Continuous Delivery the hard way with Kubernetes

Luke Marsden, Developer Experience

@lmarsden

weaveworks

Agenda

1. Why should I deliver continuously?
2. Kubernetes primer
3. GitLab primer
4. “OK, so we’ve got these pieces, how are we going to put them together?”
5. Let’s iterate on a design!
6. Conclusions

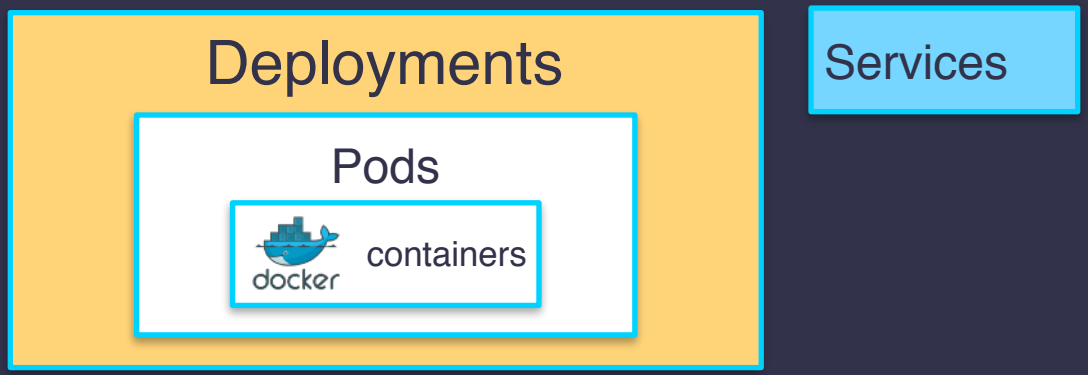
Agen

1. Why shou
2. Kube
3. GitLa
4. “OK, s e we going t
5. Let’s iter
6. Conclusio

Why should I continuously deliver?

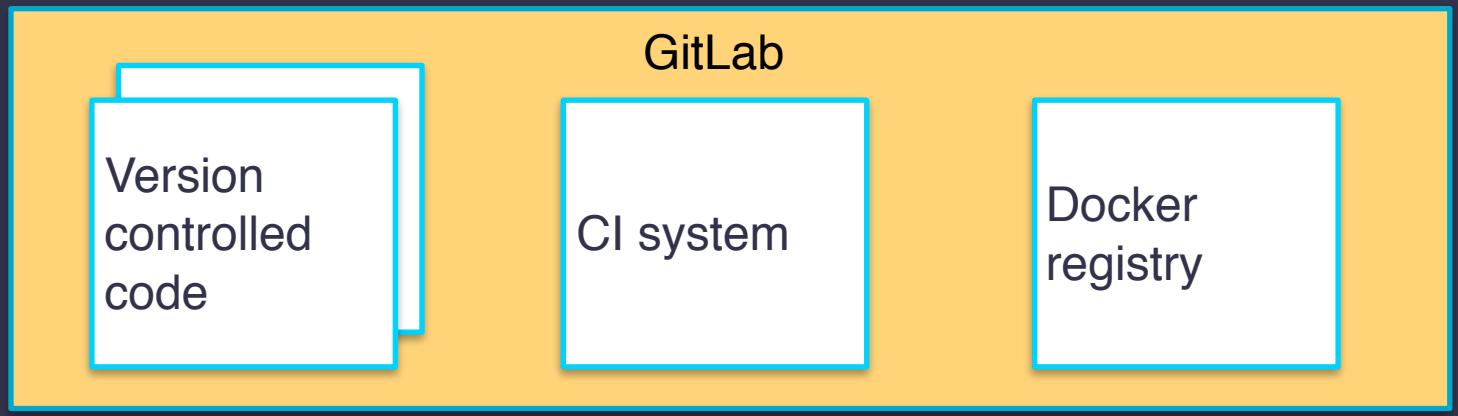
- Microservices
- Conway's law
- Scaling project, scaling team
- Velocity!

Kubernetes: all you need to know



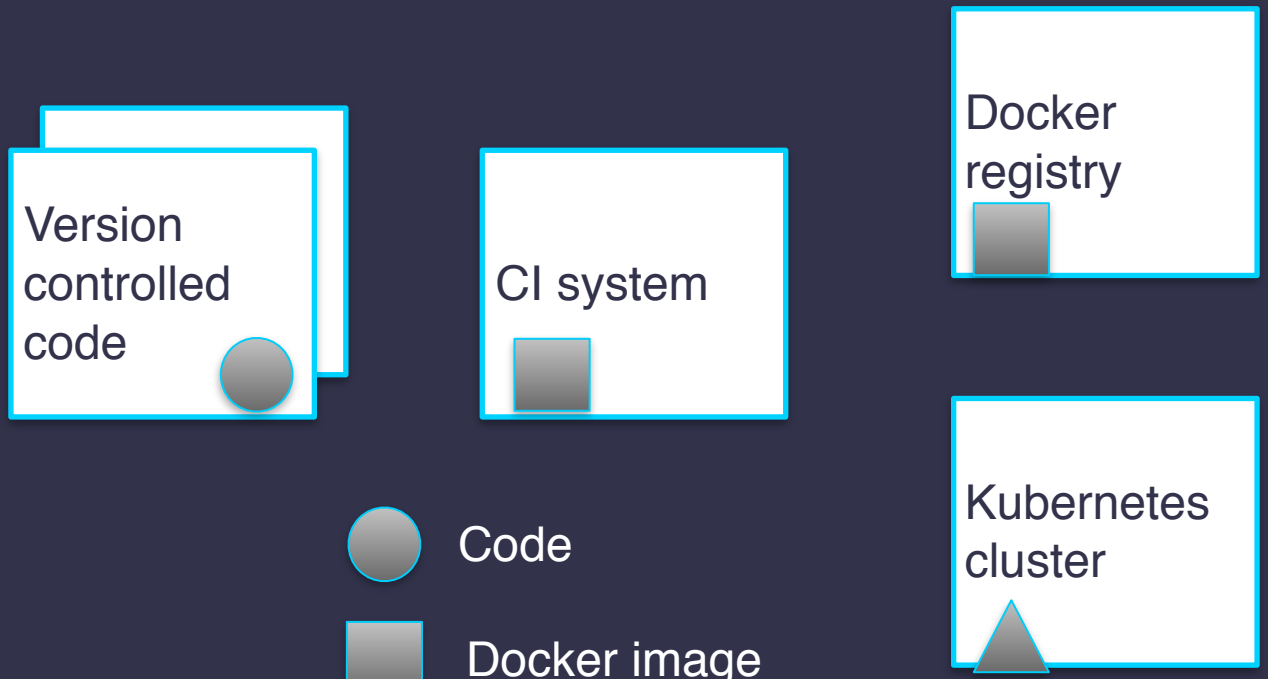
Container Image	Docker container image, contains your application code in an isolated environment.
Pod	A set of containers, sharing network namespace and local volumes, co-scheduled on one machine. Mortal. Has pod IP. Has labels.
Deployment	Specify how many replicas of a pod should run in a cluster. Then ensures that many are running across the cluster. Has labels.
Service	Names things in DNS. Gets virtual IP. Two types: ClusterIP for internal services, NodePort for publishing to outside. Routes based on labels.

GitLab primer

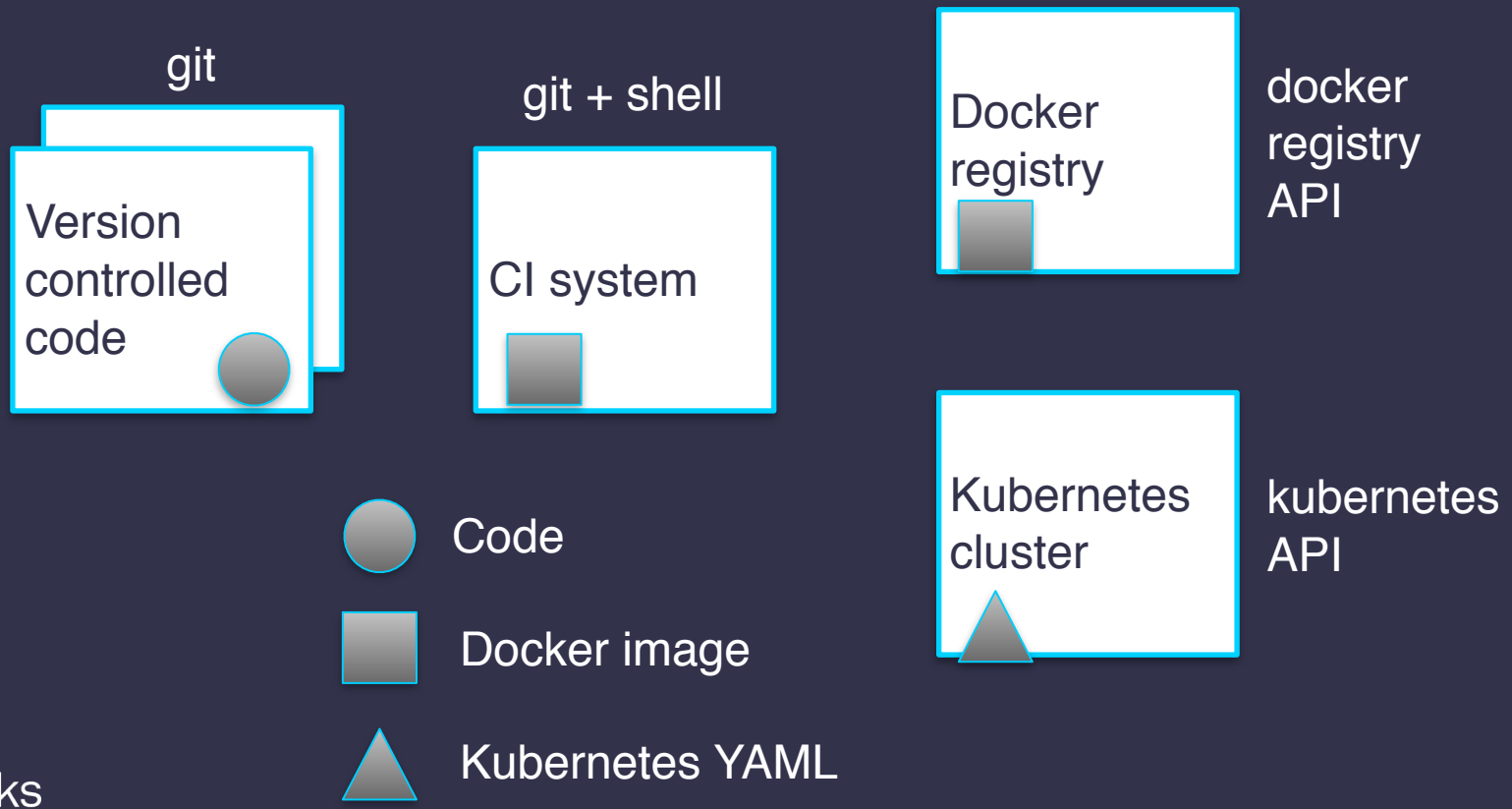


- Or you can use GitHub, Travis, Circle, Docker Hub, Quay.io, GCR…

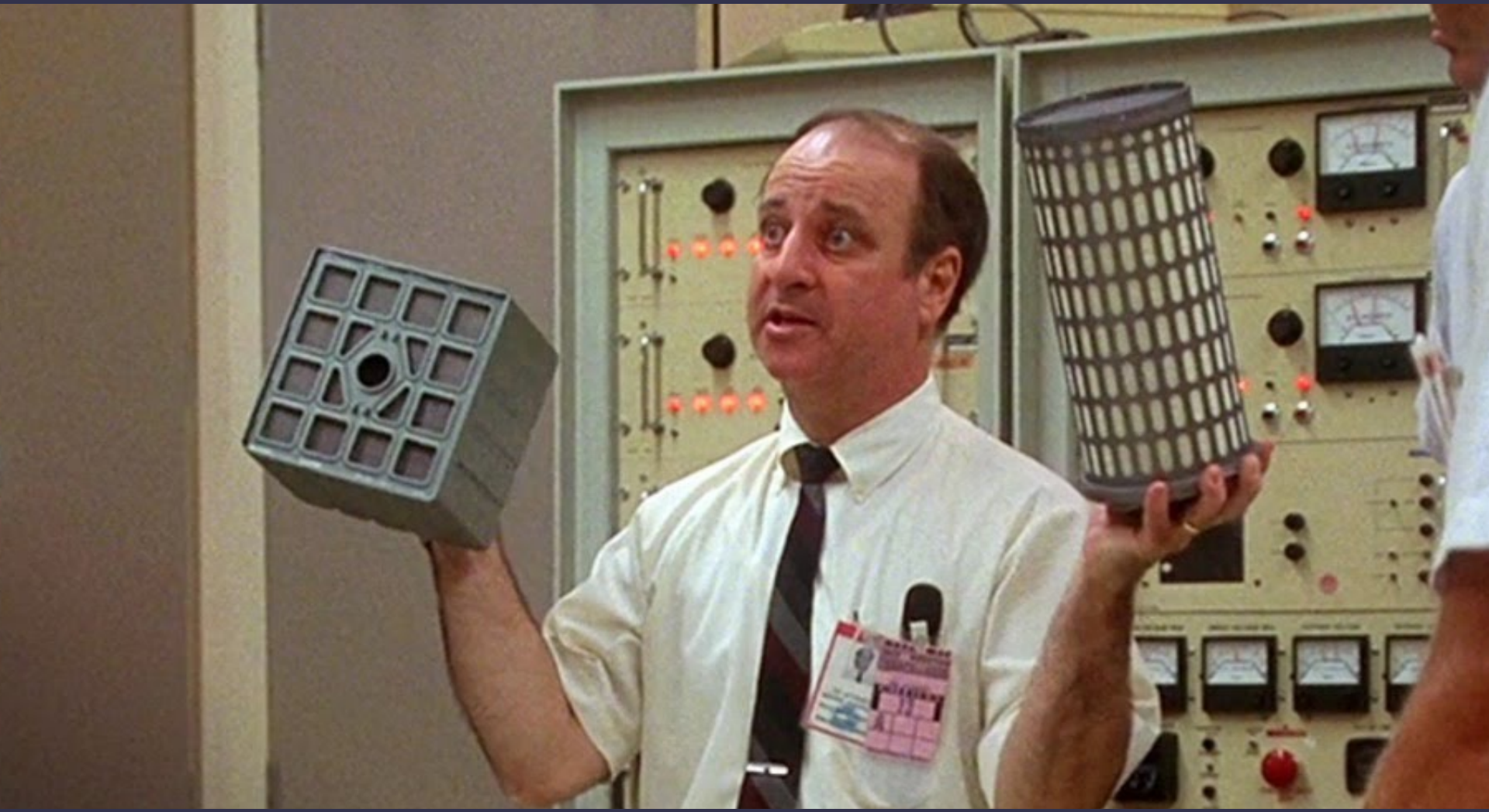
These are the things that we've got



These are the things that we've got

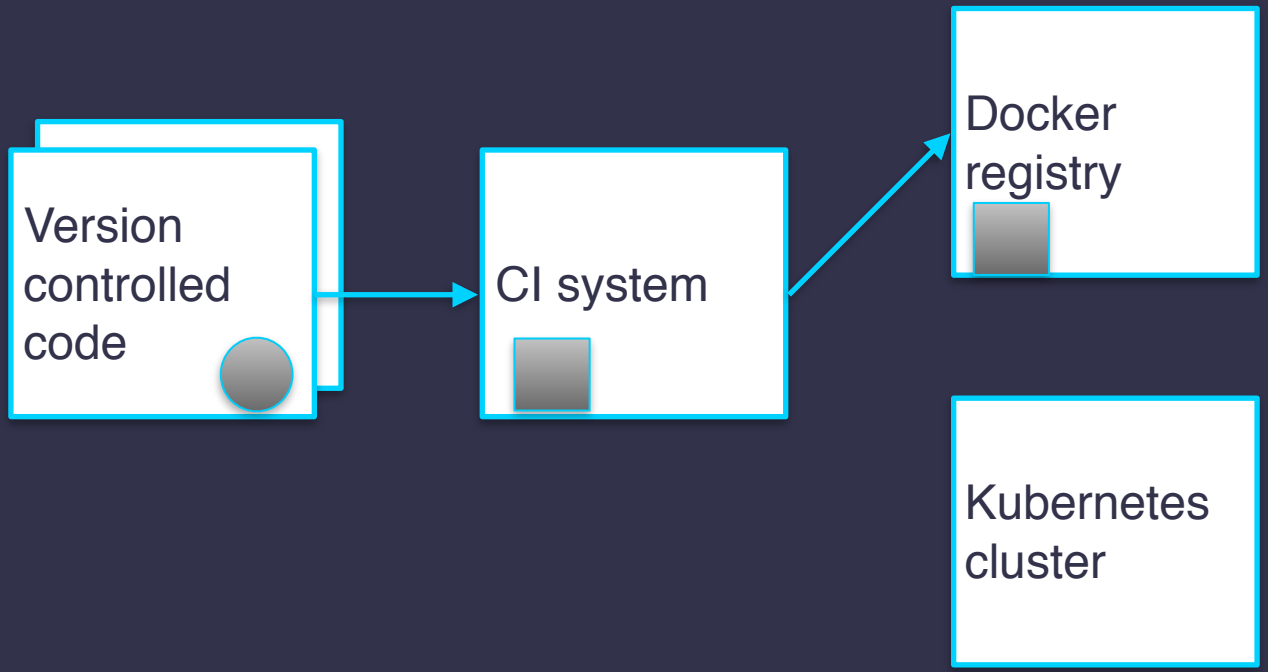


weaveworks

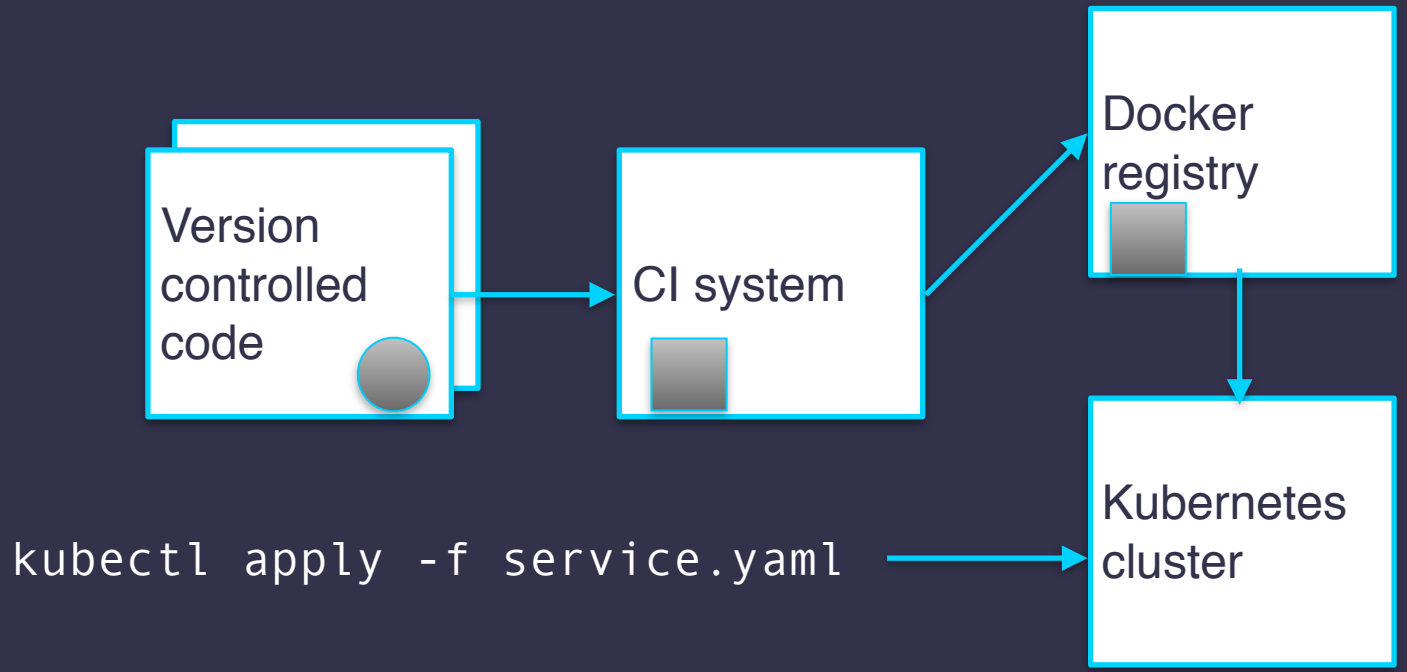


V1
Initial deploy (manually)

V1 architecture

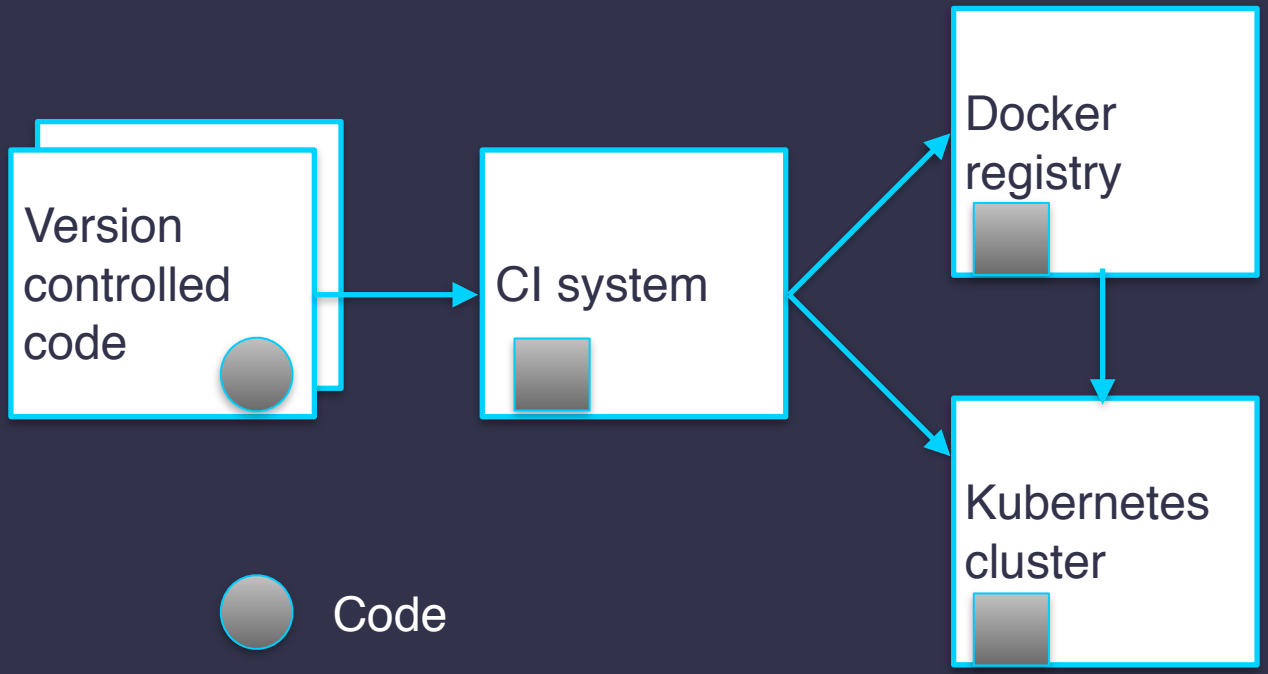


V1 architecture



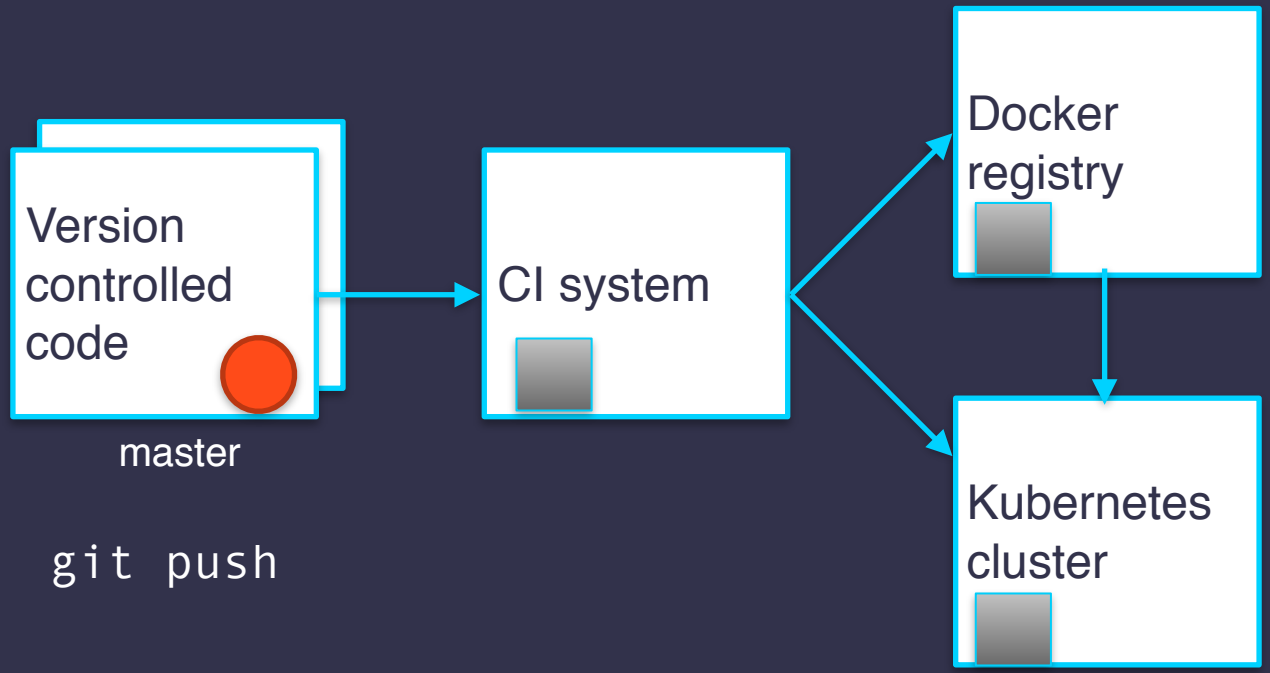
V1
Deploy update (with CI system)

V1 architecture

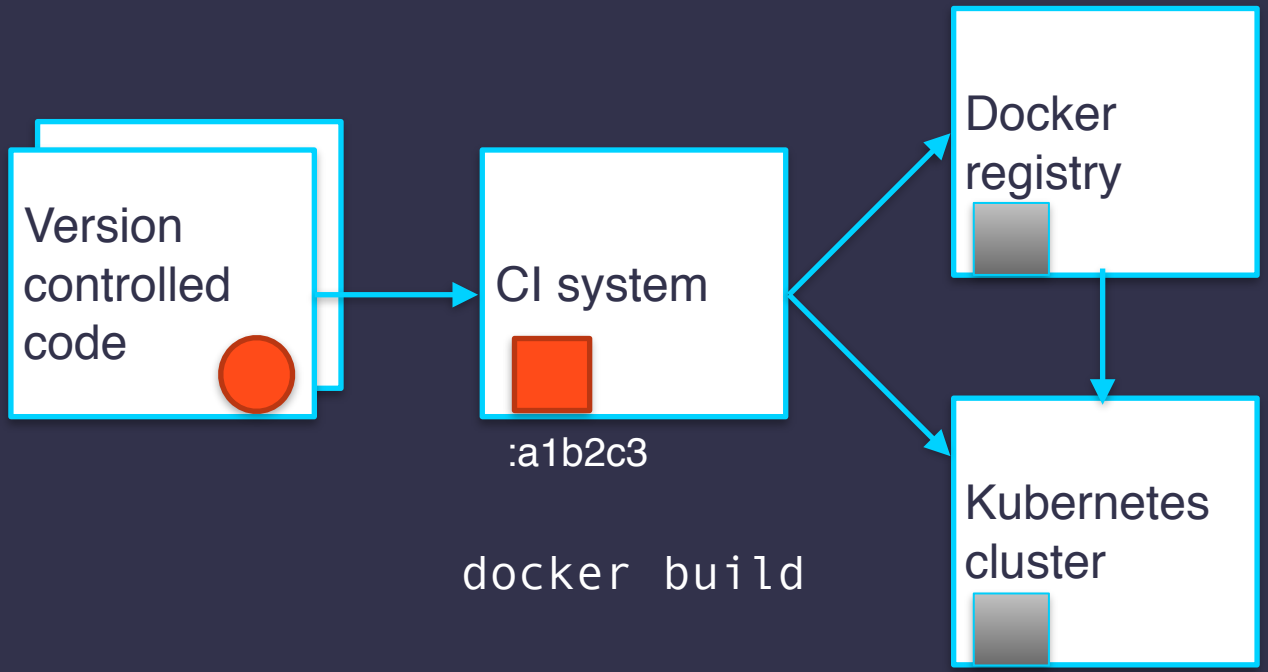


weaveworks

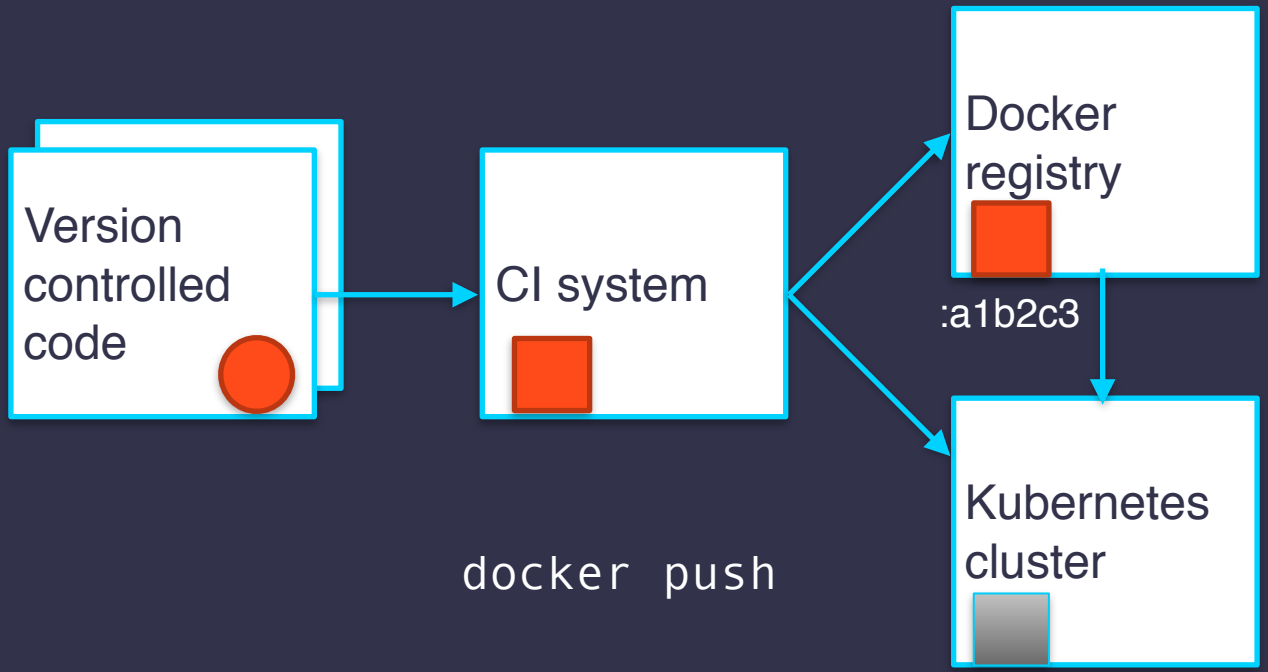
V1 architecture



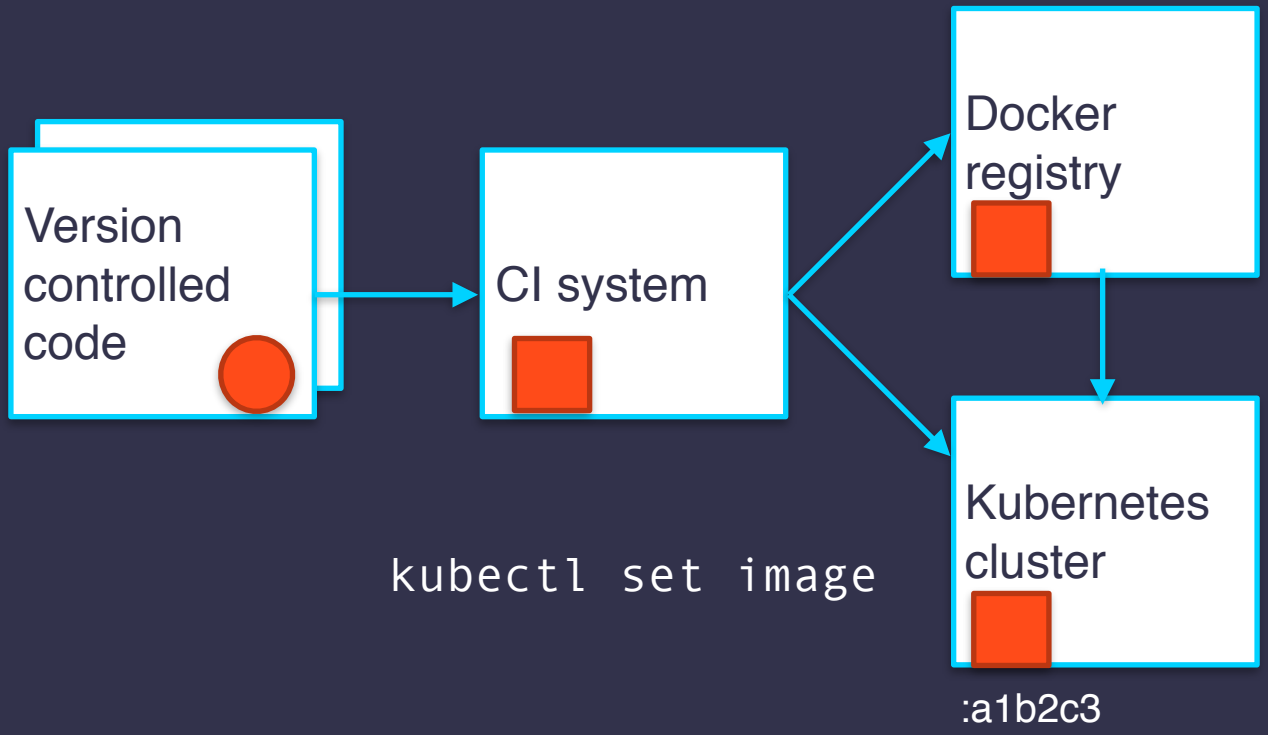
V1 architecture



V1 architecture

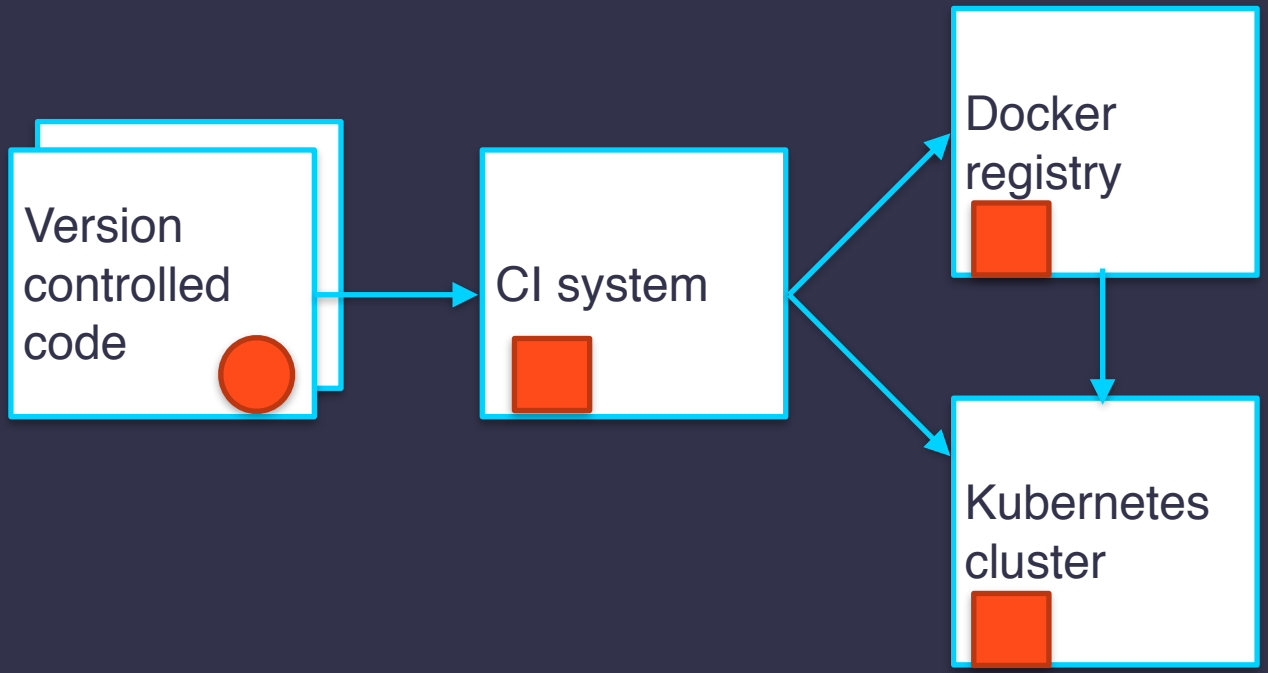


V1 architecture

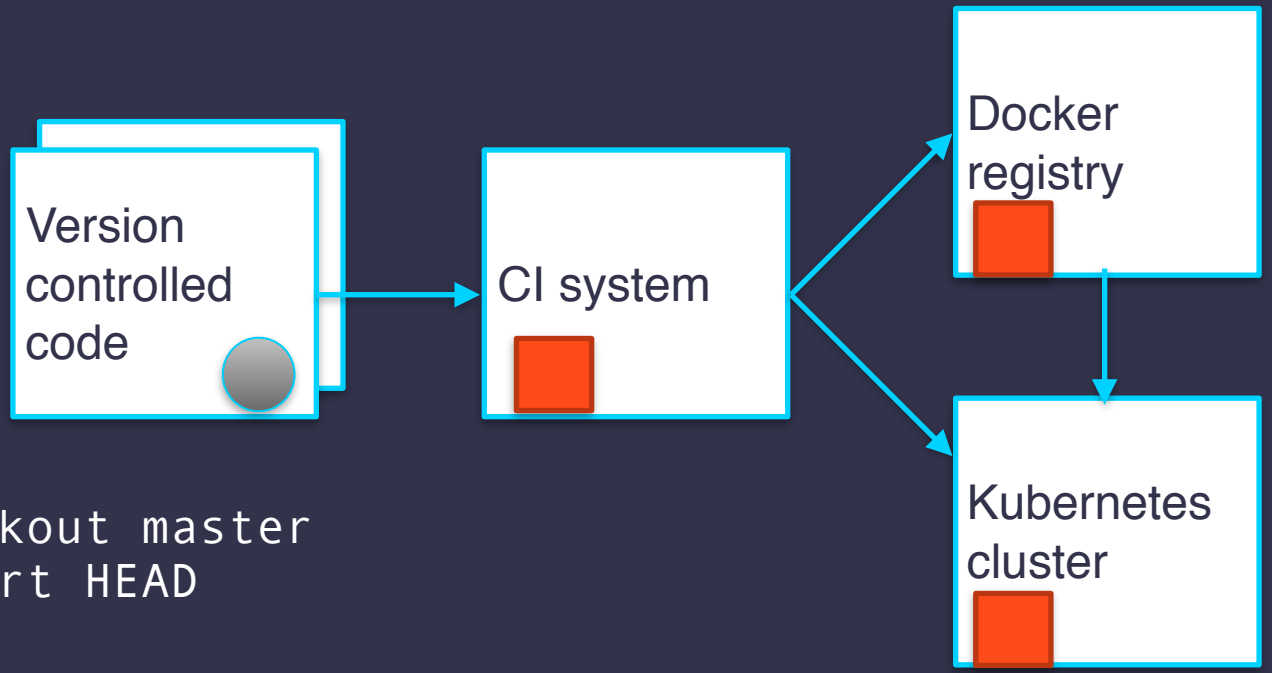


V1
Rollback

V1 architecture

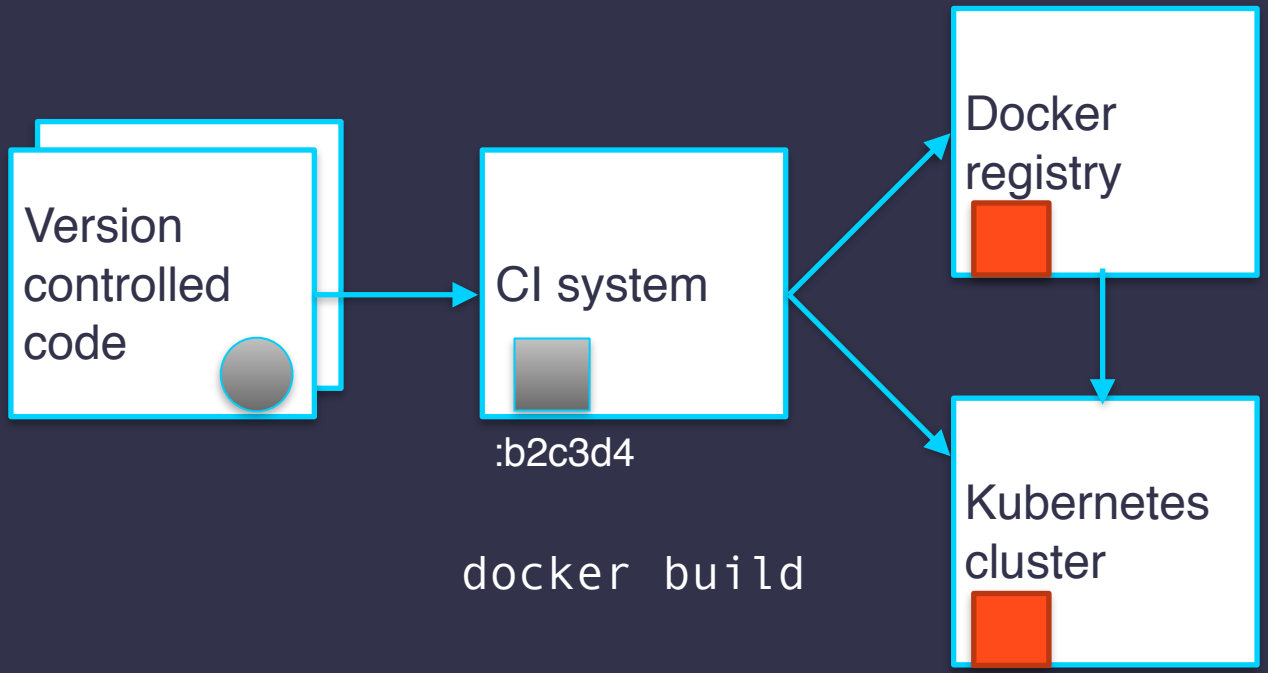


V1 architecture

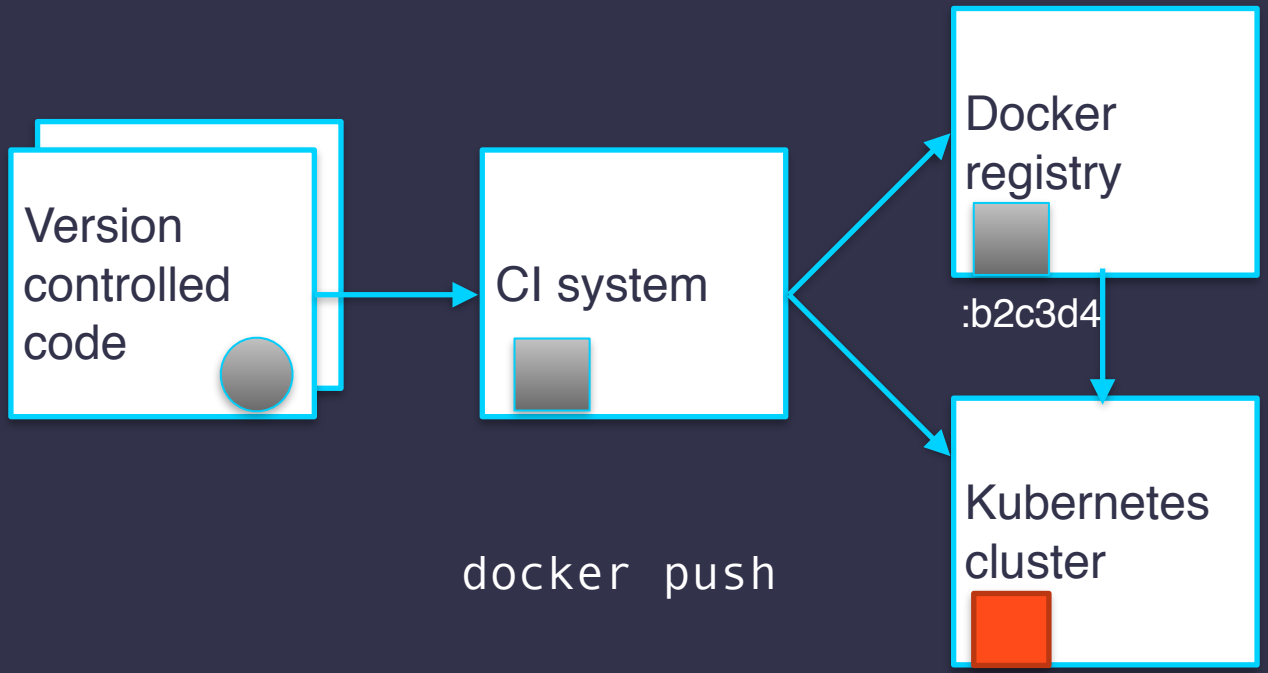



```
git checkout master
git revert HEAD
git push
```

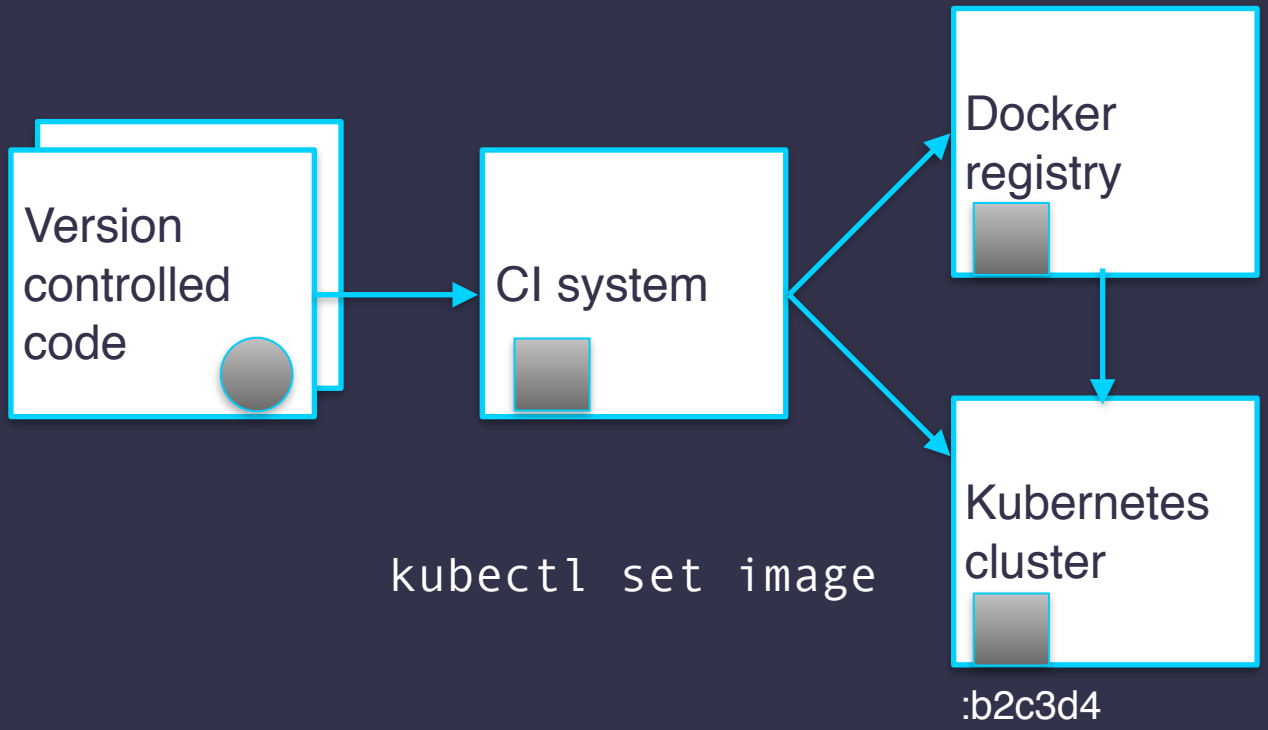
V1 architecture



V1 architecture



V1 architecture



Demo!

Downsides

- Building & pushing containers is slow (disk I/O, network), shouldn't need to this when rolling back
- Branch per environment required per microservice (explosion of branches, hard to manage & scale)
  - Only a matter of time until you get a git merge mess
- Better to decouple version of code at HEAD from version deployed…

Version controlled configuration

- Version controlled config should be the source of truth for your *whole app* (all the microservices)

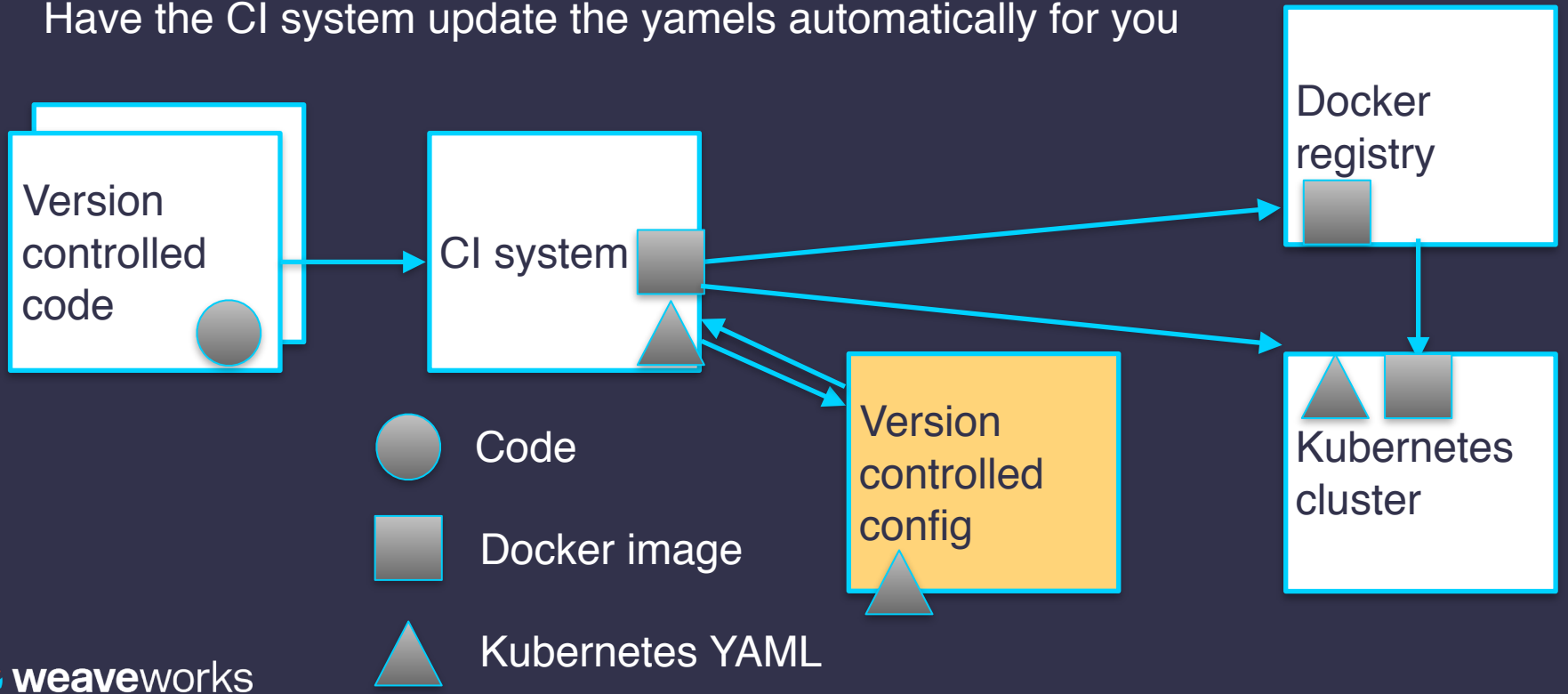
- users service
  - code for users service
  - Kubernetes YAML
- orders service
  - code for orders service
  - Kubernetes YAML

- config repo
  - Kubernetes YAML for users
  - Kubernetes YAML for orders

V2
Put all the yamels in one place

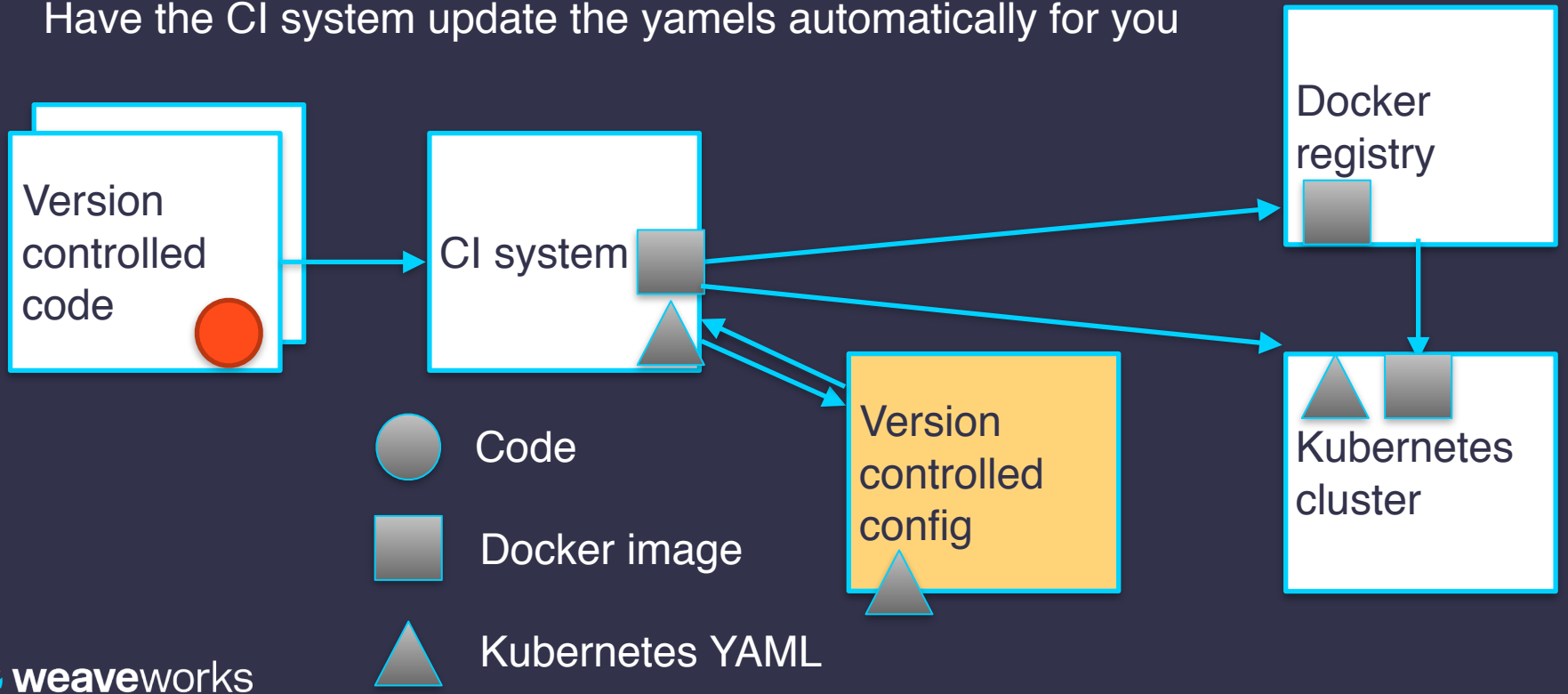
V2 architecture

Have the CI system update the yamels automatically for you



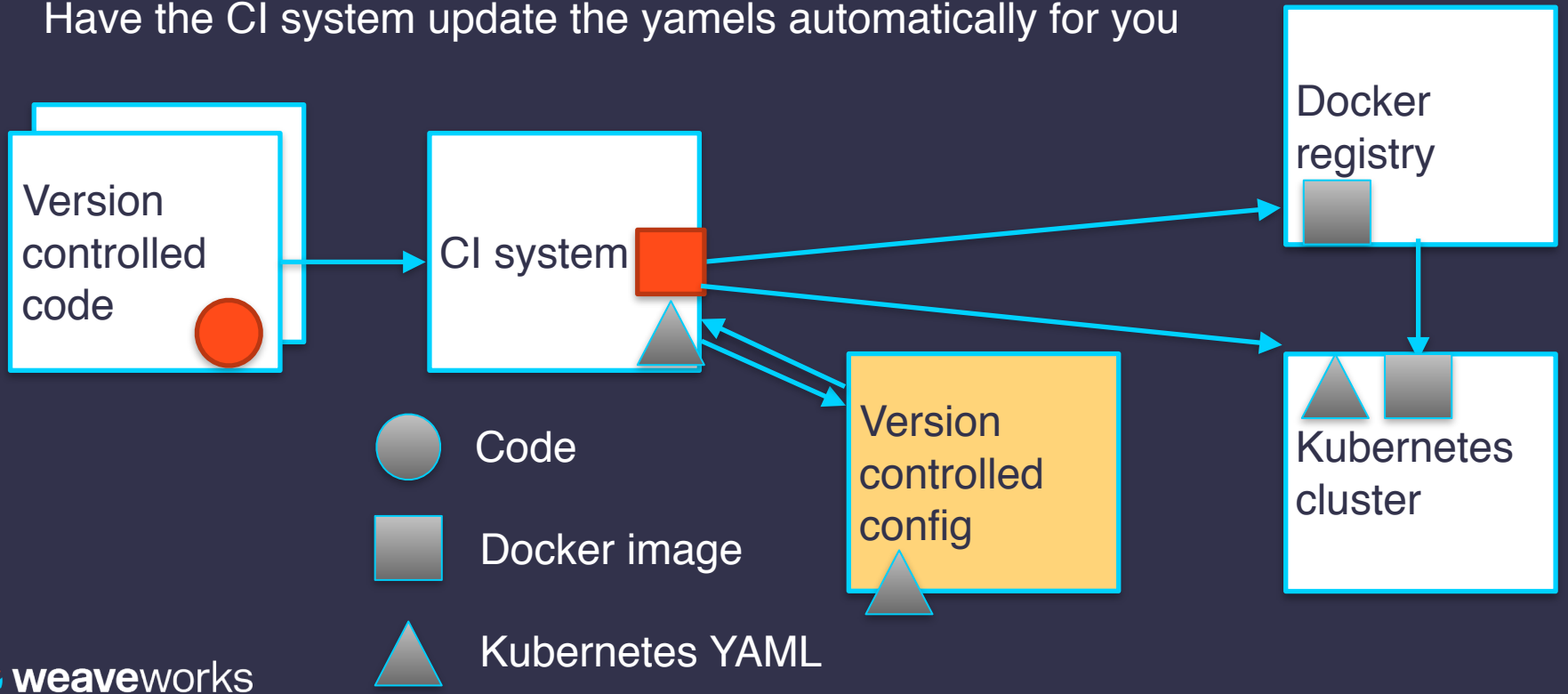
V2 architecture

Have the CI system update the yamels automatically for you



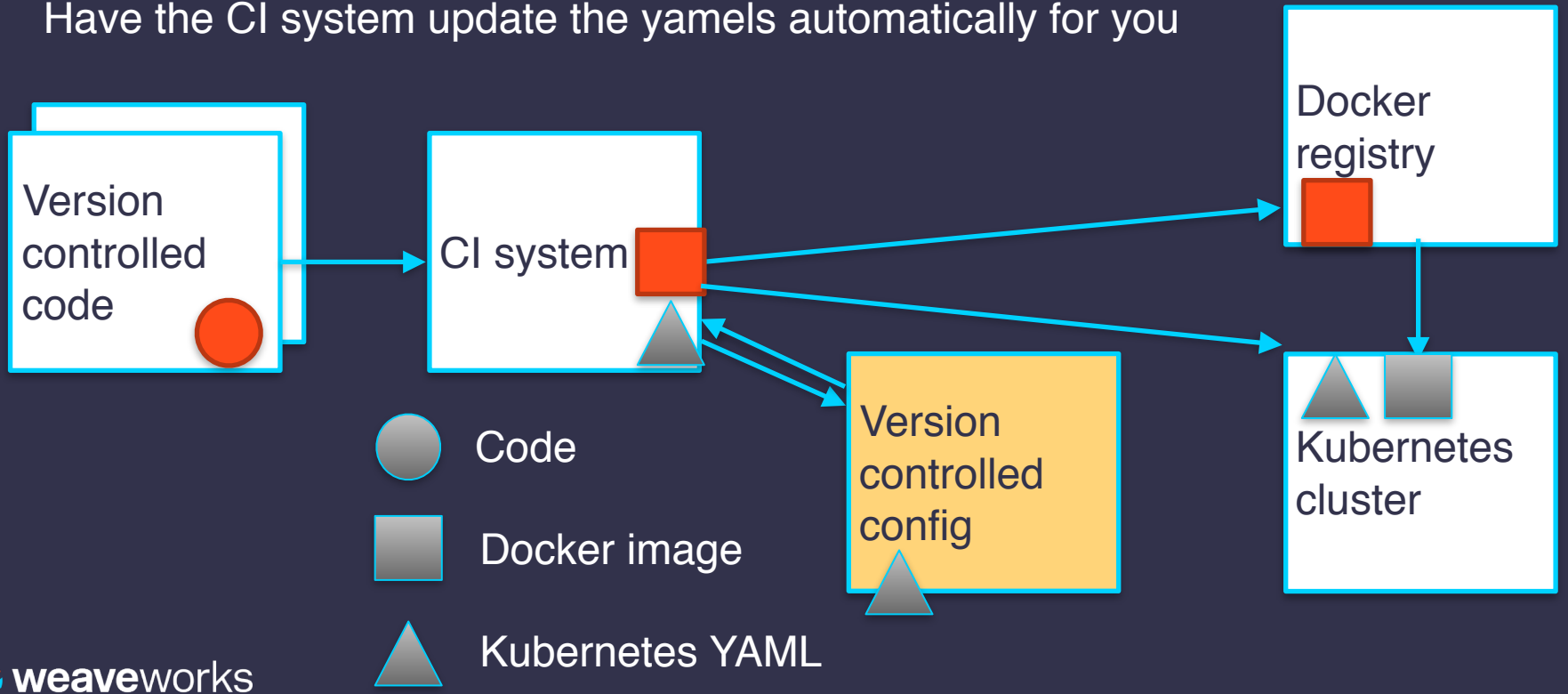
V2 architecture

Have the CI system update the yamels automatically for you



V2 architecture

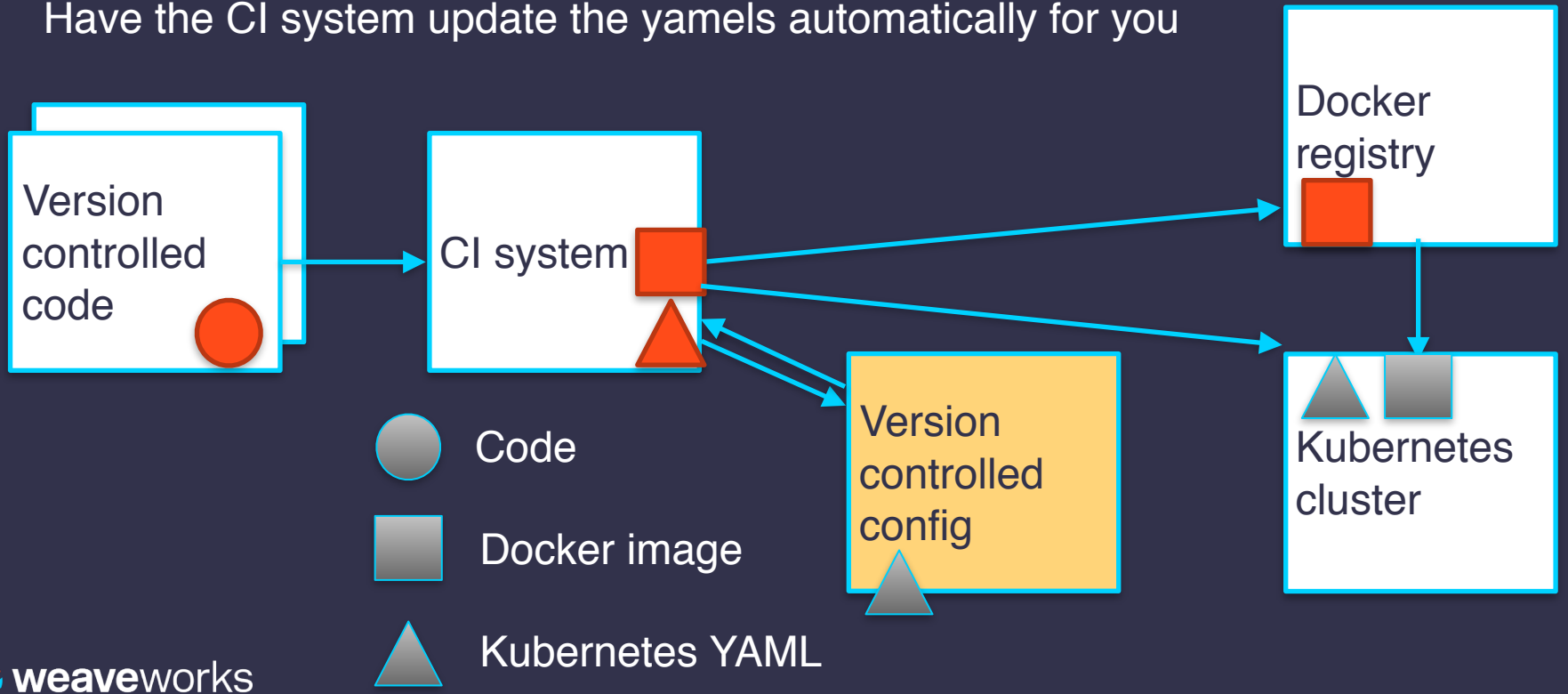
Have the CI system update the yamels automatically for you



weaveworks

V2 architecture

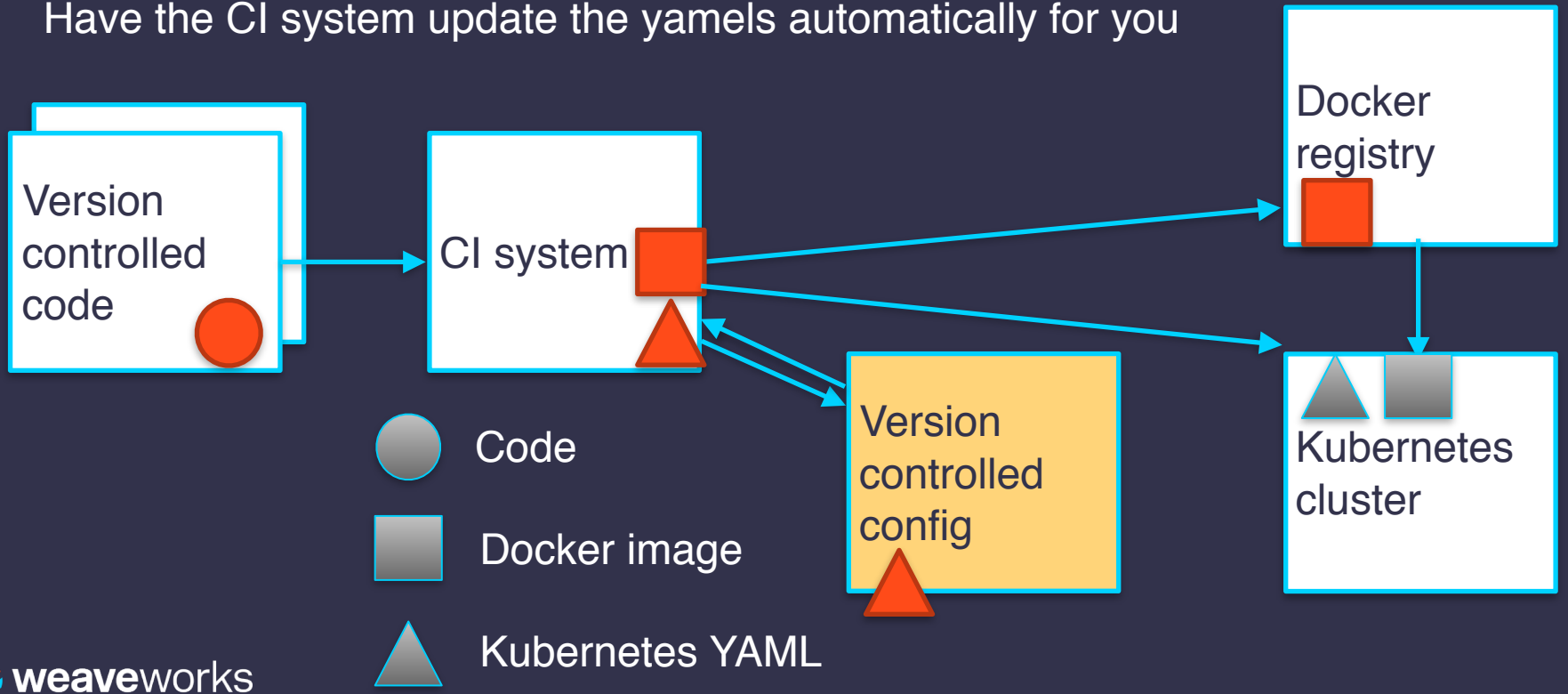
Have the CI system update the yamels automatically for you



weaveworks

V2 architecture

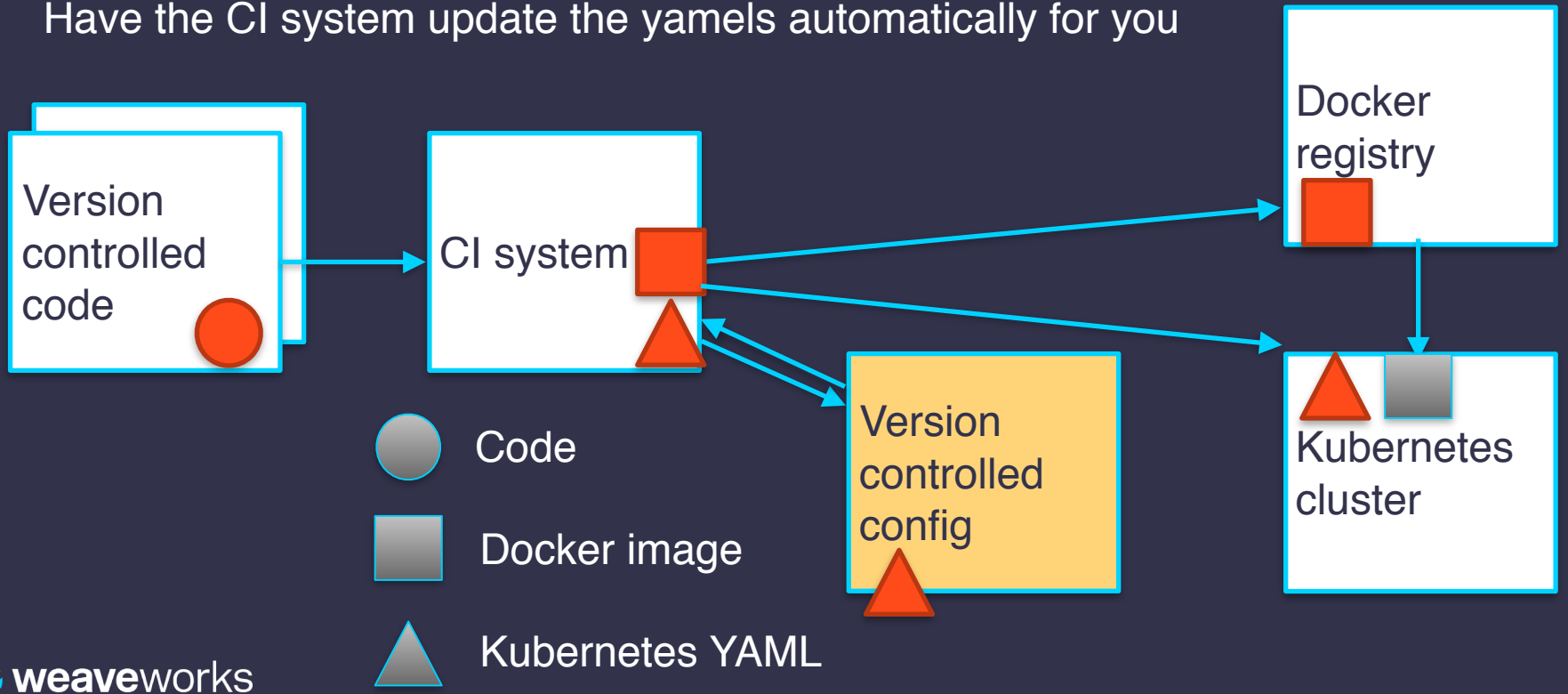
Have the CI system update the yamels automatically for you



weaveworks

V2 architecture

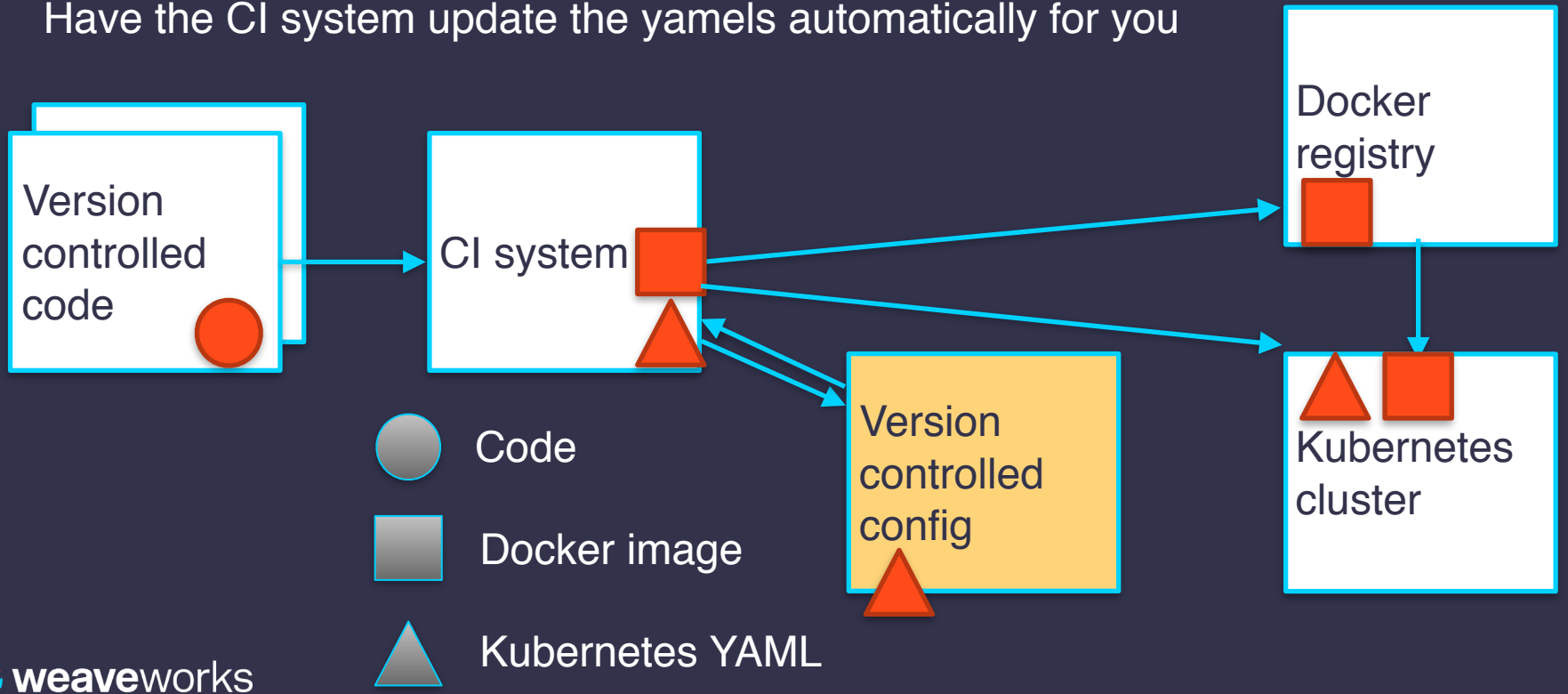
Have the CI system update the yamels automatically for you



weaveworks

V2 architecture

Have the CI system update the yamels automatically for you



weaveworks

Now you can recreate your production environment from the central YAML repository even if your entire production cluster gets deleted

Demo!

Downsides

- The CI system is responsible for a lot now (design smell – overloaded)
- You can only *trigger* the CI system by pushing code (we wanted to be able to rollback without pushing code)
    - If you rollback out of band (directly with kubectl), you have to remember to update the central configuration repo as well
- Parallel builds can tread on eachothers' toes, not atomic: race between git checkout and git push (need a global lock)
- Scripting updates of yamels can be a pain… it mangles your yamels
- Developers start asking for more release management features (rollback, pinning, automation for some envs and manual gating for others, and your once-simple script keeps growing…)

Decoupling versions from releases

Code versions (branches, tags)

- users service
  - master
  - feature_A
  - feature_B
- orders service
  - master
  - feature_A
  - feature_B
- …

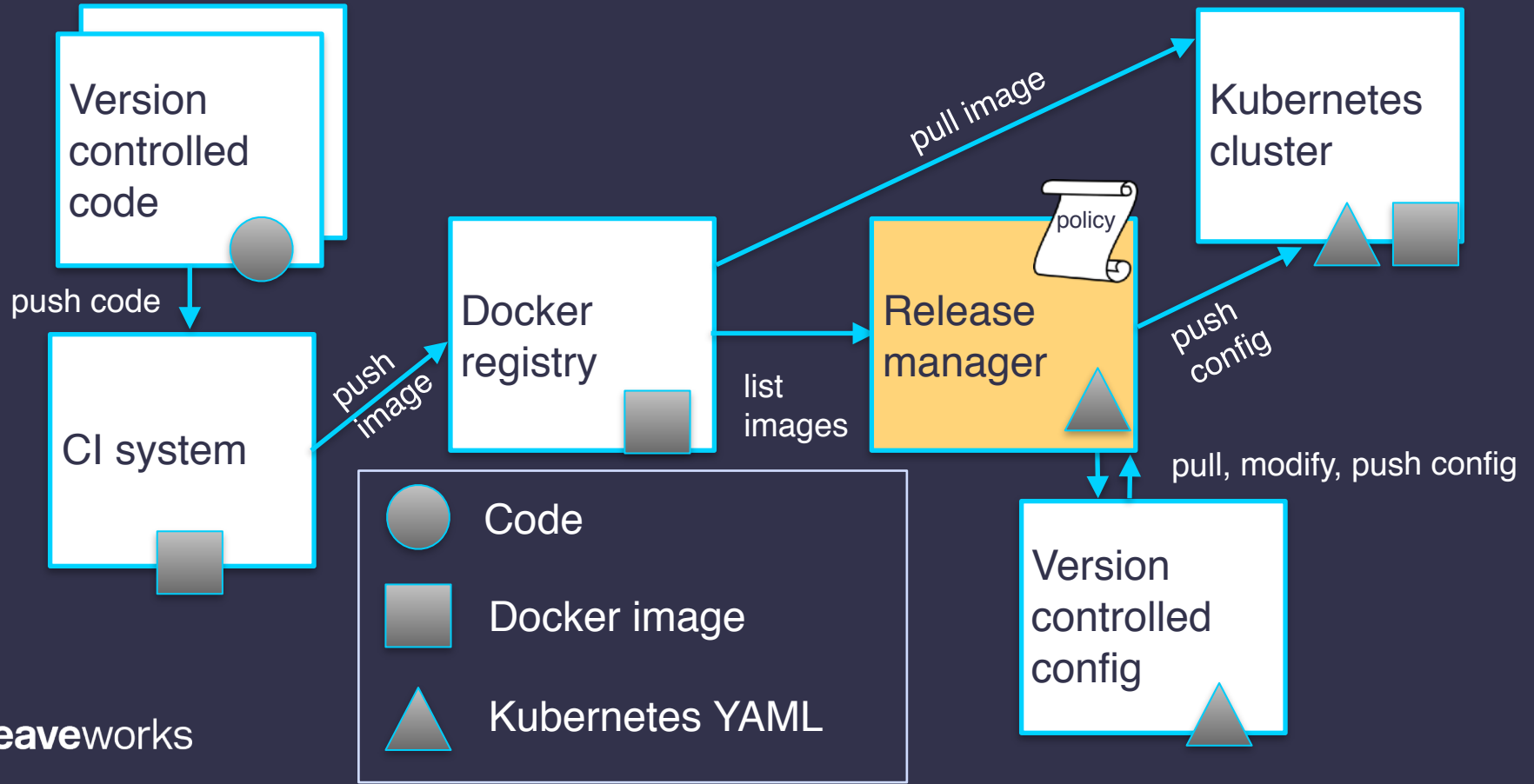
conflating per-service code branches with environments in each repo is a hack, and doesn't scale well

Environments & releases

- production
  - users -> master @ t_1
  - orders -> master @ t_1
- staging
  - orders -> master @ t_2
  - orders -> master @ t_2

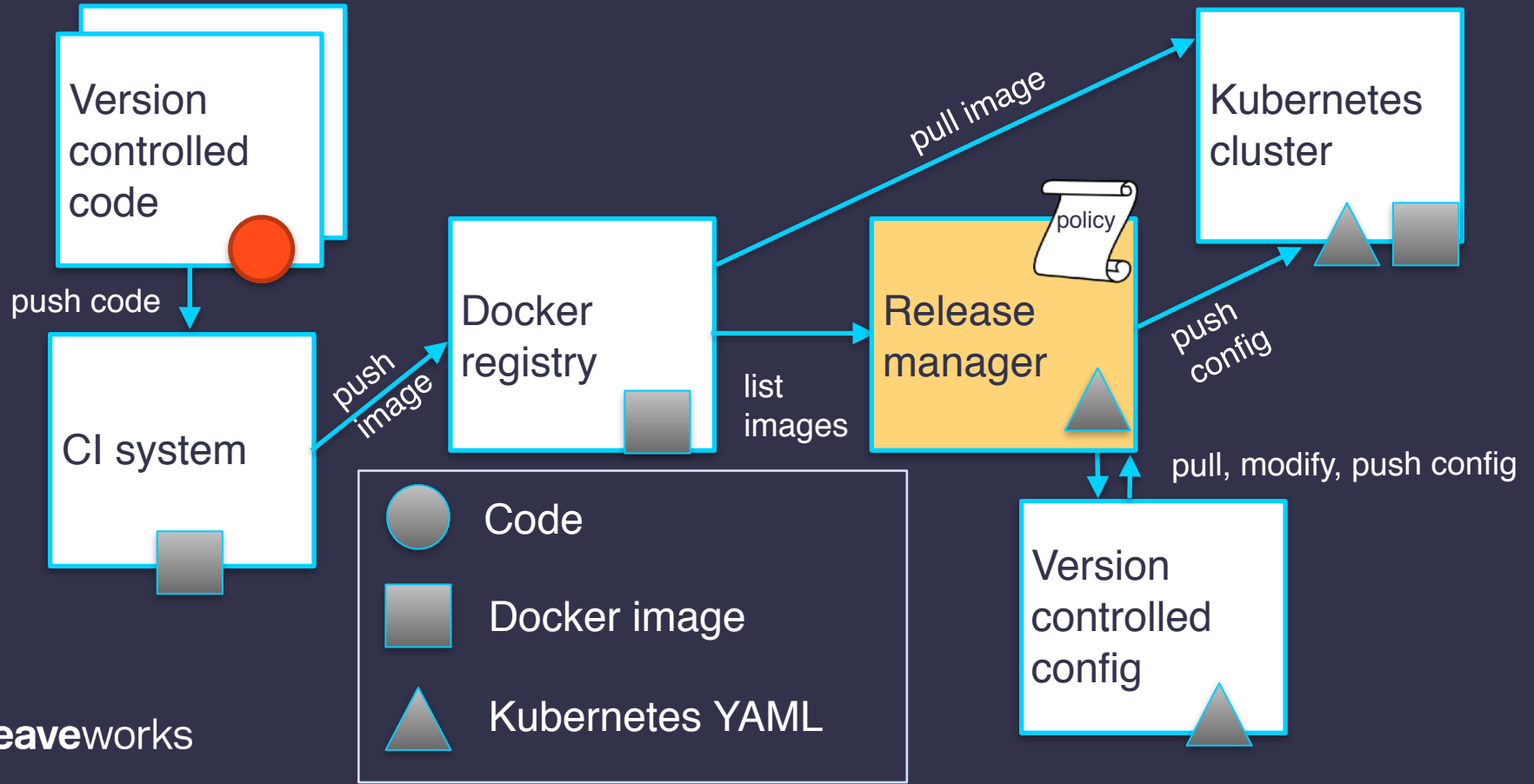
V3
Refactor architecture
Add “release manager”

V3 architecture



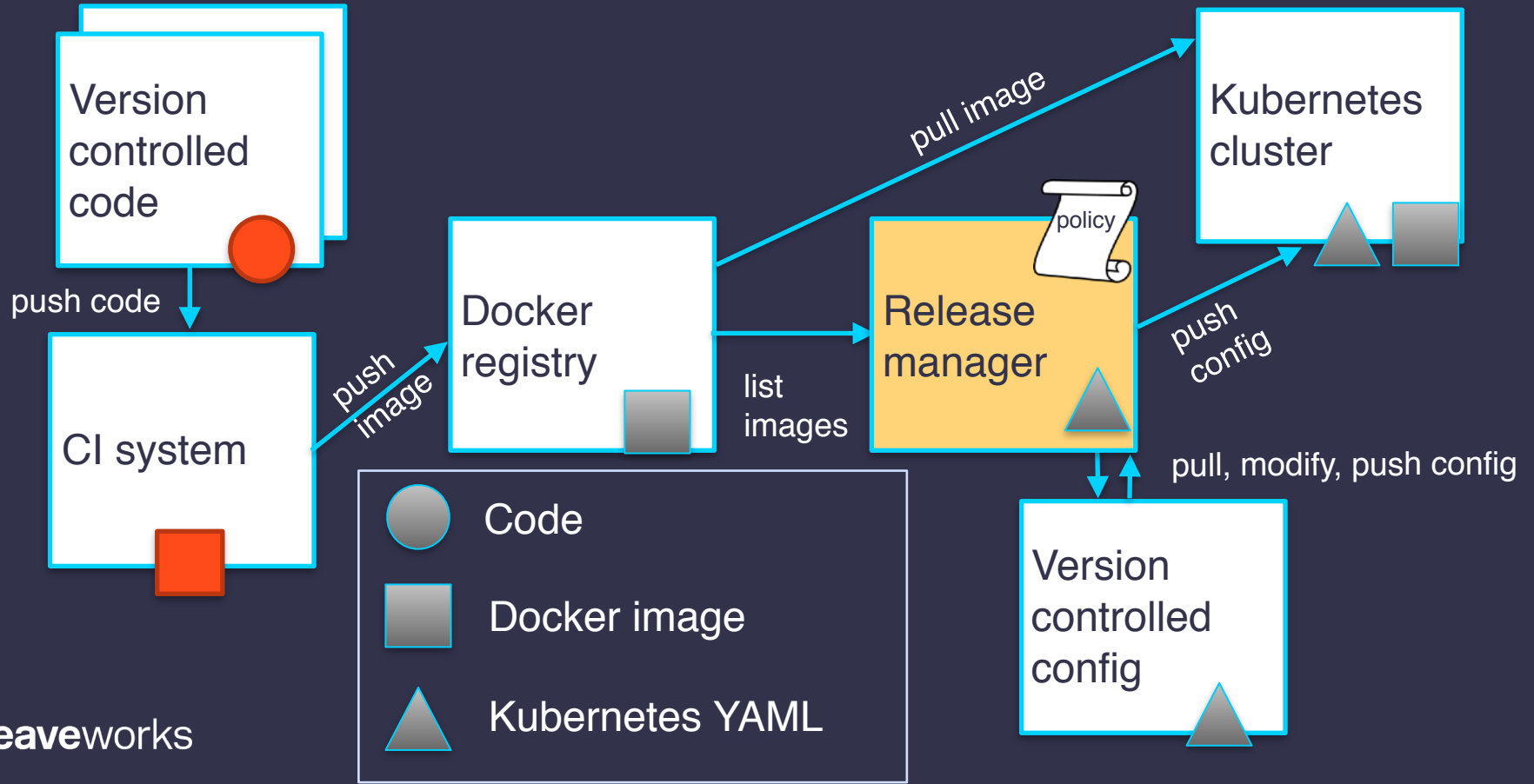
weaveworks

V3 architecture



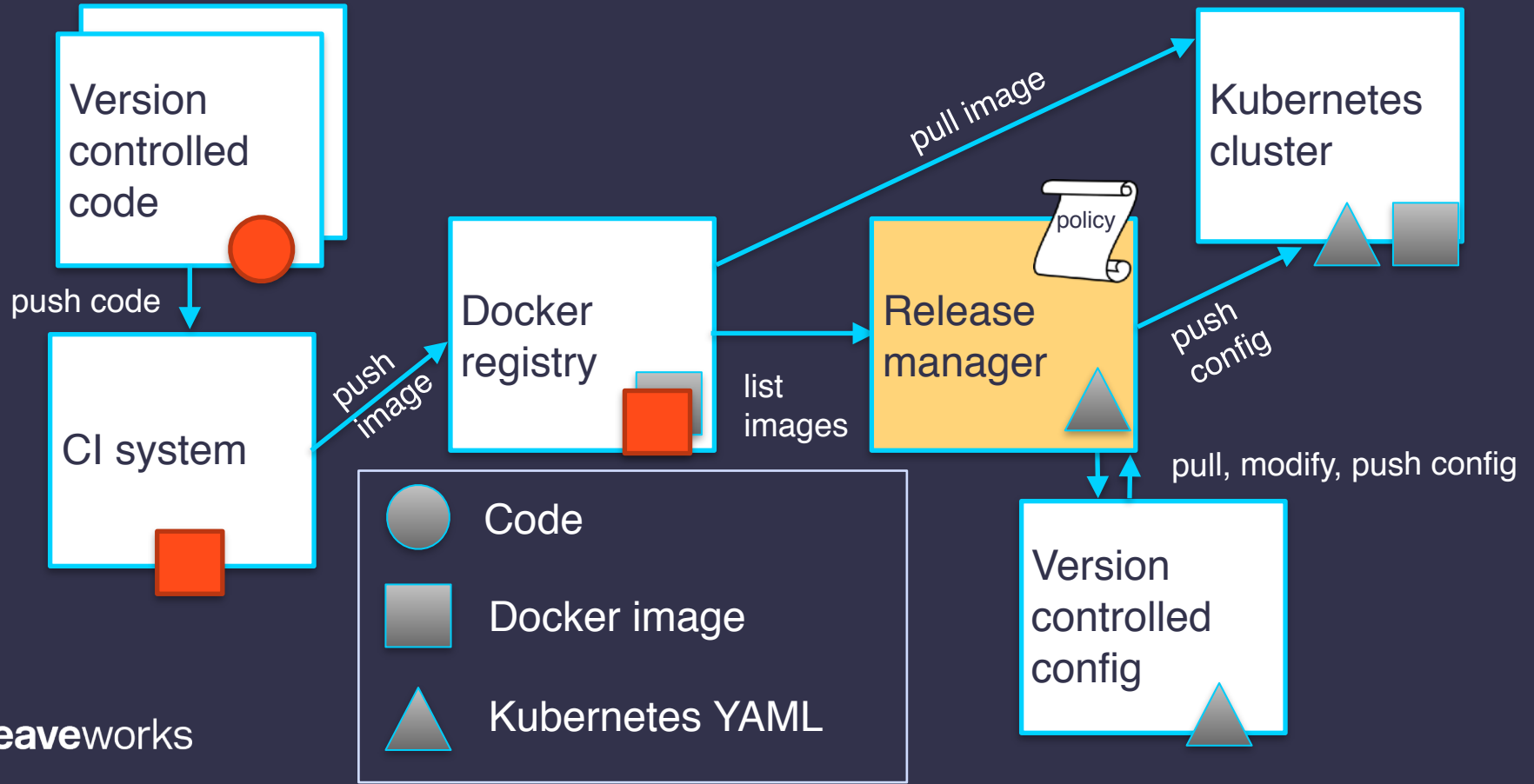
weaveworks

V3 architecture



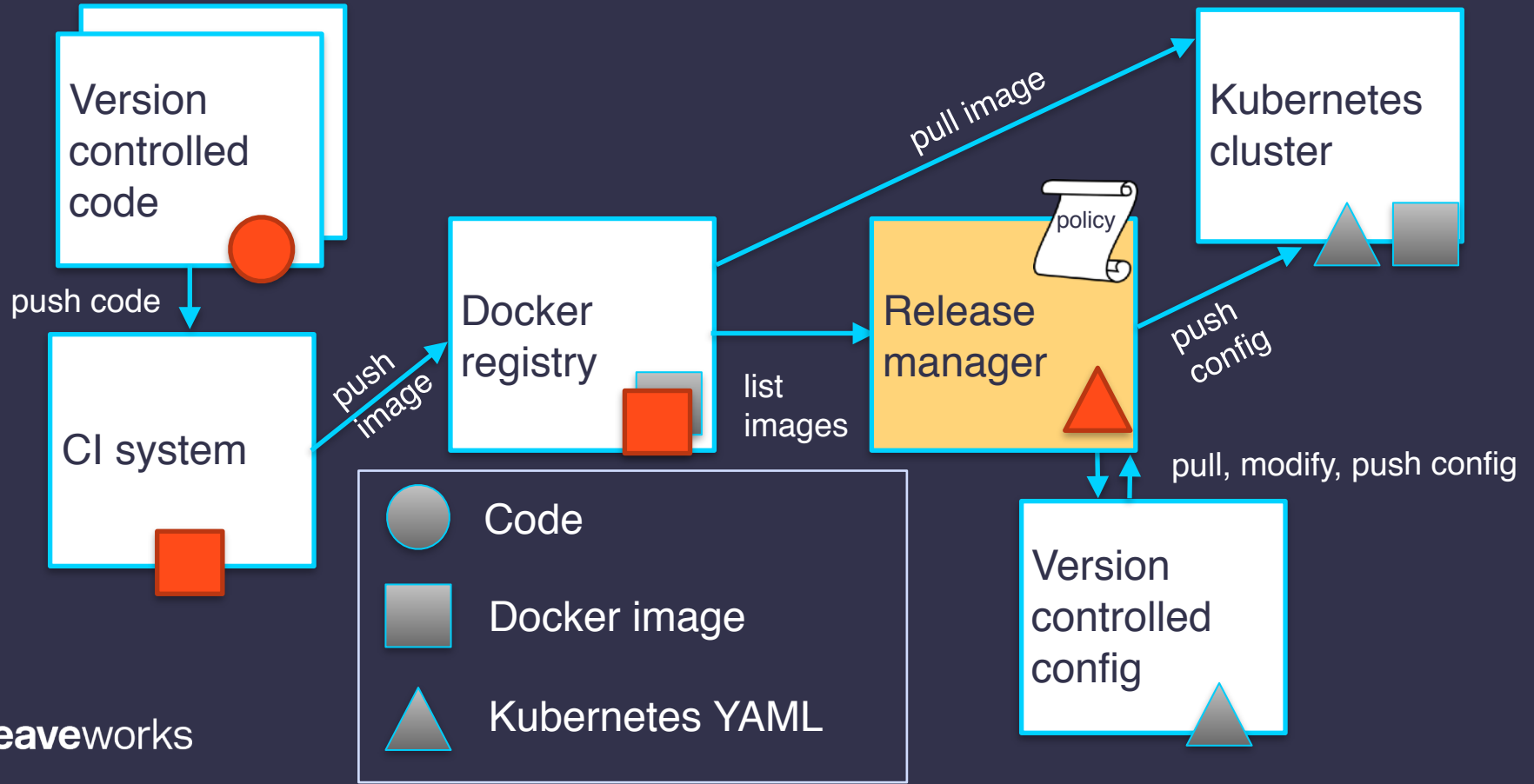
weaveworks

V3 architecture



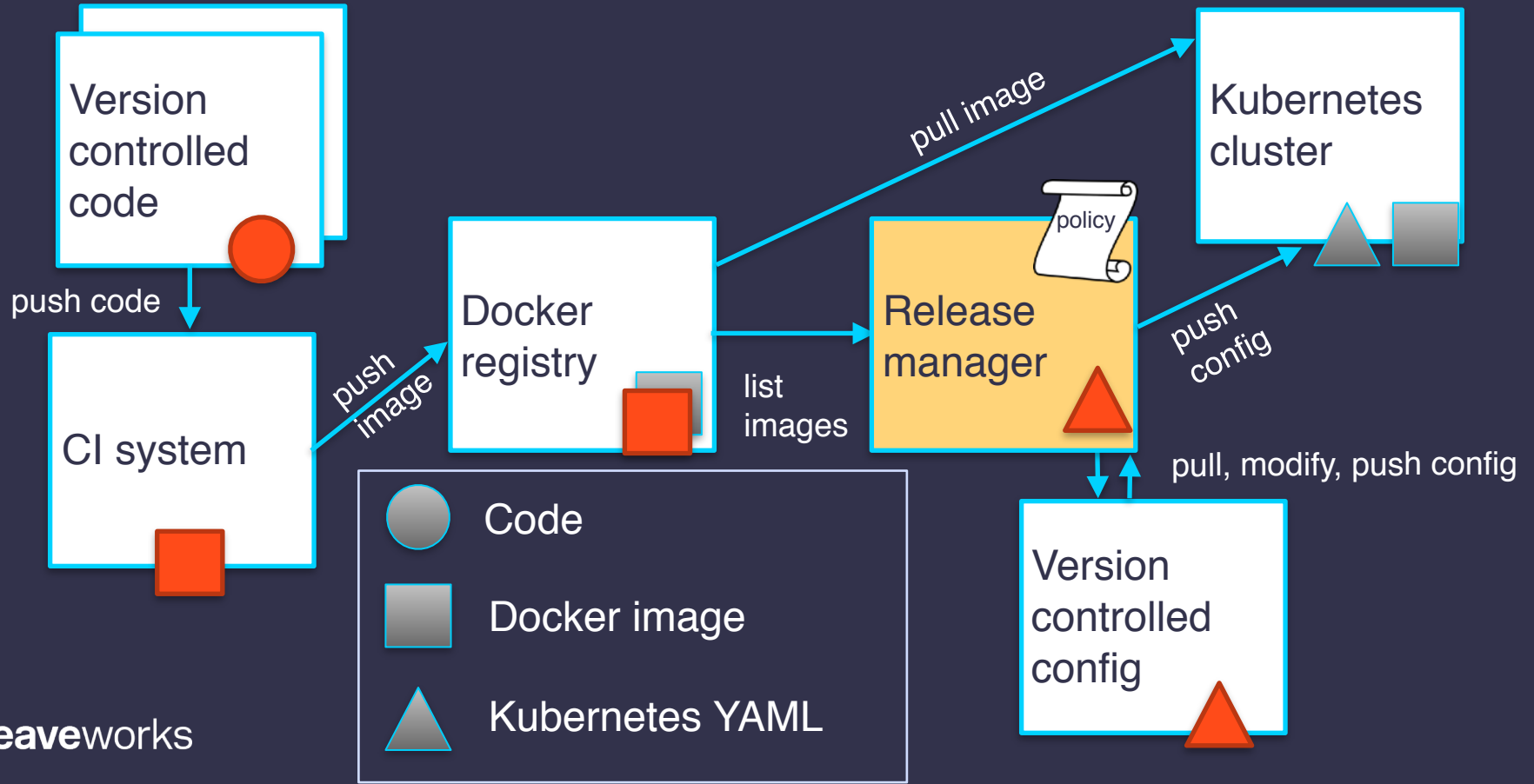
weaveworks

V3 architecture



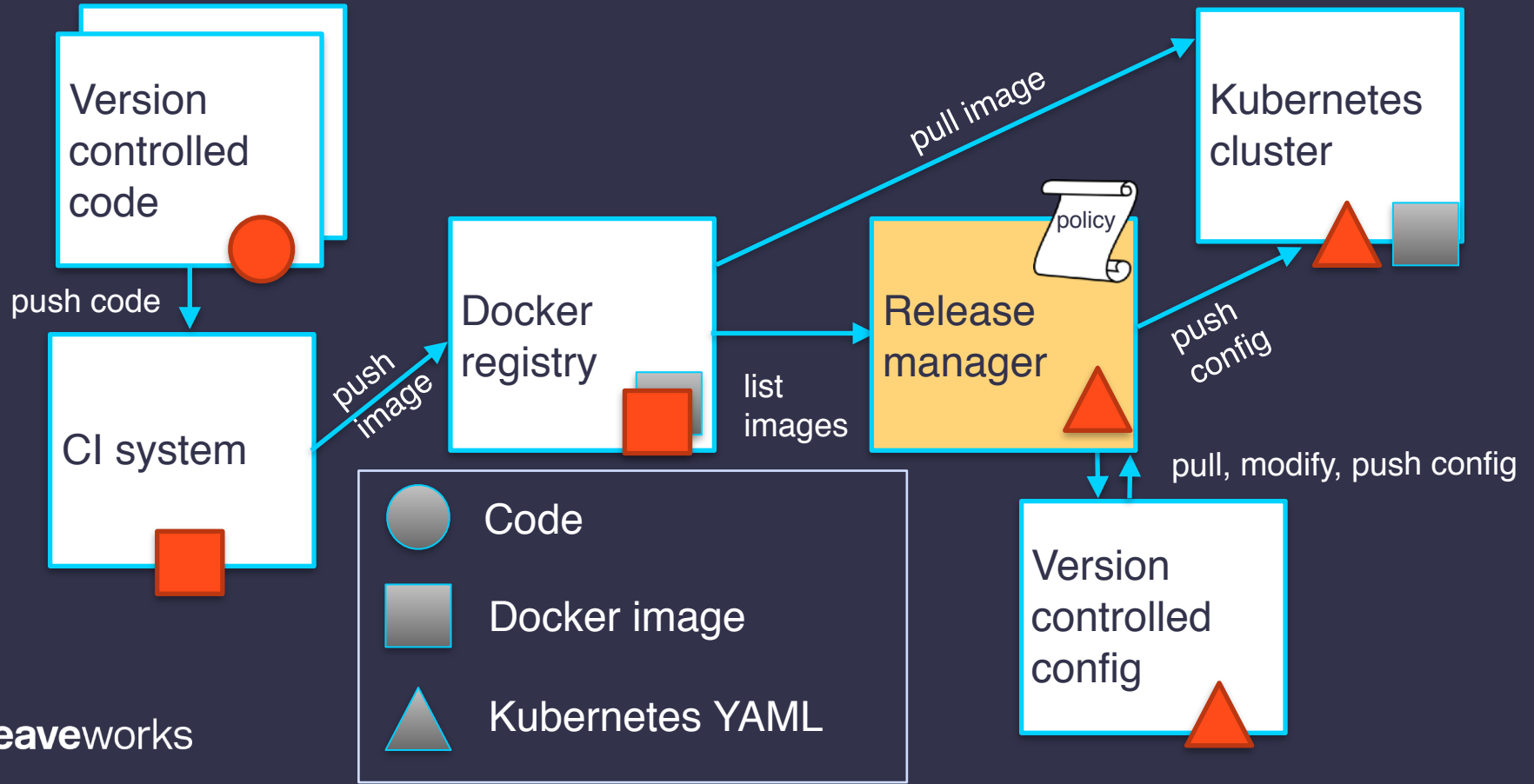
weaveworks

V3 architecture



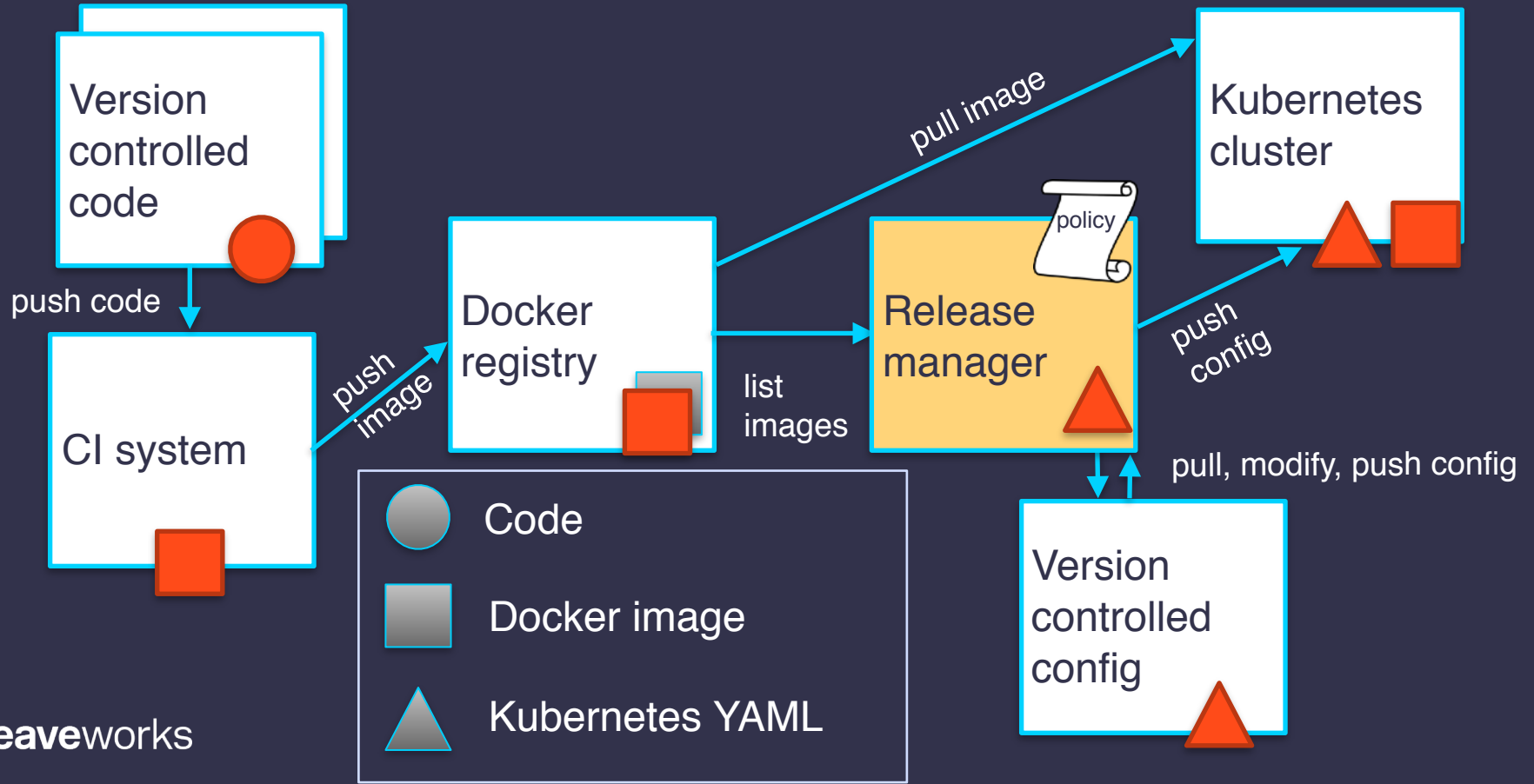
weaveworks

V3 architecture



weaveworks

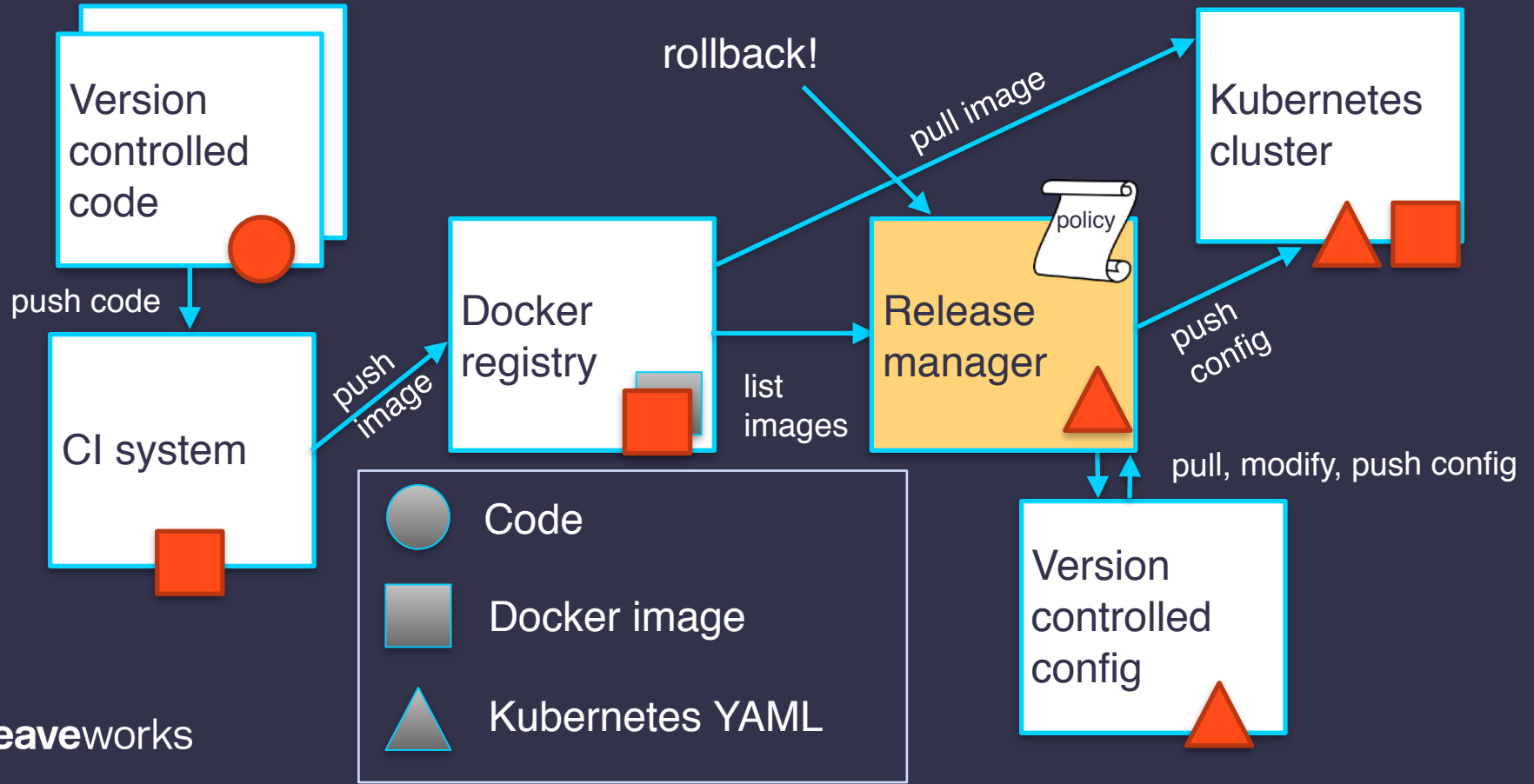
V3 architecture



weaveworks

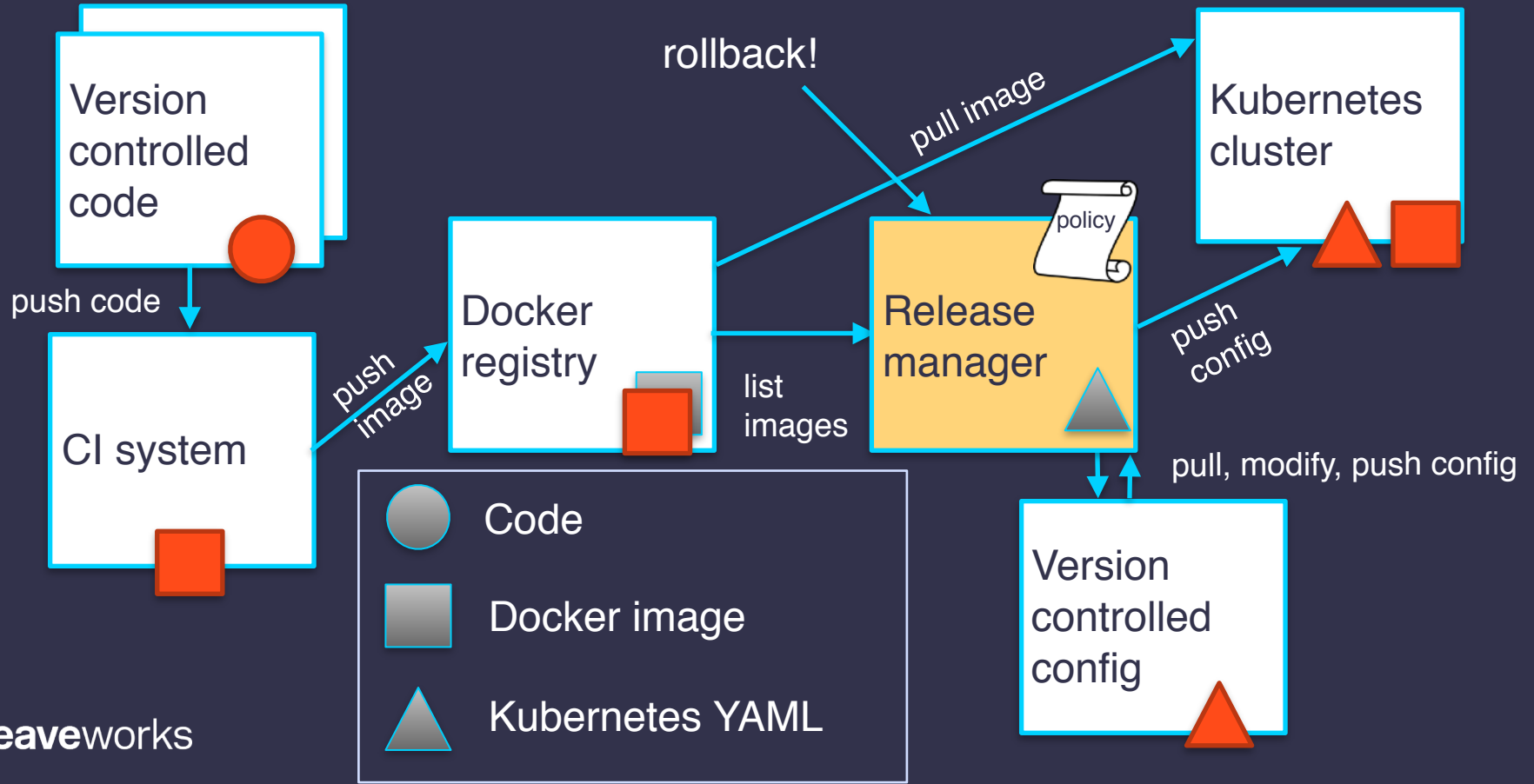
V3
Rollback doesn’t go via CI

V3 architecture



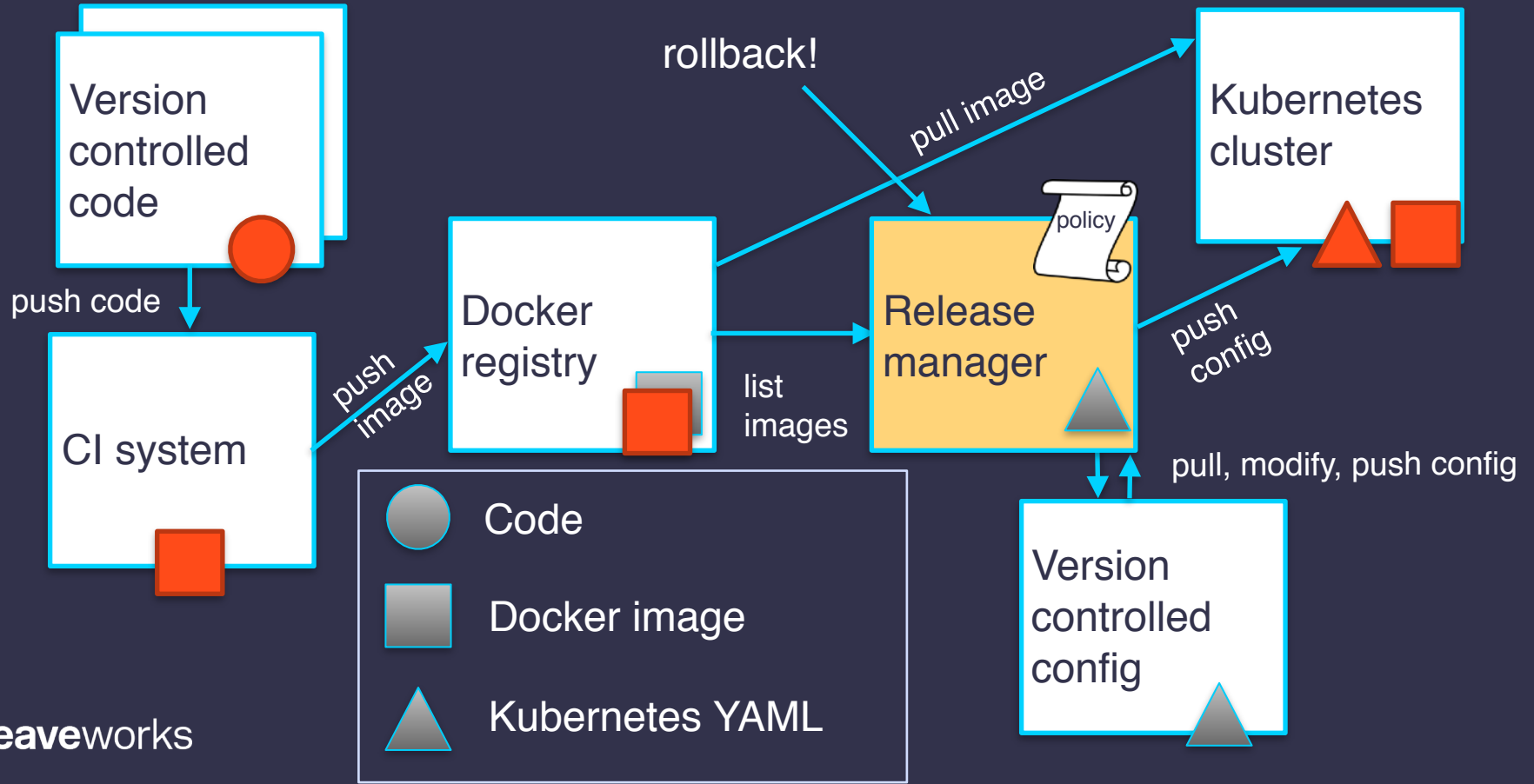
weaveworks

V3 architecture



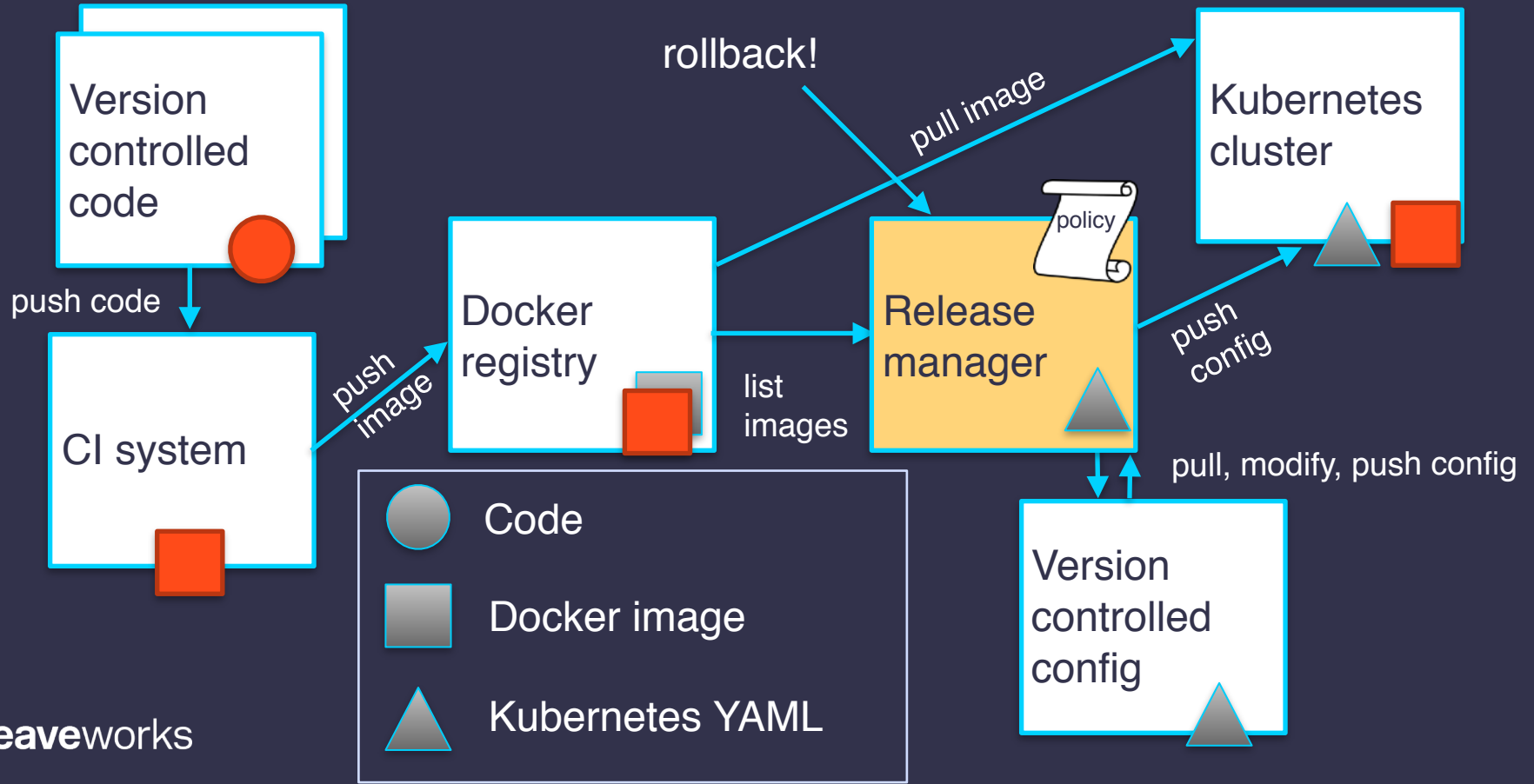
weaveworks

V3 architecture



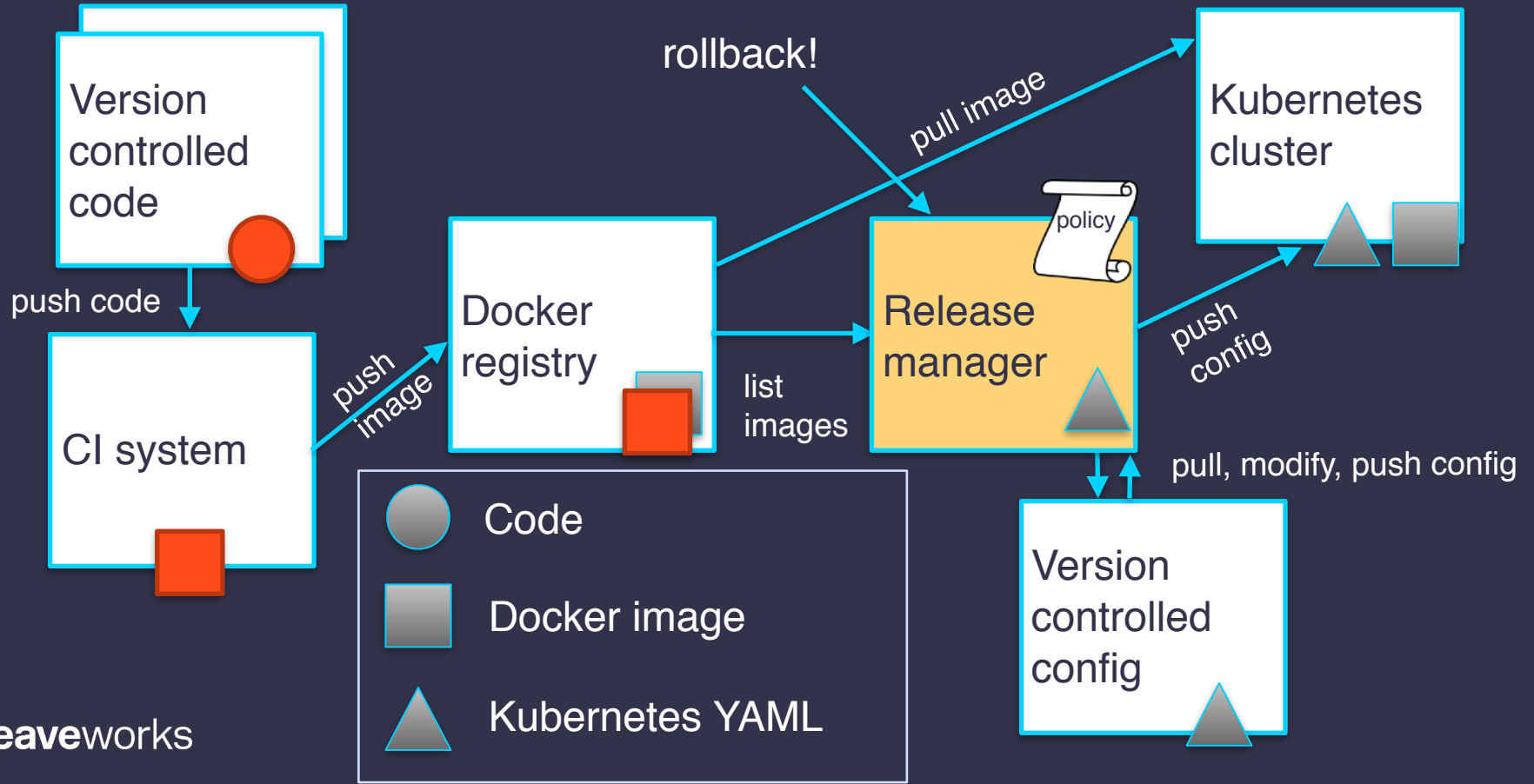
weaveworks

V3 architecture



weaveworks

V3 architecture



weaveworks

What does the release manager do?

- Watches for changes in a container registry (output of CI system)
- Makes commits for you to version controlled configuration (understands Kubernetes YAML)
- Depending on release policy (per environment), either push changes *continuously* or permit manually gated releases
- Allows releases to be rolled back by changing a pointer
- Releases can be “locked” as a social cue

Different environments can have different release policies
(no tight coupling between individual microservices repos and what's released)

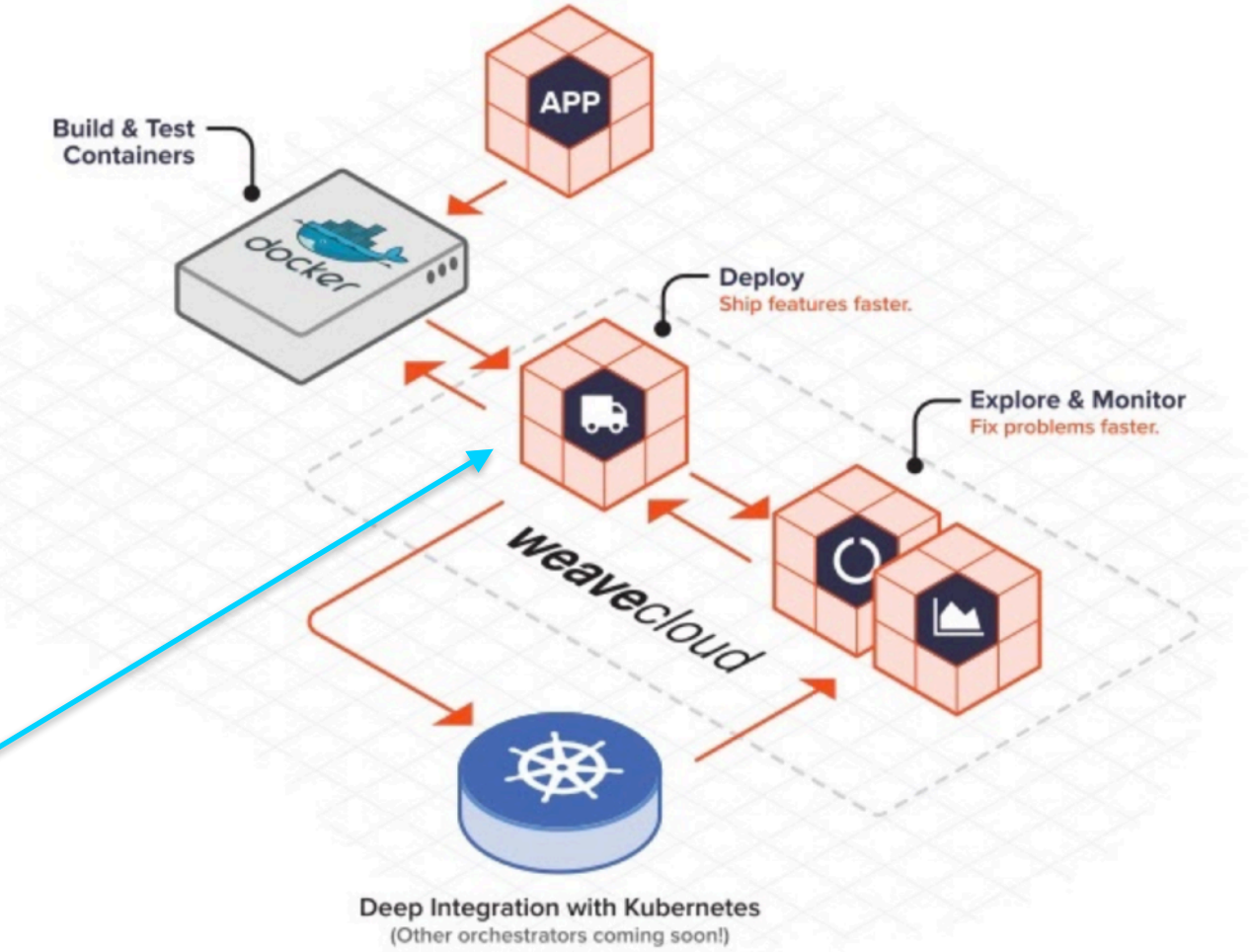
Demo!

This is how we deploy Weave Cloud

Weave Cloud helps devops iterate faster with:

- observability & monitoring
- continuous delivery
- container networks & firewalls

Weave Flux is a release manager for Kubernetes



weaveworks

Other topics

- Kubernetes 101
- How do I monitor this stuff? (Prometheus)
- Network policy for isolating & firewalling different microservices

We have talks & trainings on all these topics in the Weave user group!

Join the Weave user group!
meetup.com/pro/Weave/

Come hang out on Slack!
weave.works/help

Check out Flux on GitHub: github.com/weaveworks/flux

Thanks! Questions?

We are hiring!
DX in San Francisco
Engineers in London & SF

weave.works/weave-company/hiring