



KubeCon

CloudNativeCon

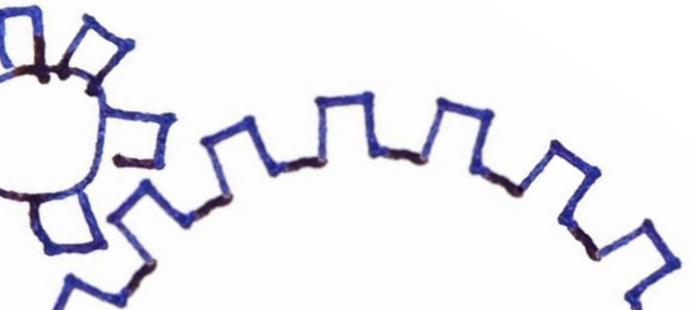
— North America 2018 —

A Vision for API Machinery

Coming to terms with the platform we built

Daniel Smith <dbsmith@google.com>
Staff Software Engineer, Google
[lavalamp@ github](https://github.com/lavalamp)
[@ originalavalamp@ twitter](https://twitter.com/originalavalamp)

A
VISION
FOR
API MACHINERY



ME

DANIEL SMITH

SIG API MACHINERY
CO-CHAR, CO-TL

STAFF SOFTWARE ENGINEER @ GOOGLE

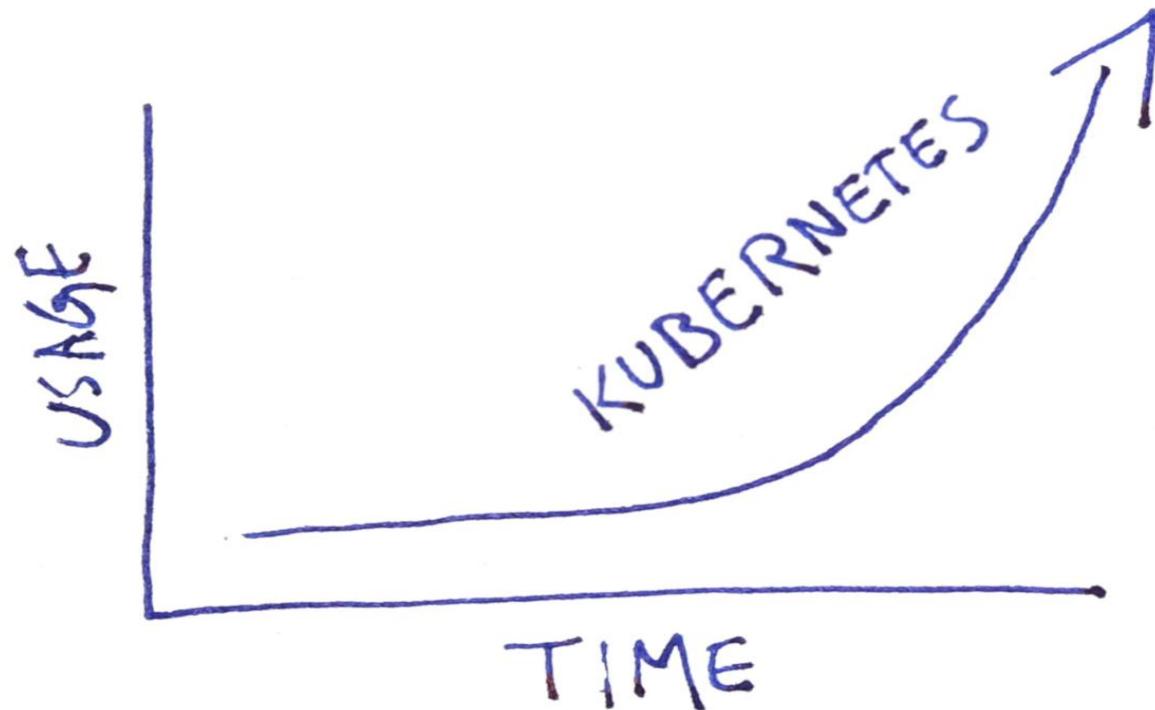


dbsmith@google.com
lavalamp@github
originallavalamp@twitter

1. WHERE WE CAME FROM
2. WHERE WE ARE
3. WHERE WE SHOULD GO

WHERE WE
CAME FROM

KUBERNETES :
UP AND TO THE RIGHT



CODE

```
// Defines the endpoints that implement the actual service, for example:  
// Name: "mysql", Endpoints: ["10.10.1.1:1909", "10.10.2.2:8834"]  
  
type Endpoints struct {  
  
    Name      string  
    Endpoints []string  
}  
  
...  
  
func ParseEndpoints(jsonString string) (api.Endpoints, error) {  
    var e api.Endpoints  
    err := json.Unmarshal([]byte(jsonString), &e)  
    return e, err  
}
```

June 5th, 2014

```
// JSONBase is shared by all objects sent to, or returned from the client
type JSONBase struct {
    Kind           string `json:"kind,omitempty" yaml:"kind,omitempty"`
    ID             string `json:"id,omitempty" yaml:"id,omitempty"`
    CreationTimestamp string `json:"creationTimestamp,omitempty"
                           yaml:"creationTimestamp,omitempty"`
    SelfLink       string `json:"selfLink,omitempty" yaml:"selfLink,omitempty"`
}
```

```
// TypeMeta is shared by all objects sent to, or returned from the client.

type TypeMeta struct {

    Kind           string `json:"kind,omitempty" yaml:"kind,omitempty"`
    ID            string `json:"id,omitempty" yaml:"id,omitempty"`
    CreationTimestamp util.Time `json:"creationTimestamp,omitempty" yaml:"creationTimestamp,omitempty"`
    SelfLink       string `json:"selfLink,omitempty" yaml:"selfLink,omitempty"`
    ResourceVersion string `json:"resourceVersion,omitempty" yaml:"resourceVersion,omitempty"`
    APIVersion     string `json:"apiVersion,omitempty" yaml:"apiVersion,omitempty"`
    Namespace      string `json:"namespace,omitempty" yaml:"namespace,omitempty"`
    UID            string `json:"uid,omitempty" yaml:"uid,omitempty"`

    // Annotations are unstructured key value data stored with a resource that may be set by
    // external tooling. They are not queryable and should be preserved when modifying
    // objects.

    Annotations map[string]string `json:"annotations,omitempty" yaml:"annotations,omitempty"`
}
```

```

// TypeMeta describes an individual object in an API response or request
// with strings representing the type of the object and its API schema version.
// Structures that are versioned or persisted should inline TypeMeta.
type TypeMeta struct {
    // Kind is a string value representing the REST resource this object represents.
    // Servers may infer this from the endpoint the client submits requests to.
    Kind string `json:"kind,omitempty"`

    // APIVersion defines the versioned schema of this representation of an object.
    // Servers should convert recognized schemas to the latest internal value, and
    // may reject unrecognized values.
    APIVersion string `json:"apiVersion,omitempty"`
}

// ObjectMeta is metadata that all persisted resources must have, which includes all objects
// users must create. A resource may have only one of {ObjectMeta, ListMeta}.
type ObjectMeta struct {
    // Name is unique within a namespace. Name is required when creating resources, although
    // some resources may allow a client to request the generation of an appropriate name
    // automatically. Name is primarily intended for creation idempotence and configuration
    // definition.
    Name string `json:"name,omitempty"`

    // Namespace defines the space within which name must be unique. An empty namespace is
    // equivalent to the "default" namespace, but "default" is the canonical representation.
    // Not all objects are required to be scoped to a namespace - the value of this field for
    // those objects will be empty.
    Namespace string `json:"namespace,omitempty"`

    // SelfLink is a URL representing this object.
    SelfLink string `json:"selfLink,omitempty"`

    // UID is the unique in time and space value for this object. It is typically generated by
    // the server on successful creation of a resource and is not allowed to change on PUT
    // operations.
    UID types.UID `json:"uid,omitempty"`

    // An opaque value that represents the version of this resource. May be used for optimistic
    // concurrency, change detection, and the watch operation on a resource or set of resources.
    // Clients must treat these values as opaque and values may only be valid for a particular
    // resource or set of resources. Only servers will generate resource versions.
    ResourceVersion string `json:"resourceVersion,omitempty"`

    // CreationTimestamp is a timestamp representing the server time when this object was
    // created. It is not guaranteed to be set in happens-before order across separate operations.
    // Clients may not set this value. It is represented in RFC3339 form and is in UTC.
    CreationTimestamp util.Time `json:"creationTimestamp,omitempty"`

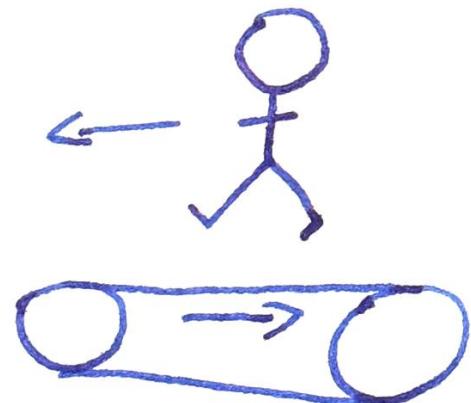
    // Labels are key value pairs that may be used to scope and select individual resources.
    // Labels must be valid JSON keys.
}

```

January 20, 2015

THE
ABSTRACTION
TREADMILL

THE ABSTRACTION TREADMILL





COMPUTERS



VMs

BINARIES
&
DEPENDENCIES



CONTAINERS

CONTAINERS



PODS

PODS



REPLICASETS

REPLICASETS



DEPLOYMENTS

CONTAINERS



PODS

PODS



REPLICASETS

DAEMONSETS

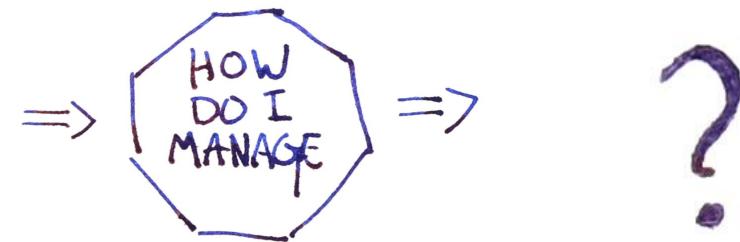
STATEFULSETS

REPLICASETS

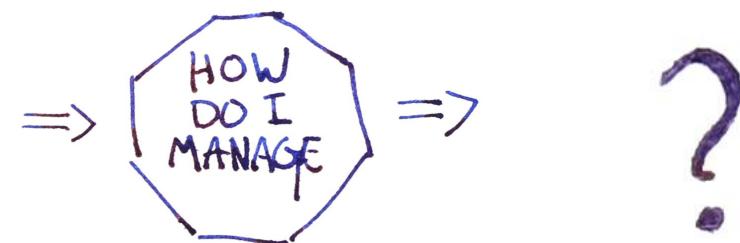


DEPLOYMENTS

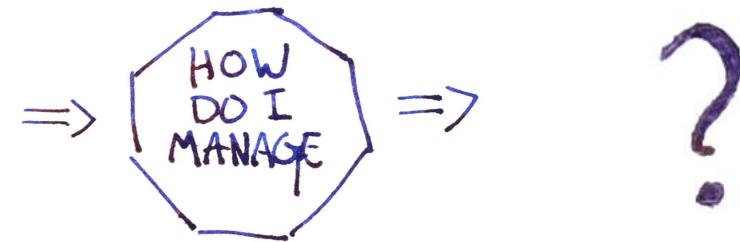
DAEMONSETS



STATEFULSETS



DEPLOYMENTS



DAEMONSETS



kubectl apply

STATEFULSETS



kubectl apply

DEPLOYMENTS



CI/CD SYSTEM

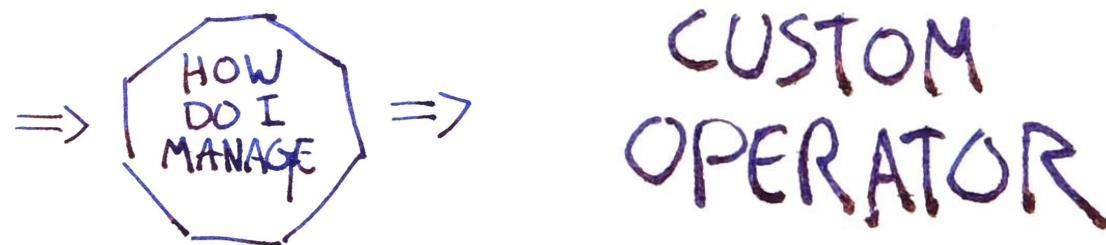
THIS DAEMONSET



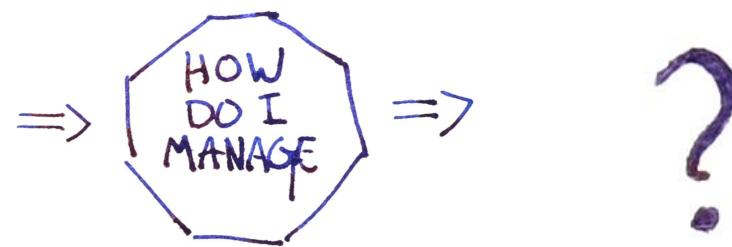
THIS STATEFUL SET.



THIS DEPLOYMENT



CUSTOM
OPERATOR



CUSTOM
OPERATOR



DEPLOYMENT

CUSTOM
OPERATOR



DEPLOYMENT

DEPLOYMENT



CUSTOM
OPERATOR

AN OPERATOR OPERATOR
IS STILL
AN OPERATOR!

NO ADDITIONAL ABSTRACTIONS ARE NECESSARY

... YOU DO NEED AN API FOR EACH OPERATOR ...

WHERE WE
ARE

KUBERNETES
HAS A CONSISTENT APT RESOURCE
MODEL

* COMPARTMENTAL
* REUSABLE

API MACHINERY
SUPPORTS AN
EXTENSIBLE API

- * TYPE
- * POLICY

WHAT
EXACTLY
IS THE
DIFFERENCE,
ANYWAY?

BOTH HAVE APIs . . .

Kubernetes APIs

Deployment

Pod

Endpoints

Node

API Machinery APIs

CustomResourceDefinition

APIService

Namespace

MutatingWebhookConfiguration

ValidatingWebhookConfiguration

ARE ROOMMATES IN kube-apiserver

Kubernetes

built in api handlers / validation
custom “subresource” handlers

API Machinery

kube-aggregator (APIService)
extensions-apiserver (CRDs)
policy hook calls
apiserver framework itself

AND PUBLISH THEIR API

Kubernetes

`k8s.io/api`

API Machinery

Multiple locations:

- `k8s.io/api`
- `k8s.io/apimachinery/pkg/apis`

BOTH HAVE CONTROLLERS

Kubernetes Controllers

Deployment

ReplicaSet

Endpoints

Node

API Machinery Controllers

Namespace Lifecycle

Garbage Collector

WHICH ARE ROOMMATES IN

kube-controller-manager

Kubernetes

Controllers for built in APIs

Cloud-specific controllers

API Machinery

Namespace / GC controllers

Controller framework:

- Reflector
- Informer (code generator!)
- workqueue

META STUFF IS ALL API MACHINERY

Kubernetes meta
none?

API Machinery meta

- ListMeta/ObjectMeta
- Optimistic concurrency
- OwnerReferences (GC)
- Watch: wire format(s)
- Proto wire format
- Status (error return format)

CONCRETE STUFF IS MOSTLY KUBERNETES

Kubernetes concrete

Liveness / readiness checks

Service selectors

Pod / node binding

PV / PVC mechanism

Ingress :)

API Machinery concrete

Flat namespace hierarchy

OPERATIONAL ISSUES

Kubernetes

API OBJECT CHANGES

VERSION UPGRADE / ROLLBACK

+

=



API Machinery

No apiserver replica coordination

Insufficient scale:

- # API Objects
- Monolithic controllers

OPERATIONAL ISSUES HAVE A CROSS PRODUCT

Suppose Kubernetes:

Adds a field in a v1 resource

Adds a new webhook

And API Machinery:

adds a new proto encoder

Adds a webhook requirement
(e.g., side effects y/n)

One might think that would block uptake, but no!
We have massive uptake of Kubernetes as a platform!

CRDs in the wild: (TODO)

policy webhooks in the wild: (TODO)

aggregated apis in the wild: (TODO)

WHERE WE
SHOULD GO

GOALS

EMBRACE THE DISTINCTIONS
TO BETTER SUPPORT

THE KUBERNETES ECOSYSTEM
THE KUBERNETES PROJECT

INTEROPERABILITY

CLIENT-SIDE INTEROPERABILITY

kubectl

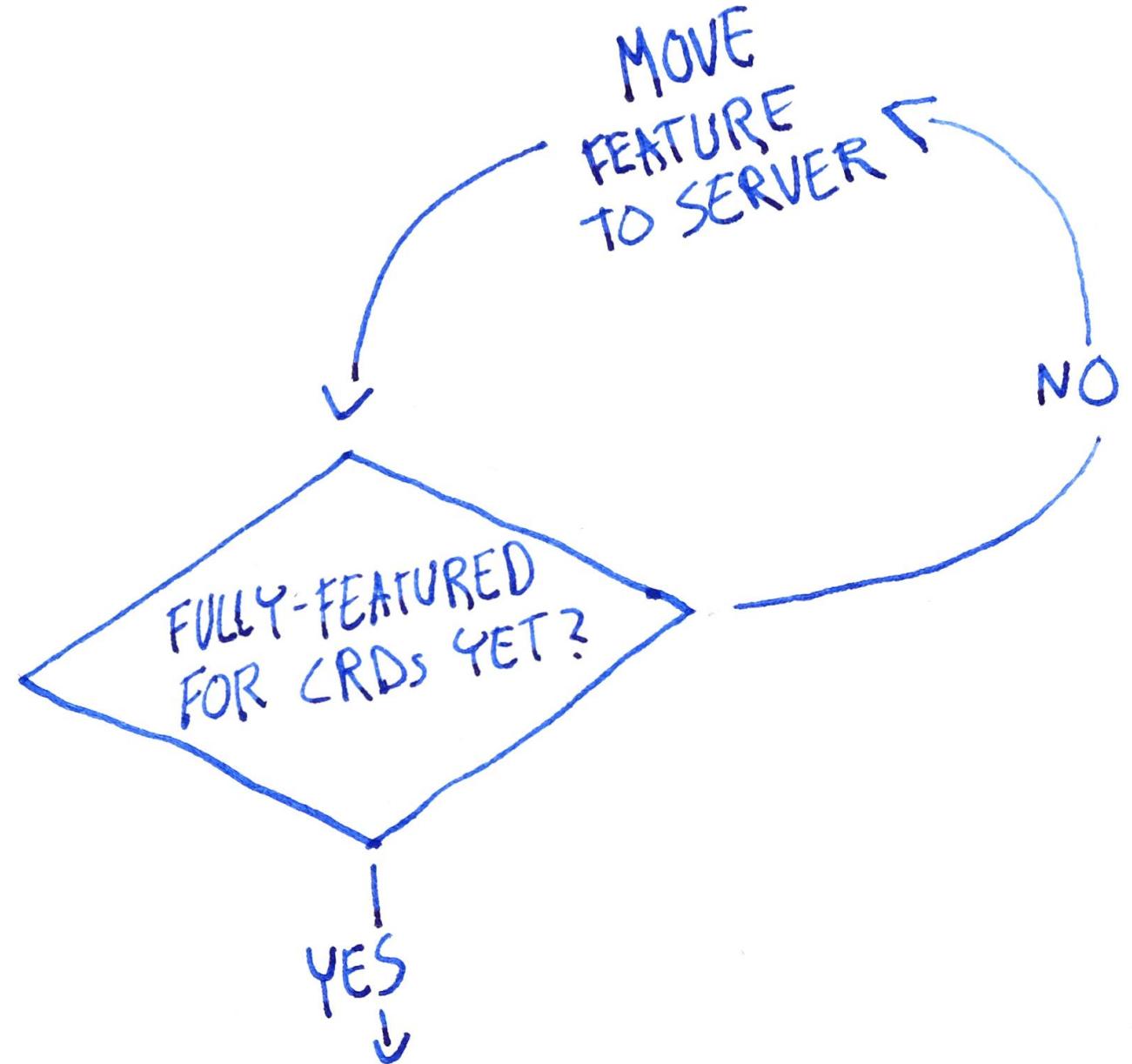
ONLY ONE CTL SHOULD BE NECESSARY

NO LANGUAGE LEFT BEHIND

CLIENT-SIDE
INTEROPERABILITY

kubectl

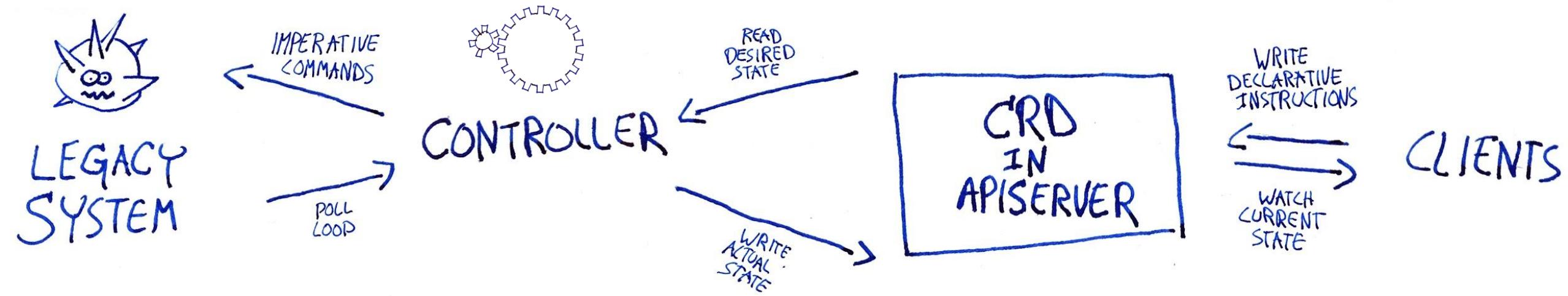
P.S. PRONOUNCED: "CUBE CONTROL"



SERVER-SIDE
INTEROPERABILITY



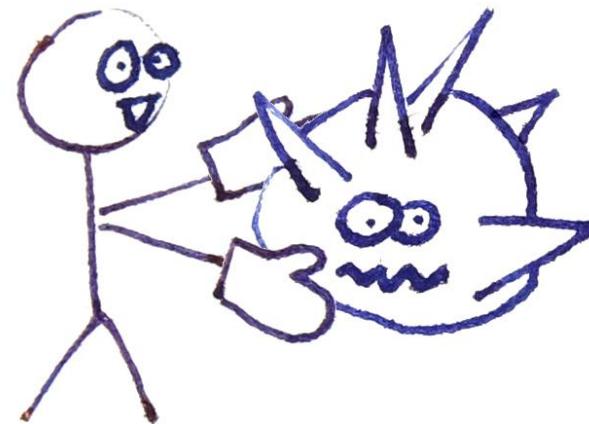
LEGACY
SYSTEM



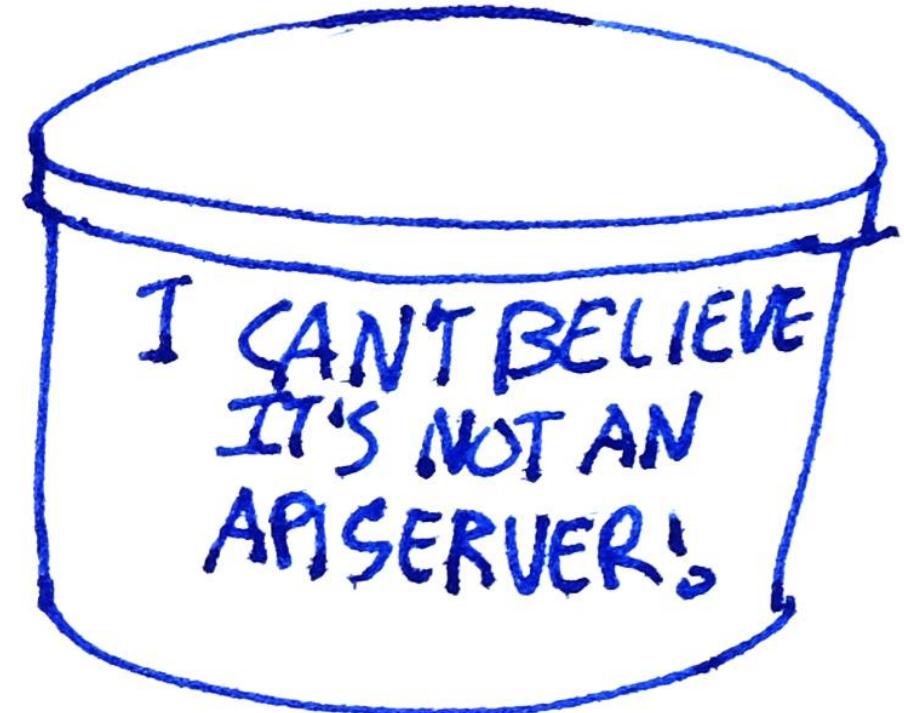
BRING
Your
Own

API SERVER!

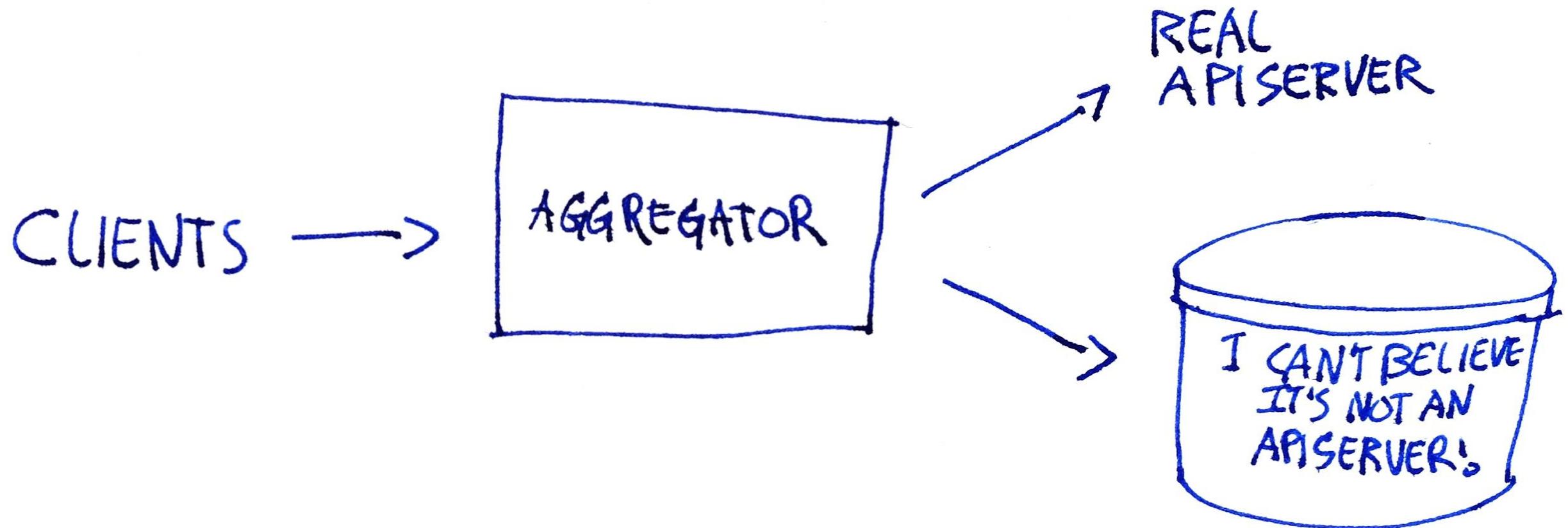
"HI! I WANT TO
HOOK MY LEGACY
APP UP AS THE
SOURCE OF TRUTH,
LIKE AN AGGREGATED
API SERVER!"

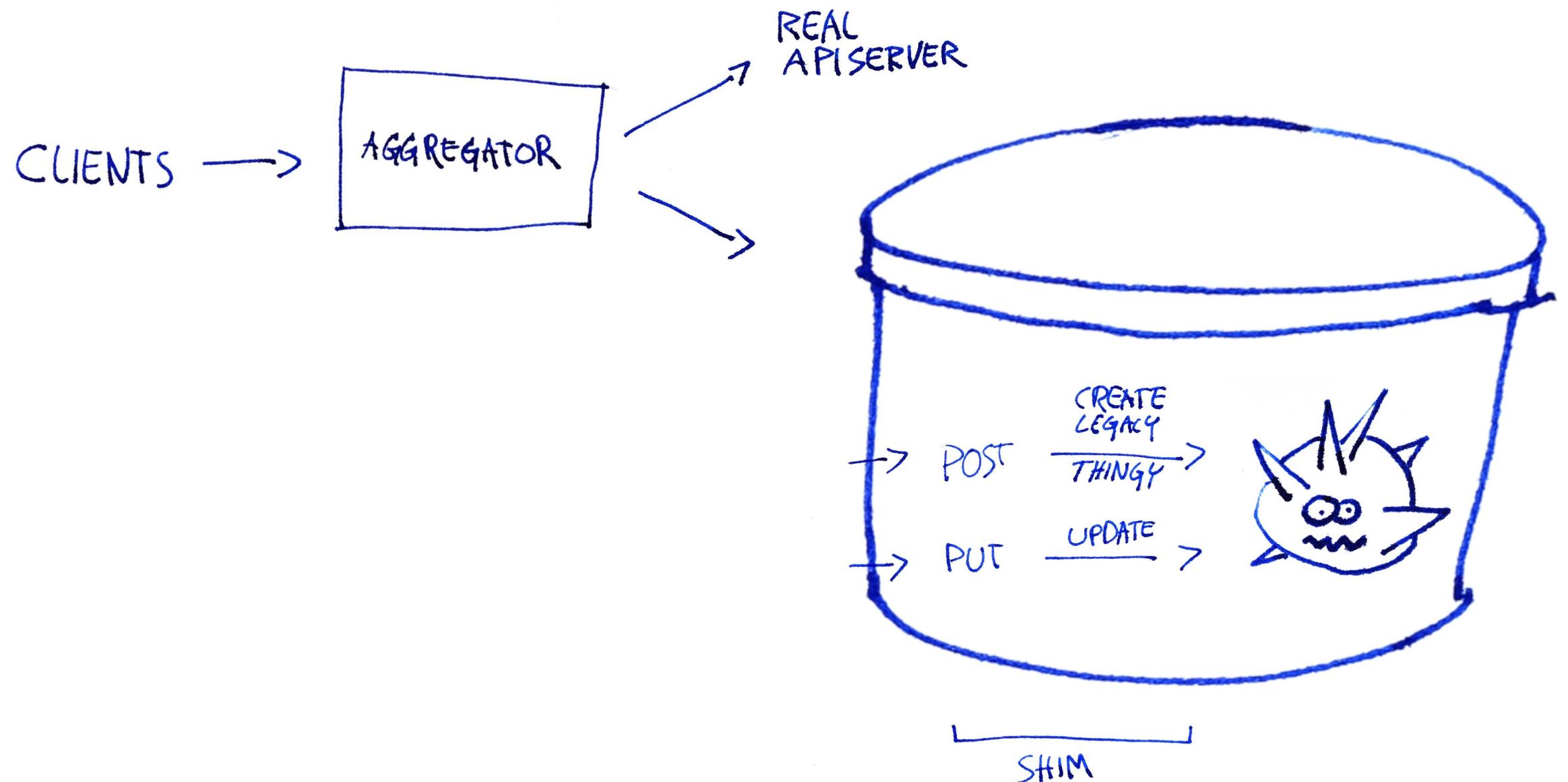


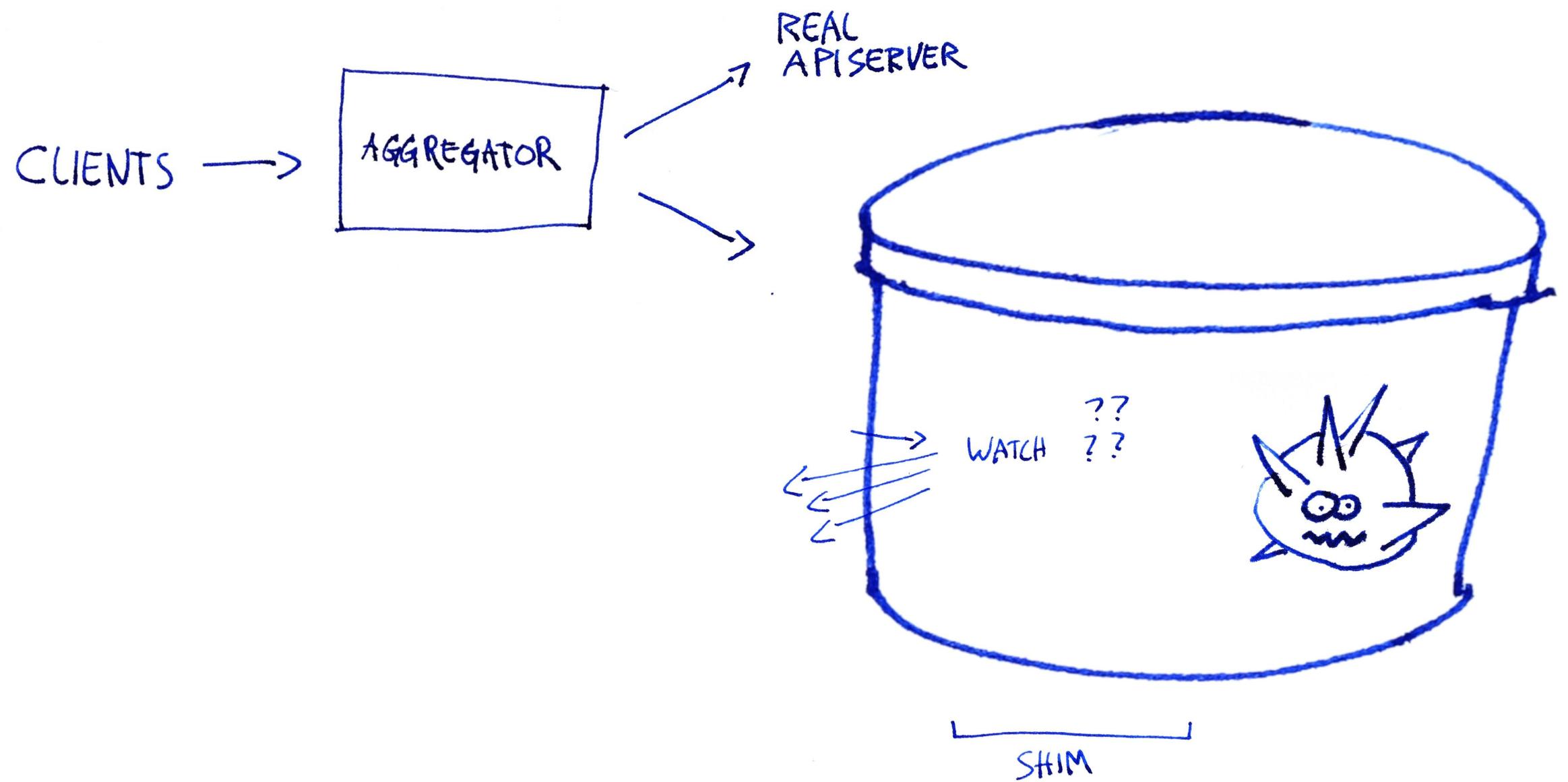
INSTANT
APISERVER!
JUST ADD
WATER!
LEGACY SYSTEM!

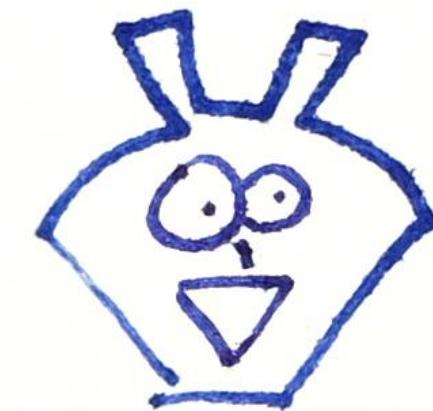
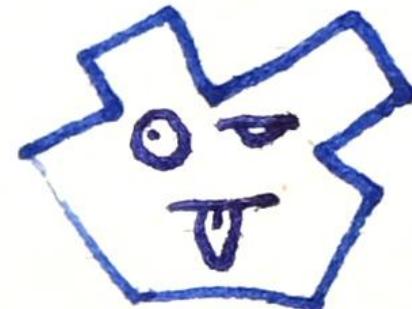
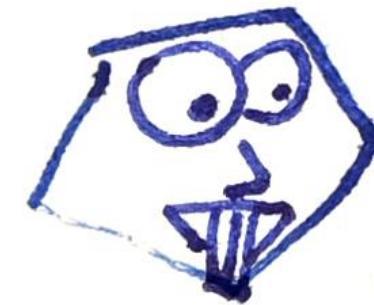
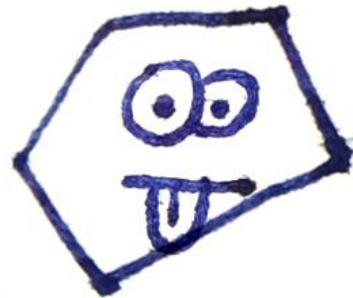


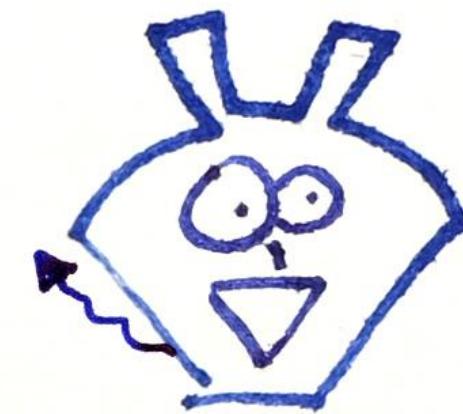
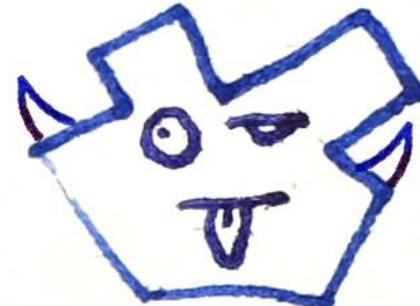
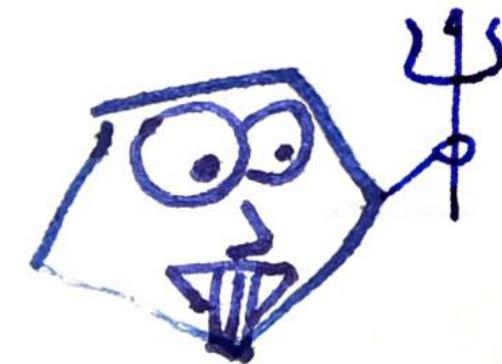
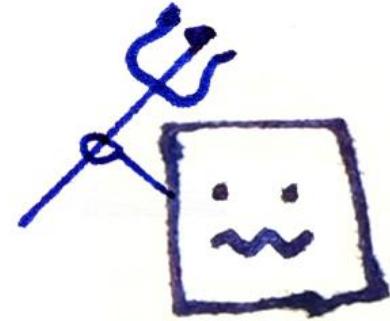
I CAN'T BELIEVE
IT'S NOT AN
APISERVER!











EMBRACE THE DISTINCTIONS
TO BETTER SUPPORT

THE KUBERNETES PROJECT

CLARIFY BOUNDARIES

- * CODE
- * LIBRARIES
- * REPOSITORIES

- * BINARIES
- * RELEASE ARTIFACTS
- * OPERATIONALLY

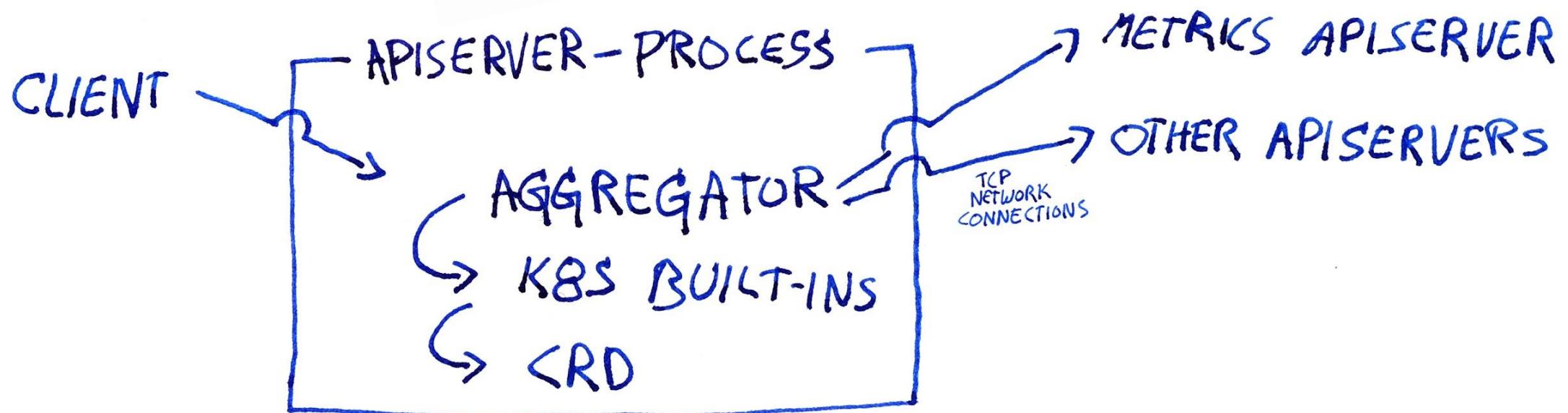
GENERAL API CONFORMANCE TESTS

- * CLIENT
- * SERVER

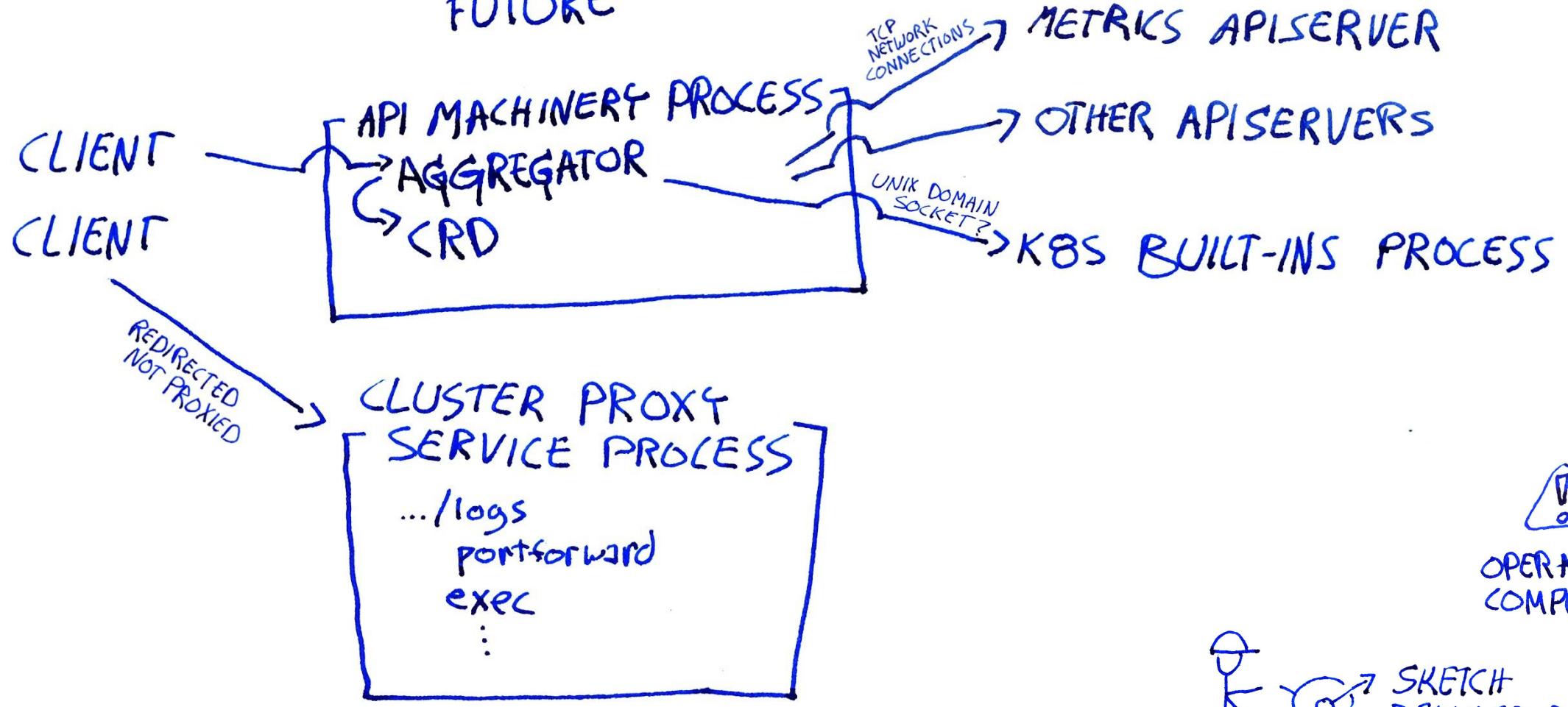
IDEAS

REFACTOR
BINARIES kube-apiserver

TODAY



REFACTOR BINARIES IN THE FUTURE kube-apiserver

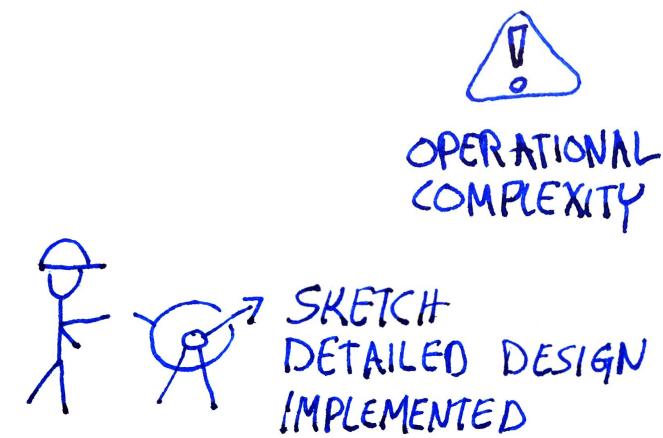
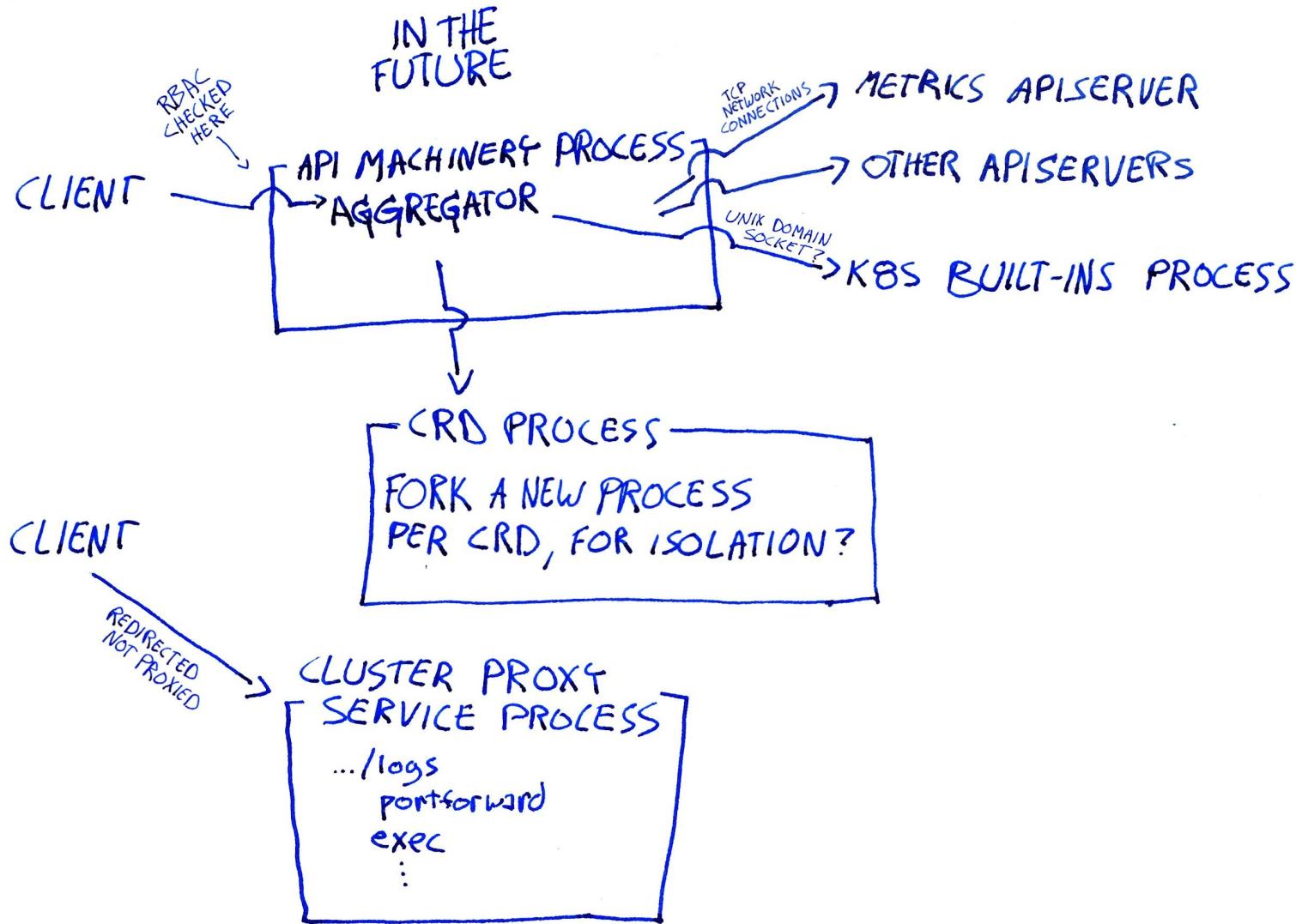


OPERATIONAL COMPLEXITY



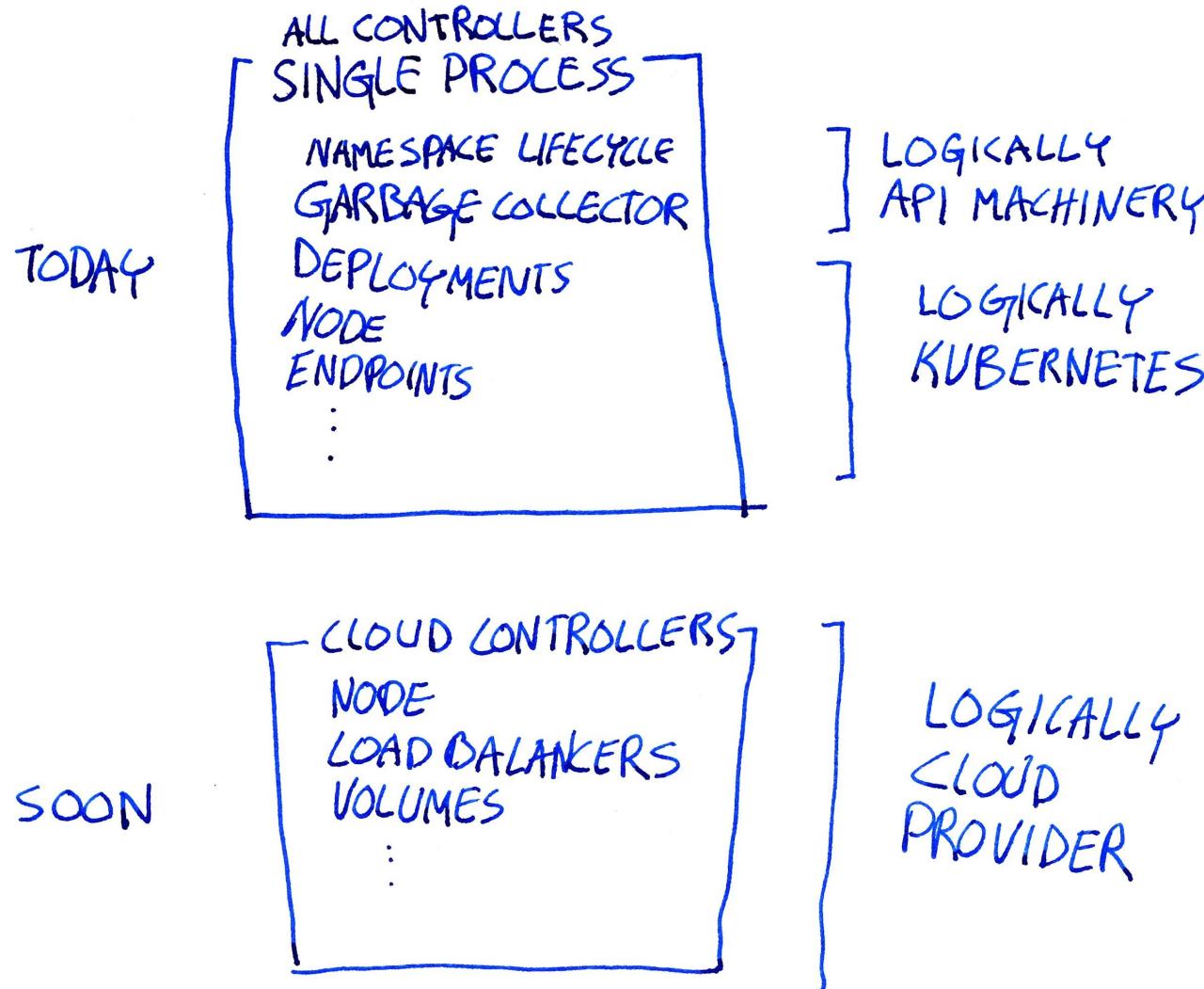
SKETCH DETAILED DESIGN IMPLEMENTED

REFACTOR BINARIES kube-apiserver



REFACTOR BINARIES

kube-controller-manager



REFACTOR
BINARIES

kube-controller-manager

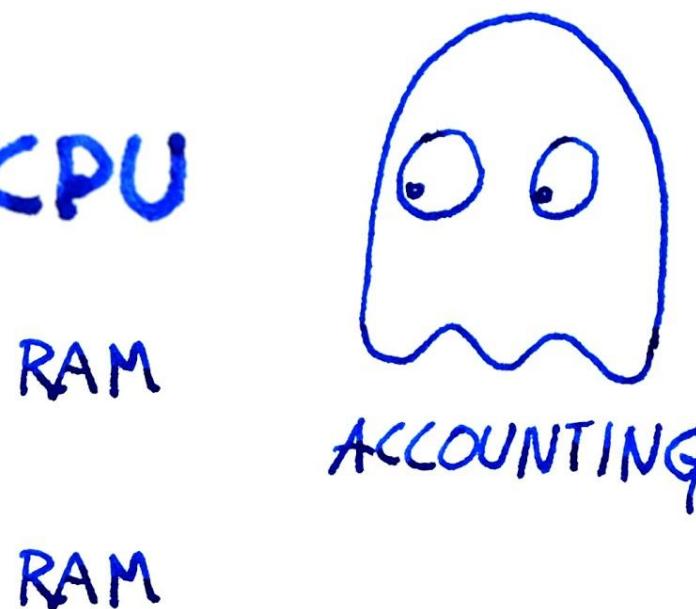
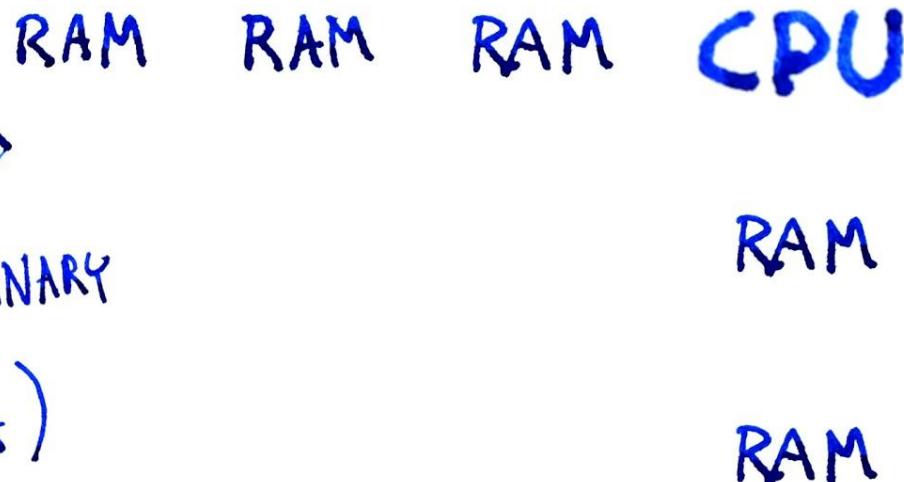
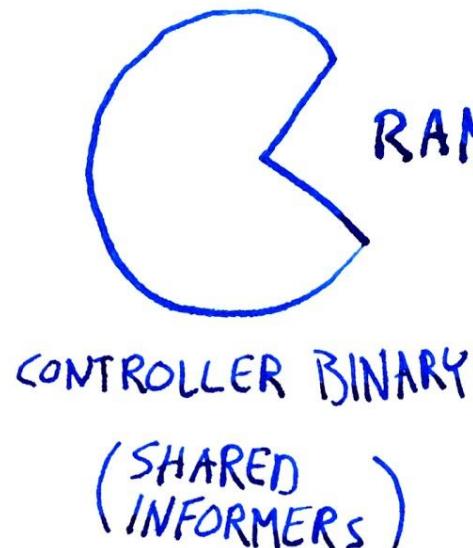
API MACHINERY
CONTROLLERS

BUILT IN
CONTROLLERS

CLOUD PROVIDER
CONTROLLERS

REFACTOR
BINARIES kube-controller-manager

RESOURCE USAGE



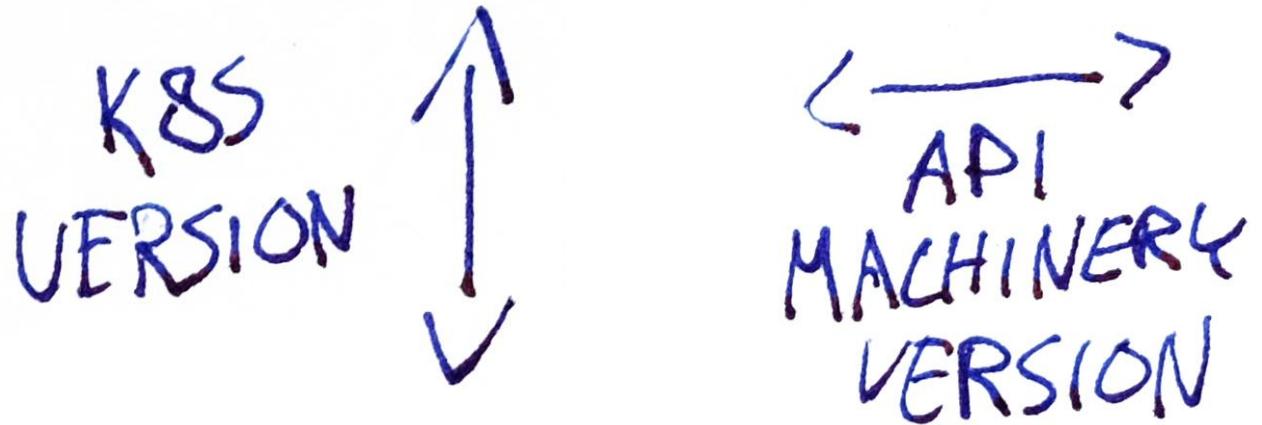
RAM RAM



OPERATIONAL
COMPLEXITY

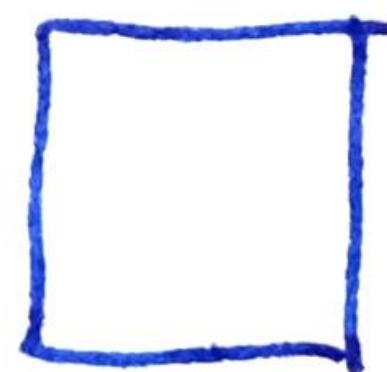
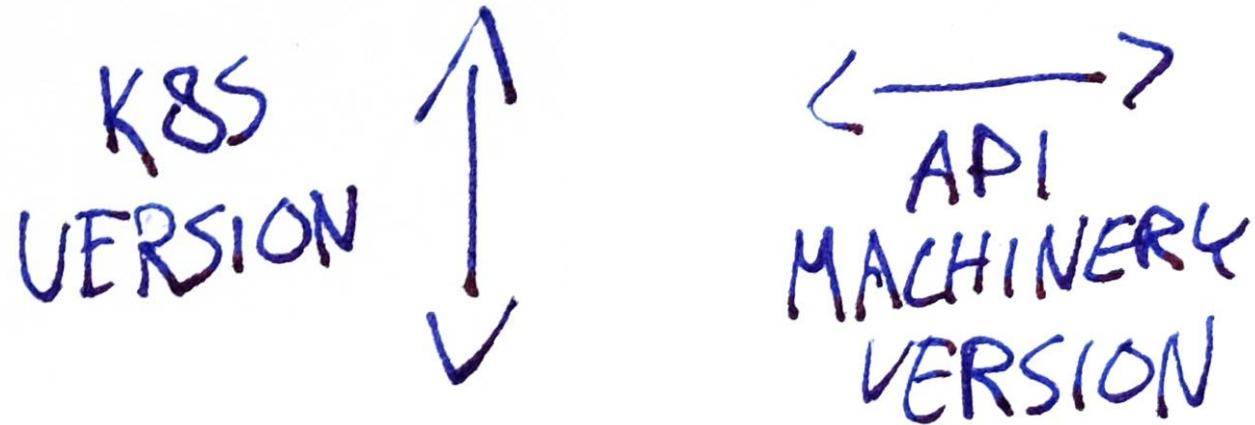
REFACTOR PROCESS

- * FREQUENT RELEASES
- * TAKE INTERFACES SERIOUSLY



- * CHANGE SCHEMAS
... but BUT NOT MECHANISMS
- * CHANGE MECHANISMS
... BUT NOT SCHEMAS

DIFFERENT
RISK
PROFILES



DAYS SINCE
LAST CVE

REFACTOR

SOCIAL STRUCTURE

NEW GITHUB ORG?

HOW
DO WE
REALIZE THIS FUTURE?

TWO
POTENTIAL
APPROACHES

/staging



/STAGING ++



TECHNICAL
DETAILS

/staging



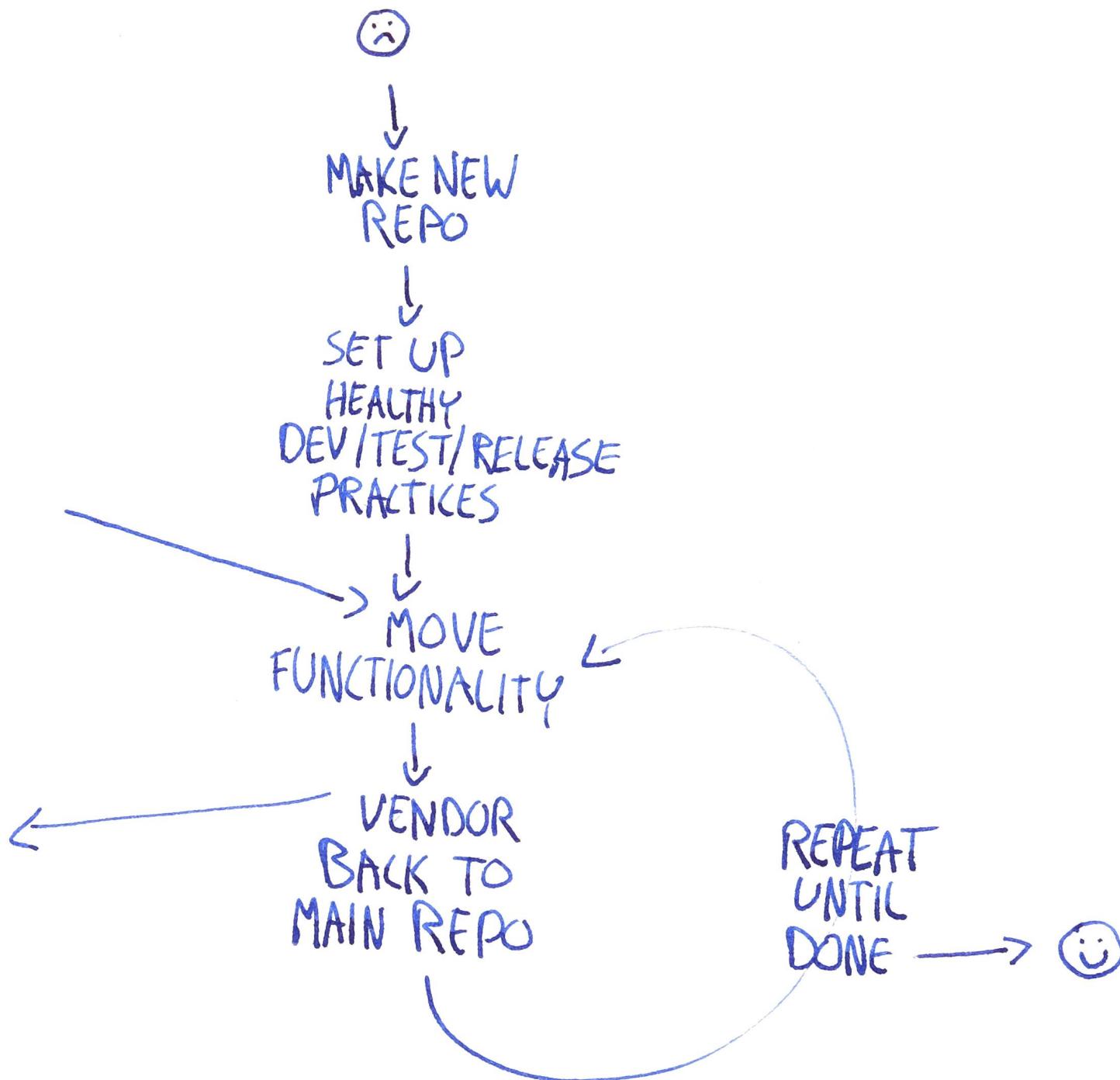
/STAGING ++

/staging

EVERYONE'S
FAVORITE
DIRECTORY!!



TECHNICAL
DETAILS



SOUNDS
\$\$\$
EXPENSIVE

IS IT
WORTH IT ???

IS IT WORTH IT ???

- * ECOSYSTEM GROWTH RATE MULTIPLIER
- * ARCHITECTURAL CLARITY FOR NEW ENTRANTS
- * IMPROVED TESTING & CONFIDENCE

VELOCITY:

→ SLOWER, THEN FASTER (NEW FEATURES)

→ FASTER (# COMMITS / LOC CODE CHANGES)

"I LIKE THESE IDEAS,
HOW CAN I HELP?"



"I HAVE CONCERNS,
WHERE CAN I LEAVE
FEEDBACK?"



"I NEED A CONFIG CHANGE"

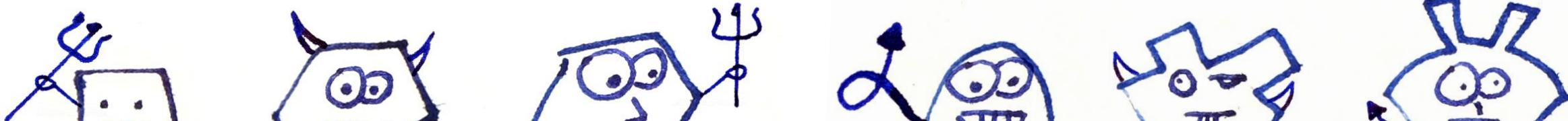


HOW DO I GET INVOLVED?

SIG API MACHINERY
&
SIG ARCHITECTURE

MEETING
&
EMAIL LIST

WE CAN'T DO THIS
WITHOUT YOU





KubeCon

CloudNativeCon

North America 2018

BOTH ARE ROOMMATES IN `client-go`

Kubernetes

Built in typed API clients

API Machinery

Config parsing

RESTClient

Dynamic client

Discovery client

... client generators

PLUGIN BOUNDARIES \Rightarrow BINARY BOUNDARIES

CALLING A LIBRARY

IS
BETTER
THAN

1. COPY DIRECTORY
2. SEARCH & REPLACE
3. RUN A TON OF RANDOM CODE GENERATORS