not used here, but needed for the cluster. It modifies the cluster state to match the desired configuration (Kubelete does it here)

ENV vars with host/port and certificates.

How to use it (1/2):

Create volume with the flocker-cli:

```
$ uft-flocker-volumes \
    --control-service=172.16.255.250
    create --node=43a06d55 \
    -m name=my-flocker-vol -s 10G

$ uft-flocker-volumes \
    --control-service=172.16.255.250
    list-nodes
```

Explain why we need to create the volume (SetUp)

How to use it (2/2):

```
spec:
  containers:
    - name: web
      image: nginx
      ports:
        - name: web
          containerPort: 80
      volumeMounts:
        - name: www-root
          mountPath: "/usr/share/nginx/html"
  volumes:
    - name: www-root
      flocker:
        datasetName: my-flocker-vol
```

Don’t forget to mention about the env vars.
You could have said that it’s a hostpath, the magic happens with “flocker”

Get ready before swimming into the code

Before going to code!

Volume vs PersistentVolume

Examples: gcePersistentDisk, awsElasticBlockStore, gitRepo (cool example!), etc…

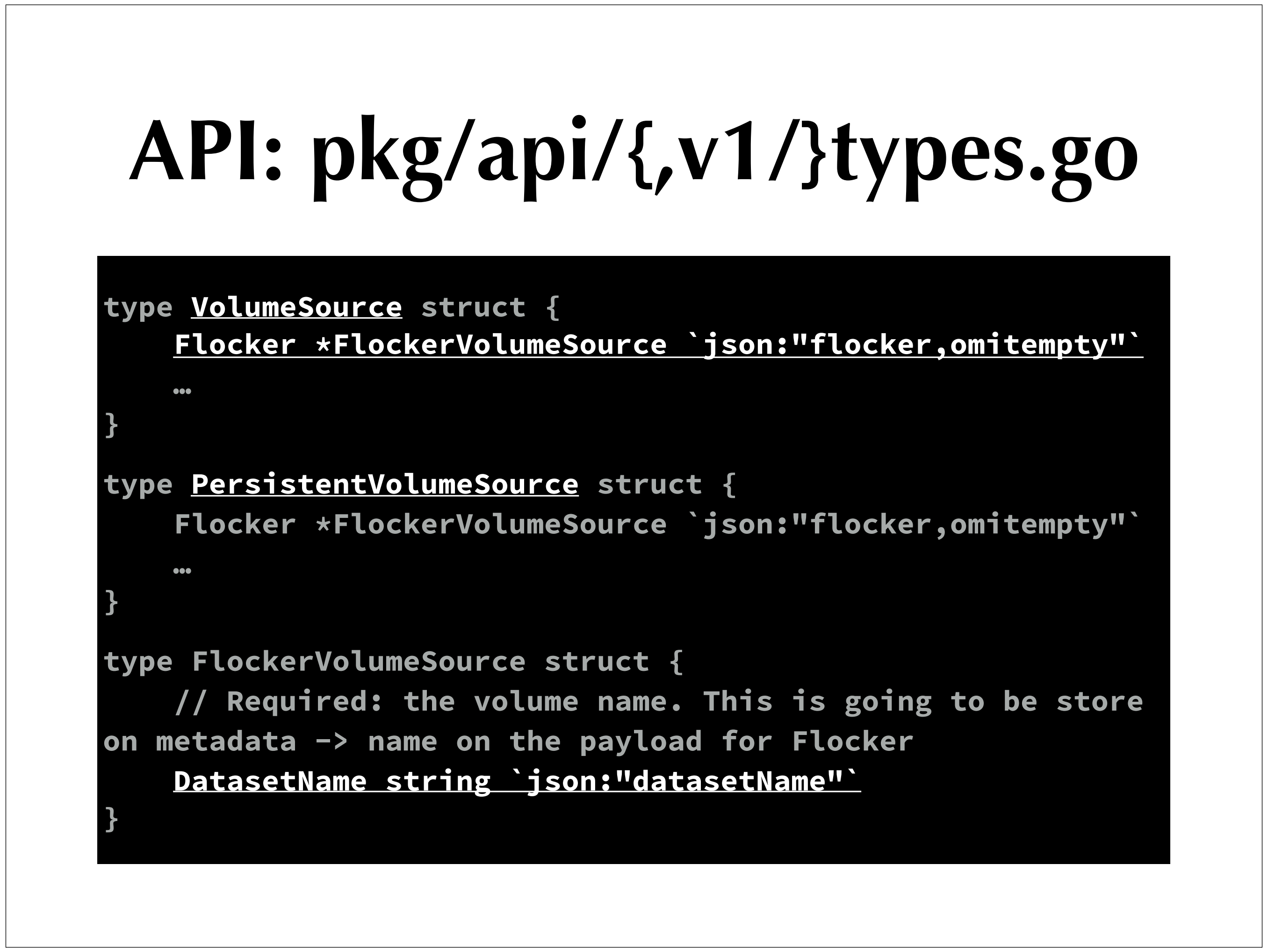
Volume: non persistent, usable for share paths between pods
PersistentVolume: well, as it says :)

gitRepo good entry point for develop plugins.

How was it made?

bit.ly/k8smeetup

I am going to go through code now, but if you are interested, there is the PR



API: pkg/api/{,v1/}types.go

```go
type VolumeSource struct {
    Flocker *FlockerVolumeSource `json:"flocker,omitempty"`
    …
}

type PersistentVolumeSource struct {
    Flocker *FlockerVolumeSource `json:"flocker,omitempty"`
    …
}

type FlockerVolumeSource struct {
    // Required: the volume name. This is going to be store
on metadata -> name on the payload for Flocker
    DatasetName string `json:"datasetName"`
}
```

Volume persistent -> we will let Flocker Agent decide.
This will generate swagger docs (with a script in hack/).
Also talk about validation.

VolumePlugin interface

```
type VolumePlugin interface {
    Init(host VolumeHost)
    Name() string
    CanSupport(spec *Spec) bool
    NewBuilder(
      s *Spec, pod *api.Pod, opts VolumeOptions
    ) (Builder, error)
    NewCleaner(
      name string, podUID types.UID
    ) (Cleaner, error)
}
```

https://github.com/kubernetes/kubernetes/blob/master/pkg/volume/plugins.go#L55-L81

Init -> called when the plugin is loaded.
Name -> to load the plugin by name, usually: kubernetes.io/XXX
CanSupport -> the spec will specify if it's volume or persistent volume, I didn't see more example. I am sure that it supports more stuff.
New{Builder,Cleaner} -> go next.

How was it used for Flocker?

NewBuilder
just creates the struct that will allow us to setup the datasets

NewCleaner
doing nothing

NewCleaner: Flocker Agent will do the work for us.

Builder interface

```
type Builder interface {
    Volume // gives GetPath()
    SetUp() error
    SetUpAt(dir string) error
    IsReadOnly() bool
}
```

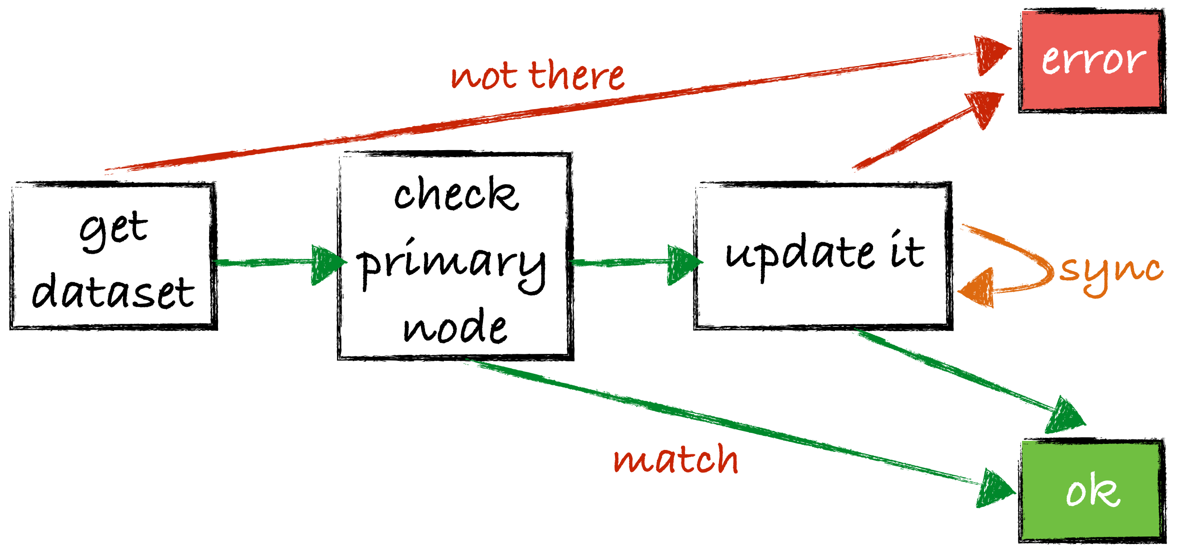
https://github.com/kubernetes/kubernetes/blob/master/pkg/volume/volume.go#L34-L48

Talk about setUp -> misleading name

Also talk about the metapath to store meta for idempotent state

How was it used for Flocker?

Setup -> Calls SetUpAt the datasetName received by API



1. Get the dataset id for the given volume name/dir
2. It should already be there, if it's not the user needs to manually create it
3. Check the current Primary UUID
4. If it doesn't match with the Primary UUID that we got on 2, then we will need to update the Primary UUID for this volume.
5. Wait until the Primary UUID was updated or timeout.

Ok, things that didn’t go so well…

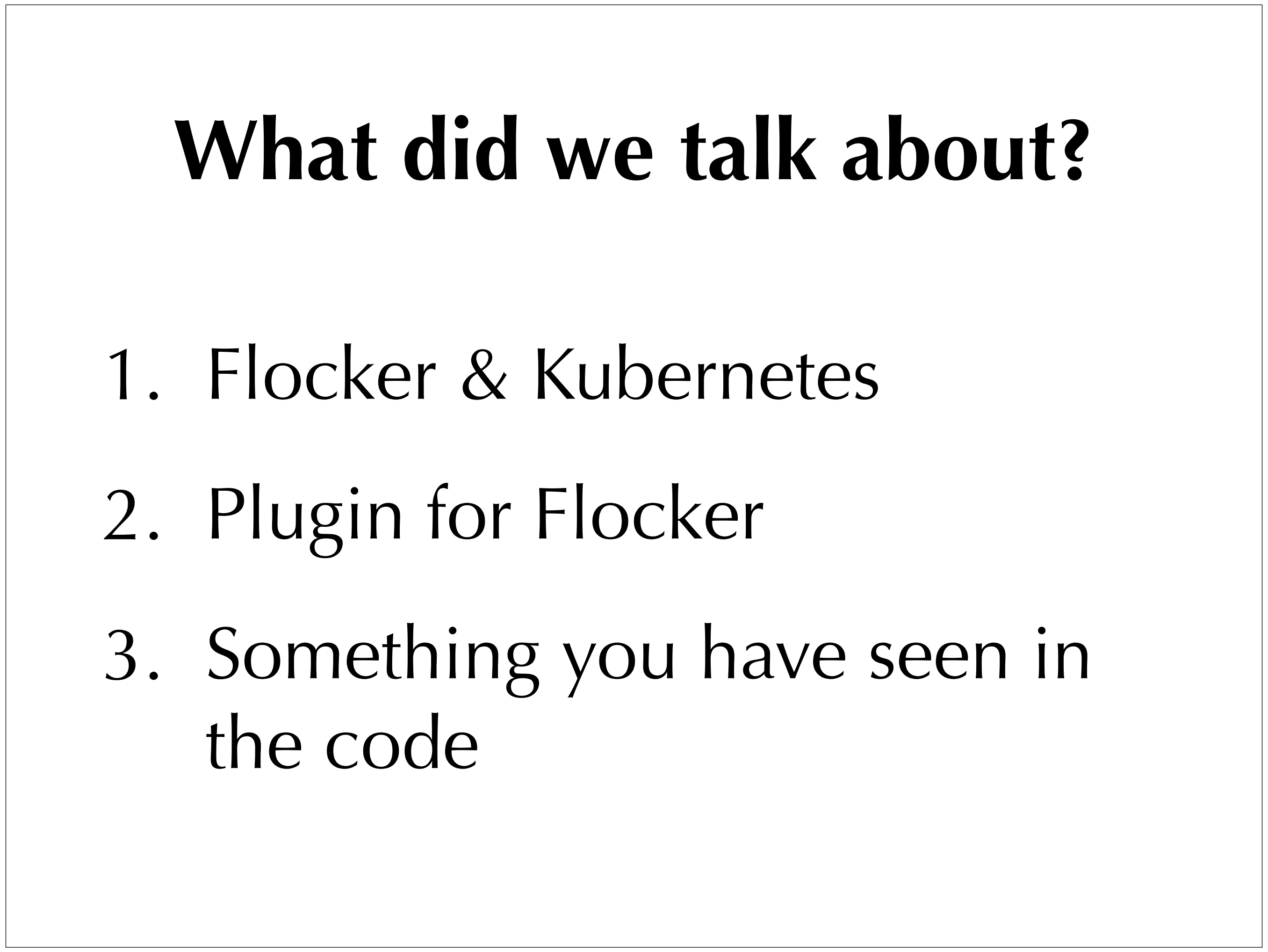
Things that didn’t go well

- OMG, CLAs! Sorry Matt!
- Shippable and Jenkins
- Creating the volume on SetUp
- `hack/` scripts

Shippable and Jenkins where flaky
They turn down the PR once because we were creating the volume just after get it merged!
hack -> update docs for a transparent pixel, api swagger documentation, etc -> pretty bad doc around it

Future!

- 1.2 has: `CreatableVolumePlugin`
- Run Flocker in a Pod
- Probably use `VolumeConfig`



What did we talk about?

1. Flocker & Kubernetes
2. Plugin for Flocker
3. Something you have seen in the code

Perhaps you have some questions about this topics, let me know.

Thanks!

@agonzalezro

richd77@flickr

To ClusterHQ for starting the project
To Jetstack for counting me in for the development
To k8s because it’s a really cool project that are making ops interesting to me
Twitter, github, gmail, whatever.