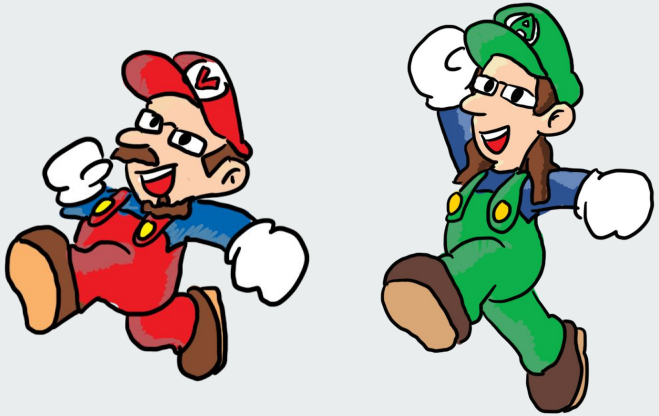
Mario's Adventures in Tekton Land

Andrea Frittoli — IBM
Vincent Demeester — Red Hat

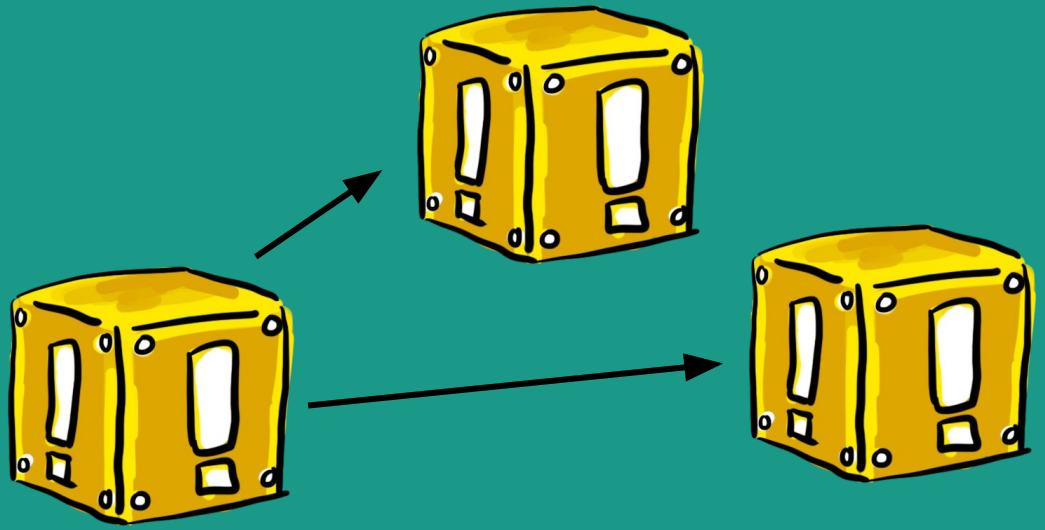


San Diego, 19.11.2019
KubeCon + CloudNativeCon NA

Agenda

- What is Tekton ?
- Mario’s plumbing
- Using our own pipes
- Future work

What is Tekton ?



Tekton

An open-source project for providing a set of shared and standard components for building Kubernetes-style CI/CD systems

Governed by the Continuous Delivery Foundation

Contributions from Google, Red Hat, Cloudbees, IBM, Pivotal and many more

Tekton in a nutshell

Standard Kubernetes-style pipelines

Declarative pipelines with standard Kubernetes custom resources (CRDs) based on Tekton*

Build images with Kubernetes tools

Use tools of your choice (source-to-image, buildah, kaniko, jib, etc) for building container images

Run pipelines in containers

Scale pipeline executions on-demand with containers on Kubernetes

Deploy to multiple platforms

Deploy applications to multiple platforms like serverless, virtual machines and Kubernetes

Powerful command-line tool

Run and manage pipelines with an interactive command-line tool

Tekton Pipeline concept

Step
Run commands in a container with volumes, env vars, etc

Task
A list of steps that run sequentially in the same pod

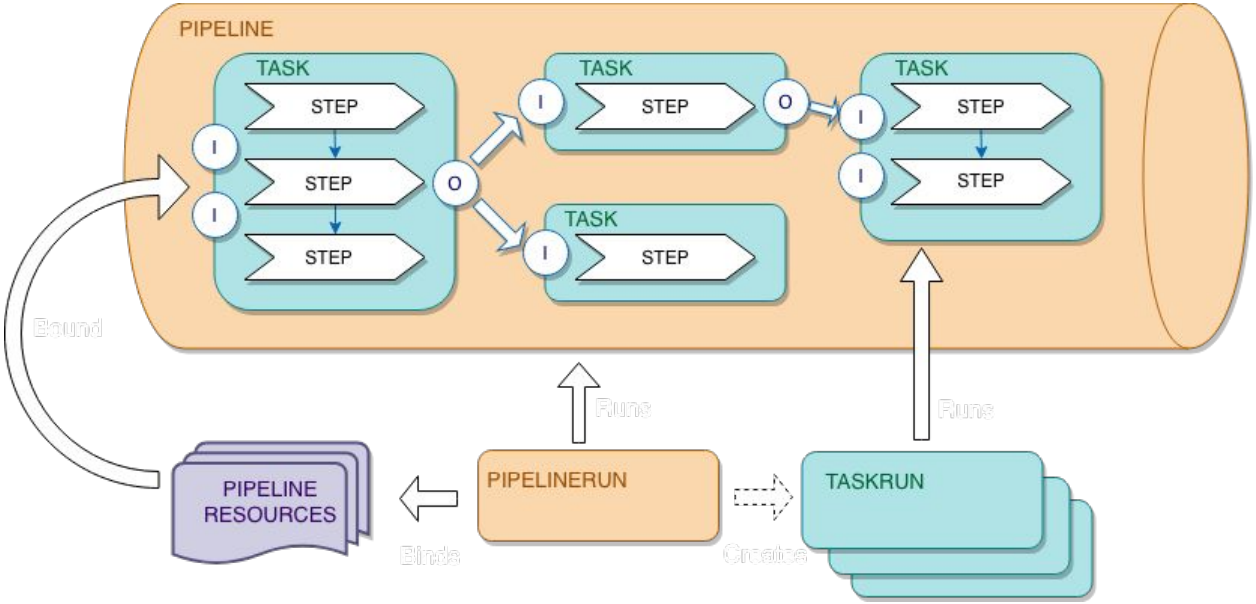
Pipeline
A graph of tasks with inputs and outputs executed in a certain order

PipelineResource
Inputs and outputs to tasks and pipelines (git, image, etc)

TaskRun
An invocation of a task with inputs and outputs

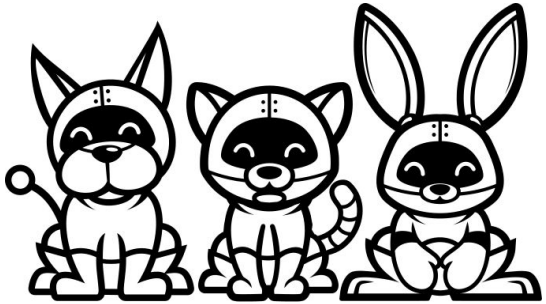
PipelineRun
An invocation of a pipeline with inputs and outputs

Tekton Pipeline concept



Tekton projects

- github.com/tektoncd/pipeline — core
- github.com/tektoncd/catalog — shareable task definition
- github.com/tektoncd/cli — command-line to interact with pipeline
- github.com/tektoncd/triggers — create tekton resources in reaction of events
- github.com/tektoncd/operator — install/upgrade you tekton
- github.com/tektoncd/dashboard — web ui for tekton



Mario’s Plumbing

Plumbing

… « the system of pipes, tanks, fittings, and other apparatus required for the water supply, heating, and sanitation in a building. »

For us this means, all tools and configuration files for the testing and automation needs of Tekton:

- Continuous integration system
- release setup
- test infrastructure
- scripts (for the CI, tests, release, infrastructure)
- GitHub issues and pull-request management (labels, /lgtm, …)
- …

Initial Plumbing

Tekton inherits its plumbing from Knative.

- Infrastructure on Google cloud
- Heavy usage of Prow
- Sharing scripts and container images with Knative

Kubernetes & Knative uses test-infra, we thought plumbing was more fun :)

Prow

« Prow is a Kubernetes based CI/CD system. Jobs can be triggered by various types of events and report their status to many different services. In addition to job execution, Prow provides GitHub automation in the form of policy enforcement, chat-ops via /foo style commands, and automatic PR merging. »

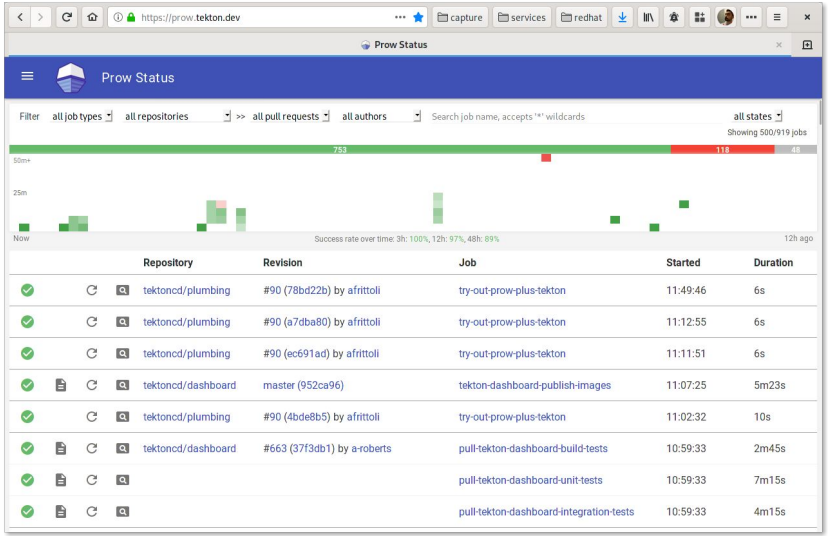
Prow is **the** CI/CD system for projects that works on top of Kubernetes.

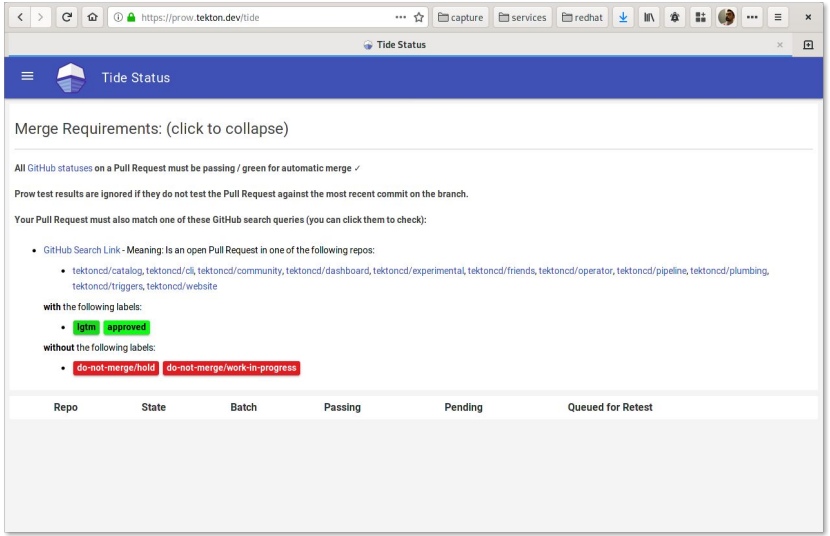
It is written and optimized for Kubernetes project's need (and OpenShift).

Prow core components

- **hook**, stateless server that listens for GitHub webhooks and dispatches them to the appropriate plugins.
- **plank**, controller that manages the job execution and lifecycle for jobs that run in k8s pods.
- **deck**, view of recent jobs, command and plugin help information, the current status and history of merge automation, and a dashboard for PR authors.
- **tide** manages retesting and merging PRs once they meet the configured merge criteria.
- **horologium** triggers periodic jobs when necessary.
- **Sinker** cleans up old jobs and pods.

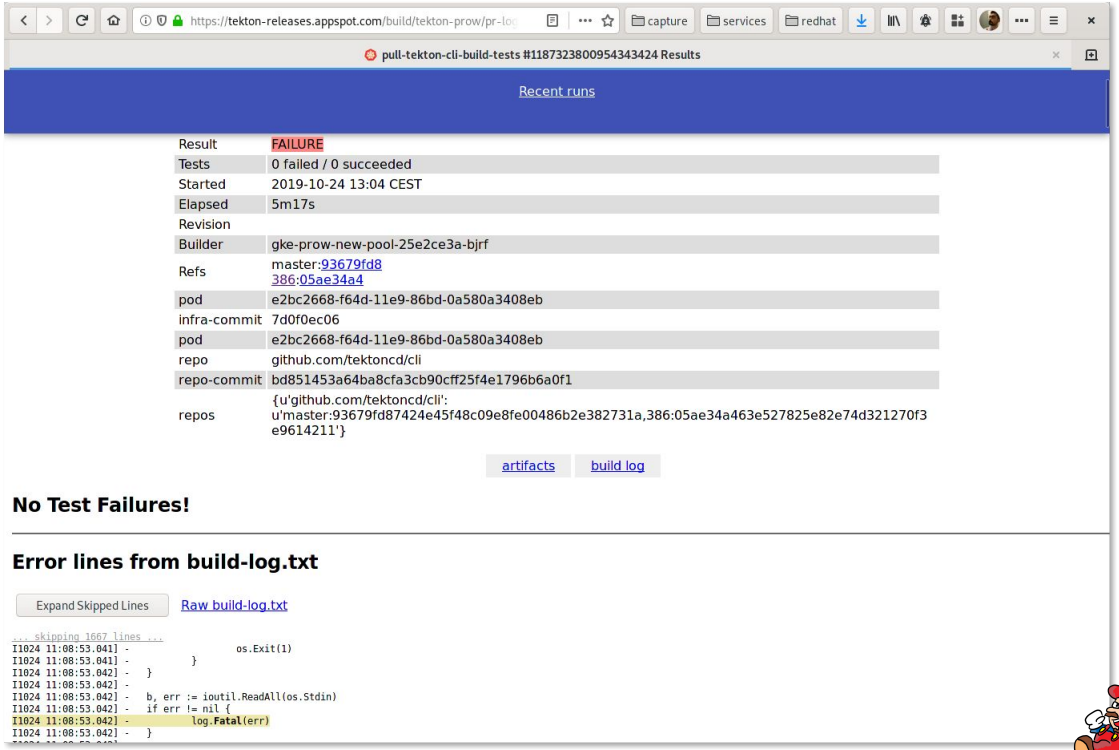
Prow dashboard (deck)





Logs

Gubernator



Use our own pipes

Step 0: Use our own scripts

Goal: First step towards independence from Knative

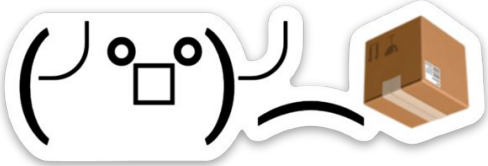
- Create of `tektoncd/plumbing`
- Document our plumbing
- Own our CI scripts
- Own our CI images

Step 1: Release Tekton with Tekton

Goal: Build and Release Tekton using Tekton

- Tasks
  - Build, test and lint Go code from `tektoncd/catalog`
  - Publish images using `ko`
  - Generate `release.yaml` from published images
  - Publish `release.yaml` to the GCS bucket
- Pipeline, PipelineResource and PipelineRun
- Executable *anywhere…*

...well, almost



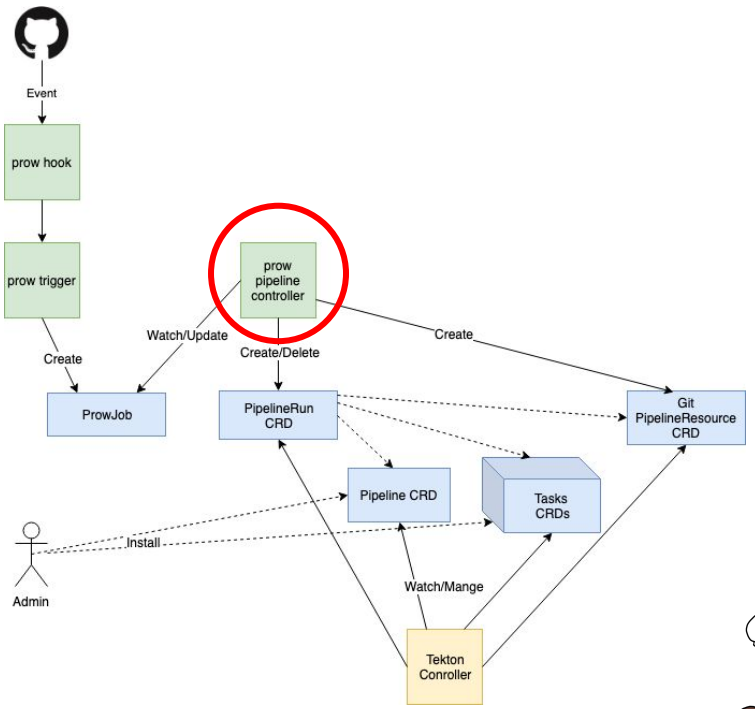
Step 2: Prow and Tekton

	Prow	Tekton
GitHub webhook	hook	triggers (with interceptors)
Job execution / lifecycle	plank	pipeline
Dashboard	deck	
Merging “bot”	tide	
Periodics job	horologium	triggers + k8s cronjobs
Garbage collection	sinker	

Step 2: Prow and Tekton

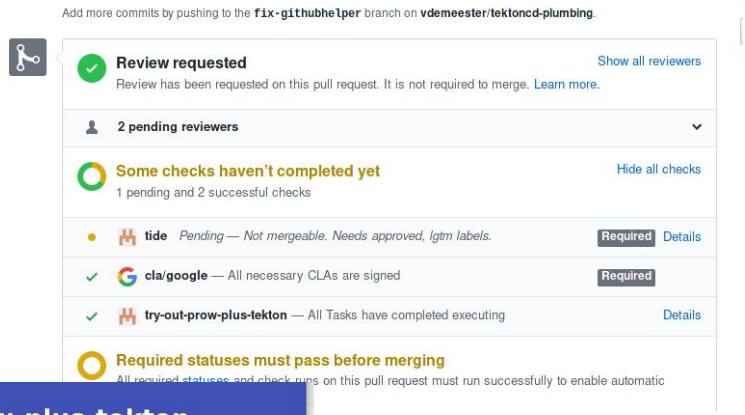
- Adds a “tekton-pipeline” agent
- Controller that watch “tekton-pipeline” jobs
- Creates Resources (PipelineResource, PipelineRun)

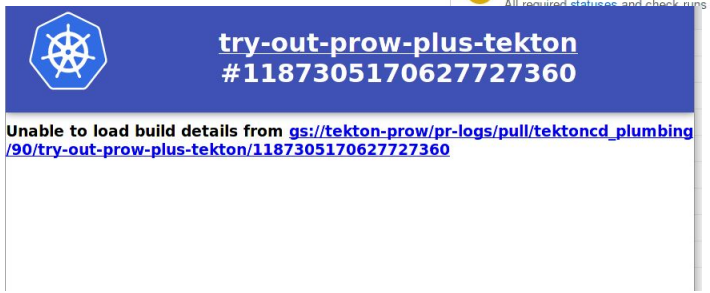
```
  - name: try-out-prow-plus-tekton
    agent: tekton-pipeline
    trigger: "(?m)^/run
(all|try-tekton),?(\\s+|$)"
    pipeline_run_spec:
      pipelineRef:
        name: special-hi-scott-pipeline
      resources:
      - name: git
        resourceRef:
        name: PROW_IMPLICIT_GIT_REF
```



Step 2: Prow and Tekton issues

- Logs are not integrated
- Limited to test-infra tekton version
  test-infra depends on 0.3.1
  current release is 0.8.0
  ^^ **incompatible**
  Test-infra has lots of inertia (k8s)
- Prow complexity





Step 3: Dogfooding cluster

Two versions of Tekton in the same cluster?

...not so easily

New Kubernetes cluster:

- **Green-field** experiment with standalone Tekton
- Introduce services in parallel to Prow
- Test new services in a **risk free** environment
- Incrementally roll-out new production services
- ...and finally **lock down**!

Step 3': Logs for everyone

Persistent and **public** access of TaskRun and PipelineRun logs

*Pods are **ephemeral** and **private**!*

No Tekton custom solution to export logs

- Dogfooding app to provide access to logs
- Nightly releases: Tekton Task to export logs to a bucket

Build "1177563153492348928" (Pipeline "special-hi-scott-pipeline")

```
[2019-09-27T12:38:30Z] 1 hello scott
[2019-09-27T12:38:30Z] 2 hello scott
[2019-09-27T12:38:30Z] 3 hello scott
[2019-09-27T12:38:30Z] 4 hello scott
[2019-09-27T12:38:30Z] 5 hello scott
[2019-09-27T12:38:30Z] 6 hello scott
[2019-09-27T12:38:30Z] 7 hello scott
[2019-09-27T12:38:30Z] 8 hello scott
[2019-09-27T12:38:30Z] 9 hello scott
[2019-09-27T12:38:30Z] 10 hello scott
[2019-09-27T12:38:30Z] 11 hello scott
[2019-09-27T12:38:30Z] 12 hello scott
[2019-09-27T12:38:30Z] 13 hello scott
[2019-09-27T12:38:30Z] 14 hello scott
[2019-09-27T12:38:30Z] 15 hello scott
[2019-09-27T12:38:30Z] 16 hello scott
[2019-09-27T12:38:30Z] 17 hello scott
[2019-09-27T12:38:30Z] 18 hello scott
[2019-09-27T12:38:30Z] 19 hello scott
[2019-09-27T12:38:30Z] 20 hello scott
[2019-09-27T12:38:30Z] 21 hello scott
[2019-09-27T12:38:30Z] 22 hello scott
```

Step 3”: Prow-less Releases

- Same tasks for both release types (Tekton v0.7+)
- Kubernetes CronJobs triggers for nightly
- Manual trigger (tkn) for full releases
- Everything “as code”
- Pre-release checks

- Asynchronous Tasks:
  - Log collection
  - Post release testing (WIP)

Name	Size	Type
pipeline-release-nightly-rvkfp-2tppc.log	1.15 MB	application/octet-stream
release.yaml	15.52 KB	application/octet-stream

Step 4: Incremental steps to CD

Continuously build images

- Reusable Tekton Trigger (event listener)
- One CronJob per image

...and garbage collection?

Continuously deploy Prow configuration

- Plumbing Issues #1
- Check for changes first
- *Configuration testing, automatic rollback*

Continuously deploy plumbing Tekton resources

- Pre and post-deploy testing
- Stale resources

Continuously deploy Tekton resources from Tekton projects

- Plumbing Cluster Resources
- One namespace per project

Continuously deploy Tekton services

Fix our own pipes

a.k.a Future Work

Step 5: Missing pipes

CI with Tekton

Testing for Plumbing
- Service Configuration
- Resources (Tasks, Pipelines, etc.)

Prow (deck) like dashboard

Chatbot integration
(reuse test-infra microservice ? roll our own ?)

Tide like management or integration

Tekton Dashboard:
- Access control
- Read-only for public access

Make tests portable

Vendor Neutral Tasks ...*run anywhere*

CD Plumbing Services
- Monitoring services
- Tracking versions of resources and configs

Missing Bolts

Finally clauses for Tasks

-> backlog. CloudEvent resource helps

Output params

-> design phase. Emulated with storage

Optional Inputs

-> backlog

Notifications

-> design phase. CloudEvent resource helps

Tasks by Reference

-> design phase. Task embedding helps

Task Hooks, Switch and Loops

-> under discussion

Demo

Was this useful?

Dogfooding is **always** good:

If we can't use it, nobody can

We made a few mistakes:

- Wrong images
- Wrong release version
- Wrong resource version
- Wrong cluster
- Wrong bucket

We experienced a few (unsurprising) things:

- CI/CD is hard ...*and Prow is complex, and useful*
- We need to get *as prow-independent as possible*
- Re-use can be hard

We learned about things we miss:

- Pipes to fix or enhance
- Missing pipes and bolt

We built new projects and tools:

- Plumbing, Triggers
- Logging application (*thanks Scott @sbwsg*)

Come and Join Us

Thank you !

Image credits: Christie Wilson (@bobcatwilson)

Links & References

- Tekton: https://tekon.dev, https://github.com/tektoncd
- Plumbing Repo: https://github.com/tektoncd/plumbing
- Tekton Community: https://github.com/tektoncd/community
- CDF: https://cd.foundation/, https://github.com/cdfoundation
- K8s Test-Infra: https://github.com/kubernetes/test-infra/
- Prow: https://github.com/kubernetes/test-infra/tree/master/prow
- Prow Pipelines (design): https://docs.google.com/document/d/1Lsp4QjzZLev4T2RttHV9srg0y1RiSEiyBjL9sFR7aKg
- KO: https://github.com/google/ko