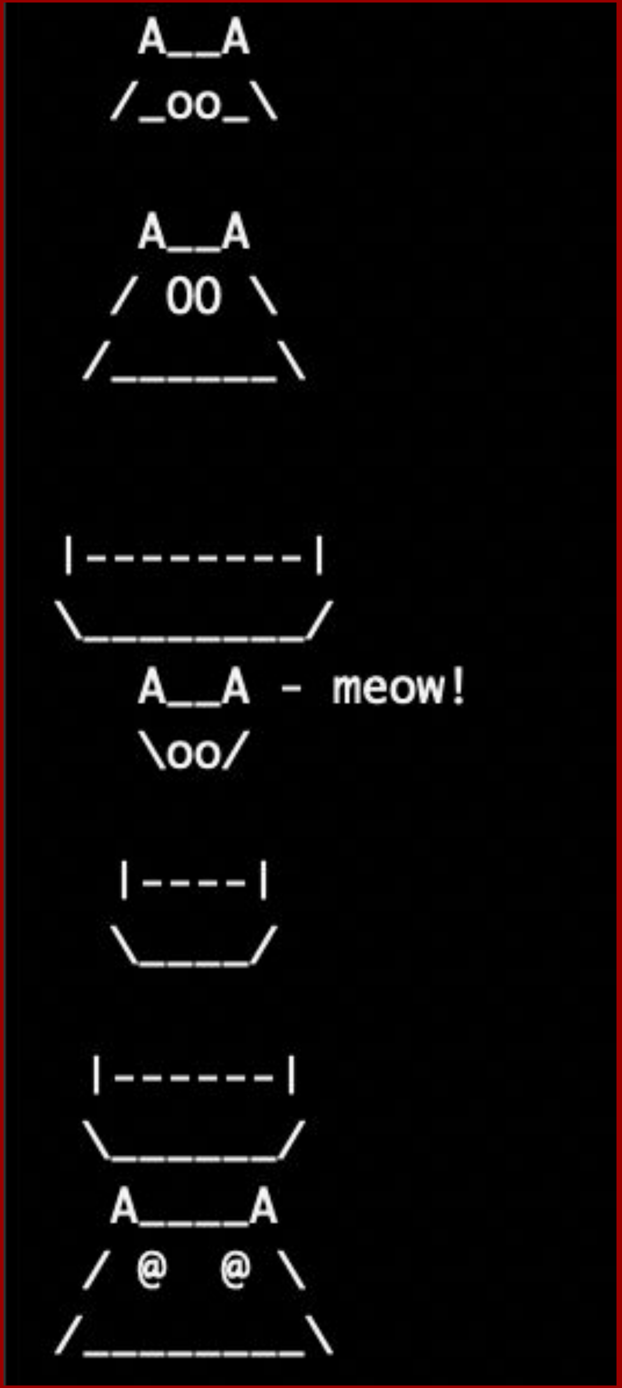
Russian Doll

Extending Containers with Nested Processes

Christie Wilson and Jason Hall (Google)

OH NOES!!1! :(



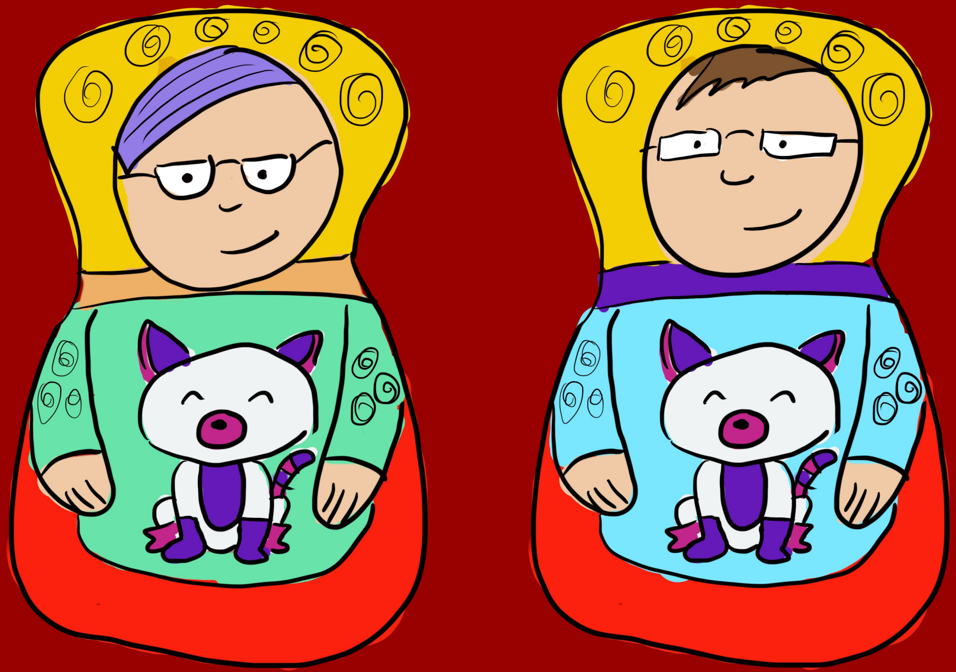
```
    A__A
   /_oo_\

    A__A
   / OO \
  /______\



 |---------|
 \_________/
    A__A  - meow!
    \oo/

   |----|
   \____/

  |------|
  \______/
   A____A
  / @  @ \
 /________\
```

Who We Are

Jason Hall

Christie Wilson

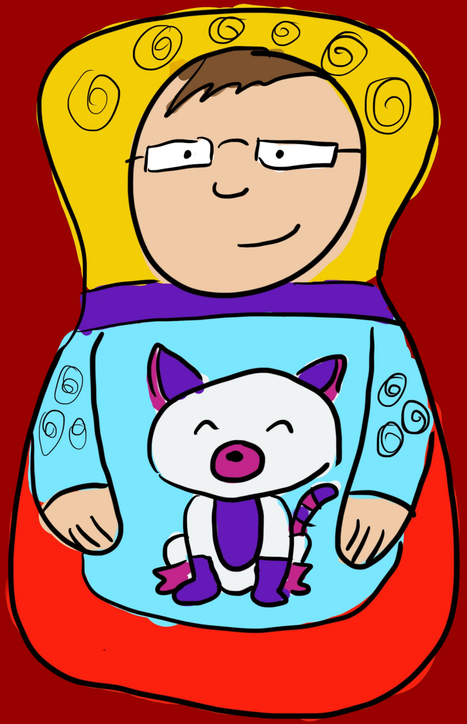
Super powerful or terrible hack? You decide!



Overview

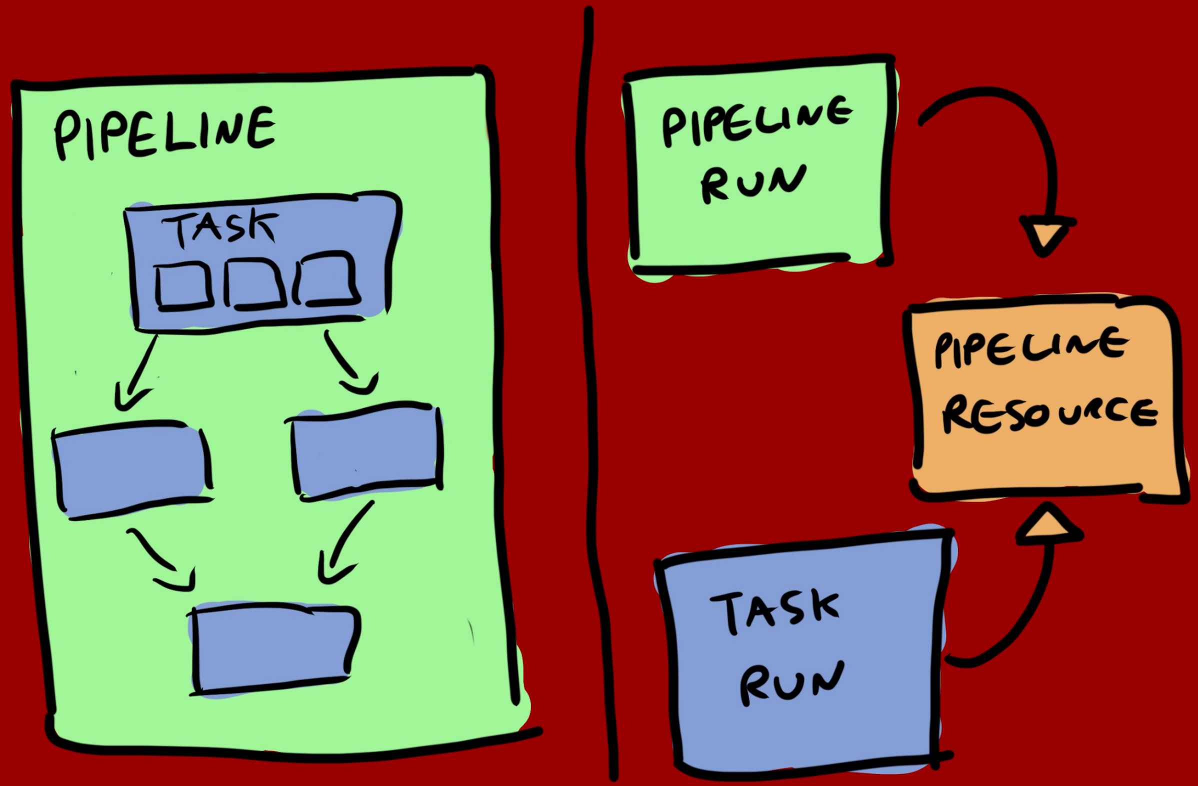
1. Tekton 101
2. But how?
3. Early attempts
4. Magic sauce
5. Demo!
6. Future work

Tekton 101



Tekton 101

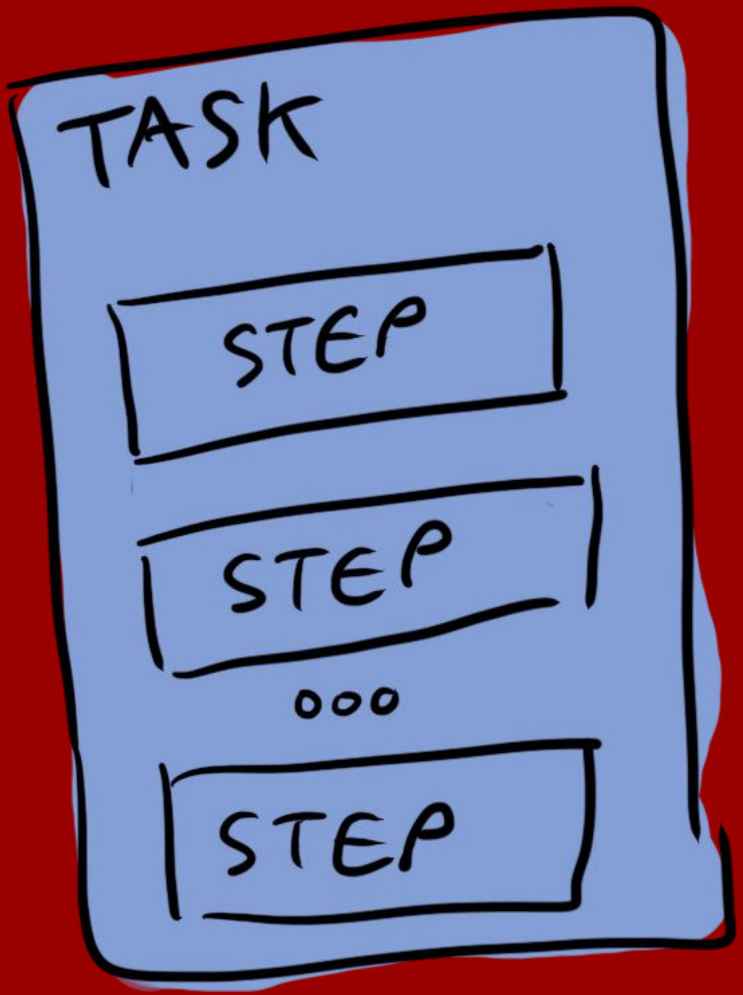
- Specification for CI/CD building blocks
  - Tasks, Pipelines, Pipeline Resources
  - e.g., Pull Request, container image, deployment target
- Maximum pluggability 🧩
- "Kubernetes-style" API, and a Kubernetes implementation
  - Builds on K8s primitives, provides higher-level abstractions



Tekton 101: Tasks

- Tasks are workflow templates
  - Defined once, invoked over and over
  - Parameterizable
- Tasks are comprised of containerized steps
- Steps run sequentially*

- Example: git clone, go vet, golint, go build ./..., go test ./…



Tekton 101: Tasks

```yaml
apiVersion: tekton.dev/v1alpha1
kind: Task
metadata:
  name: golang-build
spec:
  steps:
  - name: build
    image: golang:1.13
    command: ['go', 'build', './...']
```

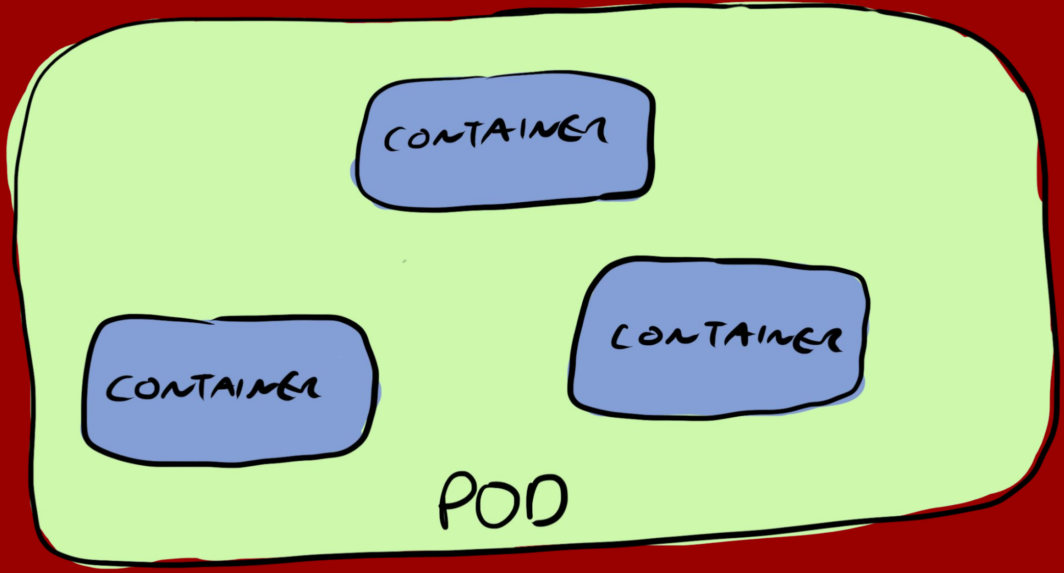
Tekton 101: Tasks

```yaml
apiVersion: tekton.dev/v1alpha1
kind: Task
metadata:
  name: buildpacks
spec:
  steps:
  - name: prepare
    image: alpine
    command: ["/bin/sh"]
    args: …

  - name: detect
    image: builder-image
    command: ["/lifecycle/detector"]
    args: …


# continued... ->
```

```yaml
  - name: analyze
    image: builder-image
    command: ["/lifecycle/analyzer"]
    args: …

  - name: build
    image: builder-image
    command: ["/lifecycle/builder"]
    args: …

  - name: export
    image: builder-image
    command: ["/lifecycle/exporter"]
    args: …

  - name: cache
    image: builder-image
    command: ["/lifecycle/cacher"]
    args: …
```

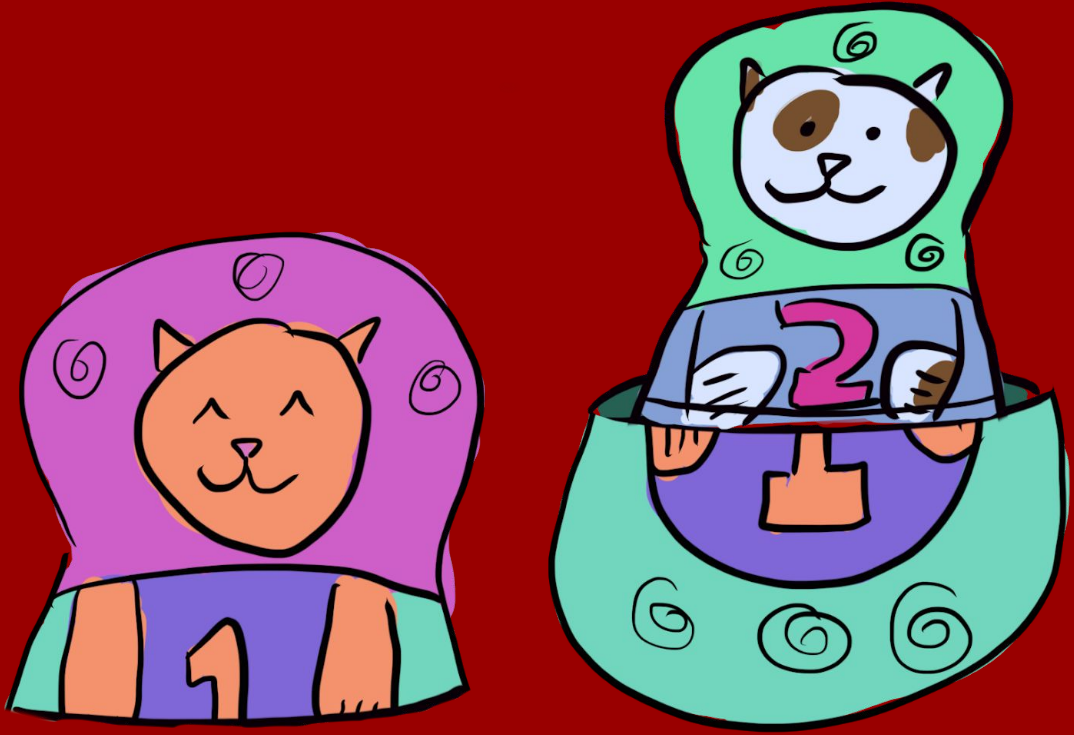
Tekton 101: Tasks

- How does a Task execute on Kubernetes?
  - We use Pods!
- Lots of things in Kubernetes are just wrappers for Pods
  - Deployments, Jobs, DaemonSets, ReplicaSets, etc.
  - ...they just create Pods that are labeled or scheduled differently
- Tekton uses Pods to run containerized steps
  - ...but Pods run containers all-at-once 🤔

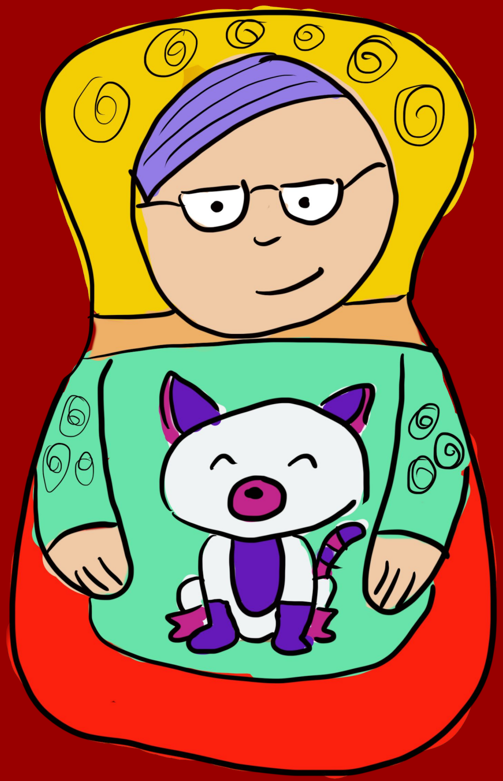


Overview

1. ~~Tekton 101~~
2. But how?
3. Early attempts
4. Magic sauce
5. Demo!
6. Future work



But how?



Requirements

Tasks

1. Run multiple containers
2. Run them in order
3. Let them share data easily

Sharing Data

Many Containers

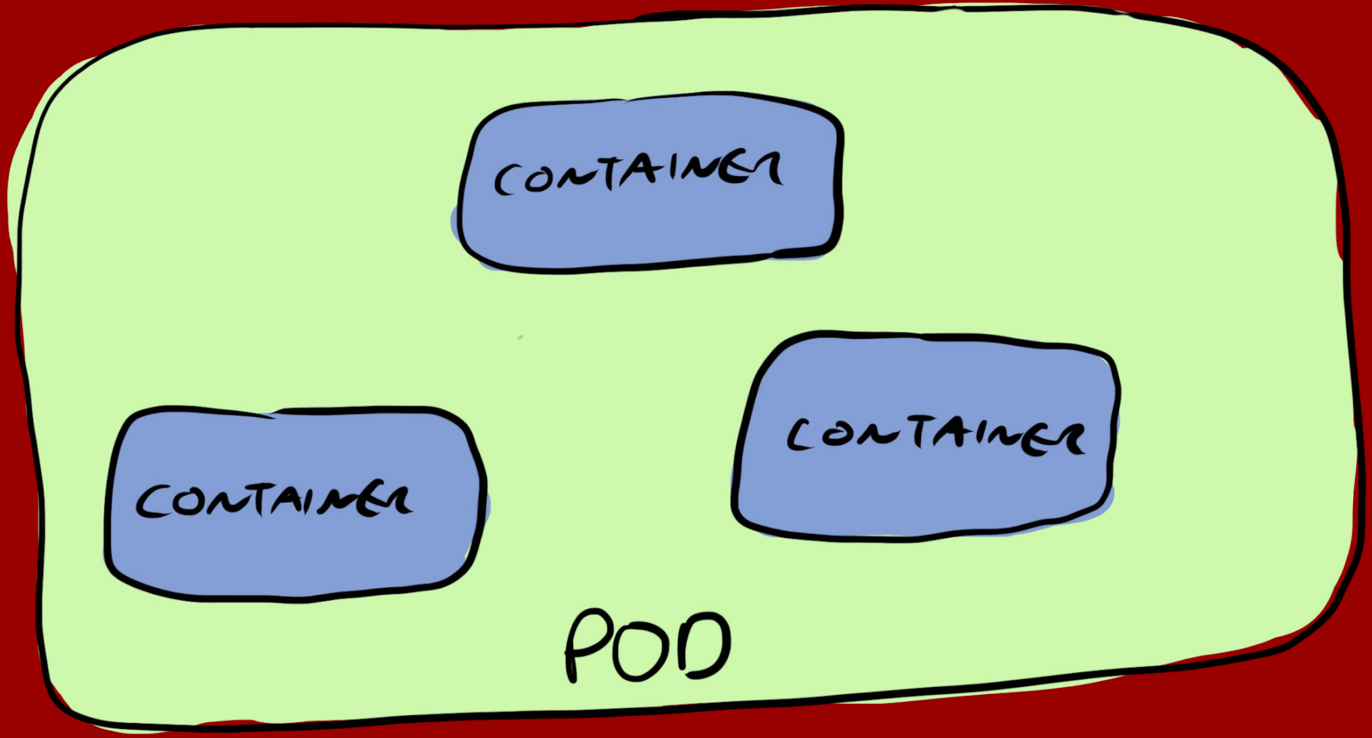
1. Naturally matches to a Pod
2. Node affinity
    a. Same node, not same disk
    b. Need to share via Volumes
3. Custom scheduler
    a. Still need Volumes
4. Jobs?

Kubernetes Jobs

Pros	Cons
"Run-to-completion" Pods	Specified containers still run all-at-once…
Can specify a deadline	
Can specify retry behavior	
Jobs retry Pod creation if it fails	

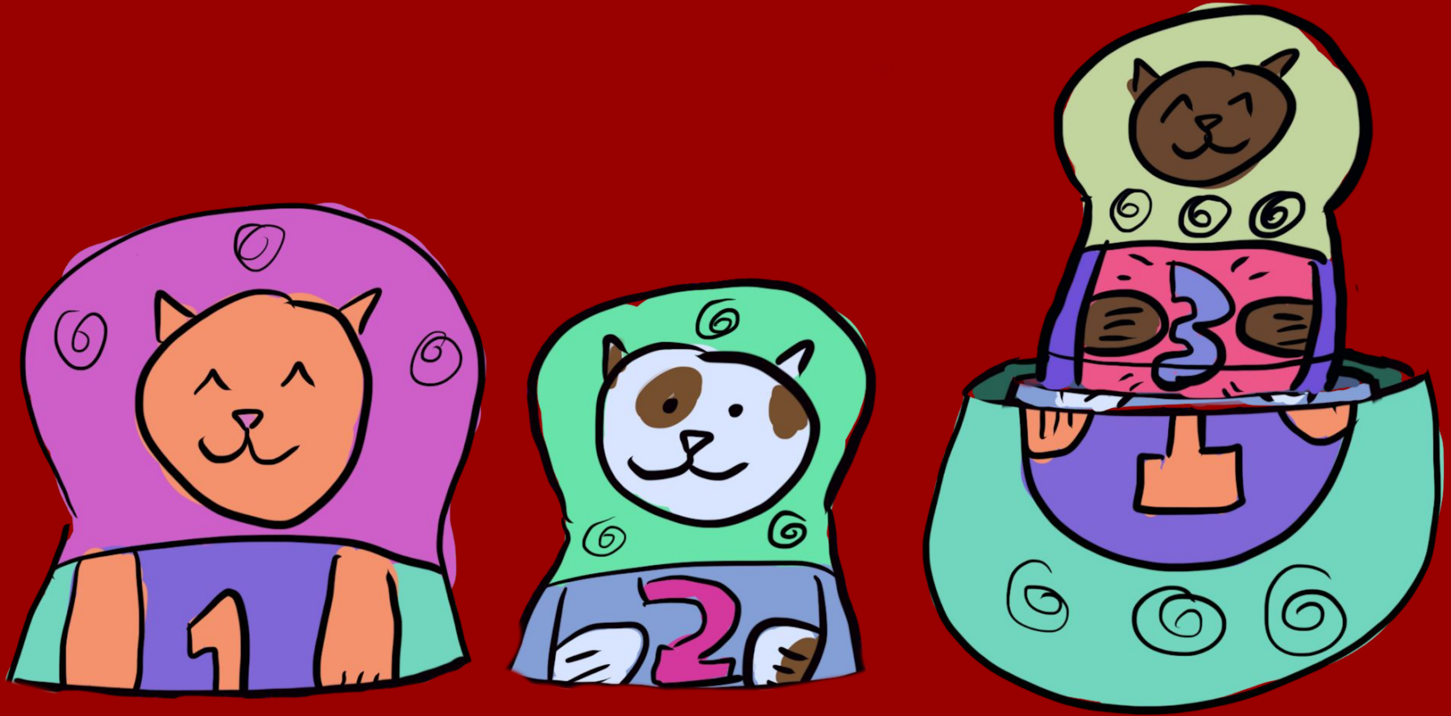
https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/

Pods

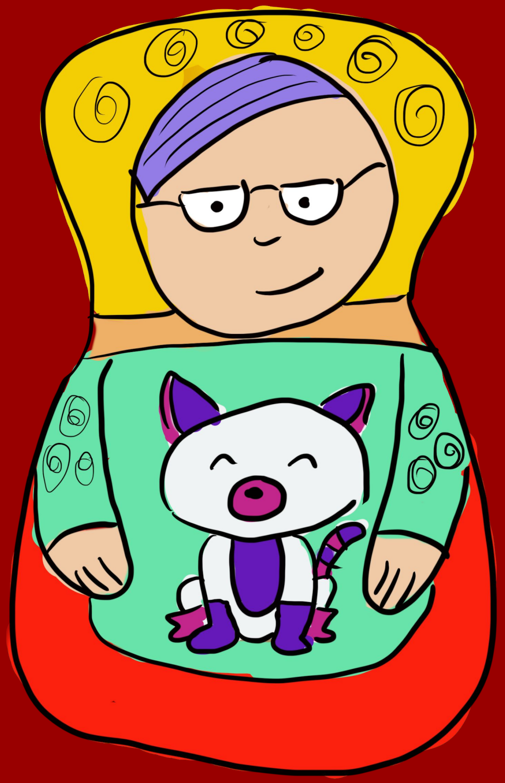


Overview

1. ~~Tekton 101~~
2. ~~But how?~~
3. Early attempts
4. Magic sauce
5. Demo!
6. Future work



initContainers



initContainers

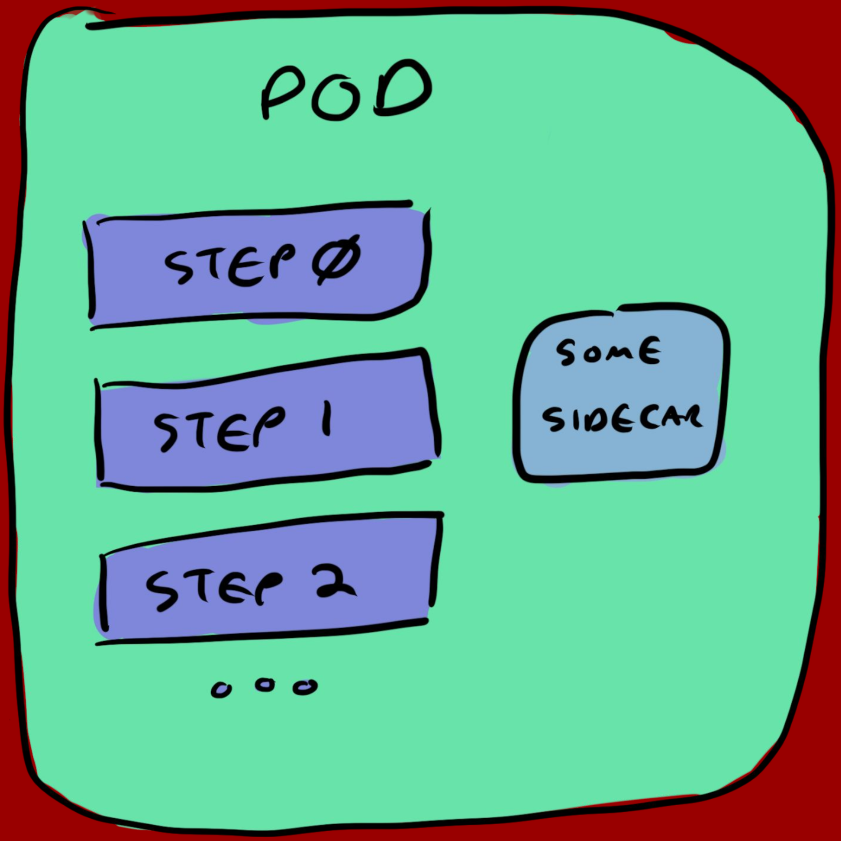
- Pods let you specify initContainers:
    - “Initialize” the pod
    - Run before the Pod's containers, sequentially!
    - A failing initContainer fails the Pod before running containers

- ...but what about Sidecars?

https://kubernetes.io/docs/concepts/workloads/pods/init-containers/

Sidecars

- Can't run anything alongside init containers
- We'd like to be able to run some containers alongside the steps
  - Integration testing
  - HTTP Proxy
  - Docker-in-Docker sidecar
- Sidecar containers should be up before steps start



https://github.com/kubernetes/enhancements/issues/753

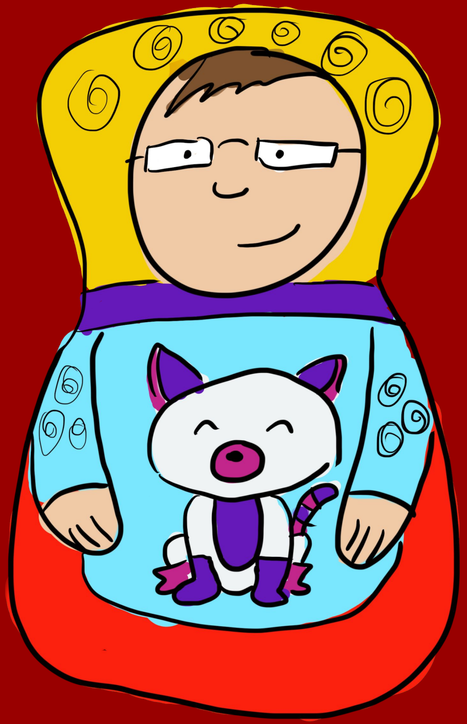
If not `initContainers`, then what?

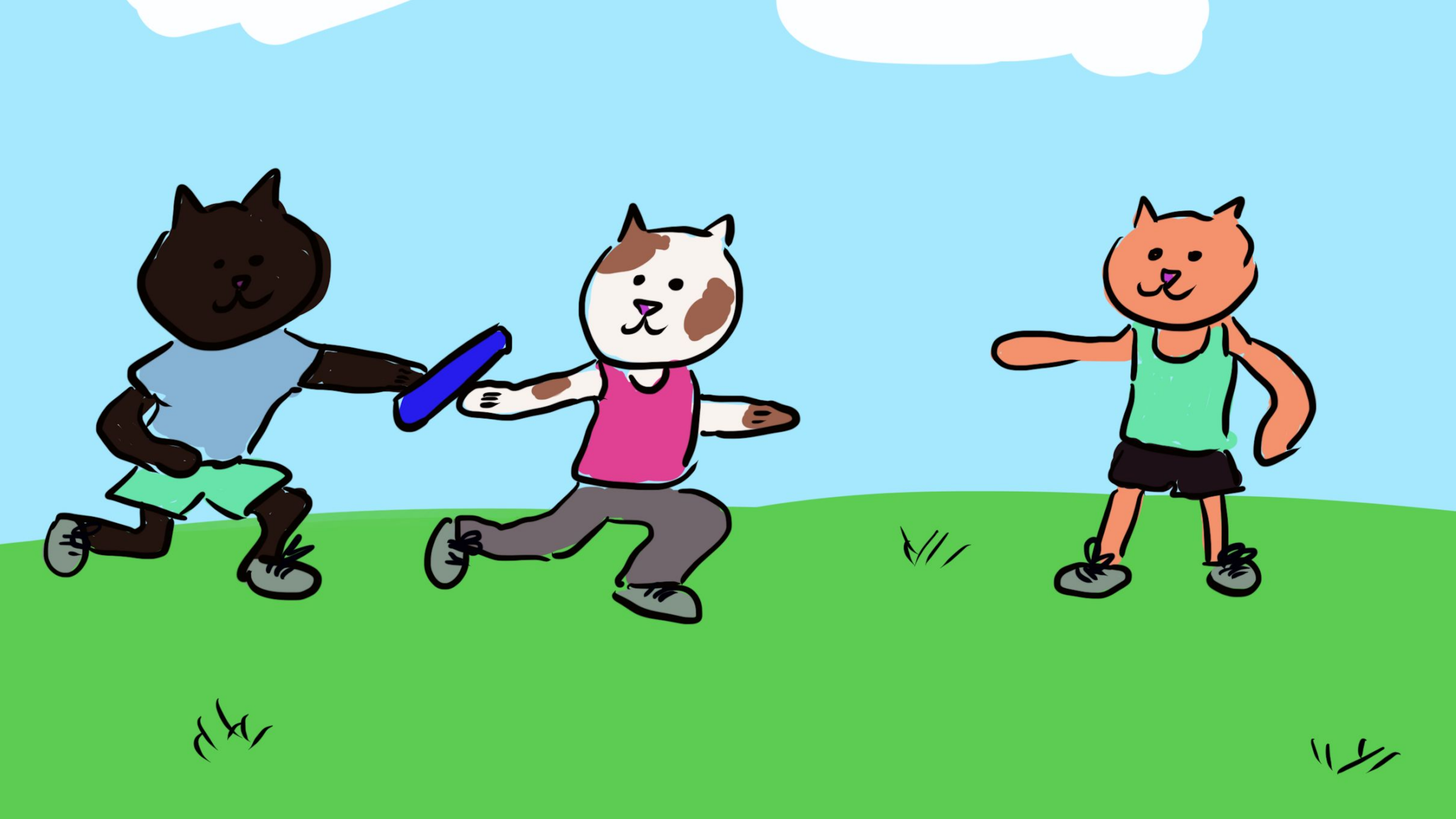
Overview

1. ~~Tekton 101~~
2. ~~But how?~~
3. ~~Early attempts~~
4. Terrible hacks!
5. Demo!
6. Future work

~~Terrible Hacks!~~

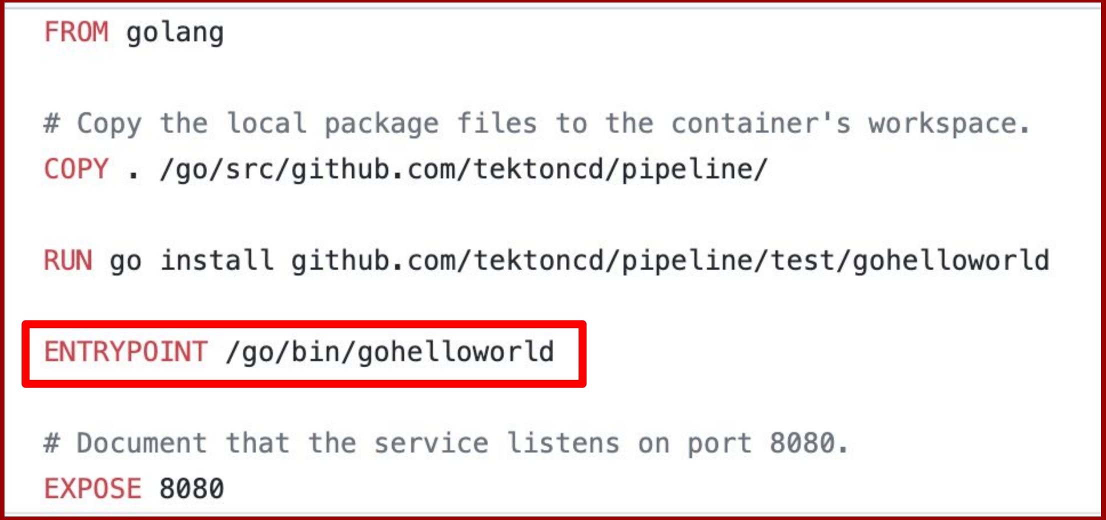
Magic Sauce!





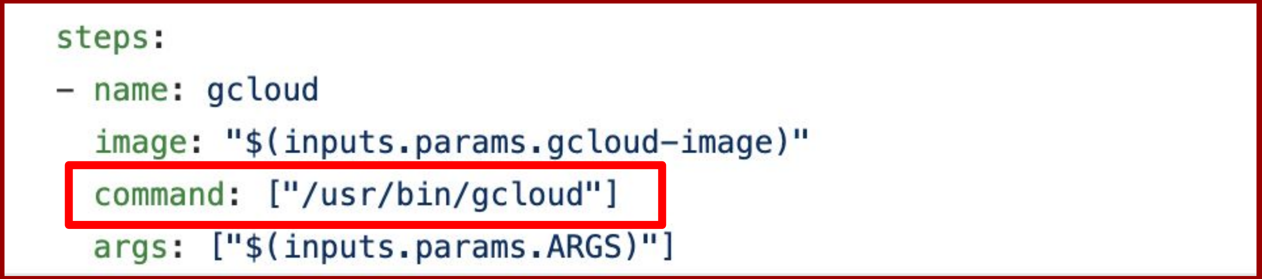
Container Entrypoint

- Container specifies what to run when it starts
- Entrypoint can be specified by the container image (`ENTRYPOINT`)
- ...or explicity when describing the container (`.containers[*].command`)



```
FROM golang

# Copy the local package files to the container's workspace.
COPY . /go/src/github.com/tektoncd/pipeline/

RUN go install github.com/tektoncd/pipeline/test/gohelloworld

ENTRYPOINT /go/bin/gohelloworld

# Document that the service listens on port 8080.
EXPOSE 8080
```

DOCKERFILE

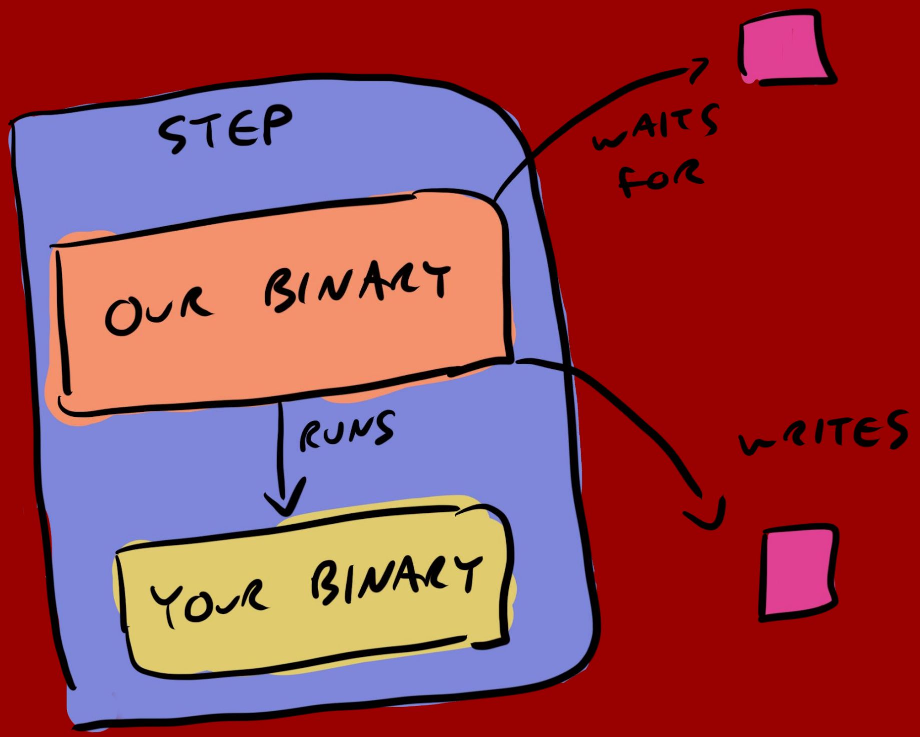


```
steps:
- name: gcloud
  image: "$(inputs.params.gcloud-image)"
  command: ["/usr/bin/gcloud"]
  args: ["$(inputs.params.ARGS)"]
```

Task Step

Entrypoint Overload

1. Override the user's specified entrypoint with one we control
2. Pass original `command+args` to our binary
3. Binary waits for some signal to start, then runs the user's `command+args`
4. When it's done, signals the next step, and so on

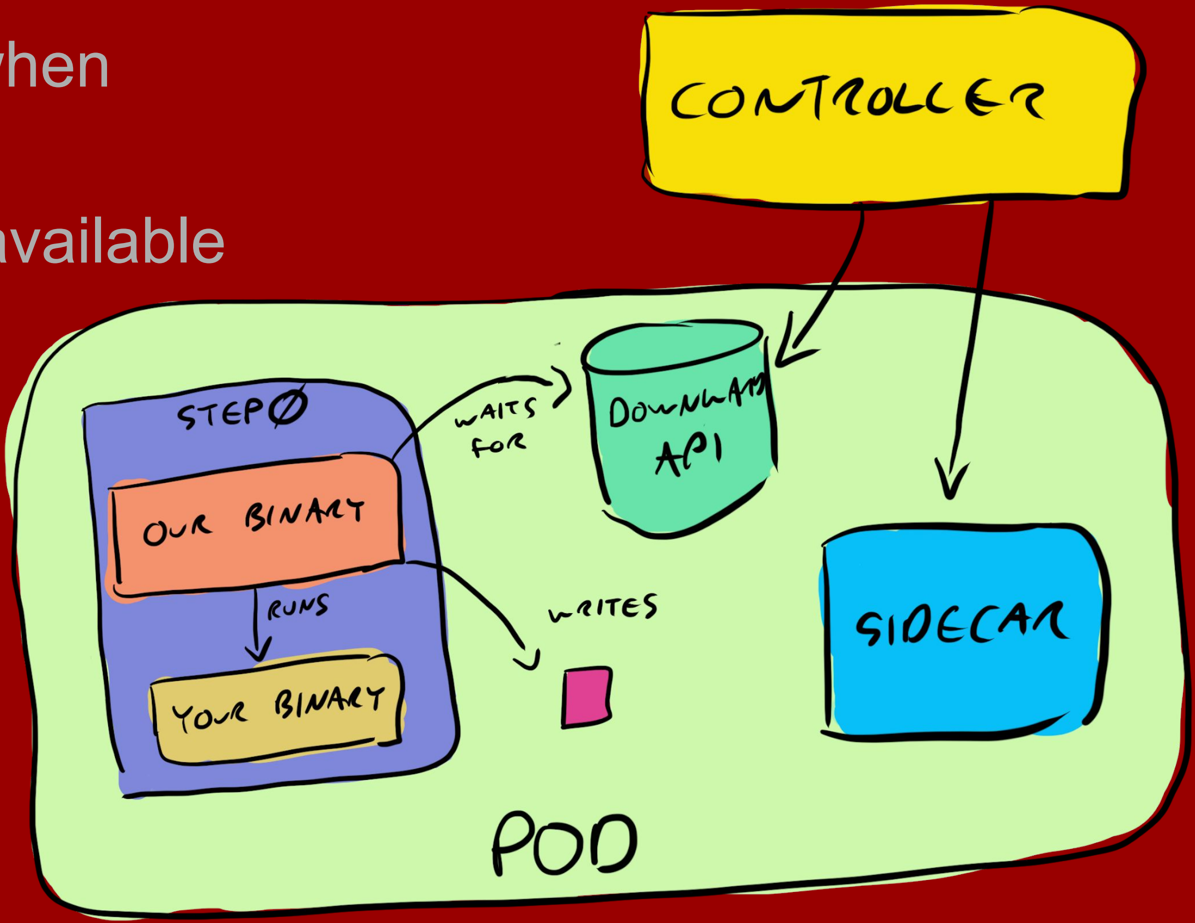


Placing the Entrypoint Binary

- Run an `initContainer` containing the binary
  - Copy it into `/builder/tools` Volume, shared with all step containers
- Override each step's `command` to point to the entrypoint binary
- Pass original `command+args` to our binary
- Entrypoint binary is statically-linked Go binary, no dependencies

Sidecar Support

1. Controller watches the Pod
2. Annotates the Pod with "READY" when sidecars are running
3. Downward API Volume makes file available when Pod is annotated
   - Downward API exposes Pod metadata to containers, as files
4. Signals step 0 to start



https://kubernetes.io/docs/tasks/inject-data-application/downward-api-volume-expose-pod-information/

KEP 753

Sidecars

KEP 753 to officially support Sidecars

- Start Pod containers only after sidecars are up
- Shut down sidecars when main Job containers finish

https://github.com/kubernetes/enhancements/issues/753

Caveats

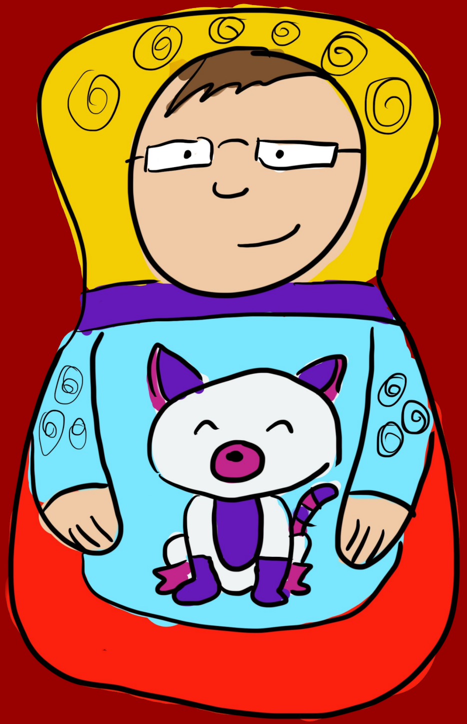
Here be dragons! 🐉

- Containers still *start* all-at-once
  - Don't get clear data when each step starts
  - Can't easily build an image in step 1 and use it in step 3
- Need to lookup the entrypoint if the step doesn't specify `command`
  - Might need credentials to read container image config

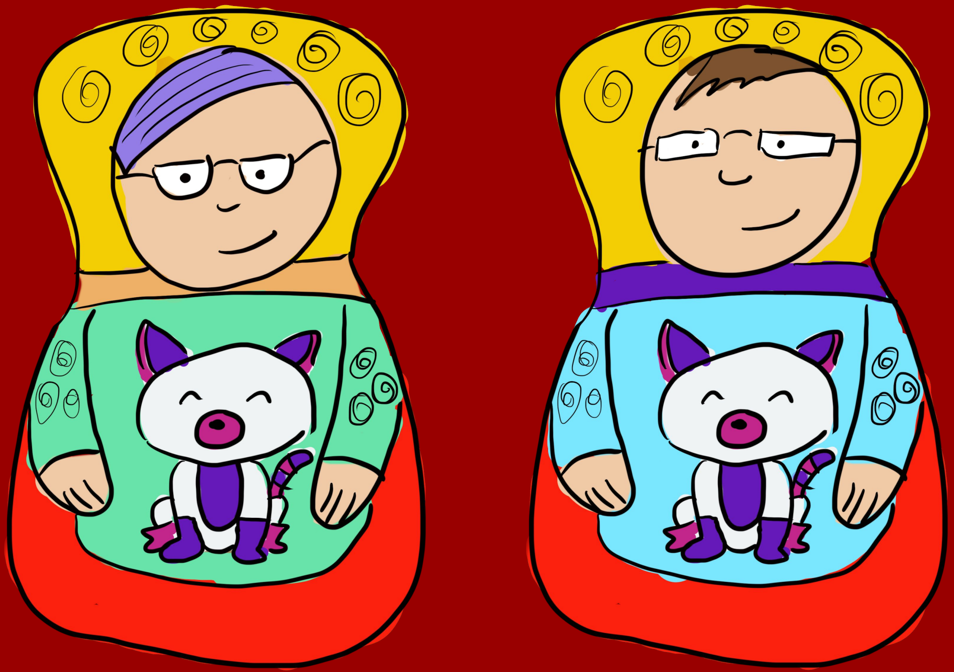
Overview

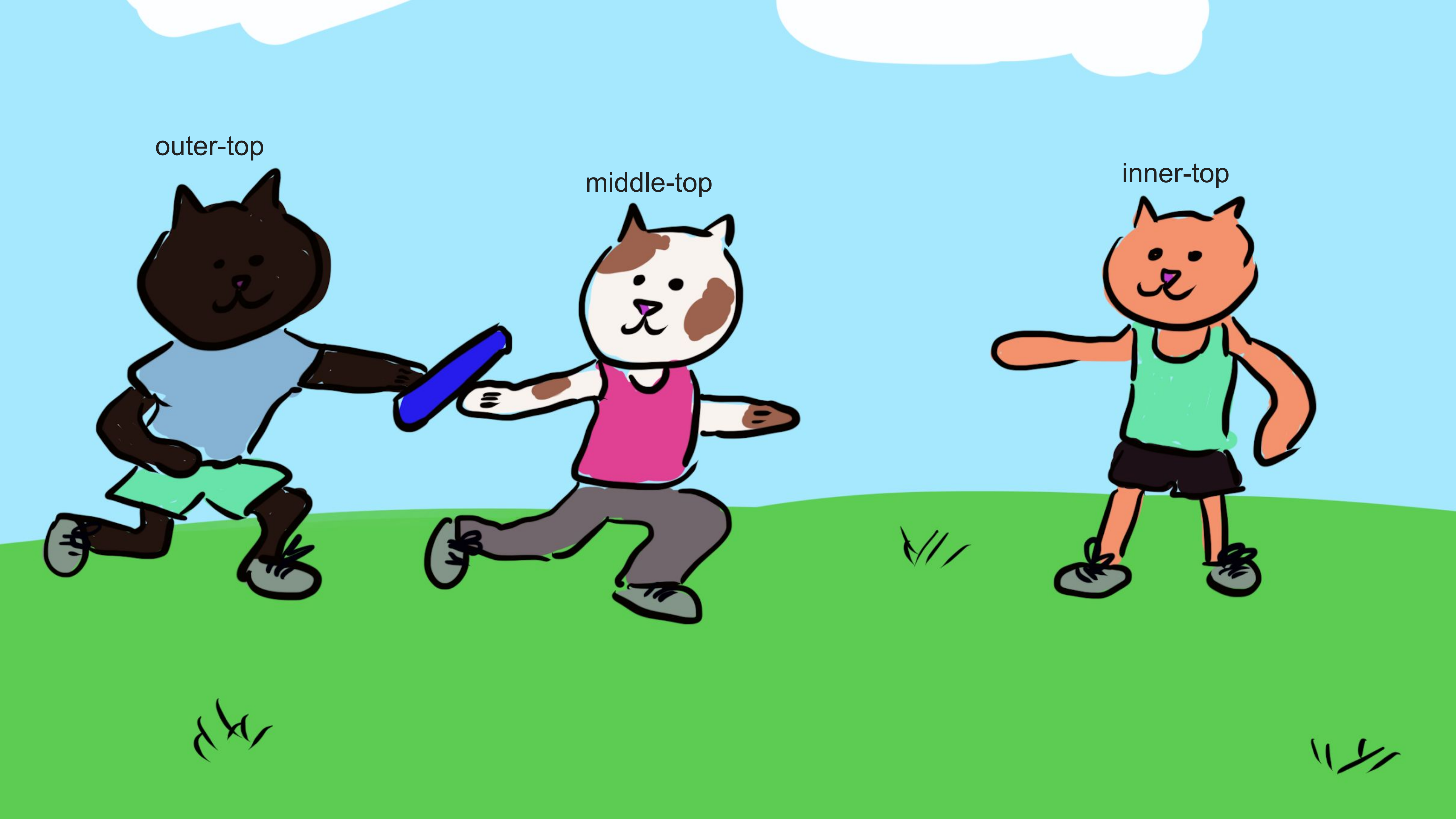
1. ~~Tekton 101~~
2. ~~But how?~~
3. ~~Early attempts~~
4. ~~Magic Sauce~~
5. Demo!
6. Future work

Demo!



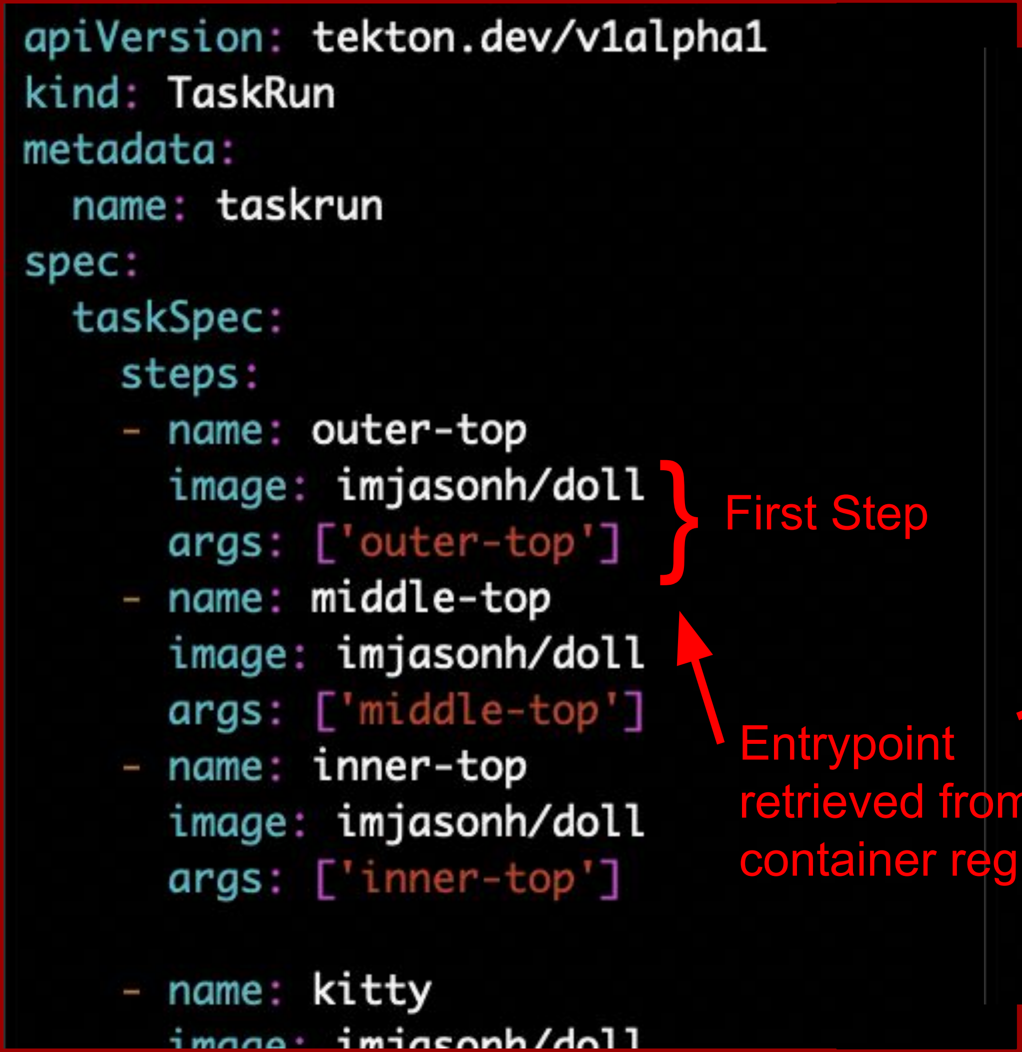
We wrote the terrible hacks so you don't have to!




outer-top
middle-top
inner-top

Step 1

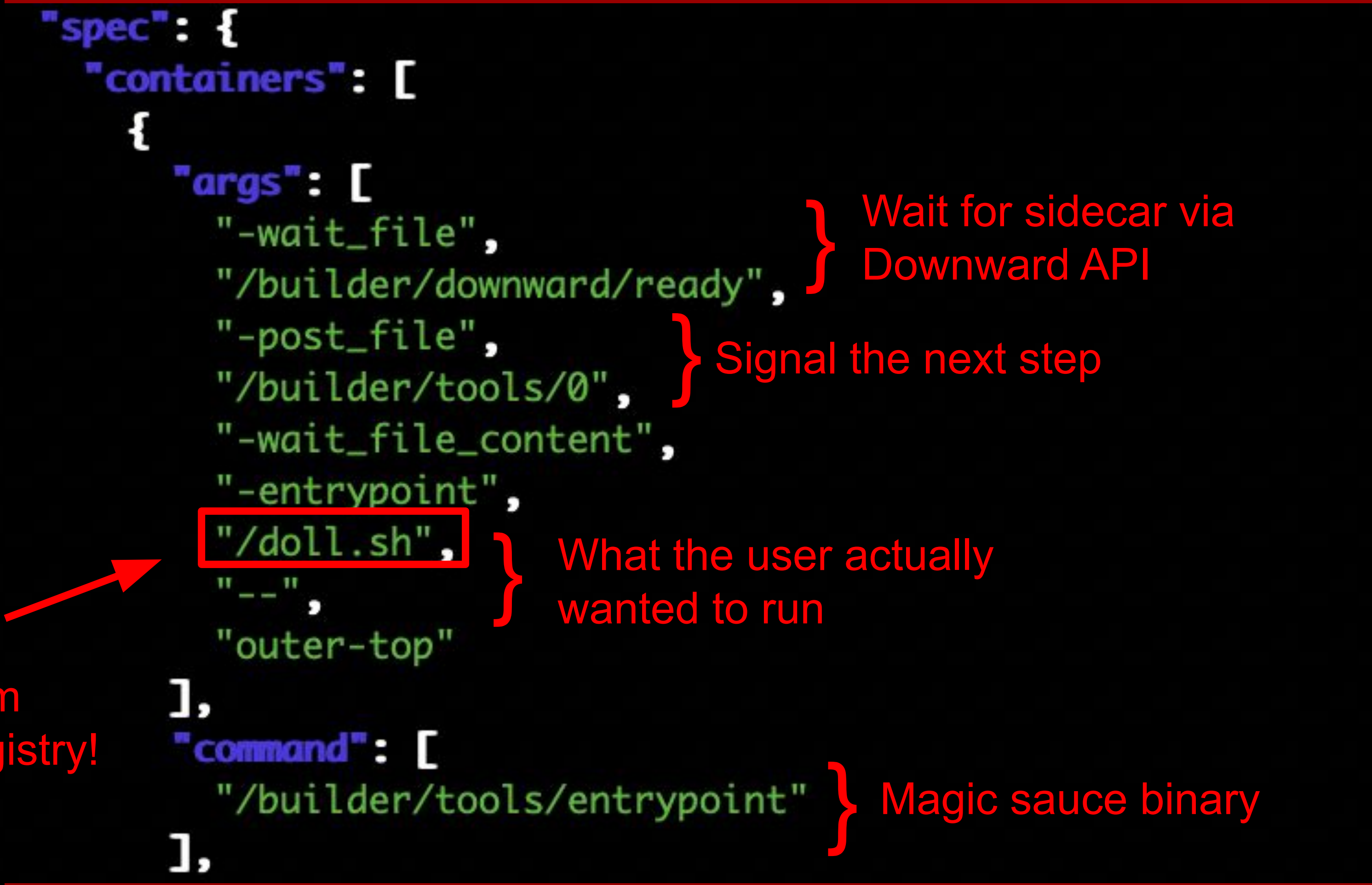
Pod



```
apiVersion: tekton.dev/v1alpha1
kind: TaskRun
metadata:
  name: taskrun
spec:
  taskSpec:
    steps:
    - name: outer-top
      image: imjasonh/doll
      args: ['outer-top']
    - name: middle-top
      image: imjasonh/doll
      args: ['middle-top']
    - name: inner-top
      image: imjasonh/doll
      args: ['inner-top']

    - name: kitty
      image: imjasonh/doll
```

First Step

Entrypoint retrieved from container registry!



```
"spec": {
  "containers": [
    {
      "args": [
        "-wait_file",
        "/builder/downward/ready",
        "-post_file",
        "/builder/tools/0",
        "-wait_file_content",
        "-entrypoint",
        "/doll.sh",
        "--",
        "outer-top"
      ],
      "command": [
        "/builder/tools/entrypoint"
      ],
```

Wait for sidecar via Downward API

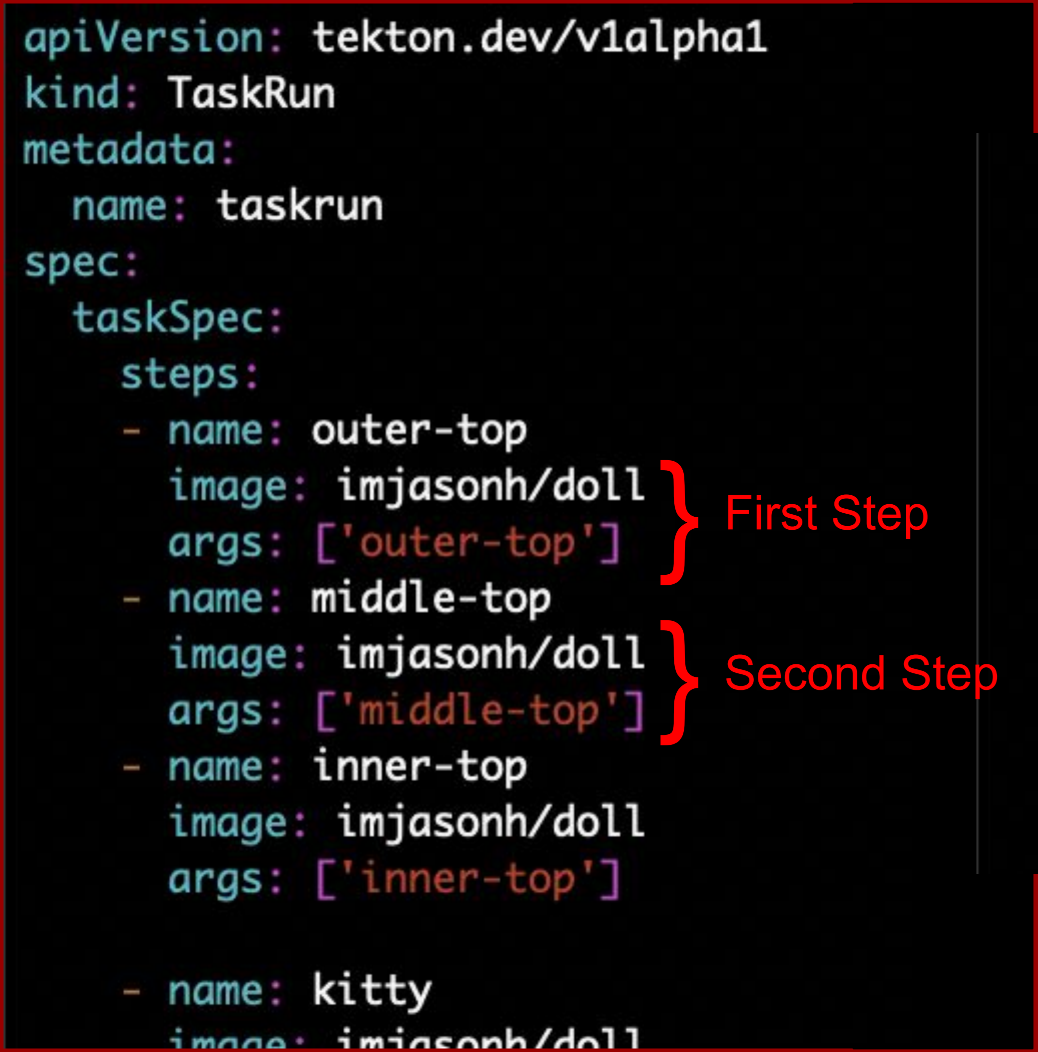
Signal the next step

What the user actually wanted to run

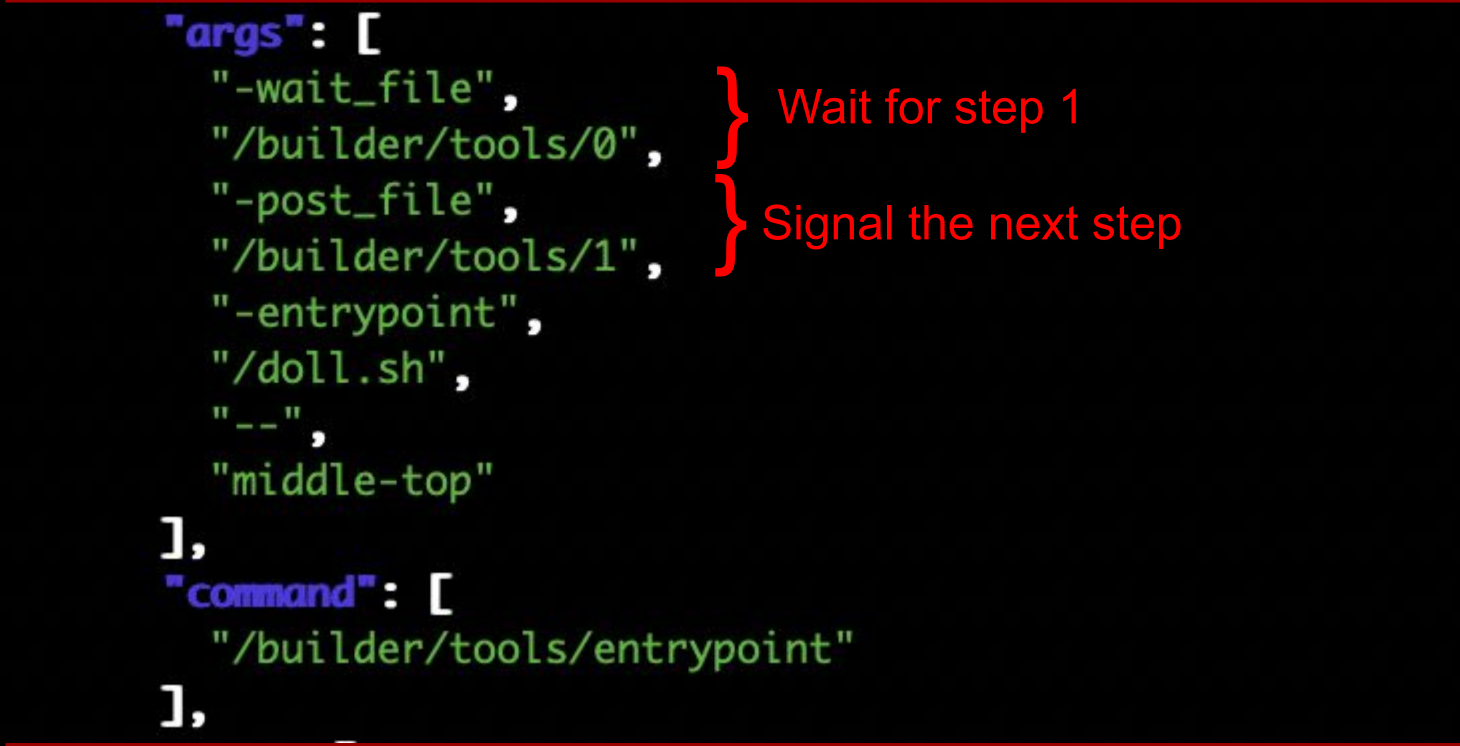
Magic sauce binary

Step 2

Pod



```
apiVersion: tekton.dev/v1alpha1
kind: TaskRun
metadata:
  name: taskrun
spec:
  taskSpec:
    steps:
    - name: outer-top
      image: imjasonh/doll
      args: ['outer-top']
    - name: middle-top
      image: imjasonh/doll
      args: ['middle-top']
    - name: inner-top
      image: imjasonh/doll
      args: ['inner-top']

    - name: kitty
      image: imjasonh/doll
```



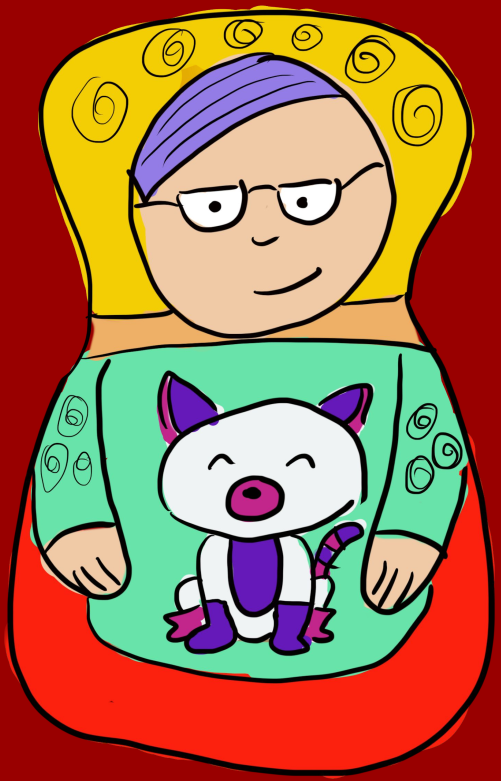
```
"args": [
  "-wait_file",
  "/builder/tools/0",
  "-post_file",
  "/builder/tools/1",
  "-entrypoint",
  "/doll.sh",
  "--",
  "middle-top"
],
"command": [
  "/builder/tools/entrypoint"
],
```

https://gfycat.com/AnchoredInfatuatedGrassspider

Overview

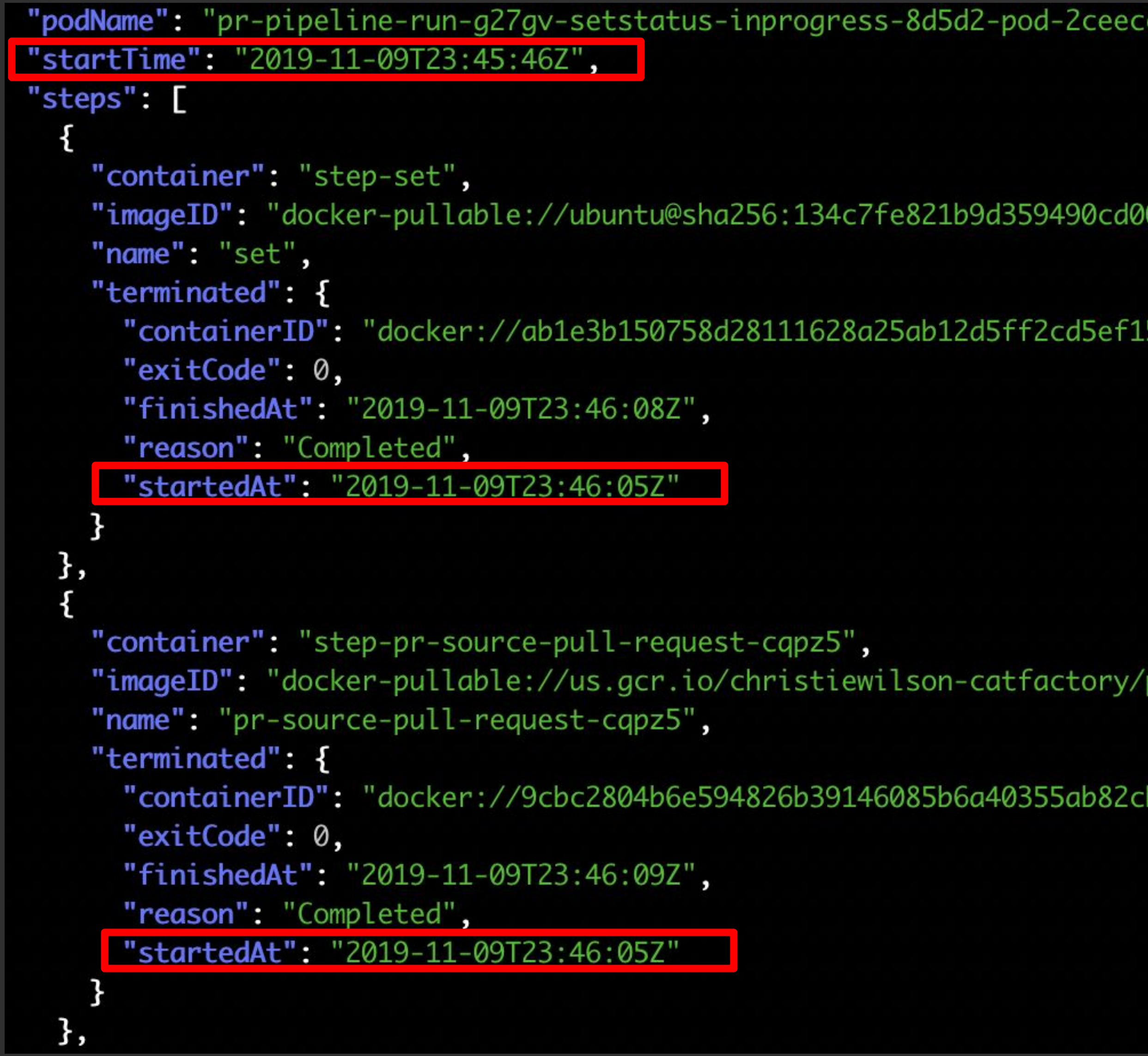
1. ~~Tekton 101~~
2. ~~But how?~~
3. ~~Early attempts~~
4. ~~Magic Sauce~~
5. ~~Demo!~~
6. Future work

Future Work



Future Work

Step Start Time



```
"podName": "pr-pipeline-run-g27gv-setstatus-inprogress-8d5d2-pod-2ceec
"startTime": "2019-11-09T23:45:46Z",
"steps": [
  {
    "container": "step-set",
    "imageID": "docker-pullable://ubuntu@sha256:134c7fe821b9d359490cd0
    "name": "set",
    "terminated": {
      "containerID": "docker://ab1e3b150758d28111628a25ab12d5ff2cd5ef1
      "exitCode": 0,
      "finishedAt": "2019-11-09T23:46:08Z",
      "reason": "Completed",
      "startedAt": "2019-11-09T23:46:05Z"
    }
  },
  {
    "container": "step-pr-source-pull-request-cqpz5",
    "imageID": "docker-pullable://us.gcr.io/christiewilson-catfactory/
    "name": "pr-source-pull-request-cqpz5",
    "terminated": {
      "containerID": "docker://9cbc2804b6e594826b39146085b6a40355ab82c
      "exitCode": 0,
      "finishedAt": "2019-11-09T23:46:09Z",
      "reason": "Completed",
      "startedAt": "2019-11-09T23:46:05Z"
    }
  },
```

Future Work

Super Sidecar

- Signal entrypoint binaries to start, from within the Pod
- "Self-driving" Pod, doesn't require input from the Controller

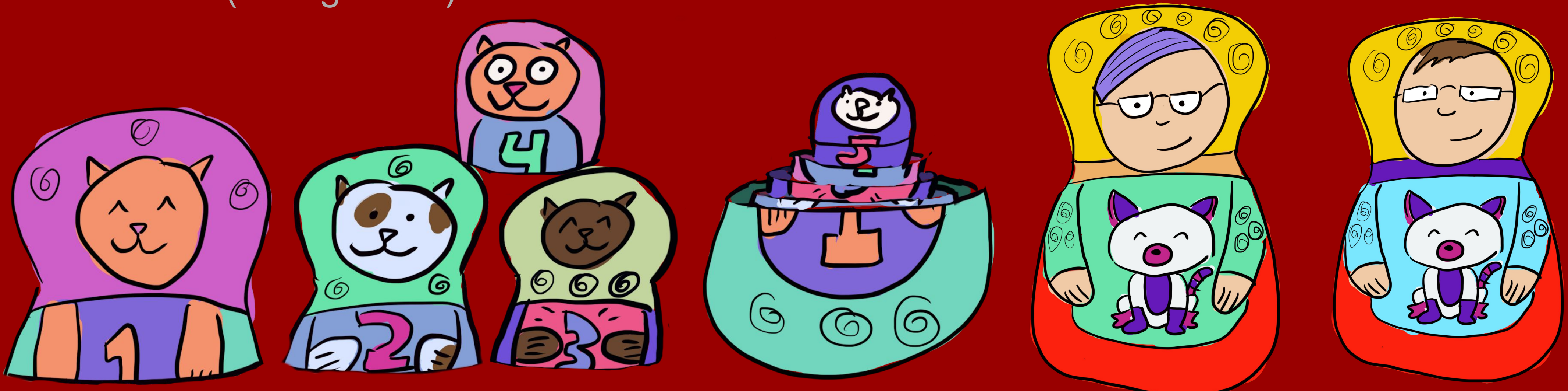
Future Work

Debug mode

- Tell entrypoint binary to keep running
- `kubectl exec` to look around and debug the step

Thanks! ❤️

- Shoutouts to some of many folks who contributed to this:
  - Matt Moore (init containers)
  - James Strachan (“just wrap the binary!”)
  - All the Prow folks (same approach)
  - Aaron Prindle (move away from init containers)
  - Alex DiCarlo (sidecar support)
  - Scott Seaward (logs)
  - Dan Lorenc (debug mode)



Closing

- See the code: github.com/tektoncd/pipeline
- Example Tasks: github.com/tektoncd/catalog
- Become a Friend: github.com/tektoncd/friends
- Ask questions on the Slack: bit.ly/2QrSksh