Dogfooding Openshift with our CI infrastructure

Michalis Kargakis
@kargakis
Software Engineer, Red Hat

Agenda

- Problems (of the past)
- Goals
- Benefits (of the present)
- CI architecture
- Problems (of the present)
- Future work

Problems (of the past)

- “CI is slow”
- Tribal knowledge of our CI tools
- Nobody wants to maintain CI infra
- Non-extensible, no multirepo support

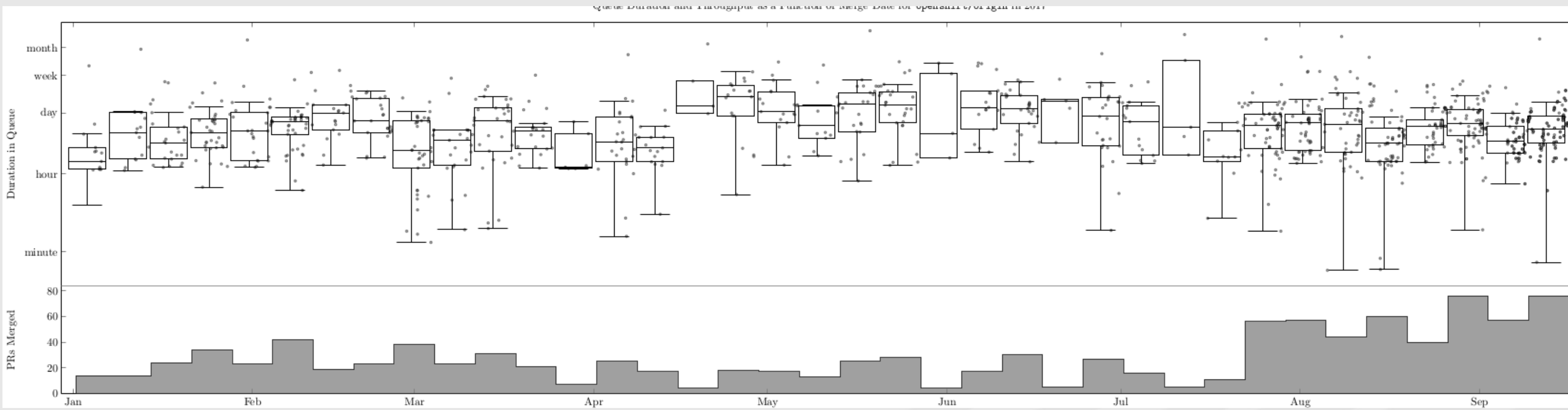
Goals

- Test and merge *efficiently*
- Support multiple repositories
- Make CI infra maintenance fun

Benefits (of the present)

- “CI is fast”
- Providing feedback to our development teams (dogfooding)
- Co-maintenance with the Kubernetes community
- CI infra shares code with core Kubernetes!
- Multirepo support, extensible, “quantum” CI

Benefits (of the present)



Duration of a PR in the queue stays the same, merge throughput quadruples

OPENSHIFT
by Red Hat

CI architecture

- Replaced the old bot with goodies from k8s/test-infra
- **prow** is responsible for testing and merging PRs and all the user interactions in between
- **Jenkins** is still in the picture (for now)
- Test results/artifacts are pushed by Jenkins/prow into GCS buckets
- **gubernator** exposes test results/artifacts

CI architecture

- prow needs to run on top of a Kubernetes cluster
- Extends the Kubernetes API with ProwJobs
- Comprised by a set of microservices that act as controller loops for ProwJobs
- Each service is responsible for a specific task

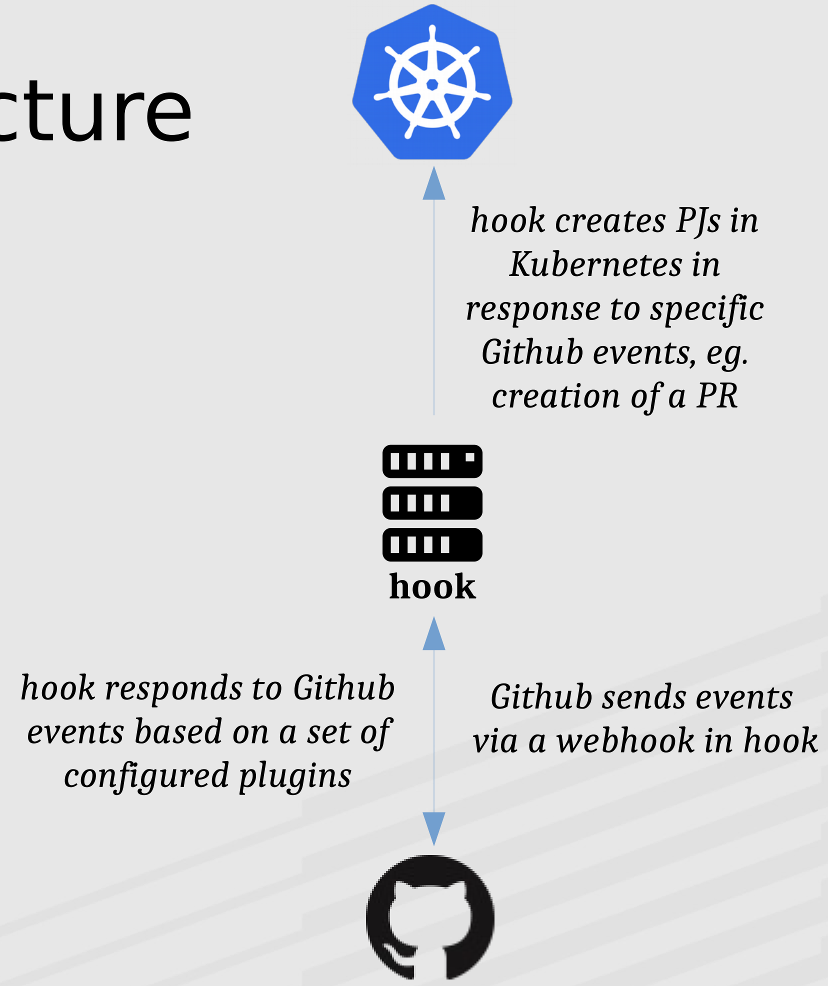
CI architecture

Entrypoint services for creating tests in the cluster:

- **hook**, based on Github events
- **horologium**, based on configured periodic jobs
- **tide** handles merging and retesting pull requests

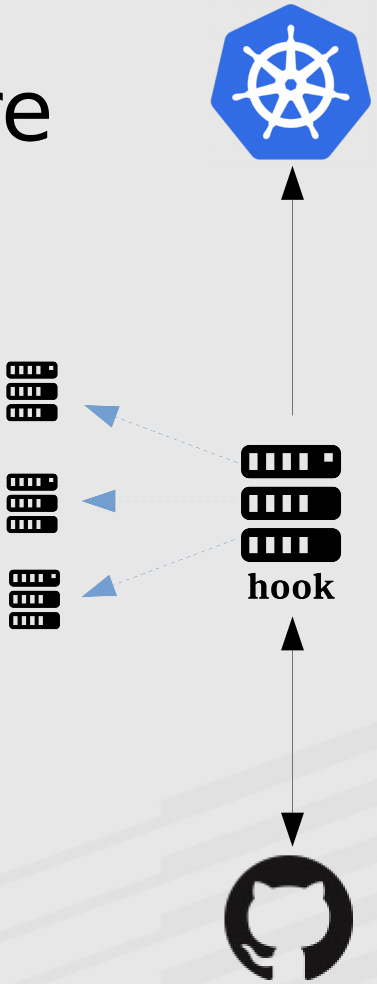
OPENSHIFT by Red Hat

CI architecture


hook creates PJs in Kubernetes in response to specific Github events, eg. creation of a PR
hook
hook responds to Github events based on a set of configured plugins
Github sends events via a webhook in hook

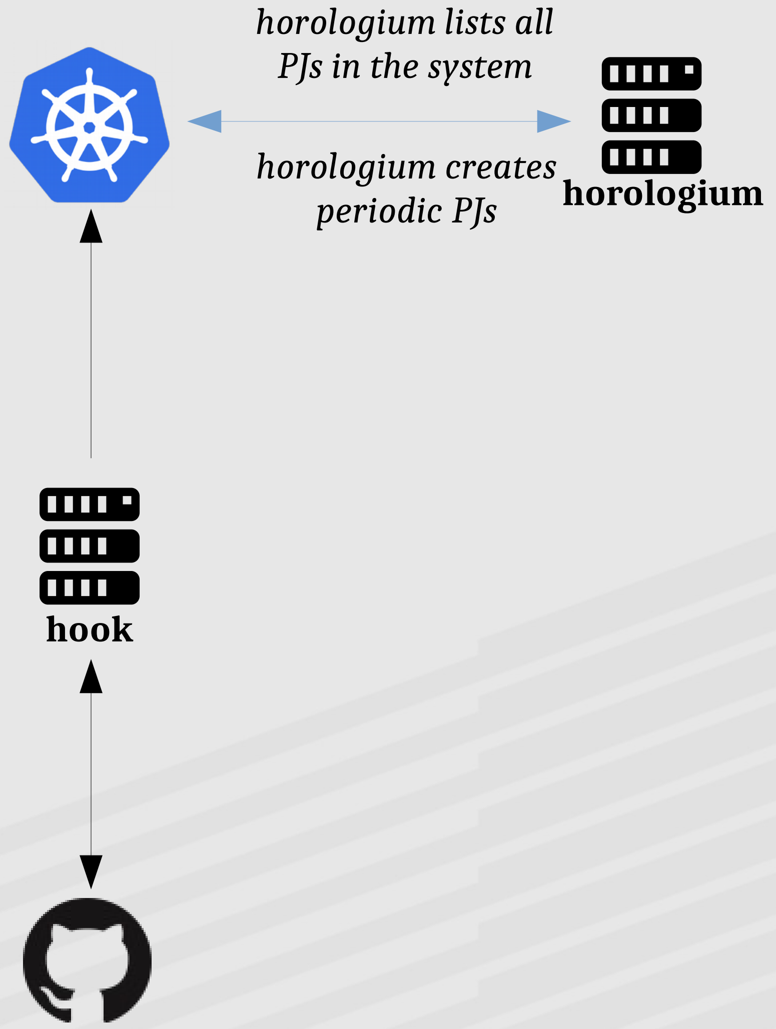
OPENSHIFT by Red Hat

CI architecture

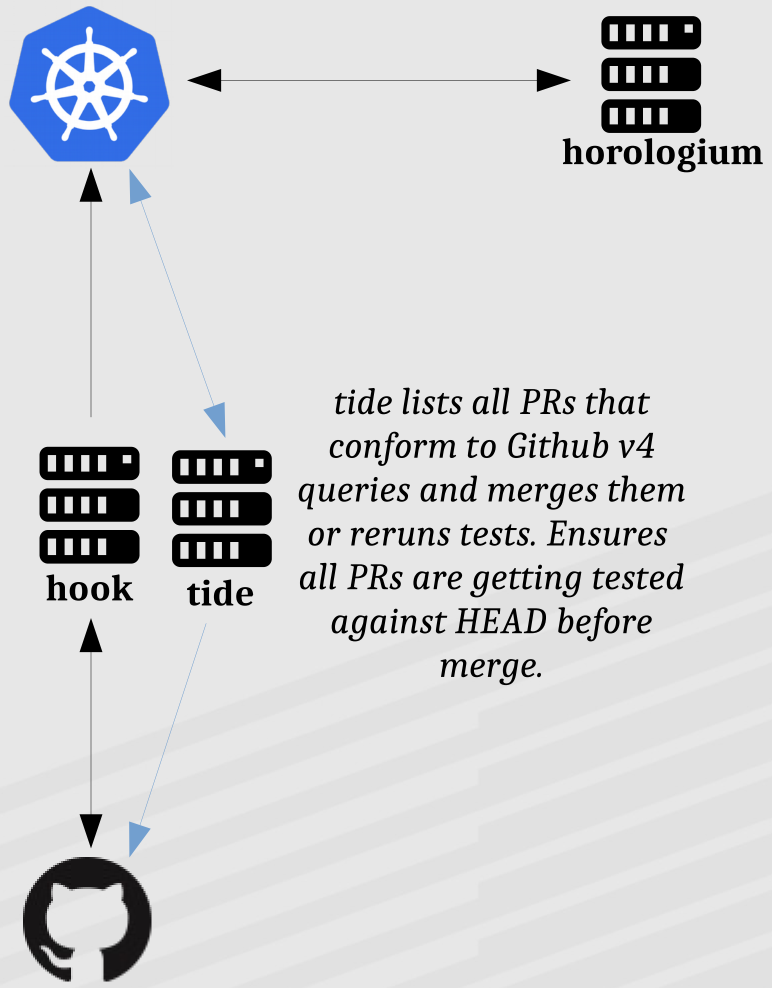


hook can demux Github events to other services - prow plugins

CI architecture


horologium lists all PJs in the system
horologium creates periodic PJs
horologium
hook

CI architecture



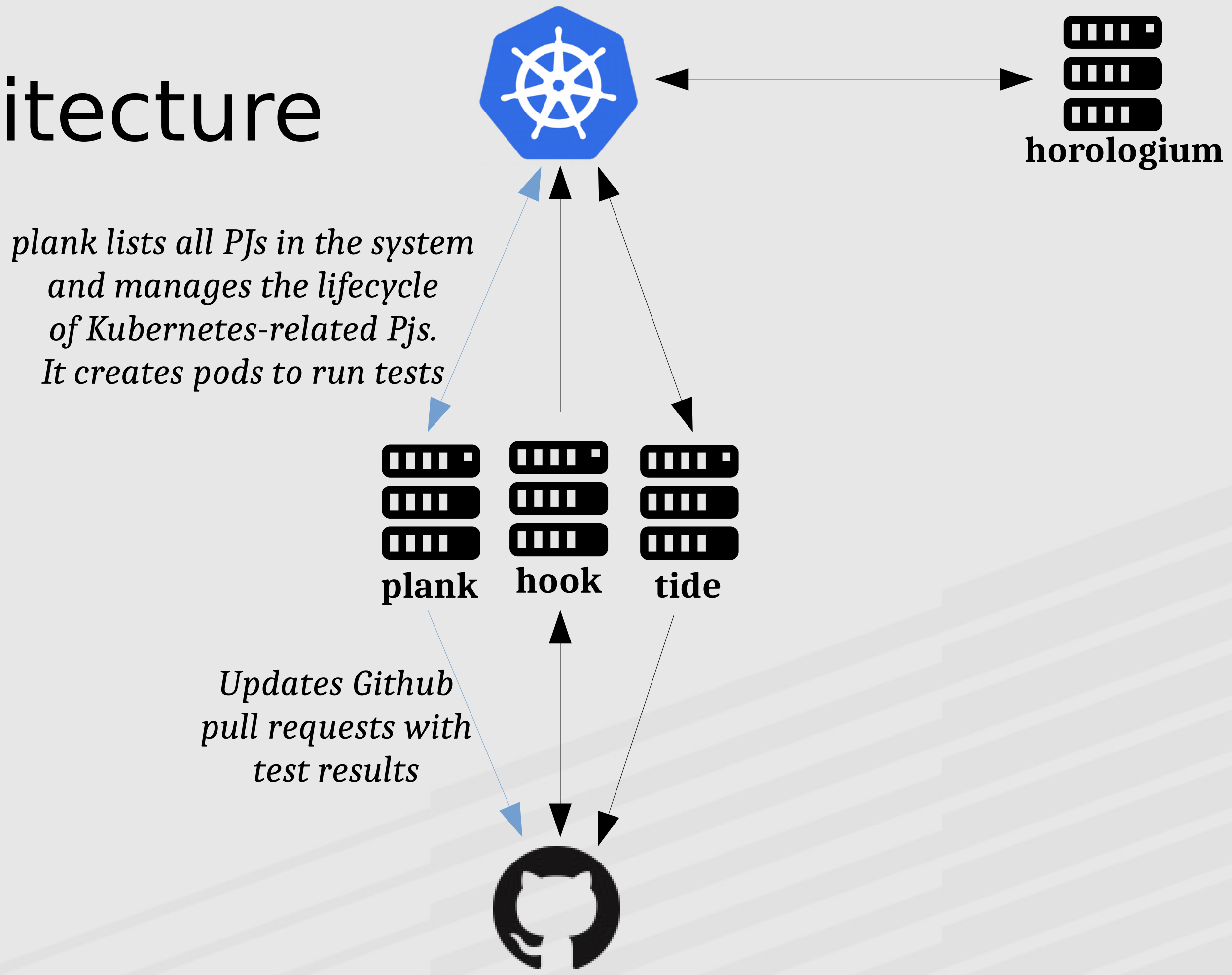
tide lists all PRs that conform to Github v4 queries and merges them or reruns tests. Ensures all PRs are getting tested against HEAD before merge.

CI architecture

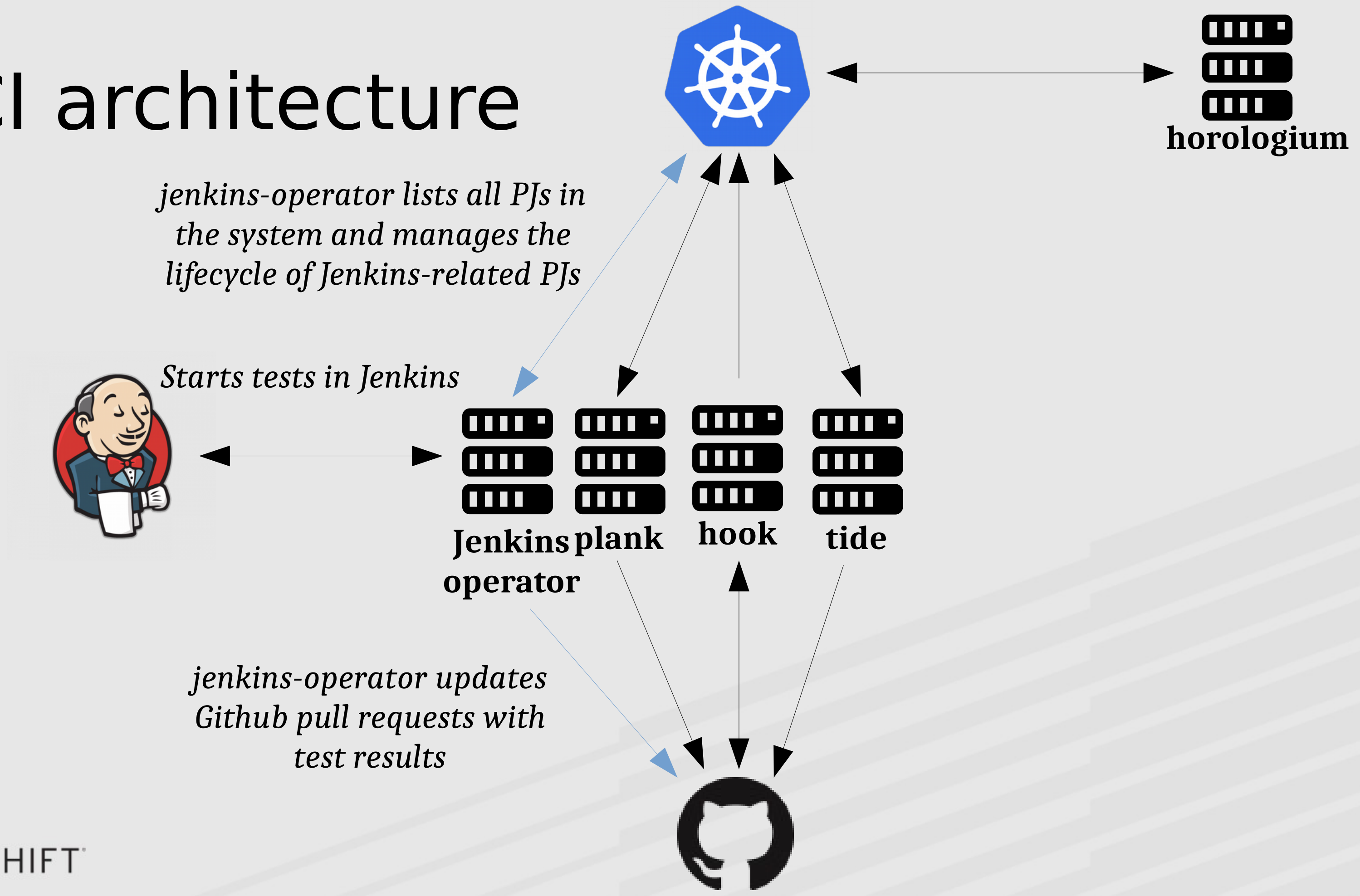
Two services for managing the lifecycle of tests:

- **plank** runs tests in Kubernetes pods
- **jenkins-operator** runs tests in Jenkins

CI architecture


horologium
plank lists all PJs in the system and manages the lifecycle of Kubernetes-related Pjs. It creates pods to run tests
plank
hook
tide
Updates Github pull requests with test results

CI architecture

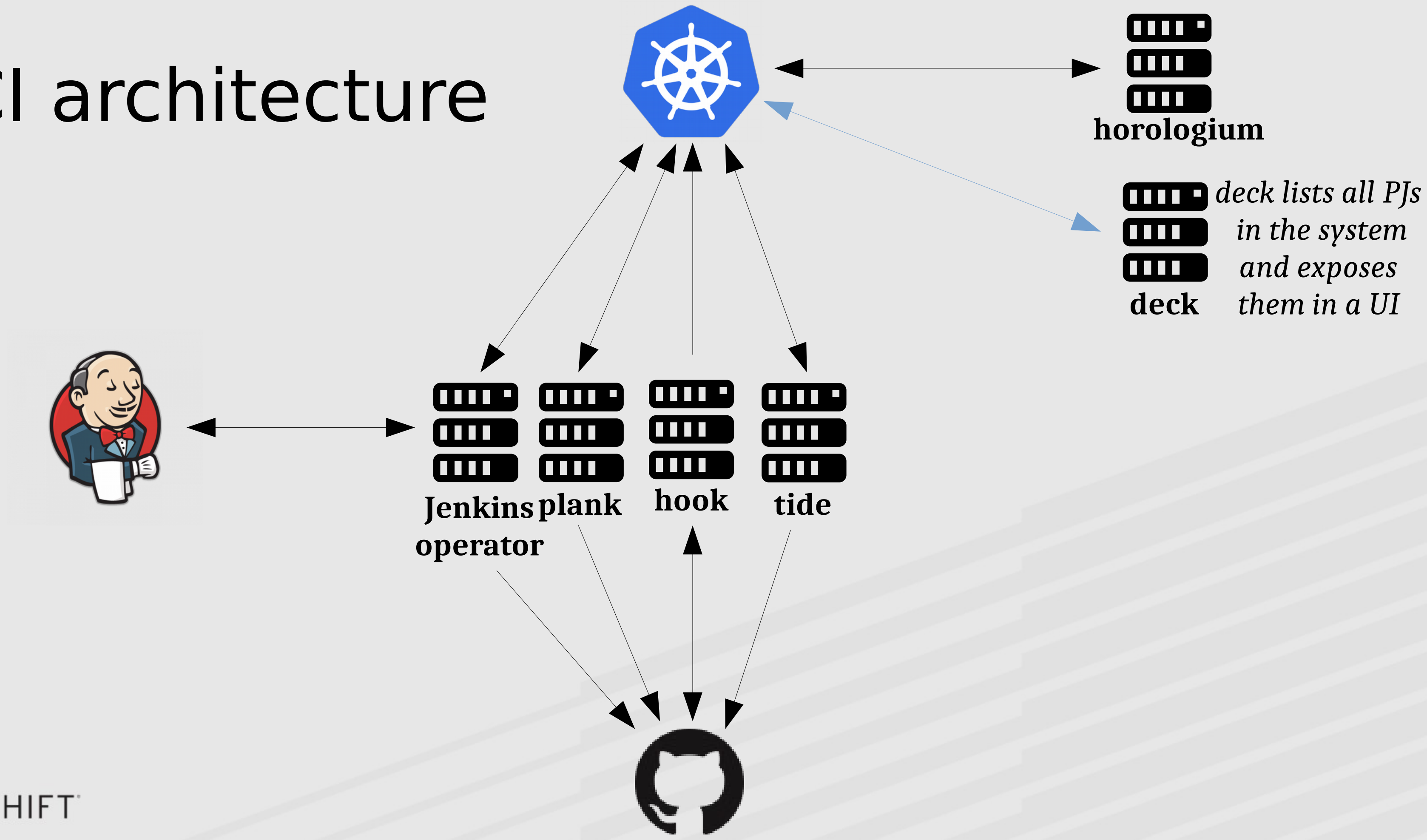

jenkins-operator lists all PJs in the system and manages the lifecycle of Jenkins-related PJs
Starts tests in Jenkins
horologium
Jenkins operator
plank
hook
tide
jenkins-operator updates Github pull requests with test results

CI architecture

Others:

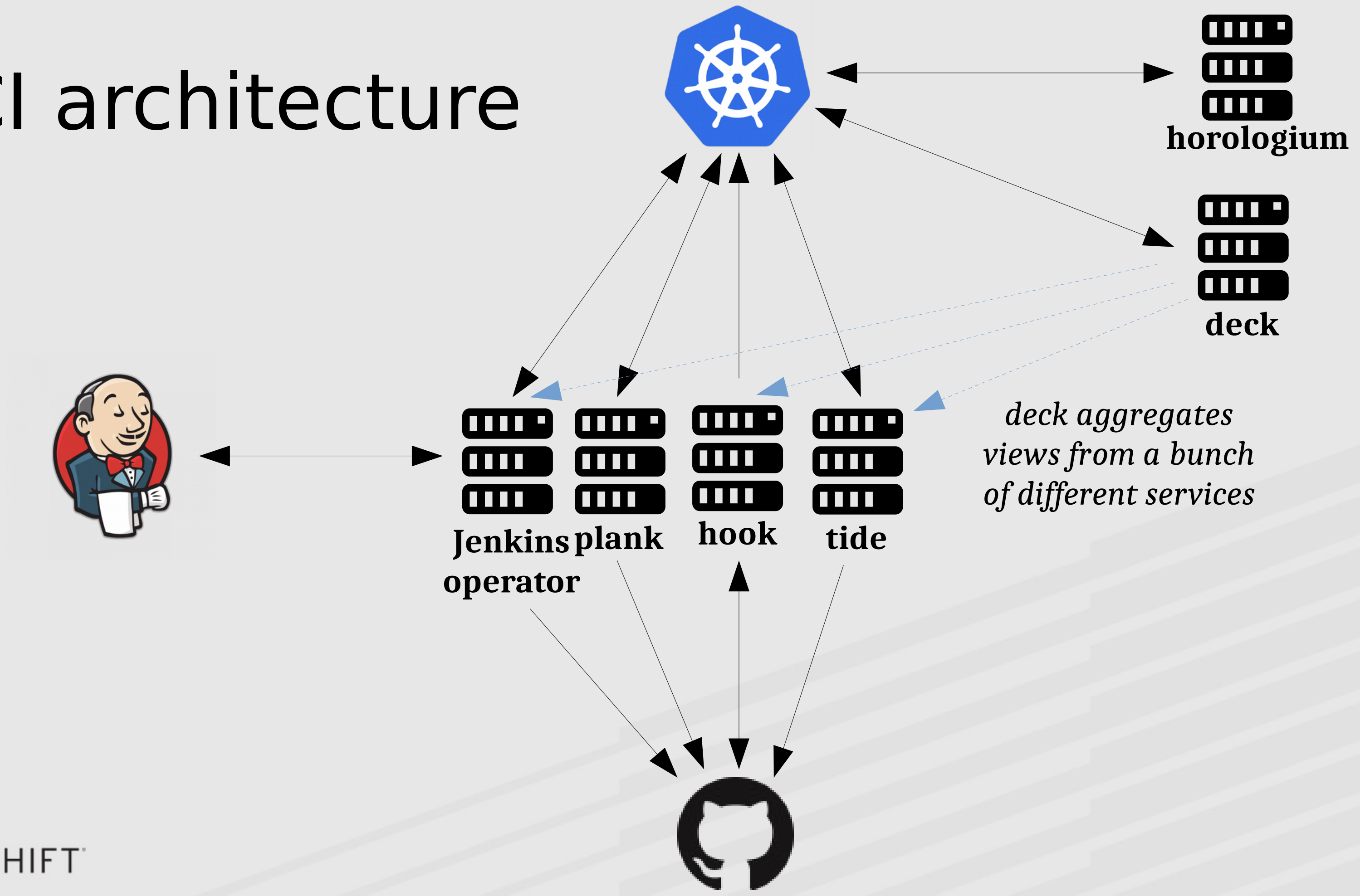
- **deck** is the front-end of prow
- **sinker** handles garbage collection of old PJs and pods

CI architecture



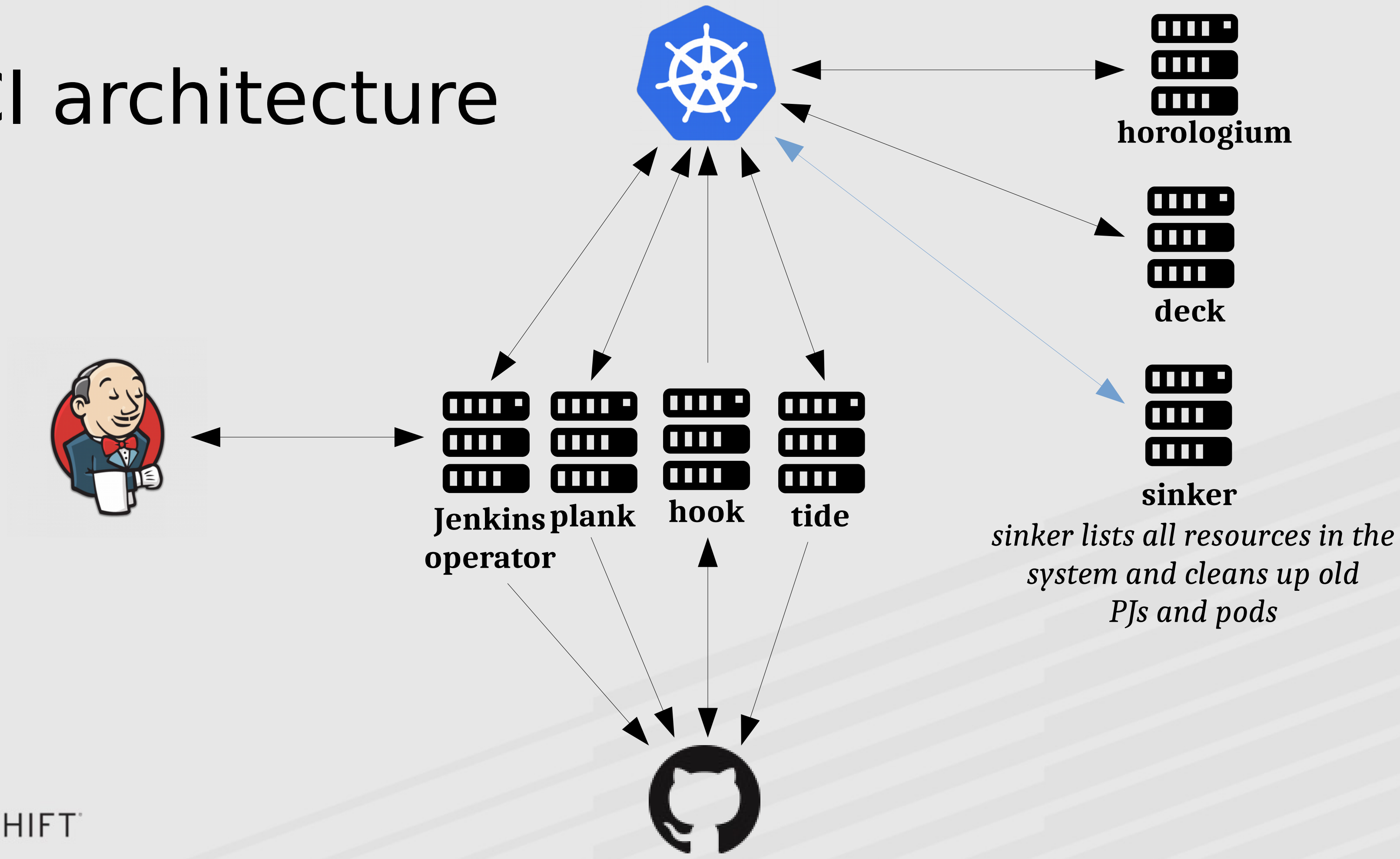
OPENSHIFT by Red Hat

CI architecture



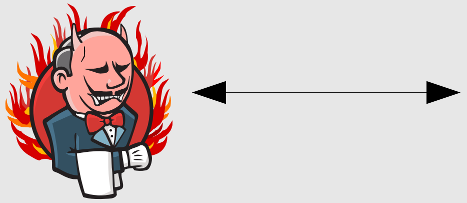
deck aggregates views from a bunch of different services

CI architecture



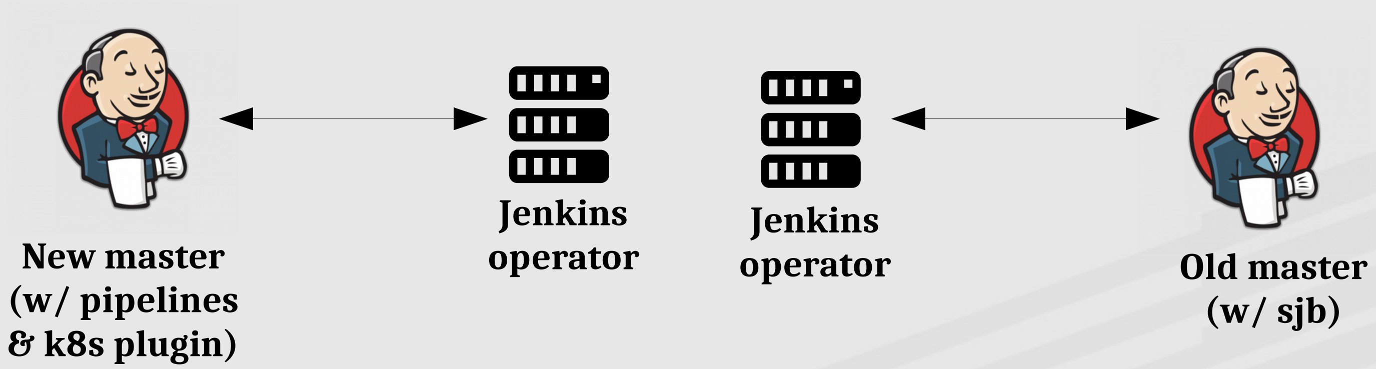
sinker lists all resources in the system and cleans up old PJs and pods

Problems (of the present)

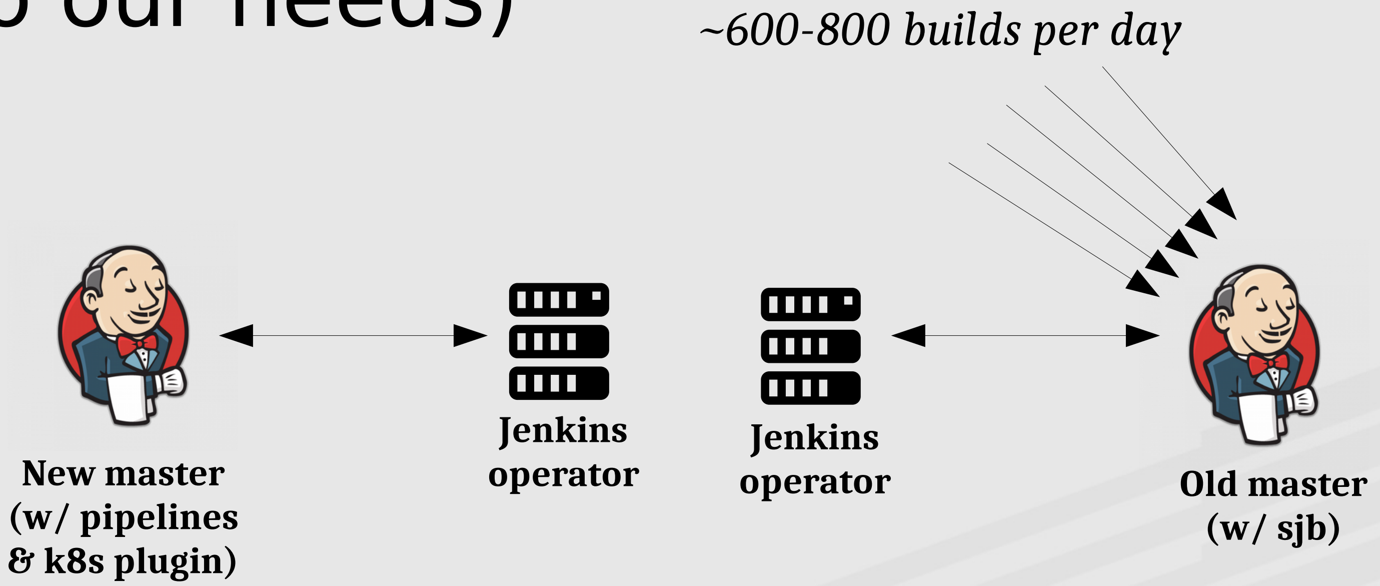


- ✓ **Jenkins k8s plugin does not scale to our needs**
- ✓ **Wrong use of Jenkins pipelines**
- ✓ Impossible to fix itself when it crashes
- ✓ Jenkins needs babysitting once it comes back after a crash
- ✓ … and more…

Problems (k8s plugin does not scale to our needs)

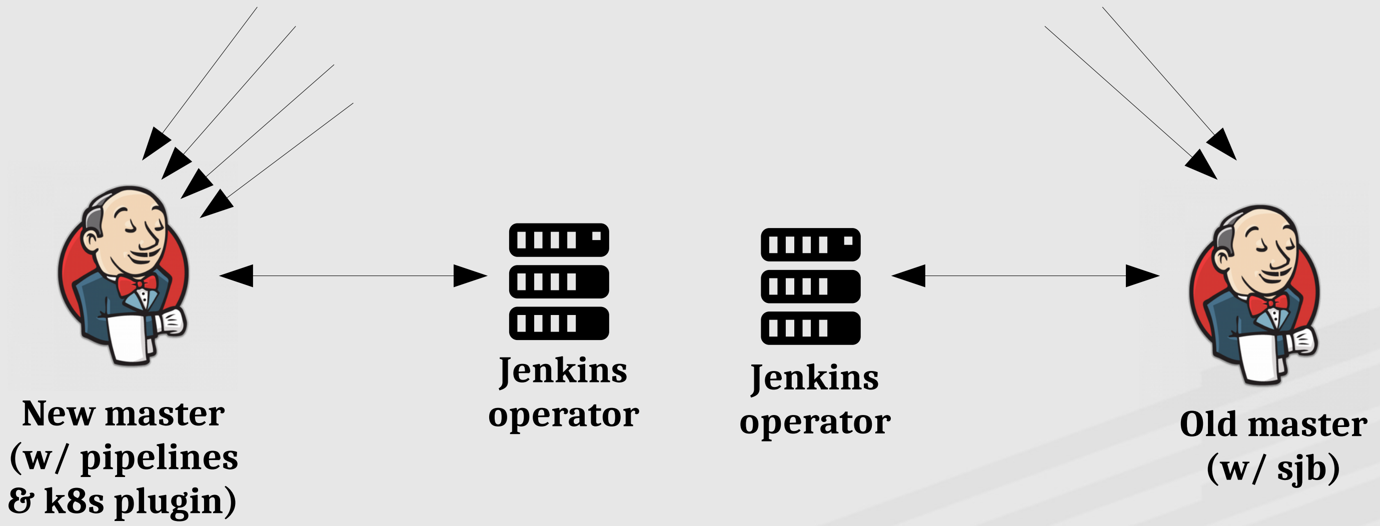


Problems (k8s plugin does not scale to our needs)

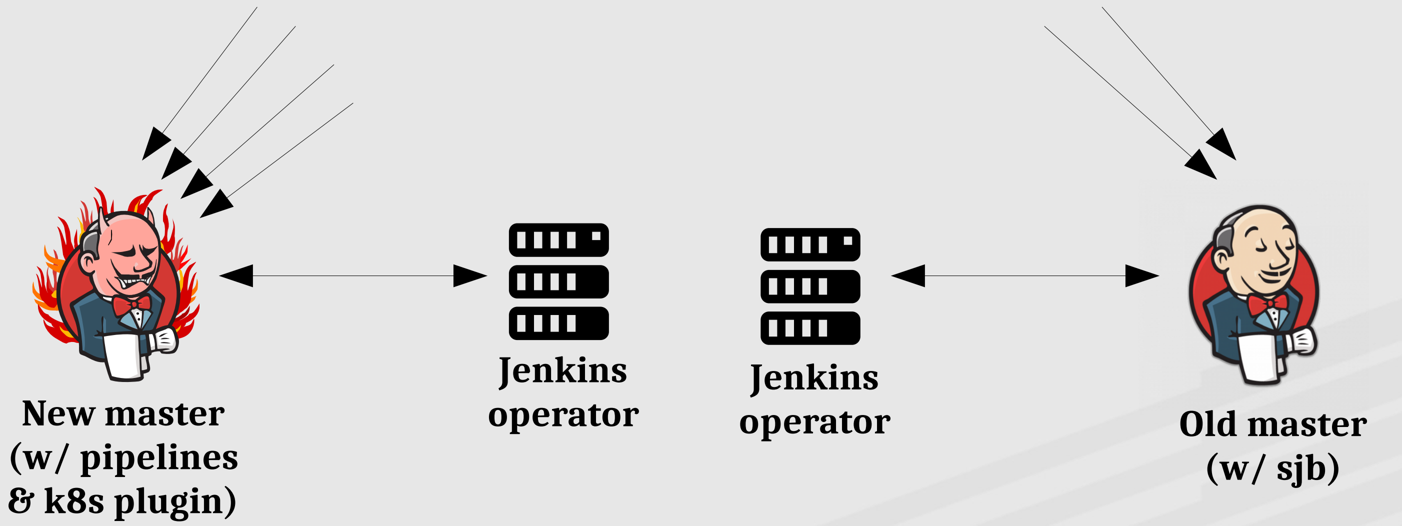


OPENSHIFT by Red Hat

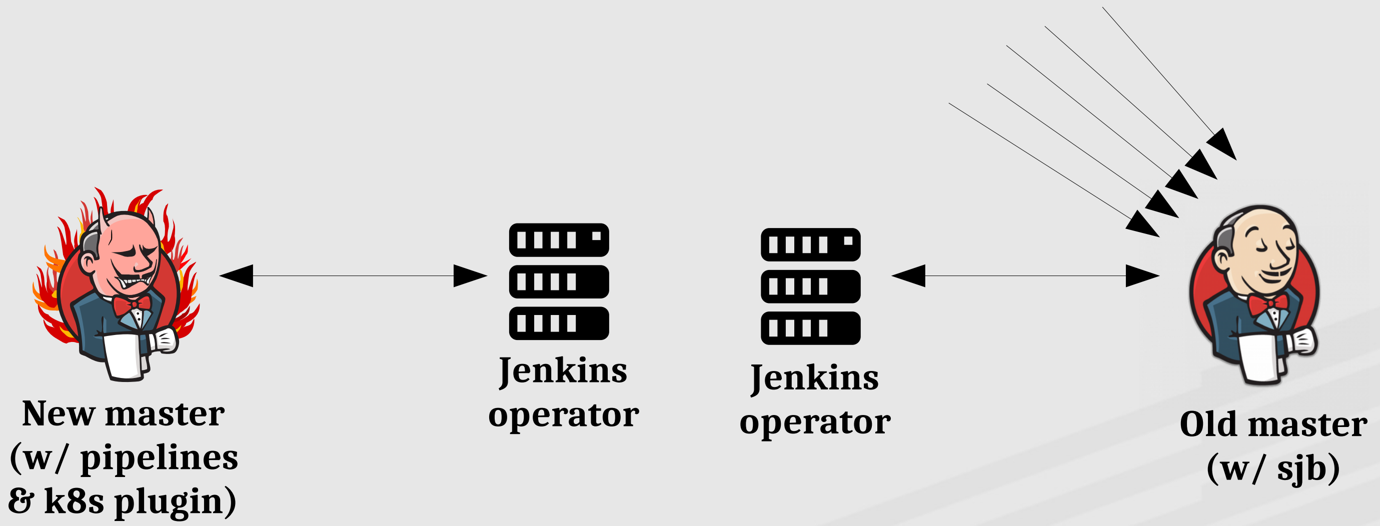
Problems (k8s plugin does not scale to our needs)



Problems (k8s plugin does not scale to our needs)



Problems (k8s plugin does not scale to our needs)

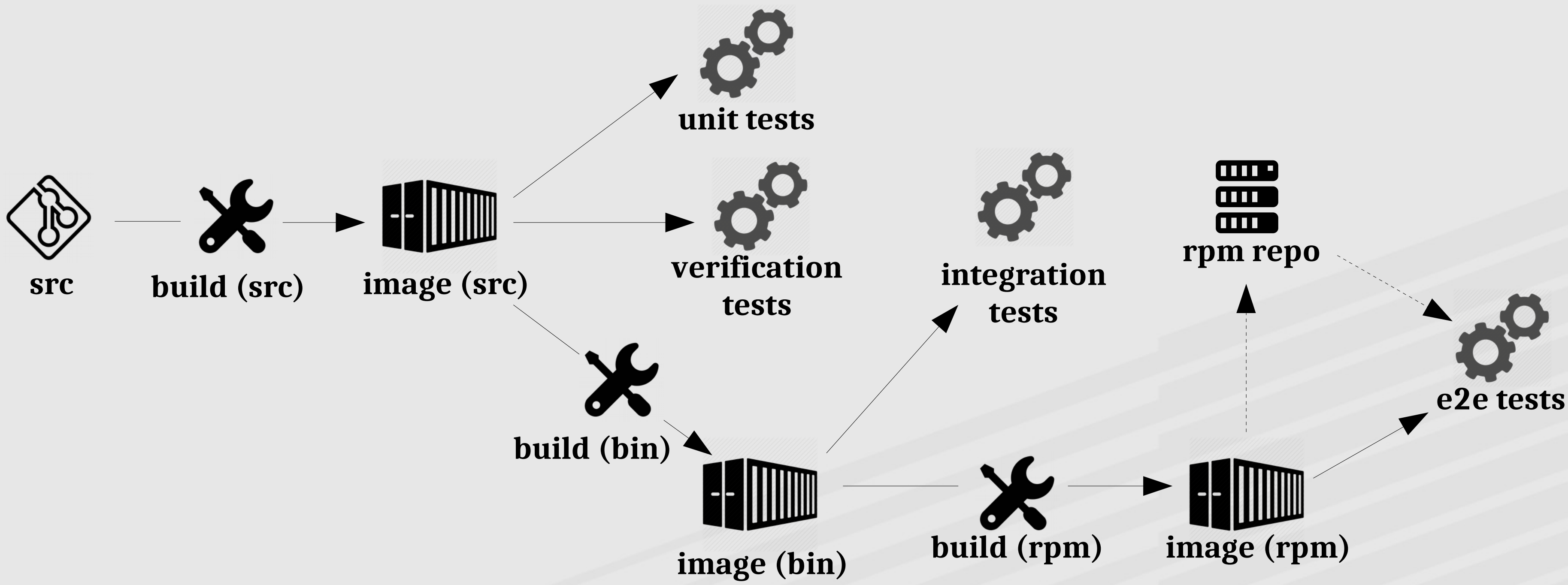


Problems (Wrong use of pipelines)

Goal: Use Openshift to test Openshift:

- **Builds** can run builds
- Tests can run in **Pods**
- **ImageStreams** can store images
- Serve RPMs with a **DeploymentConfig**

Problems (Wrong use of pipelines)



Problems (Wrong use of pipelines)

- Hard to express intent with such a flow
- Ended up with a lot of **imperative** steps
- Reached Groovy to its limits
- Jenkins pipelines are meant to be used **declaratively**

Future work

- Move all Openshift repos to use prow
- Re-architect Jenkins pipelines into Golang pipelines
- Release a declarative API to users that want to setup testing for their projects

OPENSHIFT
by Red Hat

OPENSHIFT
by Red Hat

Fin