The Control Loop As An App Development Framework

Nick Santos

KubeCon + CloudNativeCon North America 2021

Motivation

How Kubernetes design choices influenced our tool!

And how it should influence yours!

We'll discuss...

- Control Loops
- Kubernetes
- Case Studies
- Lessons Learned

What? Why?

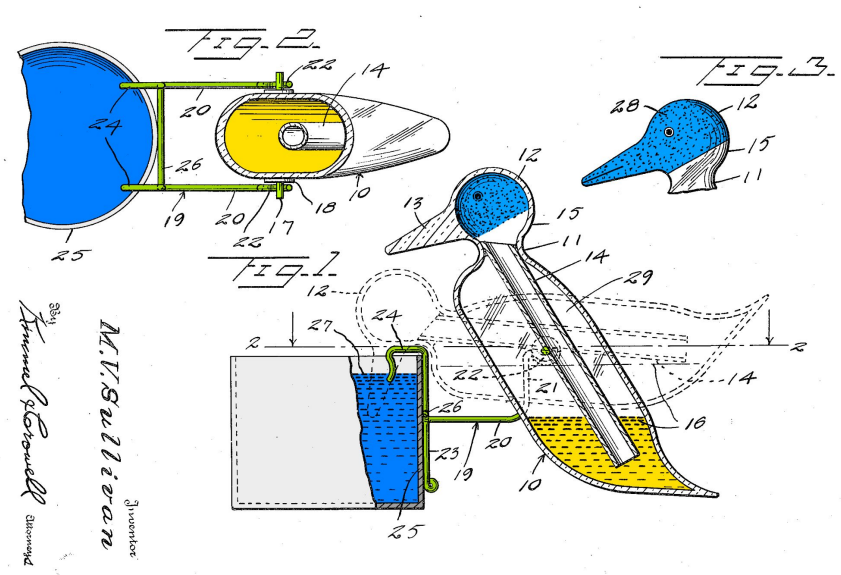
Control Loops

What is a control loop?

Any system that needs to react to changes at runtime to avoid obstacles.

- Thermostat
- ABS brakes

Don’t you wish you had a control loop to water your plants?



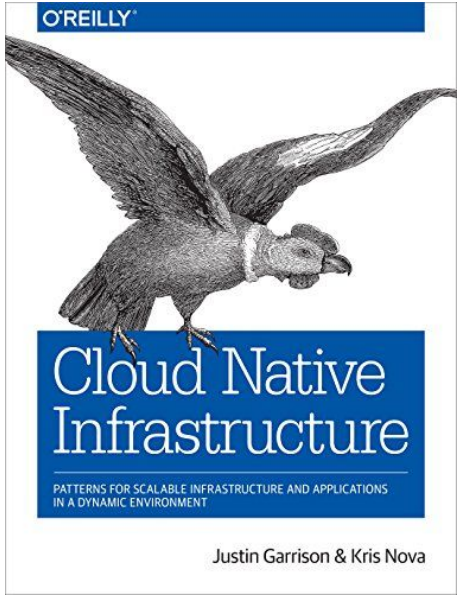
US Patent US2402463, a bird controlled by ambient temperature differentials, Miles V Sullivan, 1945

Software control loops...

- **Zoom**: connection quality throttling
- **entr, nodemon**: filewatching
- **IDEs**: lint/check scheduling
- **React**: hot-reloading

Kubernetes!

- Kubernetes!
  - Kubernetes!
- **Declarative state management!**
  - See: Cloud-Native Infrastructure, by Kris Nóva & Justin Garrison



Why is this relevant now?

- The philosophy of...
  - Unix: tons of **files**
  - Distributed systems: tons of **servers**
- Servers have **runtime inputs** as well as "static" inputs
- Control loops are great at reacting to runtime conditions
- Kubernetes is both a driver of this trend, and reacting to this trend
- Anything that works with or like Kubernetes needs to be a control loop too

Adding Some Nitty To This Gritty

Kubernetes

The Kubernetes API Doesn't Have to Be Scary!

kubectl / CLI

HTTP APIs

YAML

Pods

SPDY

Protobuf/GRPC

Client libs

DNS

Kubernetes - Also A Control Loop Library!

[“Kubernetes is so Simple You Can Explore it with Curl”]

Kubernetes has a simple, genius idea for how to configure control loops.

Start with simple Go structs.

Everything builds around the Go structs.

"Curling,--a Scottish Game, at Central Park" by John George Brown. Via Wikipedia.

Kubernetes is so Simple You Can Explore it with Curl

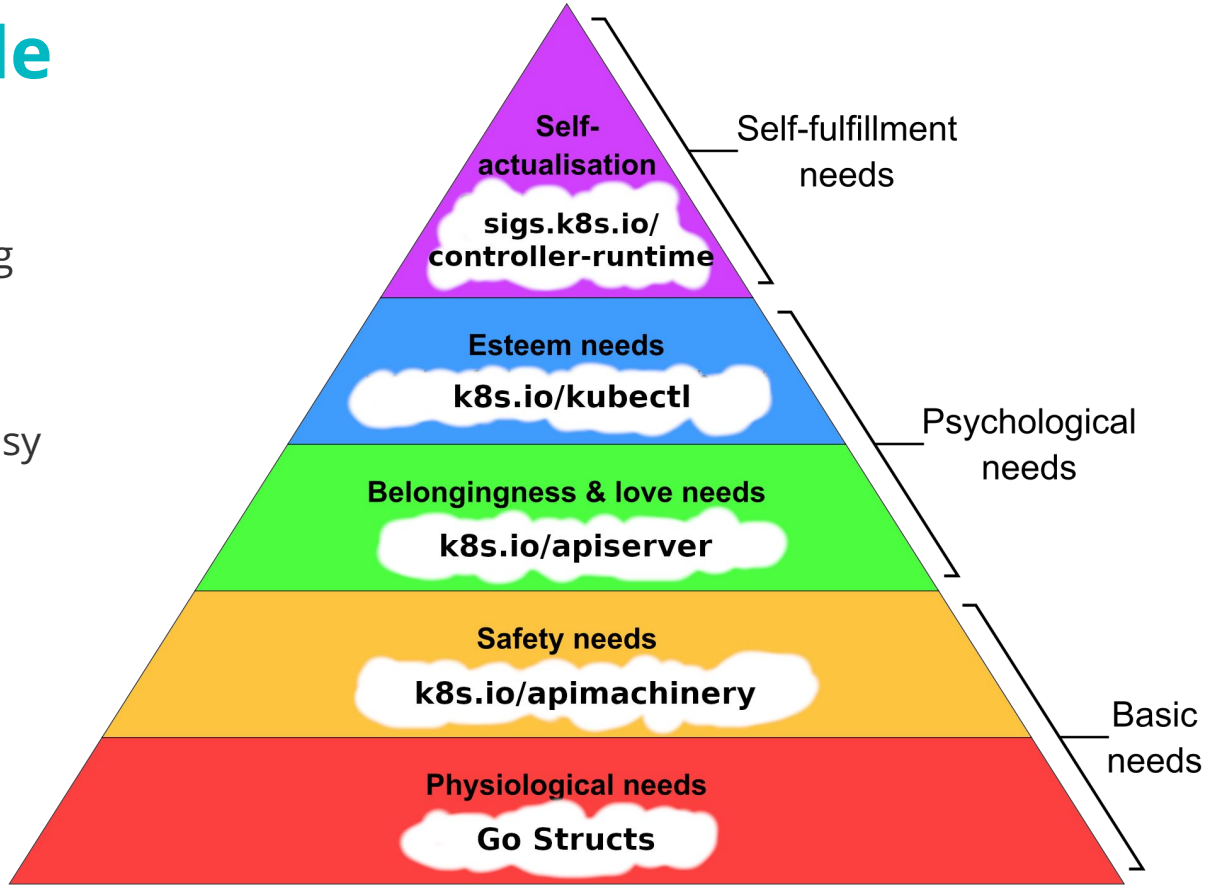
Published on 18 March 2021
Nick Santos

Kubernetes - Code

The Kubernetes community has done a lot of good work breaking it up into a stack of repos!

Here are some repos that are easy to re-use.

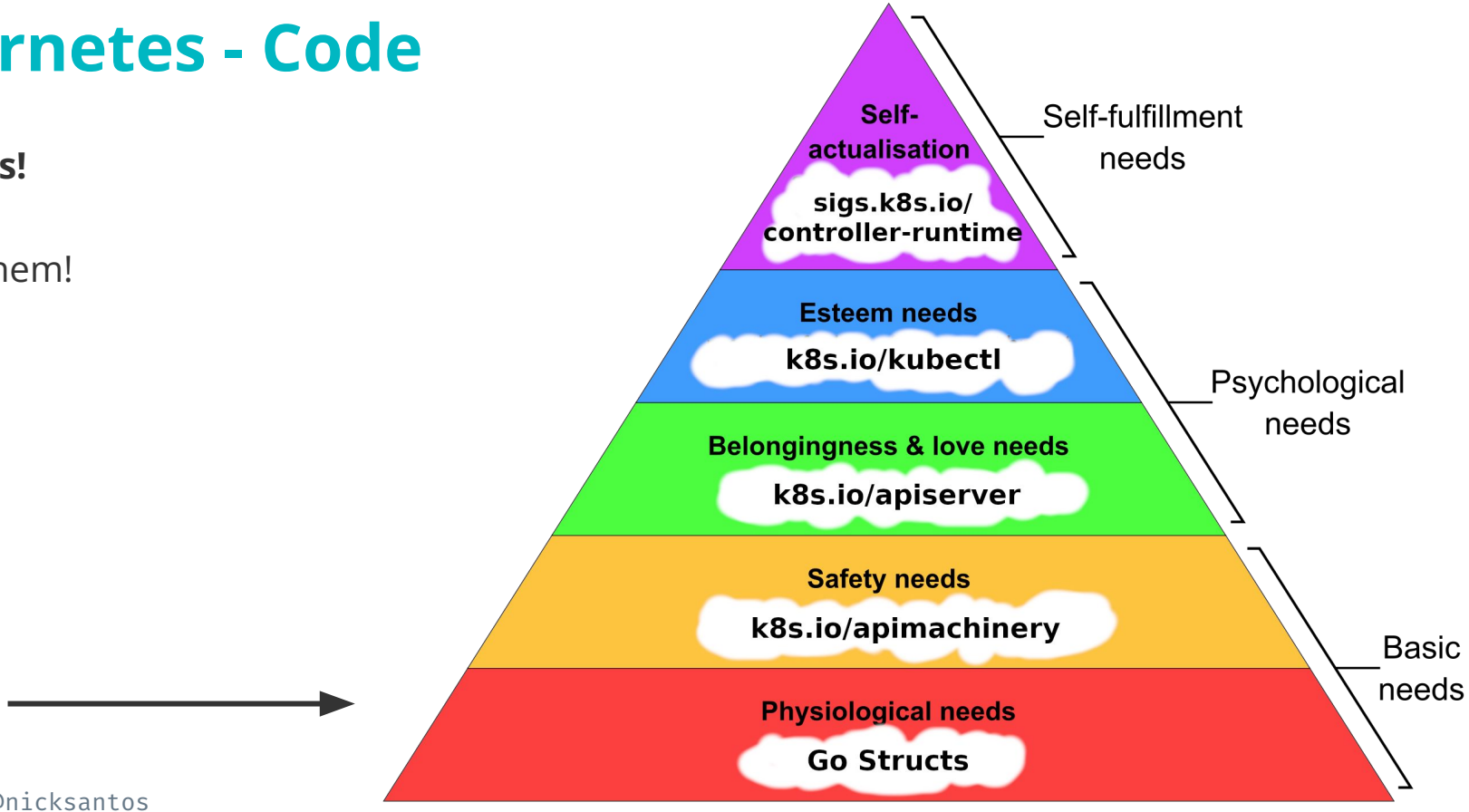
Let's go through them.



Kubernetes - Code

Go Structs!

Go read them!



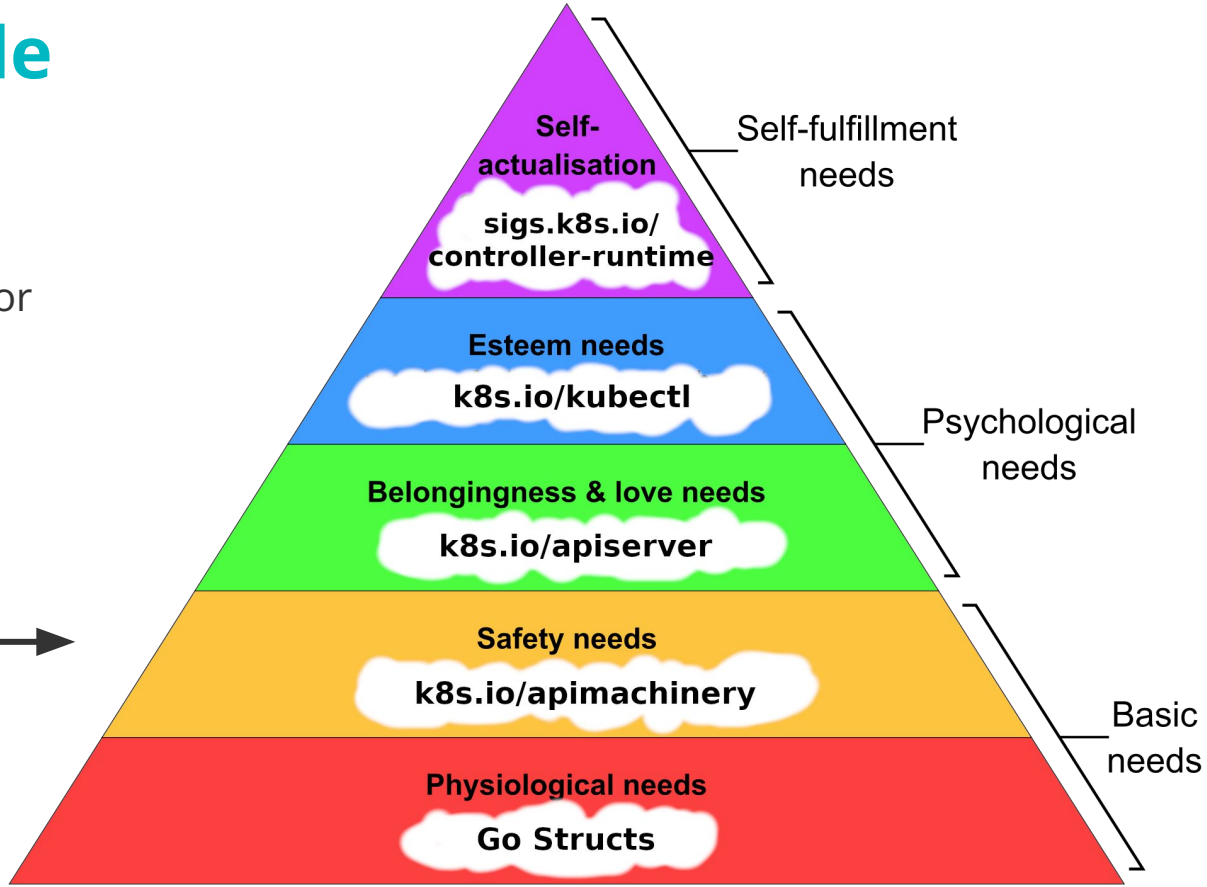
```go
type Service struct {
	metav1.TypeMeta `json:",inline"`
	// Standard object's metadata.
	// +optional
	metav1.ObjectMeta `json:"metadata,omitempty" protobuf:"bytes,1,opt,name=metadata"`

	// Spec defines the behavior of a service.
	//
	// +optional
	Spec ServiceSpec `json:"spec,omitempty" protobuf:"bytes,2,opt,name=spec"`

	// Most recently observed status of the service.
	// Populated by the system.
	// Read-only.
	// +optional
	Status ServiceStatus `json:"status,omitempty" protobuf:"bytes,3,opt,name=status"`
}
```

Kubernetes - Code

k8s.io/apimachinery

Defines the common structure for referencing the Go structs.

(Name, Namespace, Labels, etc)

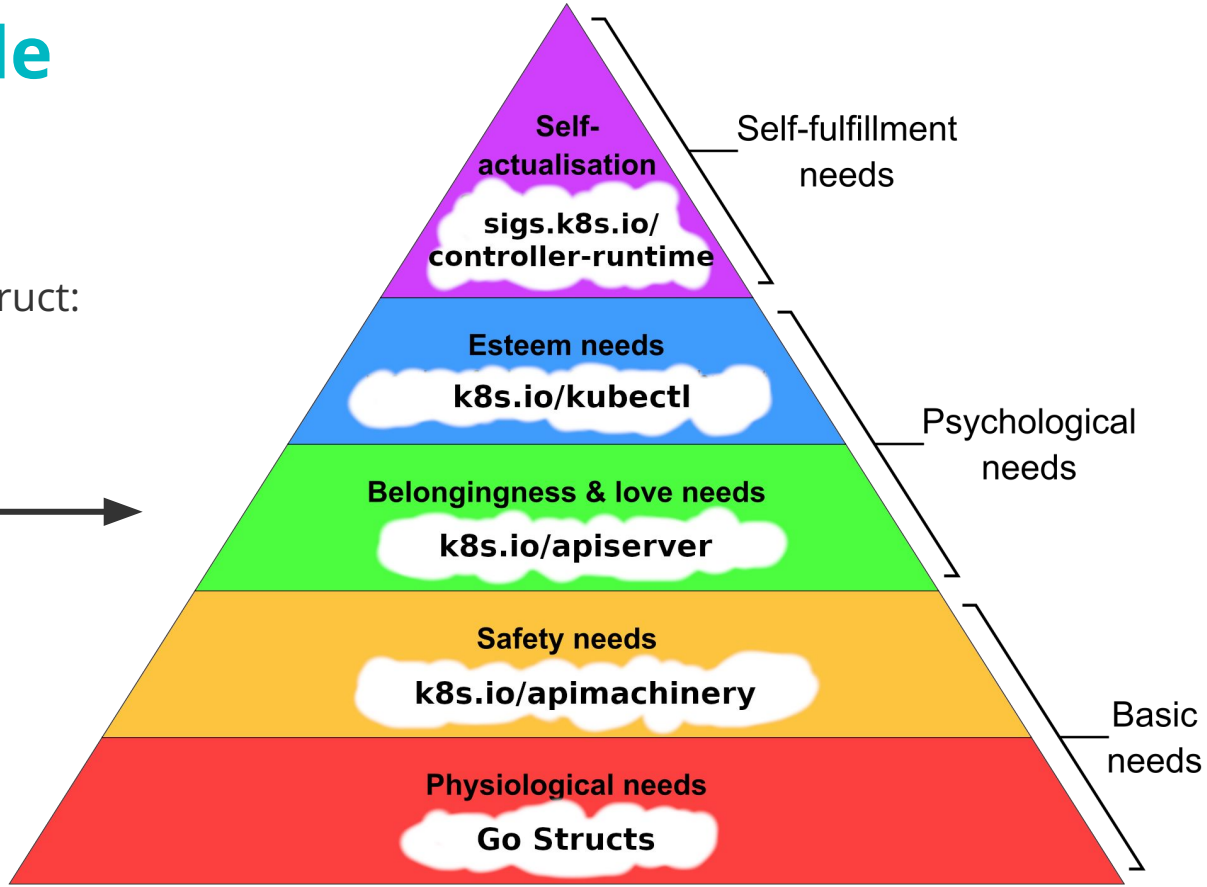


Kubernetes - Code

k8s.io/apiserver

Generates HTTP APIs for each struct:

Create
Update
Get
Delete
Patch
List
Watch

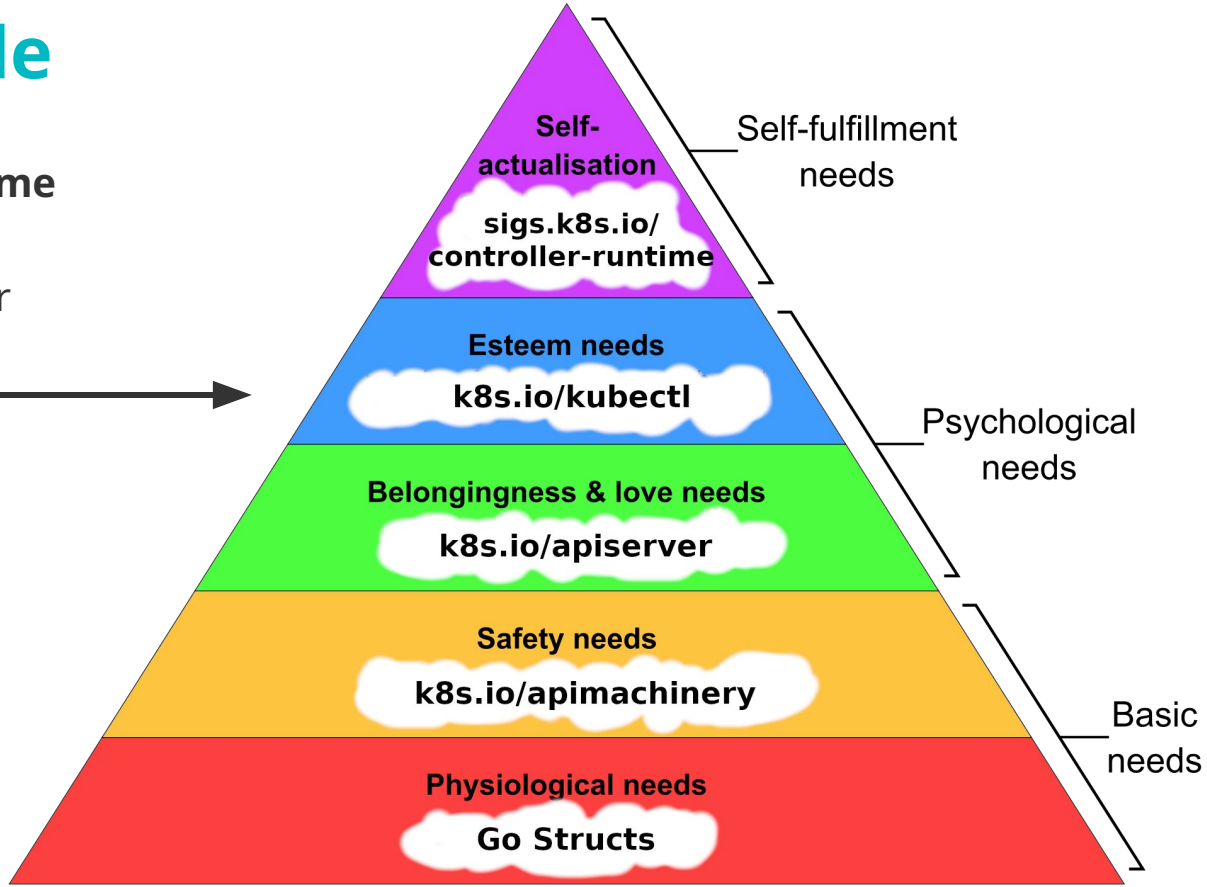


Kubernetes - Code

k8s.io/kubectl, k8s.io/cli-runtime

Command-line tools platform for working with each Go struct:

kubectl get
kubectl describe
kubectl edit
kubectl apply
kubectl patch



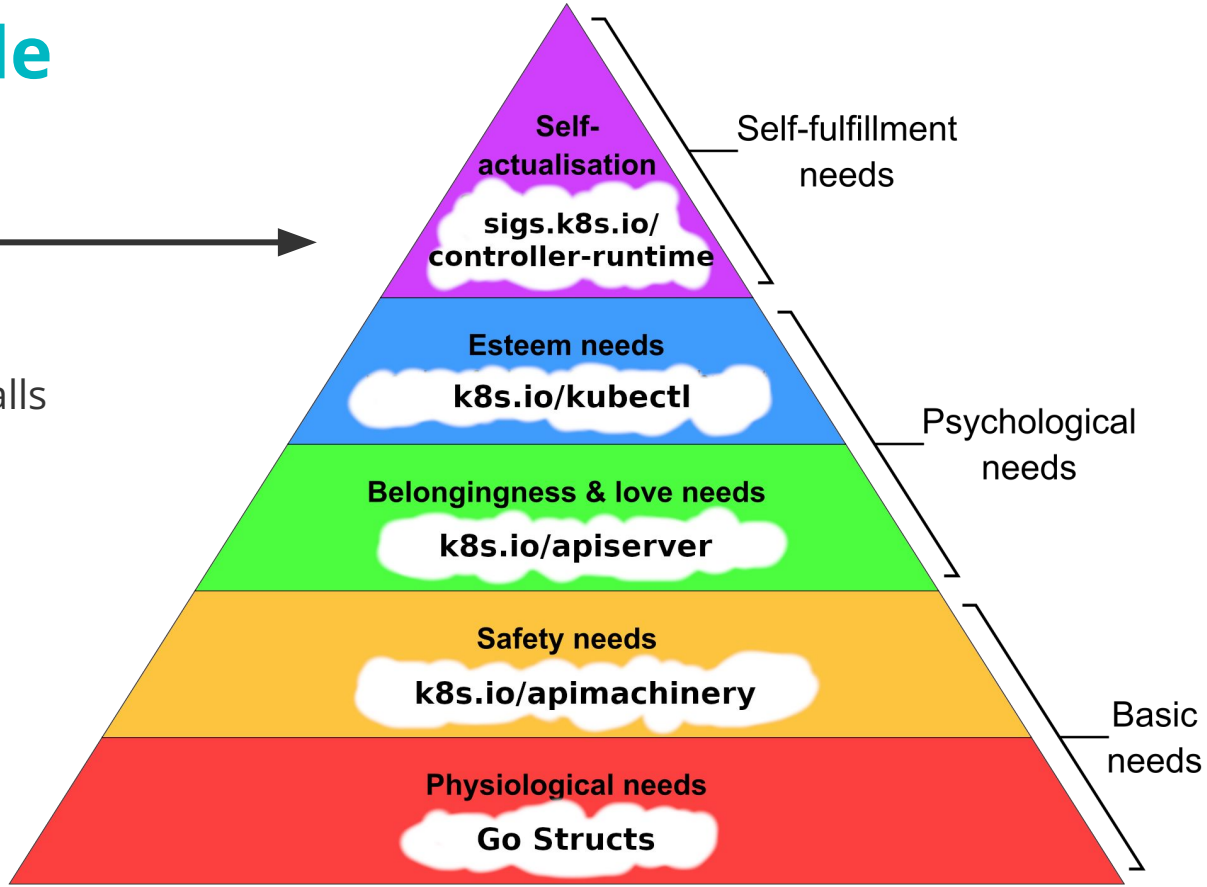
Kubernetes - Code

sigs.k8s.io/controller-runtime

A framework for turning HTTP calls into a fully realized control loop.

Listens to changes with Watch.

Determines when you need to respond to a change.



Kubernetes - Object Access Model

Classic REST uses CRUD:

Create, Read, Update and Delete

All this infra builds around CRUD++:

CRUD plus **List** and **Watch**

List and Watch are building blocks that let you build tools that react to changes.

Feat. Tilt!

Case Studies

What Kind of Tools Care About Control Loops?

What Tilt does:

- Watches your source code!
- Auto-deploys and restarts services when the source code changes.
- Raises alerts when services are unhealthy.

Building Your Own Control Loop

When we started, we built our own control loop!

An artisanal, hand-rolled control loop with big gaps.

Hard to support:

- Swap out implementations for common tasks
- CLIs to diagnose misconfiguration
- Display status as part of alternative dashboards

Kubernetes Already Solved These Problems

We didn't need to build:

- An API Server
- A CLI framework
- A New Control Loop Runtime

Because it was way easier to use what the Kubernetes community already built!

Case #1: Filewatching

- "File watcher as a daemon" has lots of prior art: entr, nodemon, watchman
- This problem space is a good match for the **reconciler pattern**:
  - **Declarative data model** i.e. watch all files matching *.go!
  - Needs to continuously **react to events at runtime**
- Many possible implementations
  - Operating systems seem to invent a new filewatch API every few years

    i.e. inotify, kqueue, fsevents, ReadDirectoryChangesW

```go
// FileWatchSpec defines the desired state of FileWatch
type FileWatchSpec struct {

    // WatchedPaths are paths of directories or files to watch for changes to.
    // It cannot be empty.
    WatchedPaths []string `json:"watchedPaths"`

    // Ignores are optional rules to filter out a subset of changes
    // matched by WatchedPaths.
    Ignores []IgnoreDef `json:"ignores,omitempty"`

}
```

```go
// FileWatchStatus defines the observed state of FileWatch
type FileWatchStatus struct {

	// FileEvents summarizes batches of file changes (create, modify, or delete)
	// that have been seen in ascending chronological order.
	// Only the most recent 20 events are included.
	FileEvents []FileEvent `json:"fileEvents,omitempty"`

	// Error is set if there is a problem with the filesystem watch.
	// If non-empty, consumers should assume that no filesystem events
	// will be seen and that the file watcher is in a failed state.
	Error string `json:"error,omitempty"`
}
```

Case #1: Filewatching

Before:

Your file watch doesn't watch the files you expect. Let's ask 20 questions.

- Are you hitting out of inotify nodes?
- Is there a symlink?
- Is there a typo in the file name?
- Did you get the pattern syntax right?
- Is the path relative or absolute?
- Is the working directory of your pattern what you expect?
- Is the filesystem case-sensitive or case-insensitive?
- Is there an ignore pattern?
- Can you check the .dockerignore?

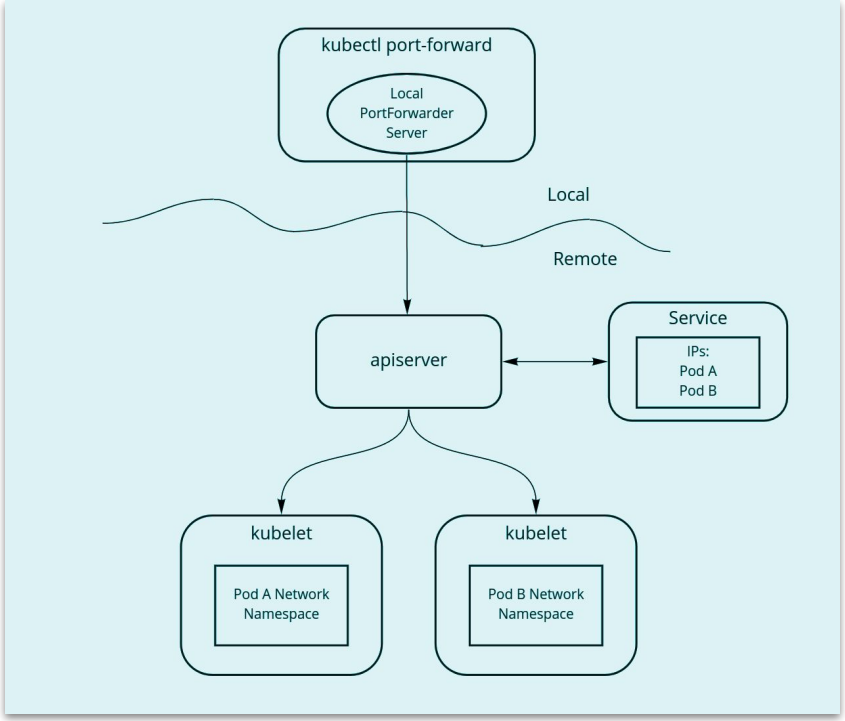
Case #1: Filewatching

After:

- Interactive experiments!
  - Change the spec
  - Touch a file
  - Watch the status
- k8s.io/kubectl generates CLIs (get/explain/describe)
- The control loop pattern enables fast feedback!

Case #2: kubectl port-forward

- `kubectl port-forward service/frontend`
- Resolves service to a specific pod.
- Creates a network tunnel to the pod.
- Kubernetes networking is a great adventure!



Case #2: kubectl port-forward

Before:

- Port-forwarding problems get lumped together into One Big Megabug®
- “It’s not working!”

Anything goes wrong? Restart it.

You can see this play out in the Kubernetes issue tracker! Here’s a fun thread.

Case #2: kubectl port-forward

- Tilt has its own port-forwarding controller, forked from `kubectl port-forward`
- It adapts `kubectl port-forward` to the reconciler pattern (takes in a spec, reports a status)
- Much easier to debug and narrow down which part is failing
- The controller restarts the port-forwarder on any error.

Case #2: kubectl port-forward

After

Users can:

- Diagnose their own problems with spec/status information.
- Use the CLI to delete portforwards that are wedged and the system will self-heal.

Maintainers can:

- Split The Megabug® into distinct issues
- Recommend more accurate fixes

What have we learned?

Lessons

Joys and Pains of Building with apiserver

When you run Tilt today, you run a Kubernetes apiserver locally.

More and more of our basic primitives use the Kubernetes control loop infrastructure.

Pod Discovery, UI Buttons, Log Streams

What did we learn?

Joy #1

Many users have used kubectl!

We get lots of good verbs for free:

- kubectl edit, kubectl describe, kubectl explain, kubectl apply
- Robust, familiar
- Don’t underestimate the social conventions of devtools!

Joy #2

"Picking someone else's libraries means **you get a bunch of opinions for free** that you don't have to worry about.

"Kubernetes has some good opinions.

"It's not a matter of having the best opinions but a matter of **agreeing with an architecture** to follow."

Joy #3

Anything can interact/watch any element in the system in the same way

- Users on CLI
- “In-binary” code
- Separate-binary code (dashboards, visualizations, etc.)

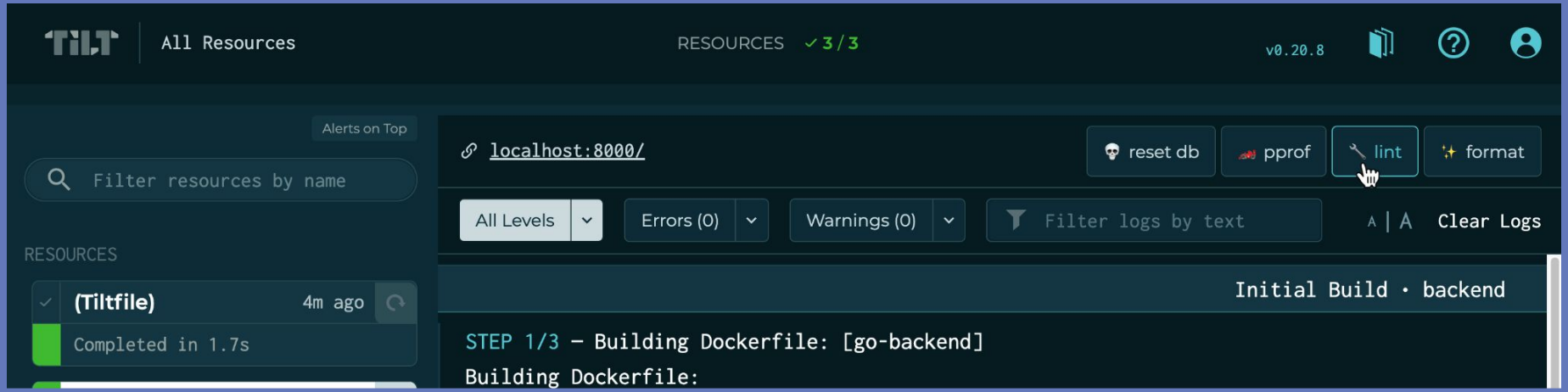
I’ve been thinking on a new blog post:

“Kubernetes is so simple you can implement

new features with Bash and Curl.”

```bash
#!/bin/bash

# Starts a controller that watches for new Cmds, and
# creates buttons to cancel them.
set -eou pipefail
tilt get cmd --watch -o name | while read -r cmd_full_name; do
    cmd_short_name=${cmd_full_name#cmd.tilt.dev/}
    ./reconcile_cancel_btn.sh "$cmd_short_name"
done
```

```
#!/bin/bash

# Reconcile function: creates a cancel button for resources that support it.
cancel_button_name="$cmd_name:cancel"
resource=$(echo "$cmd" | jq -r '.metadata.annotations["tilt.dev/resource"]')
cat <<EOF | tilt apply -f -
apiVersion: tilt.dev/v1alpha1
kind: UIButton
metadata:
  name: $cancel_button_name
spec:
  text: Cancel
  location:
    componentType: resource
    componentID: $resource
```


TiLT
All Resources
RESOURCES ✓3/3
v0.20.8
Alerts on Top
Filter resources by name
RESOURCES
(Tiltfile)
4m ago
Completed in 1.7s
localhost:8000/
reset db
pprof
lint
format
All Levels
Errors (0)
Warnings (0)
Filter logs by text
A | A
Clear Logs
Initial Build • backend
STEP 1/3 — Building Dockerfile: [go-backend]
Building Dockerfile:

Pains #1

Kubernetes/Go dependency management is its own eng challenge

> See DeTiberus' talk, who also spent a lot of time on this!
>
> It's a non-trivial concern!

"gRPC is a toxic dependency in the Go ecosystem, and should be blocklisted from any project" — @dave_universetf

(We **so** appreciate recent work to break up kubernetes/kubernetes into smaller repos! Thanks, y'all!)

Pains #2

Kubernetes also brings in a lot of things we don't need.

Storage: Kubernetes really likes etcd!

When Tilt plugged in an in-memory storage system, we hit lots of gotchas!

Security/Networking: Kubernetes really likes TLS and certs!

Further study

We're not the first team that noticed this trend!

Interesting talks from Kubecon EU 2021:

- KCP
- BadIdea

Also see: Kubebuilder

What else should be a CRUD++ app?

Thank you!

Nick Santos

→ nick@tilt.dev
→ @nicksantos
→ #tilt - slack.k8s.io

Thanks to:

→ L Körbes (@ellenkorbes)
→ The Tilt Dev Team

Questions