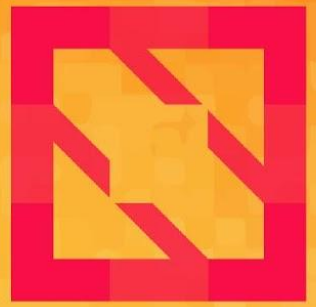
Mastering Multi-version CRDs From YAML to a Serious Development Project

This is an interactive tutorial!

As you arrive, start getting ready!

https://bit.ly/2JWsbxC



KubeCon | CloudNativeCon

North America 2019

KubeCon | CloudNativeCon

North America 2019

Mastering Multi-version CRDs
From YAML to a Serious Development Project

Stefan Schimanski, Red Hat
Joe Betz, Google

Getting Ready

This is an interactive tutorial!

Following along is **optional** but **encouraged**!

More detailed Instructions at:
https://bit.ly/2JWsbxC

Getting Ready

Step 1: Create a Kubernetes cluster

Step 2: Checkout the git repo

Step 3: Setup go development tools

More detailed Instructions at:
https://bit.ly/2JWsbxC

Getting Ready

Step 1: Create a Kubernetes Cluster

Option 1

Use a provided Google Cloud Temporary Account to create a GKE Cluster

Accounts are being handed out. Look around for one or raise your hand.

Option 2

Use your own Kubernetes cluster (You'll need a docker repo too)

Start getting it ready now!

More detailed Instructions at: https://bit.ly/2JWsbxC

Getting Ready

Step 1: Create a Kubernetes Cluster: GKE Cluster Creation

via Web Console

- Visit https://console.cloud.google.com
- Log in using the provided temporary account.
- Select the “Multi-version CRDs…” project from the “select a project” top nav bar
- Go to: Kubernetes Engine -> Clusters
- Click “Create Cluster”
- Give the cluster a name. e.g. “**kubecon-crd-tutorial**”
- **Click “Use release channels”**
- Select a “1.15.4-gke.18” cluster version
- Click “create”

via gcloud CLI

```
$ gcloud auth login <account-username>
$ gcloud config set project <project-id>
$ gcloud container clusters create \
    <cluster-name> \
    --zone us-central1-a \
    --cluster-version 1.15.4-gke.18
```

More detailed Instructions at:
https://bit.ly/2JWsbxC

Getting Ready

Decide if you will use your local terminal, or a cloud shell

Option 1

Use a Google Cloud Shell

No installation steps required. Might not work well in large conference room with limited wifi.

Option 2

Use your terminal

More reliable. Requires gcloud and go 1.12+.

More detailed Instructions at: https://bit.ly/2JWsbxC

Getting Ready

KubeCon | CloudNativeCon
North America 2019

Step 1: Create a Kubernetes Cluster: kubectl access for GKE

Option 1: Cloud Shell

- click “Connect” on the same screen the cluster was created on
- If using cloud shell for development click “Run In Cloud Shell”
- When the shell opens, press enter to run the “gcloud” command that has been automatically pasted into the shell.

Option 2: Local Shell

Install gcloud (https://cloud.google.com/sdk/install)

```
$ gcloud auth login <account-username>
$ gcloud config set project <project-id>
$ gcloud container clusters get-credentials \
    <cluster-name> --zone us-central1-a
```

(devstar####@gcplab.me)
(multi-crd-kubecon19-san-####)
(kubecon-crd-tutorial)

Name	Location	Cluster size	Total cores	Total memory	Notifications	Labels	
kubecon-crd-tutorial	us-central1-a	3	3 vCPUs	11.25 GB			Connect

More detailed Instructions at:
https://bit.ly/2JWsbxC

Getting Ready

Step 2: Checkout the git repository

```
$ git clone https://github.com/jpbetz/KoT
$ cd KoT
```

This will be our working directory for the rest of the Tutorial.

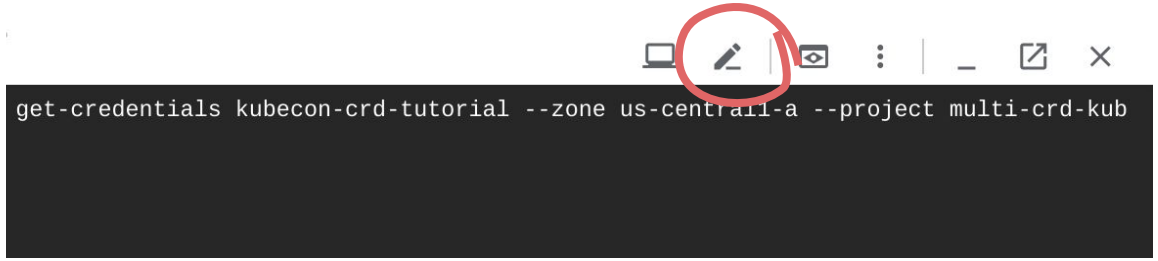
More detailed Instructions at: https://bit.ly/2JWsbxC

Getting Ready

Step 3: Setup the go development tools

Option 1: Develop in Google Cloud Shell

Click on the “Launch Editor” button at the top of the Cloud Shell window to open a basic text editor



Option 2: Local Development Tools

run **go version**. If the version is 1.12+, you’re ready to go.
Otherwise, download go 1.12+
https://golang.org/dl/

Open The KoT project in an editor.

More detailed Instructions at:
https://bit.ly/2JWsbxC

Getting Ready

Sanity Check

$ kubectl version

Kubernetes server version should be 1.15 or 1.16.

$ go version

Version should be 1.12+.

More detailed Instructions at:
https://bit.ly/2JWsbxC

Advanced CRDs

What is an advanced CRD?


```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: deepsea.kubecon.io
webhooks:
- name: deepsea.kubecon.io
  sideEffects: None
  failurePolicy: Fail
  ...
```

admission

```
validation:
  openAPIV3Schema:
    type: object
    properties:
      spec:
        type: object
        ...
```

OpenAPI schemas

```
conversion:
  strategy: Webhook
  webhook:
    clientConfig:
      caBundle: LS0tLS1CRUdJTiBDRV...
      service:
        namespace: things
        name: conversion-webhook
        path: /convert/v1/devices
```

conversion

```
crd.yaml:
apiVersion: apiextensions/v1
kind: CustomResourceDefinition
spec:
  group: …
  names: ...
```

code generation

controller-tools **controller-gen:**
types.go => Go client + OpenAPI schema

```
versions:
- name: v1alpha1
  strorage: true
  schema:
    openAPIV3Schema:
      …
- name: v1
  storage: false
```

versioning

```
spec:
  preserveUnknownFields: false
```

pruning

```
validation:
  openAPIV3Schema:
    …
    replicas:
      type: integer
      minimum: 0
    type:
      type: string
      enum: ["Foo","Bar"]
```

value validation

What is an advanced CRD?


```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: deepsea.kubecon.io
webhooks:
- name: deepsea.kubecon.io
  sideEffects: None
  failurePolicy: Fail
  ...
```

admission

```
validation:
  openAPIV3Schema:
    type: object
    properties:
      spec:
        type: object
        ...
```

OpenAPI schemas

```
conversion:
  strategy: Webhook
  webhook:
    clientConfig:
      caBundle: LS0tLS1CRUdJTiBDRV...
      service:
        namespace: things
        name: conversion-webhook
        path: /convert/v1/devices
```

conversion

```
crd.yaml:
apiVersion: apiextensions/v1
kind: CustomResourceDefinition
spec:
  group: …
  names: ...
```

code generation

```
controller-tools controller-gen:
  types.go => Go client + OpenAPI schema
```

```
validation:
  openAPIV3Schema:
    …
    replicas:
      type: integer
      minimum: 0
    type:
      type: string
      enum: ["Foo","Bar"]
```

value validation

```
versions:
- name: v1alpha1
  strorage: true
  schema:
    openAPIV3Schema:
      …
- name: v1
  storage: false
```

versioning

```
spec:
  preserveUnknownFields: false
```

pruning

Agenda

- What are advanced CRDs?
- Our example for the next 90 min
- Code generation & controller

- OpenAPI generation
- OpenAPI value validation
- Pruning

- Versioning
- Admission

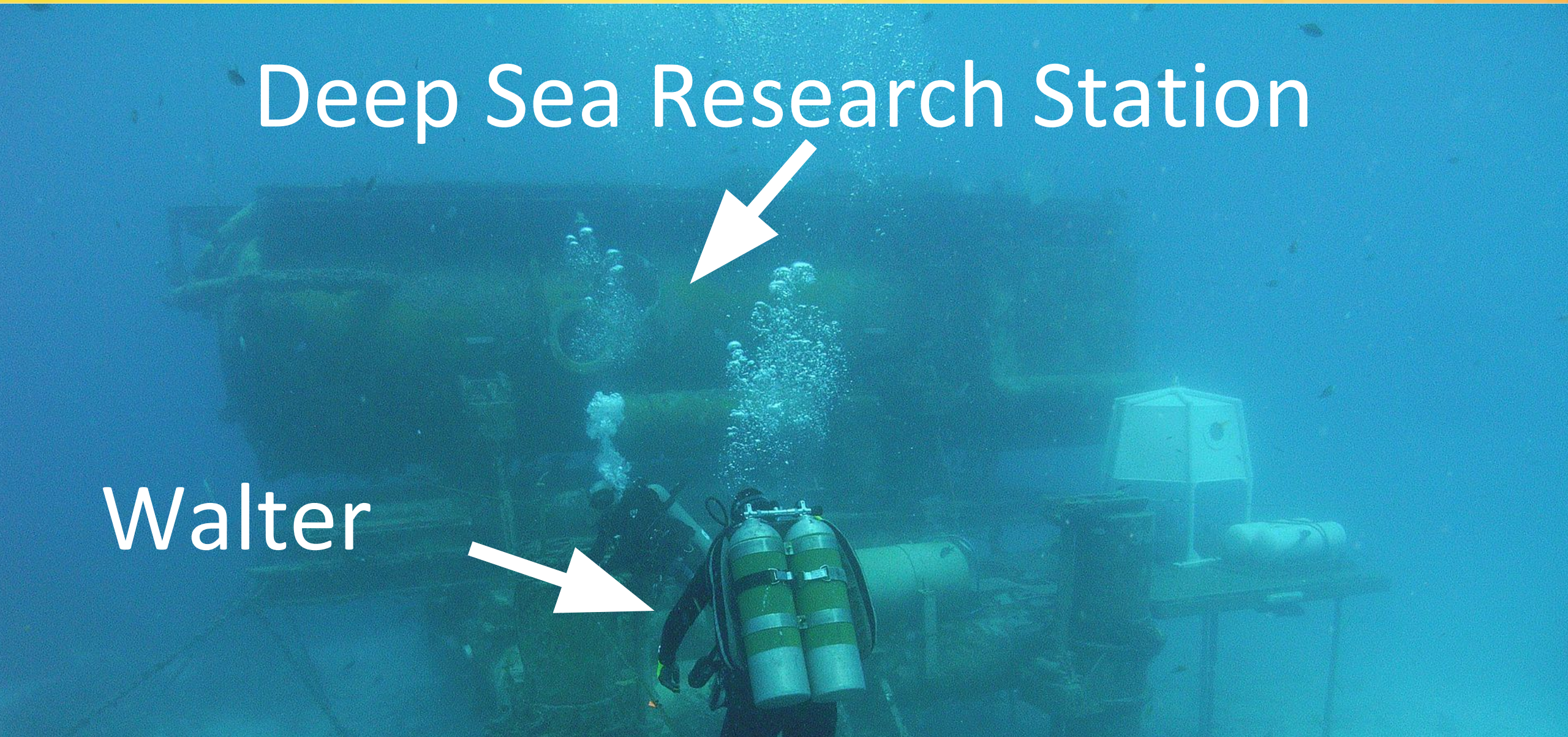
⇦ Exercise: setup your cluster

⇦ Exercise: start simulator

⇦ Exercise: validate values and union type

⇦ Exercise: v1 <-> v1alpha1 conversion
⇦ Exercise: validate module devices
⇦ Exercise: write controller

Our Task Today

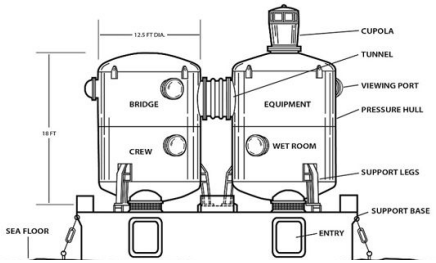


Our Task Today

Atmospheric pressure at sea level

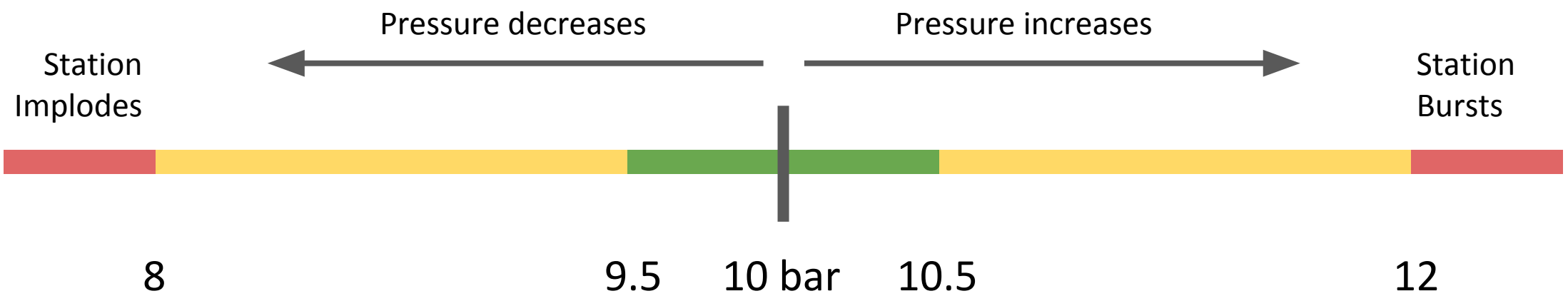
1 bar

Pressure at our Research Station, 90 meters (297ft) below



10 bar

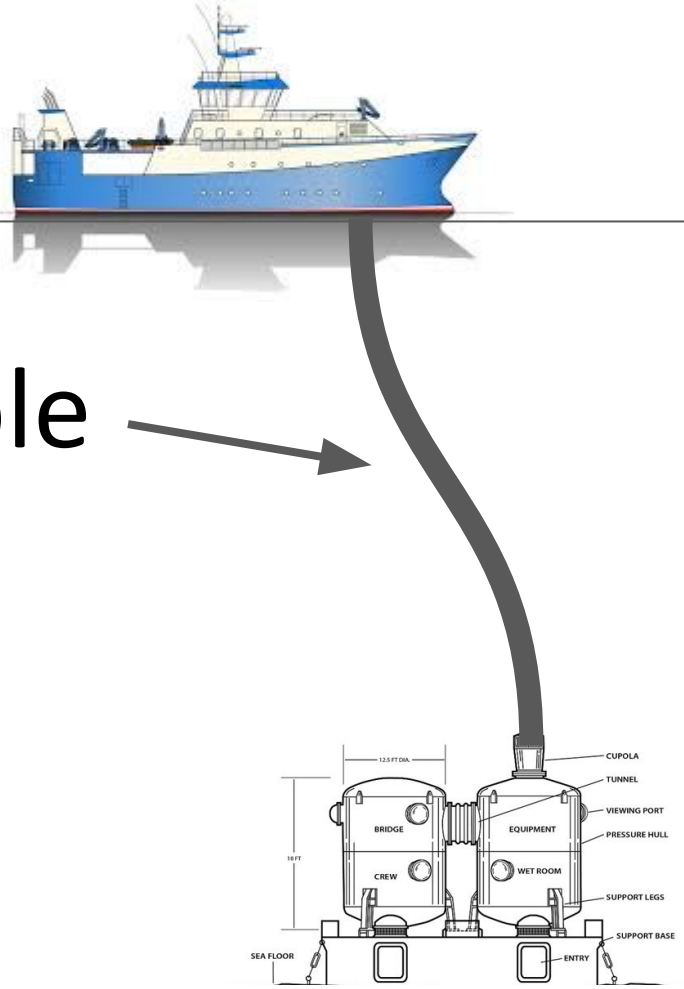
Our Task Today



We don’t want either of these things to happen.

Walter is inside.

Our Task Today



Umbilical Cable

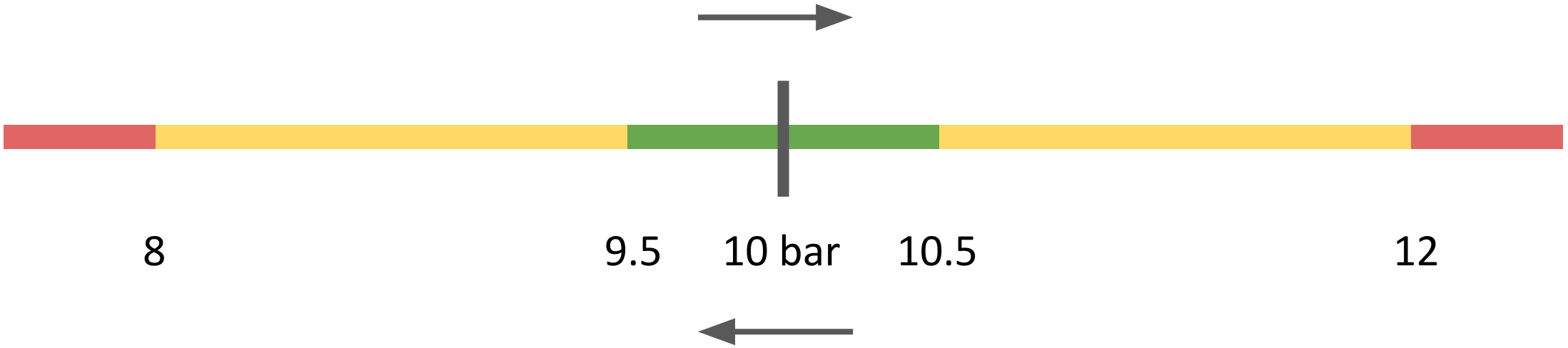
- Electricity
- Data
- Air

...and airflow == pressure changes!

Fortunately, we've got pumps we can use to manage our pressure.

Our Task Today

Run more pumps to increase pressure.



Run fewer pumps to decrease pressure.

Deepsea CRDs


```yaml
apiVersion: things.kubecon.io/v1alpha1
kind: Device
metadata:
  name: pump-1
spec:
  inputs:
  - name: activeCount
    value: "3"
    type: "Integer"
status:
  observedInputs: …
  outputs: …
```

Devices exist as v1 and v1alpha1.

```yaml
apiVersion: deepsea.kubecon.io/v1alpha1
kind: Module
spec:
  devices:
    pump: pump-1
    waterAlarm: alarm-1
    pressureSensor: sensor-1
```

Modules only exist as v1alpha1.

Devices in 2 versions

Different representations of the same object:

- **There is no:**

  "object of version X in the cluster"

- All objects exist in all defined versions.

```
apiVersion: things/v1
kind: Device
metadata:
  name: pump-1
spec:
  inputs:
  - name: activeCount
    integer: 3
```

lossless conversion

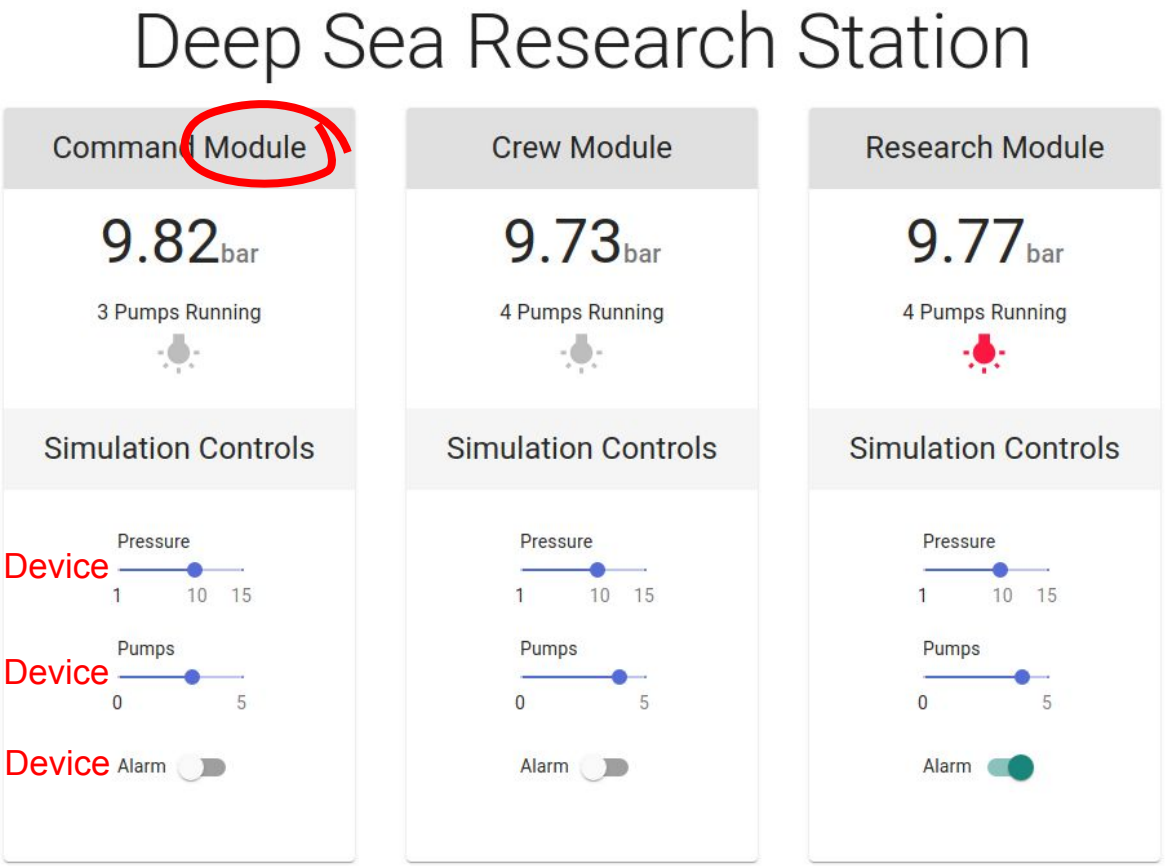
```
/v1alpha1
spec:
  inputs:
  - name: activeCount
    value: "3"
    type: "Integer"
```

In etcd:

- Each object is versioned.
- Etcd can have mixture of versions.
- Storage version defines version of future writes.

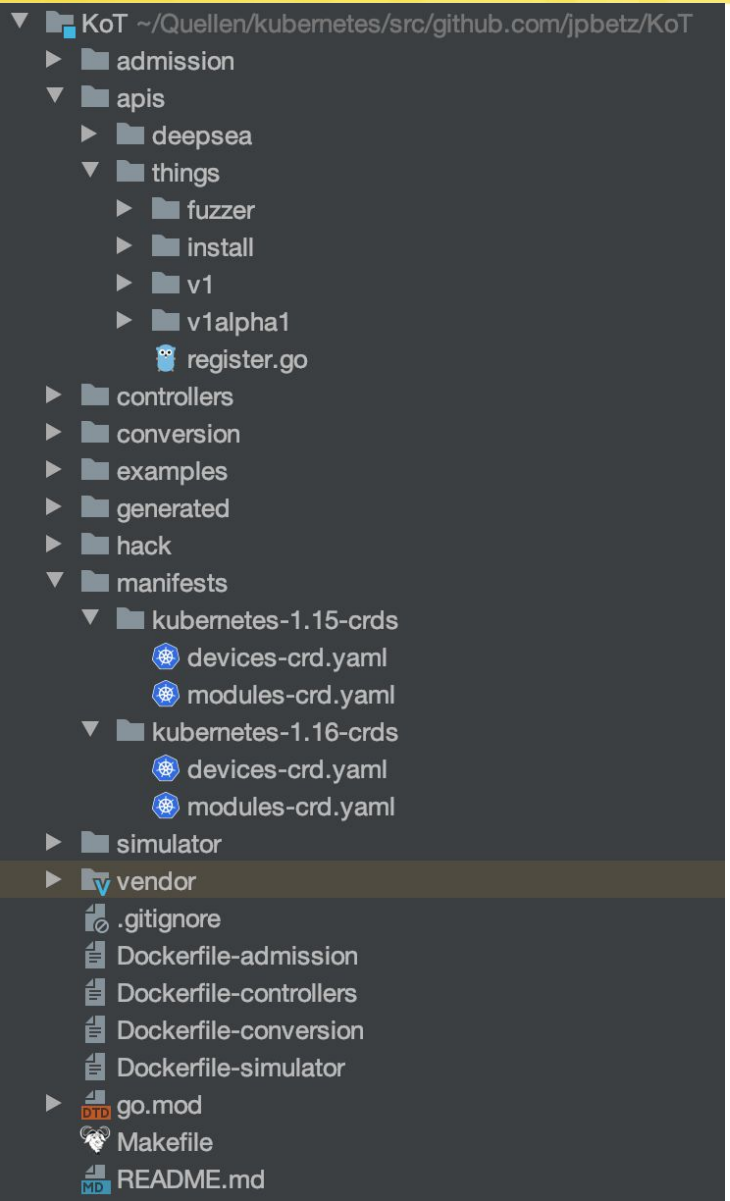
→ github.com/kubernetes-sigs/kube-storage-version-migrator

Simulator


Deep Sea Research Station
Command Module
9.82bar
3 Pumps Running
Simulation Controls
Device
Pressure
1
10
15
Device
Pumps
0
5
Device
Alarm
Crew Module
9.73bar
4 Pumps Running
Simulation Controls
Pressure
1
10
15
Pumps
0
5
Alarm
Research Module
9.77bar
4 Pumps Running
Simulation Controls
Pressure
1
10
15
Pumps
0
5
Alarm

Kubernetes-of-Things Repository

github.com/jpbetz/KoT



admission – validating admission webhook
apis – the API types for devices and modules
controllers – pressure regulation controller
conversion – CRD conversion webhook
examples – sample device and module yaml files
generated – code-generation output, client, informers
manifests – CRD manifest for Kube 1.15 and 1.16
simulator – the web UI

{conversion,admission,controllers}/manifests.yaml – kube yamls

hack/update-codegen.sh – regenerate clients
hack/update-crds – update OpenAPI schemas

```
DOCKER_ORG=docker.io/sttts \
    make build-conversion push-conversion
```

or admission or simulator

CRD and conversion setup – manifests/

or 1.16 if you have it, to get defaulting

```
$ kubectl apply -f manifests/kubernetes-1.15-crds
customresourcedefinition.apiextensions.k8s.io/devices.things.kubecon.io created
customresourcedefinition.apiextensions.k8s.io/modules.deepsea.kubecon.io created

$ kubectl apply -f conversion/manifests.yaml
...

$ kubectl apply -f examples/research-module
module.deepsea.kubecon.io/research created
device.things.kubecon.io/research-water created
...
$ kubectl apply -f examples/crew-module
$ kubectl apply -f examples/command-module

$ kubectl get devices,modules
NAME                                          PRESSURE    PUMPS    ALARM
device.things.kubecon.io/research-pressure
...

NAME                                    AGE
module.deepsea.kubecon.io/research      14s
```

Simulator Setup

Let's install the simulator.

```
$ kubectl apply -f simulator/manifests.yaml
```

(a)

```
$ kubectl apply -f simulator/ingress.yaml
ingress.extensions/simulator-ingress created
$ kubectl -n deepsea get ingress
NAME                HOSTS   ADDRESS          PORTS   AGE
simulator-ingress   *       xx.xxx.xxx.xxx   80      75s
```

<wait ~5 minutes>

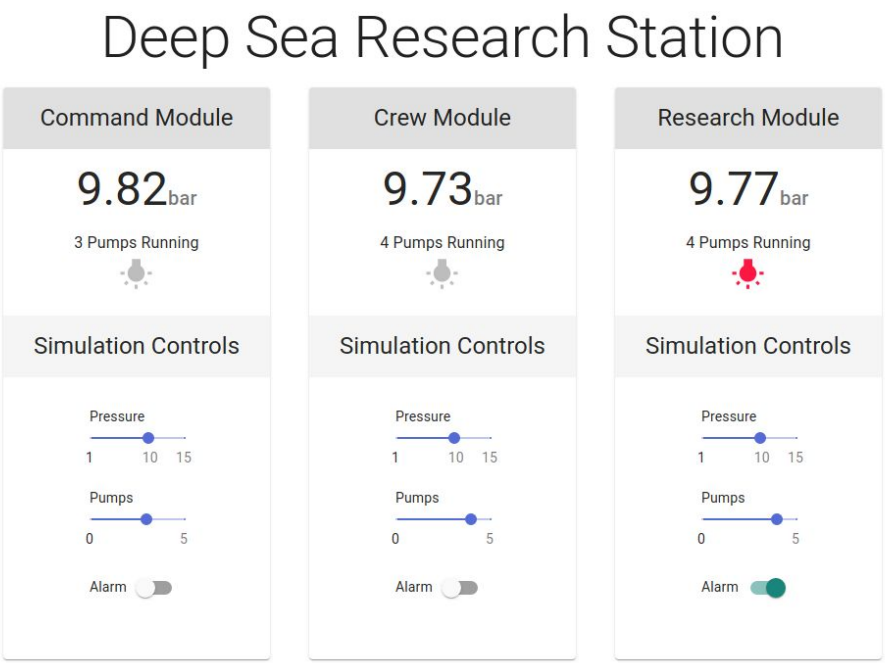
(b)

```
$ kubectl port-forward -n deepsea service/deepsea-simulator-service 8080:80
```

Visit http://<ADDRESS>/

Simulator Setup

http://<ADDRESS>:8080/




```
$ watch -n 0.1 kubectl get devices

NAME                  PRESSURE   PUMPS   ALARM
command-pressure      10219m
command-pump                     1
command-water
crew-pressure         10194m
crew-pump                        2
crew-water
research-pressure     10250e-3
research-pump                    1
research-water
```

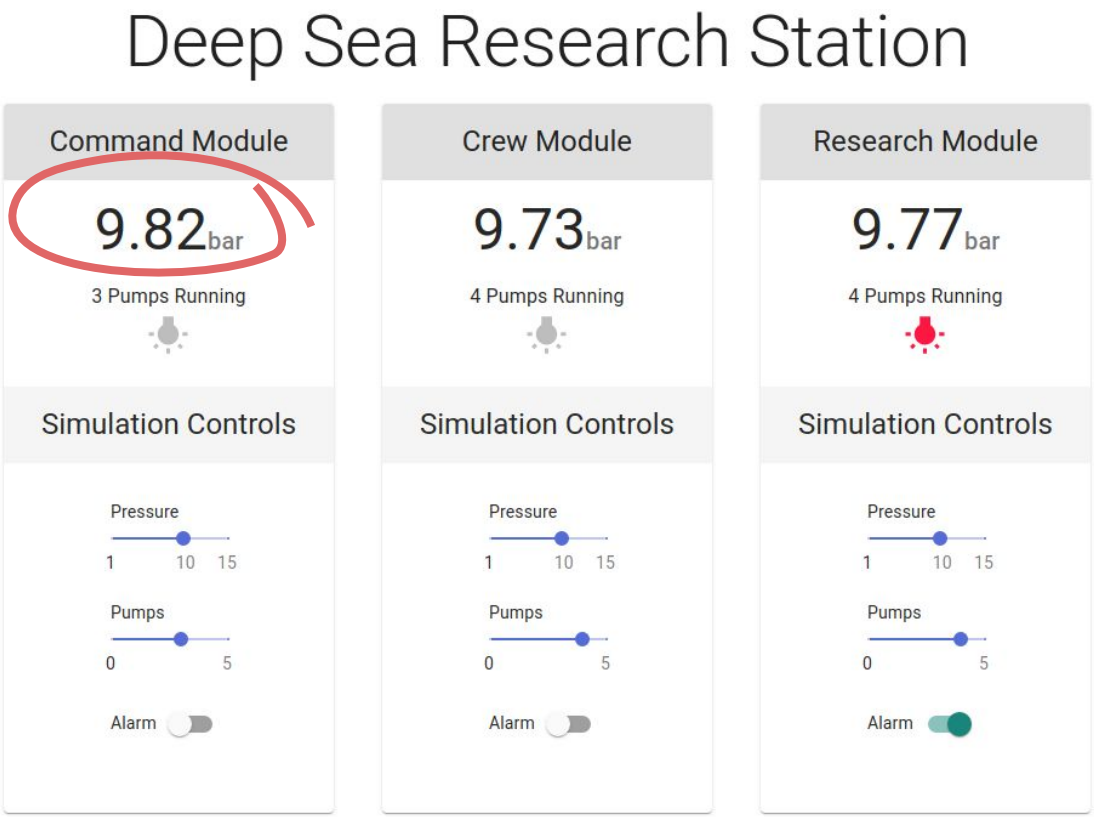
Simulation Controller

```
$ kubectl apply -f controllers/manifests.yaml
```

Pressure should start to change over time!

The “simulation” controller is changing the pressure.

Later we will implement a controller to manage the pressure changes.



CRD and conversion setup – manifests/

or 1.16 if you have it, to get defaulting

```
$ kubectl apply -f manifests/kubernetes-1.15-crds
customresourcedefinition.apiextensions.k8s.io/devices.things.kubecon.io created
customresourcedefinition.apiextensions.k8s.io/modules.deepsea.kubecon.io created

$ kubectl apply -f conversion/manifests.yaml
...

$ kubectl apply -f examples/research-module
module.deepsea.kubecon.io/research created
device.things.kubecon.io/research-water created
...
$ kubectl apply -f examples/crew-module
$ kubectl apply -f examples/command-module

$ kubectl get devices,modules
NAME                                          PRESSURE    PUMPS    ALARM
device.things.kubecon.io/research-pressure
...

NAME                                    AGE
module.deepsea.kubecon.io/research      14s
```

OpenAPI

Deepsea CRDs


```yaml
apiVersion: things.kubecon.io/v1alpha1
kind: Device
metadata:
  name: pump-1
spec:
  inputs:
  - name: activeCount
    value: "3"
    type: "Integer"
status:
  observedInputs: …
  outputs: …
```

Devices exist as v1 and v1alpha1.

```yaml
apiVersion: deepsea.kubecon.io/v1alpha1
kind: Module
spec:
  devices:
    pump: pump-1
    waterAlarm: alarm-1
    pressureSensor: sensor-1
```

Modules only exist as v1alpha1.

OpenAPI schema – manifests/kubernetes-1.16-crds/devices-crd.yaml


```yaml
apiVersion: things.kubecon.io/v1
kind: Device
metadata:
  name: pump-1
spec:
  inputs:
  - name: activeCount
    integer: 3
status:
  observedInputs: …
  outputs: …
```

```yaml
apiVersion: things.kubecon.io/v1alpha1
kind: Device
```

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: devices.things.kubecon.io
spec:
  versions:
  - name: v1
    schema:
      openAPIV3Schema:
        type: object
        properties:
          apiVersion:
            type: string
          kind:
            type: string
          metadata: ← mostly implicit
            type: object
          spec:
            properties:
              inputs:
                …
          status:
            …
    served: true
    storage: false
  - name: v1alpha1
    schema:
      openAPIV3Schema:
        …
```

Generating OpenAPI v3 schema – kuberbuilder controller-tools


```go
type Value struct {
    // name is the name of this input value.
    // +kubebuilder:validation:Required
    Name string `json:"name"`

    // value is the floating point input value.
    // +kubebuilder:validation:Required
    Value resource.Quantity `json:"value"`

    // +kubebuilder:default=Float
    // +kubebuilder:validation:Enum={"Integer","Float","Boolean"}
    Type Type `json:"type"`
}

type Type string

const (
     IntegerType Type = "Integer"
     BooleanType Type = "Boolean"
     FloatType   Type = "Float"
)
```

```yaml
type: object
properties:
  name:
    type: string
  value:
    type: string
  type:
    type: string
    enum:
    - Integer
    - Float
    - Boolean
    default: Float
required:
- name
- value
```

← only in apiextensions/v1

read CRD yamls from here: find matching Golang types: write result back:

```
$ controller-gen schemapatch:manifests=./manifests paths=./apis/... output:dir=./manifests
```

Or in `github.com/jpbetz/KoT`:

```
$ hack/update-crds.sh
```

Controlling kubebuilder's controller-tools

apis/things/v1/doc.go:

```
// +k8s:deepcopy-gen=package
// +groupName=things.kubecon.io
// +versionName=v1
// +kubebuilder:validation:Optional
package v1
```

generate DeepCopy methods
this is "things.kubecon.io", not "things"
this version v1
the default for fields is to be optional, not required

Documentation:
https://book.kubebuilder.io/reference/markers/crd-validation.html

Pruning

Opt-in for **v1beta1** CRDs

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: devices.things.kubecon.io
spec:
  group: things.kubecon.io
  preserveUnknownFields: false
```

Default for **v1** CRDs

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: devices.things.kubecon.io
spec:
  group: things.kubecon.io
```

Pruning


```yaml
apiVersion: things.kubecon.io/v1
kind: Device
metadata:
  name: pump-1
spec:
  inputs:
  - name: activeCount
    integer: 3
  unknown: 42
status:
  observedInputs: …
  outputs: …
```

```yaml
apiVersion: things.kubecon.io/v1alpha1
kind: Device
```

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: devices.things.kubecon.io
spec:
  versions:
  - name: v1
    schema:
      openAPIV3Schema:
        type: object
        properties:
          apiVersion:
            type: string
          kind:
            type: string
          metadata:
            type: object
          spec:
            properties:
              inputs:
                …
          status:
            …
    served: true
    storage: false
  - name: v1alpha1
    schema:
      openAPIV3Schema:
        …
```

This must be complete!

Generating OpenAPI v3 schema – kuberbuilder controller-tools


```go
type Value struct {
    // name is the name of this input value.
    // +kubebuilder:validation:Required
    Name string `json:"name"`

    // value is the floating point input value.
    // +kubebuilder:validation:Required
    Value resource.Quantity `json:"value"`

    // +kubebuilder:default=Float
    // +kubebuilder:validation:Enum={"Integer","Float","Boolean"}
    Type Type `json:"type"`
}

type Type string

const (
     IntegerType Type = "Integer"
     BooleanType Type = "Boolean"
     FloatType   Type = "Float"
)
```

```yaml
type: object
properties:
  name:
    type: string
  value:
    type: string
  type:
    type: string
    enum:
    - Integer
    - Float
    - Boolean
    default: Float
required:
- name
- value
```

read CRD yamls from here:

```
$ controller-gen schemapatch:manifests=./manifests
```

find matching Golang types:

```
paths=./apis/...
```

write result back:

```
output:dir=./manifests
```

Or in `github.com/jpbetz/KoT`:

```
$ hack/update-crds.sh
```

Pruning

Opt-in for **v1beta1** CRDs

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: devices.things.kubecon.io
spec:
  group: things.kubecon.io
  preserveUnknownFields: false
```

Default for **v1** CRDs

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: devices.things.kubecon.io
spec:
  group: things.kubecon.io
  # preserveUnknownFields:false is the default
```

Exercise 1

- Restrict `Value.Name`
  - to be non-empty.
  - to only consist of a-z,A-Z,0-9.

must be done for v1 and v1alpha1

Compare https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md#schema-object and https://book.kubebuilder.io/reference/markers/crd-validation.html

- Regenerate CRD schemas: `hack/update-crds.sh`
- Verify validation in cluster.

Stretch goals:

- Specify that in v1 devices only one of float, integer, boolean can be set.
- Try to restrict Value.Value to "1.0" and "0.0" for boolean type.

Multi-version

Devices in 2 versions

Different representations of the same object:

- There is no

  "object of version X in the cluster"

- All objects exist in all defined versions.

```
apiVersion: things/v1
kind: Device
metadata:
  name: pump-1
spec:
  inputs:
  - name: activeCount
    integer: 3
```

lossless conversion

```
/v1alpha1
  inputs:
  - name: activeCount
    value: "3"
    type: "Integer"
```

In etcd:

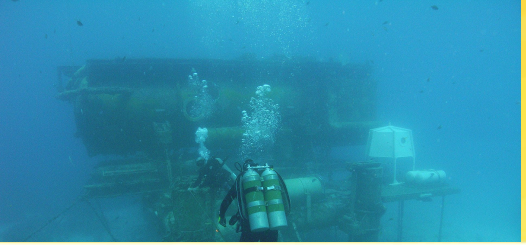
- Each object is versioned.
- Etcd can have mixture of versions.
- Storage version defines version of future writes.

→ github.com/kubernetes-sigs/kube-storage-version-migrator

Version history

Multiple versions in CRDs: since 1.12

Conversion via **webhooks**: beta since 1.15, GA since 1.16



v1alpha1 and v1 – apis/things/{v1,v1alpha1}

v1alpha1:

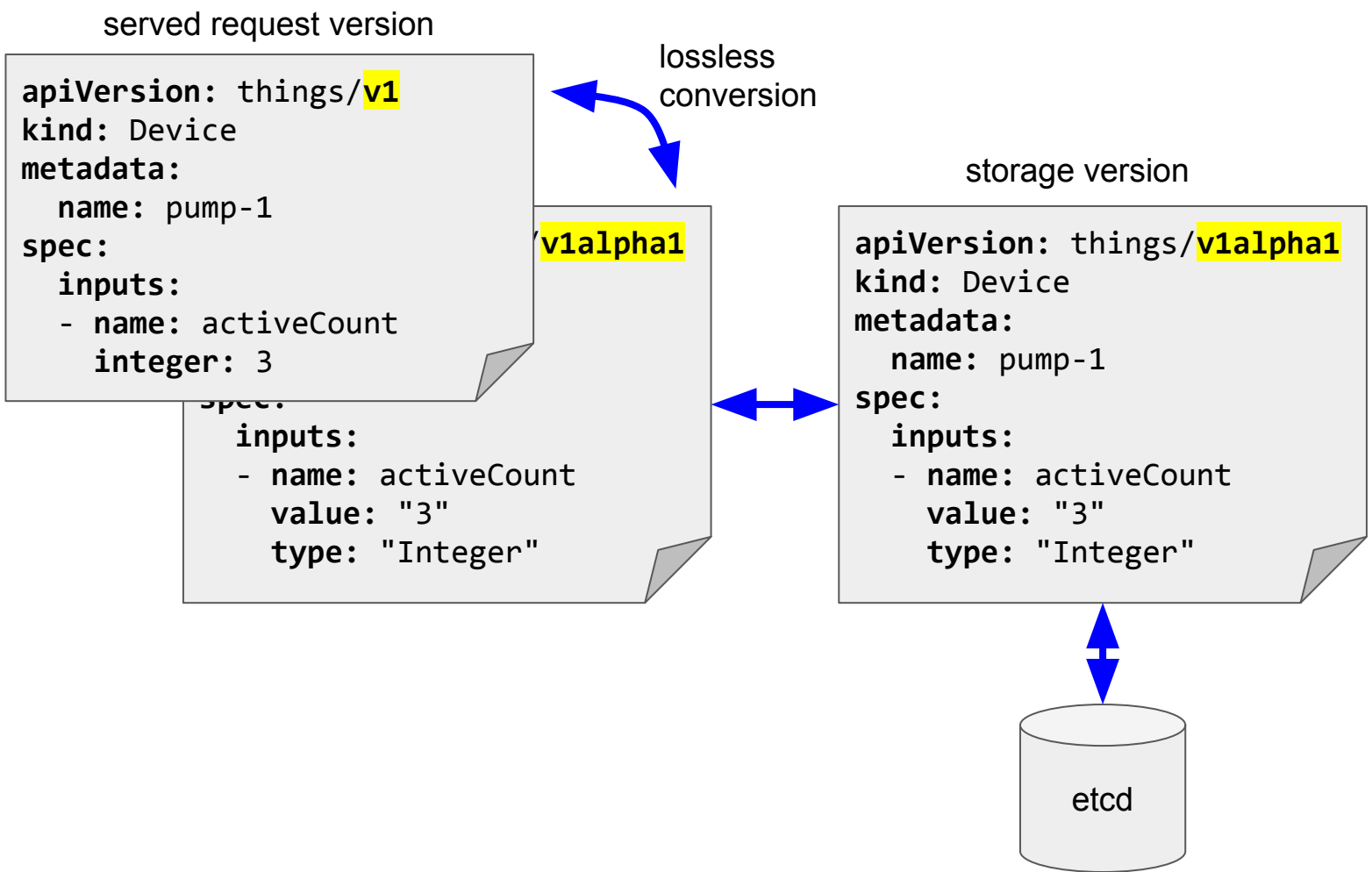
```
type Value struct {
    // name is the name of this input value.
    // +kubebuilder:validation:Required
    Name string `json:"name"`

    // value is the floating point input value.
    // +kubebuilder:validation:Required
    Value resource.Quantity `json:"value"`

    // +kubebuilder:default=Float
    // +kubebuilder:validation:Enum={"Integer","Float","Boolean"}
    Type Type `json:"type"`
}

type Type string

const (
     IntegerType Type = "Integer"
     BooleanType Type = "Boolean"
     FloatType   Type = "Float"
)
```

v1:

```
type Value struct {
    // name is the name of this input value.
    // +kubebuilder:validation:Required
    Name string `json:"name"`

    // float is a floating point input value.
    Float *resource.Quantity `json:"float,omitempty"`

    // boolean is a true or false value.
    Boolean *bool `json:"boolean,omitempty"`

    // integer is a integer value.
    Integer *int32 `json:"integer,omitempty"`
}
```

CRD Versioning


```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata: …
spec:
  …
  versions:
  - name: v1
    served: true
    storage: false
    schema: {"openAPIV3Schema": …}
  - name: v1alpha1
    served: true
    storage: true
    schema: {"openAPIV3Schema": …}
 conversion:
    strategy: Webhook
    webhook:
      clientConfig:
        caBundle: …
        service:
          namespace: things
          name: conversion-webhook
          path: /convert/v1beta1/devices
      conversionReviewVersions:
        - v1beta1
```



kubectl and versions


```
$ kubectl get devices --v=7
I1109 15:05:02.647993 GET https://127.0.0.1:52303/apis/things.kubecon.io/v1/namespaces/default/devices?limit=500
```

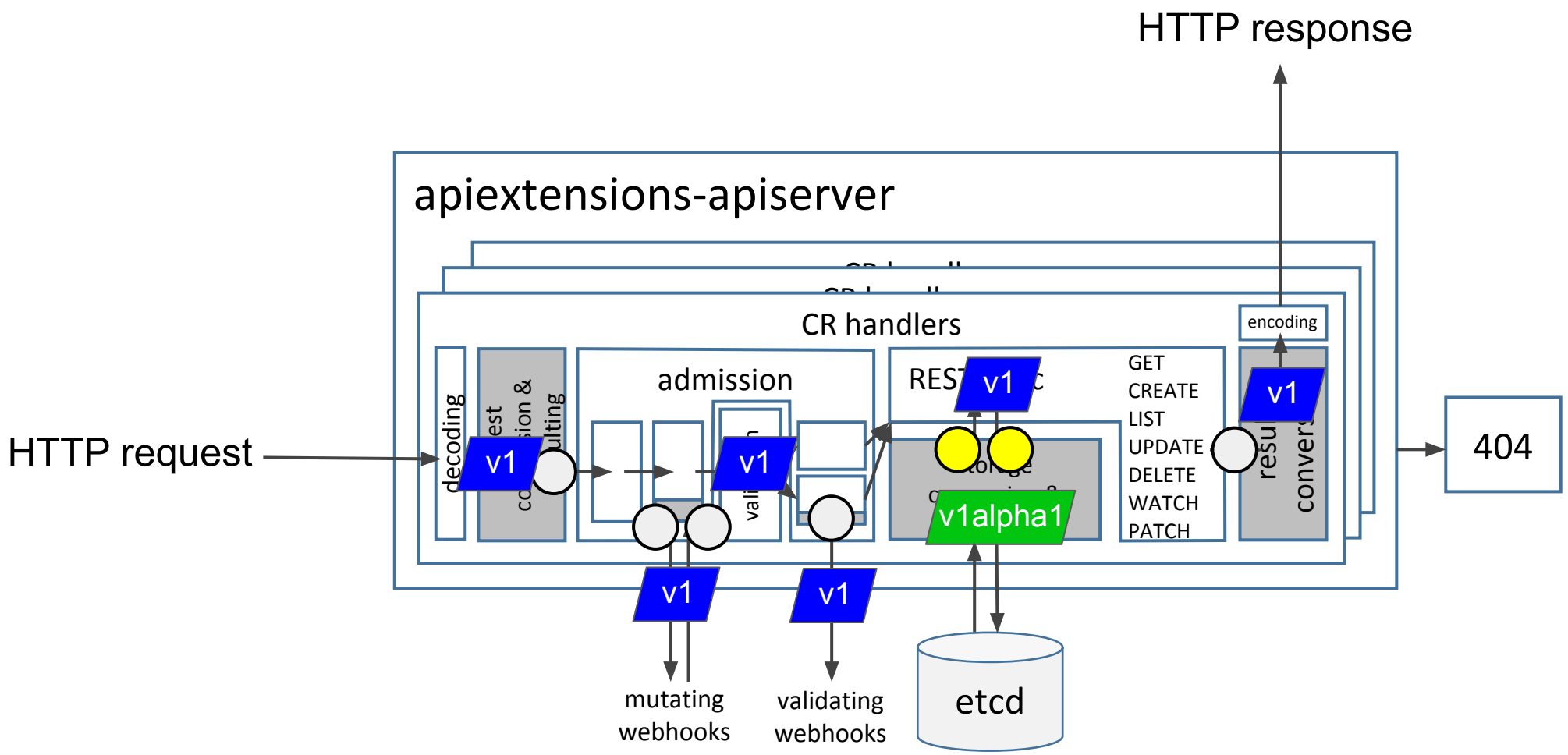
```
$ kubectl get devices.v1alpha1.things.kubecon.io --v=7
I1109 15:06:49.595604 GET https://127.0.0.1:52303/apis/things.kubecon.io/v1alpha1/namespaces/default/devices?limit=500
```

```
$ kubectl api-versions | grep things
things.kubecon.io/v1
things.kubecon.io/v1alpha1
```

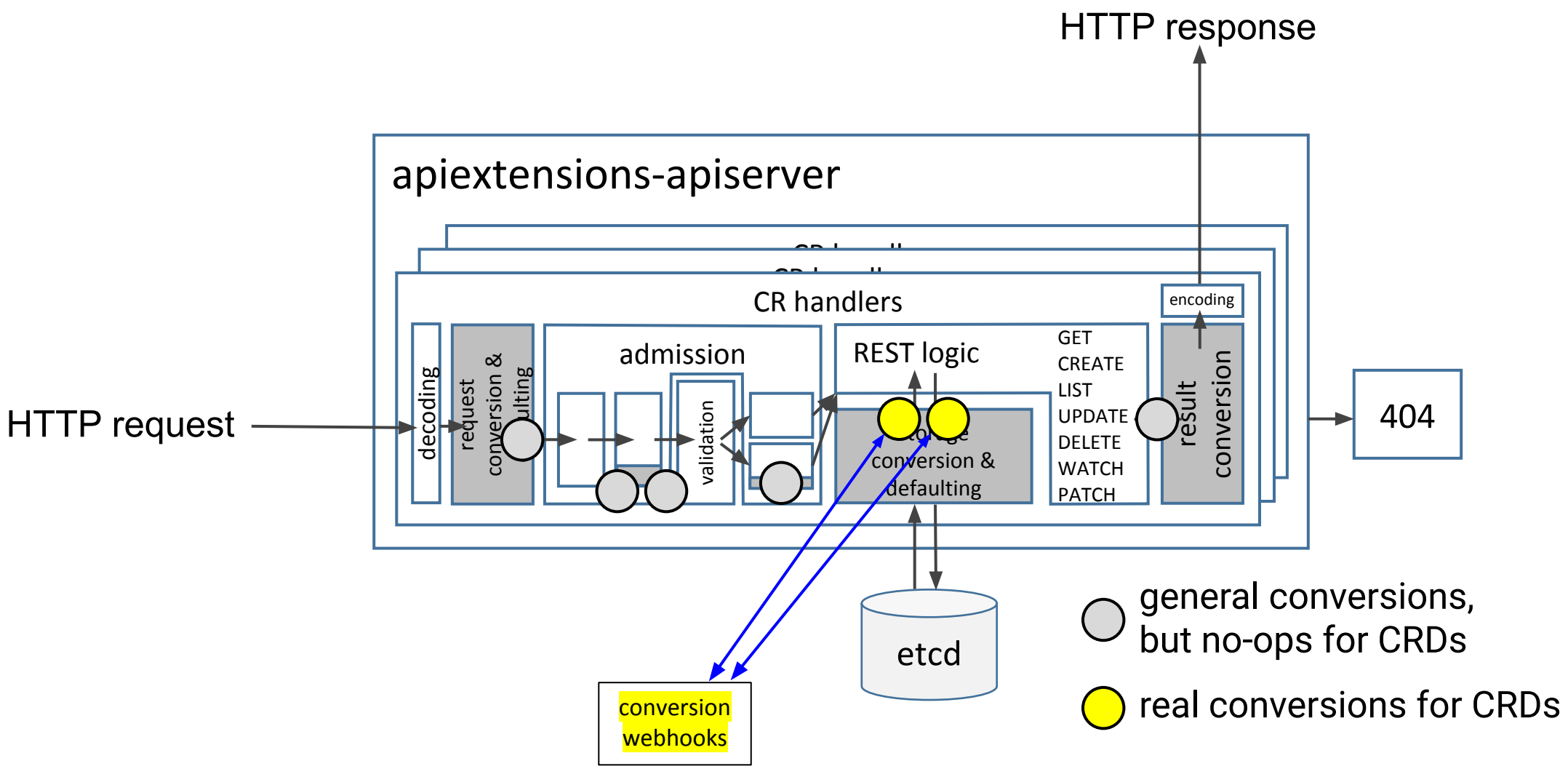
```
$ kubectl get --raw /apis/things.kubecon.io
{"kind":"APIGroup","apiVersion":"v1","name":"things.kubecon.io","versions":[{"groupVersion":"things.kubecon.io/v1","version":"v1"},{"groupVersion":"things.kubecon.io/v1alpha1","version":"v1alpha1"}],"preferredVersion":{"groupVersion":"things.kubecon.io/v1","version":"v1"}}
```

v2 > v1 > v2beta1 > v1beta2 > v1beta1 > v2alpha1 > v1alpha1 > foo > bar

CRD request pipeline



Conversion invocations

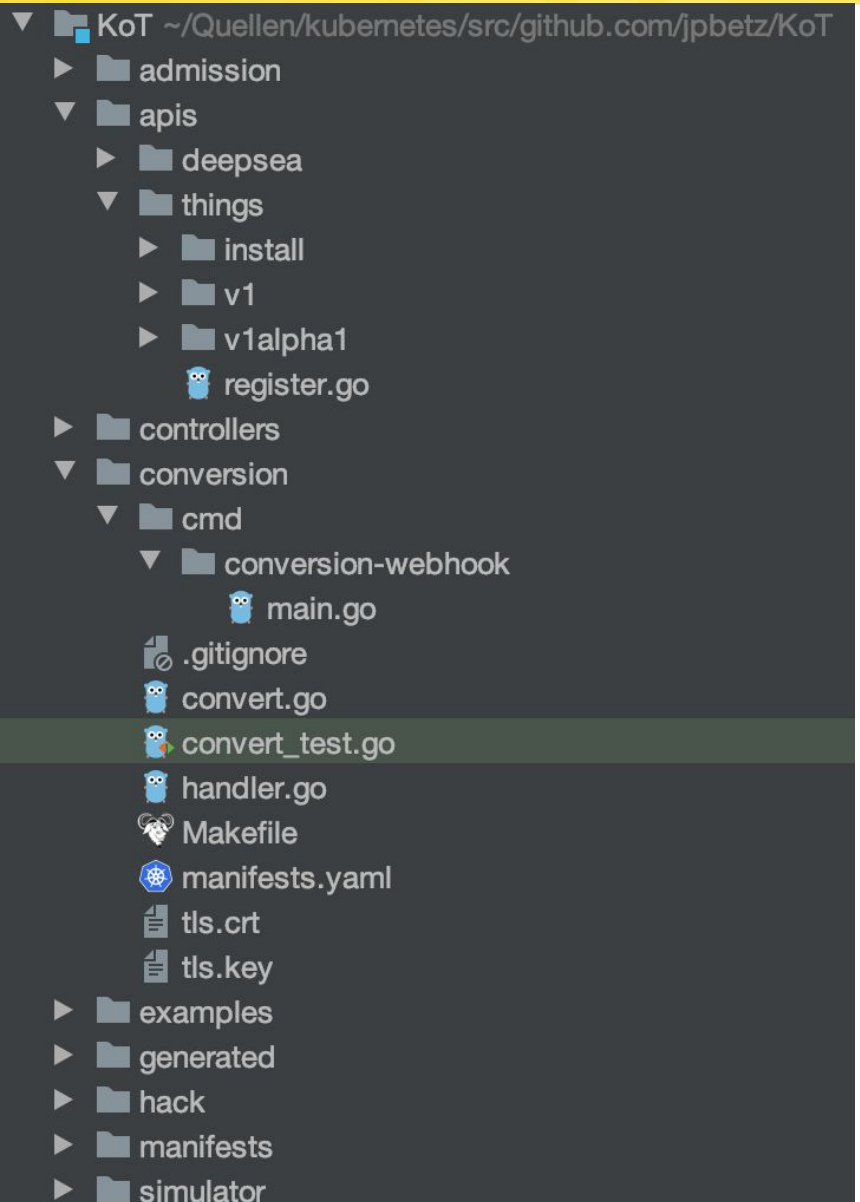


```
type ConversionReview struct {
    metav1.TypeMeta
    Request *ConversionRequest
    Response *ConversionResponse
}
```

```
type ConversionRequest struct {
    UID types.UID
    DesiredAPIVersion string
    Objects []runtime.RawExtension
}
```

```
type ConversionResponse struct {
    UID types.UID
    ConvertedObjects []runtime.RawExtension
    Result metav1.Status
}
```

Conversion webhook for devices



github.com/jpbetz/KoT/apis/things/v1
github.com/jpbetz/KoT/apis/things/v1alpha1

main.go – webhook main func

convert.go

```
func convert(in runtime.Object, apiVersion string) (runtime.Object, error)
func convertValueToV1alpha(in *v1.Value) *v1alpha1.Value
func convertValueToV1(in *v1alpha1.Value) *v1.Value
```

handler.go – serving `/convert/v1beta1/devices`

```
func Serve(w http.ResponseWriter, req *http.Request)
```

Deploy: `kubectl apply -f conversion/manifests.yaml`

Conversion – conversion/convert.go


```go
func convert(in runtime.Object, apiVersion string) (runtime.Object, error) {
    switch in := in.(type) {
    case *v1alpha1.Device:
        if apiVersion != v1.SchemeGroupVersion.String() {
            return nil, fmt.Errorf("cannot convert to %s", apiVersion)
        }

        out := &v1.Device{
            TypeMeta:   in.TypeMeta,
            ObjectMeta: in.ObjectMeta,
            Spec:       v1.DeviceSpec{
                Inputs: convertValuesToV1(in.Spec.Inputs),
            },
            Status: v1.DeviceStatus{
                ObservedInputs: convertValuesToV1(in.Status.ObservedInputs),
                Outputs:        convertValuesToV1(in.Status.Outputs),
            },
        }

        out.TypeMeta.APIVersion = apiVersion
        return out, nil

    case *v1.Device:
        …
}
```

lossless conversion

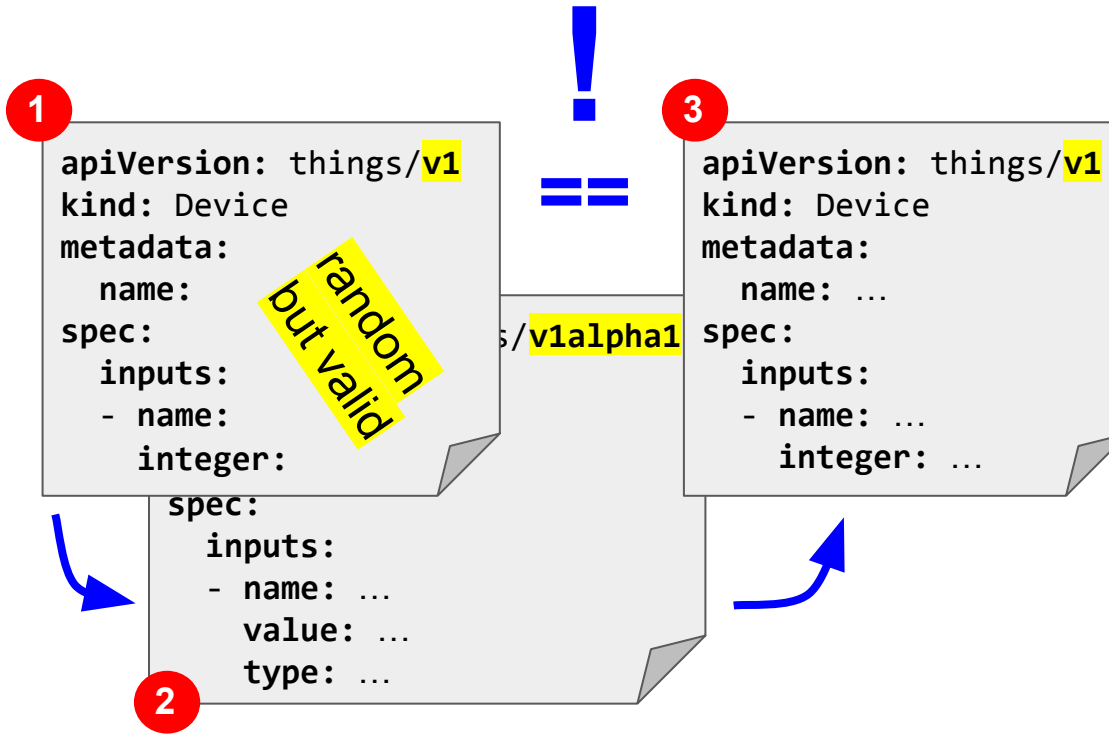
```yaml
apiVersion: things/v1
kind: Device
metadata:
  name: pump-1
spec:
  inputs:
  - name: activeCount
    integer: 3
```

```yaml
v1alpha1
spec:
  inputs:
  - name: activeCount
    value: "3"
    type: "Integer"
```

Roundtrip testing – conversion/convert_test.go


```go
func TestRoundTrip(t *testing.T) {
  f := apitestingfuzzer.FuzzerFor(
    apitestingfuzzer.MergeFuzzerFuncs(metafuzzer.Funcs, fuzzer.Funcs),
    rand.NewSource(rand.Int63()),
    codecs,
  )

  for _, kind := range []string{"Device"} {
    for _, version := range []string{"v1", "v1alpha1"} {
      gvk := schema.GroupVersionKind{things.GroupName, version, kind}
      for i := 0; i < 1000; i++ {
        x, _ := scheme.New(gvk)
        f.Fuzz(x)
        x.GetObjectKind().SetGroupVersionKind(gvk)

        otherVersion := thingsv1.SchemeGroupVersion
        if gvk.Version == "v1" {
          otherVersion = thingsv1alpha1.SchemeGroupVersion
        }

        other, _ := convert(x, otherVersion.String())
        back, _ := convert(other, gvk.GroupVersion().String())

        if !reflect.DeepEqual(x, back) {
          t.Errorf("roundtrip failed (a expected, b got): %s", diff.ObjectReflec
        }
      })
    }
  }
}
```

ObjectMeta fuzzer *Device fuzzer*




```
$ go test -mod vendor ./conversion -run TestConvert
--- FAIL: TestRoundTrip/things.kubecon.io.v1.Device (0.04s)
    roundtrip_test.go:74: roundtrip failed:
      object.Spec.Inputs:
        a: []v1.Value{}
        b: []v1.Value(nil)
```

Fuzzing – apis/things/fuzzer/fuzzer.go

1 2

Exhaustive[1] randomizing of objects which validate

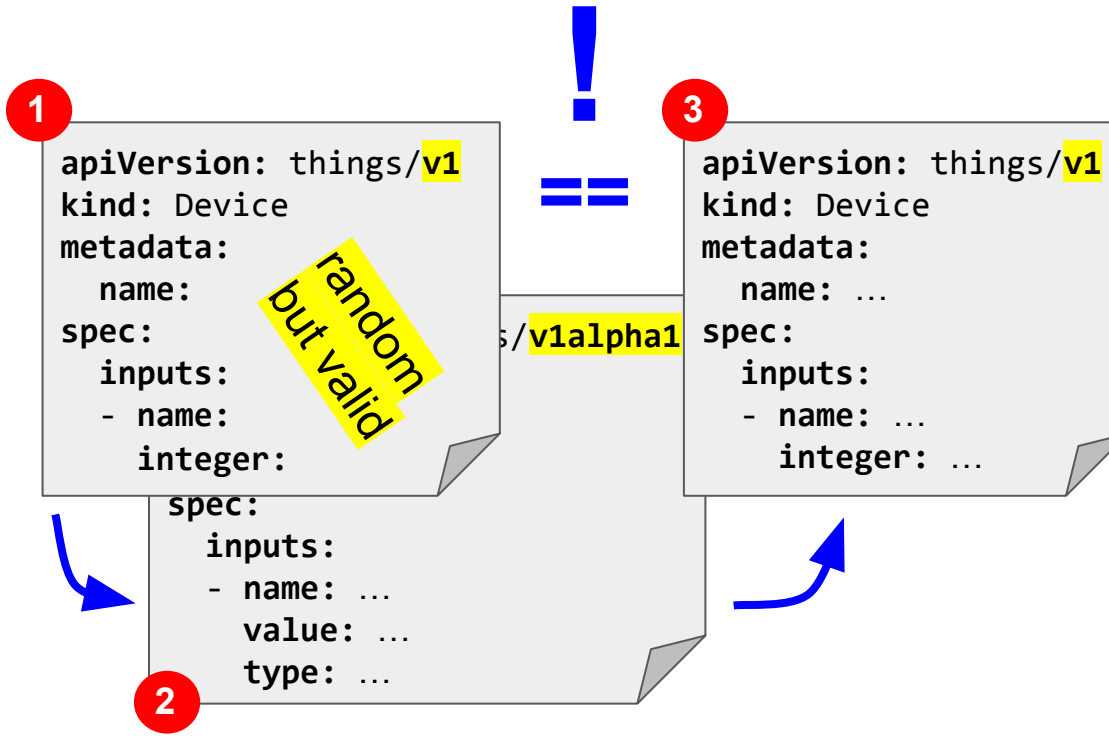
```
// Funcs returns the fuzzer functions for the things api group.
func Funcs(codecs runtimeserializer.CodecFactory) []interface{} {
	return []interface{}{
		func(v *v1.Value, c fuzz.Continue) {
			// c.FuzzNoCustom(v) – we could use this to pre-randomize all fields

			v.Name = c.RandString()

			switch c.RandUint64() % 3 {
			case 0:
				v.Integer = pointer.Int32Ptr(c.Rand.Int31())
			case 1:
				v.Boolean = pointer.BoolPtr(c.RandBool())
			case 2:
				v.Float = resource.NewMilliQuantity(c.Int63(), resource.DecimalSI)
			}
		},

		func(v *v1alpha1.Value, c fuzz.Continue) {
			…
		},
	}
}
```

1) with size limits or depth limits

Roundtrip testing – conversion/convert_test.go


```go
func TestRoundTrip(t *testing.T) {
  f := apitestingfuzzer.FuzzerFor(
    apitestingfuzzer.MergeFuzzerFuncs(metafuzzer.Funcs, fuzzer.Funcs),
    rand.NewSource(rand.Int63()),
    codecs,
  )

  for _, kind := range []string{"Device"} {
    for _, version := range []string{"v1", "v1alpha1"} {
      gvk := schema.GroupVersionKind{things.GroupName, version, kind}
      for i := 0; i < 1000; i++ {
        x, _ := scheme.New(gvk)
        f.Fuzz(x)
        x.GetObjectKind().SetGroupVersionKind(gvk)

        otherVersion := thingsv1.SchemeGroupVersion
        if gvk.Version == "v1" {
          otherVersion = thingsv1alpha1.SchemeGroupVersion
        }

        other, _ := convert(x, otherVersion.String())
        back, _ := convert(other, gvk.GroupVersion().String())

        if !reflect.DeepEqual(x, back) {
          t.Errorf("roundtrip failed (a expected, b got): %s", diff.ObjectReflec
        }
      })
    }
  }
}
```

ObjectMeta fuzzer *Device fuzzer*




```
$ go test -mod vendor ./conversion -run TestConvert
--- FAIL: TestRoundTrip/things.kubecon.io.v1.Device (0.04s)
    roundtrip_test.go:74: roundtrip failed:
      object.Spec.Inputs:
        a: []v1.Value{}
        b: []v1.Value(nil)
```

Exercise 2

1. Run `TestRoundTrip`

```
go test -mod vendor ./conversion -run TestConvert
 --- FAIL: TestRoundTrip/things.kubecon.io.v1.Device (0.04s)
     roundtrip_test.go:74: roundtrip failed:
         object.Spec.Inputs:
           a: []v1.Value{}
           b: []v1.Value(nil)
```

2. Fix conversion to roundtrip well.
3. Redeploy conversion webhook and verify roundtripping of `nil`.

Docker build and publish instructions at:
https://bit.ly/2JWsbxC

Admission

Deepsea CRDs

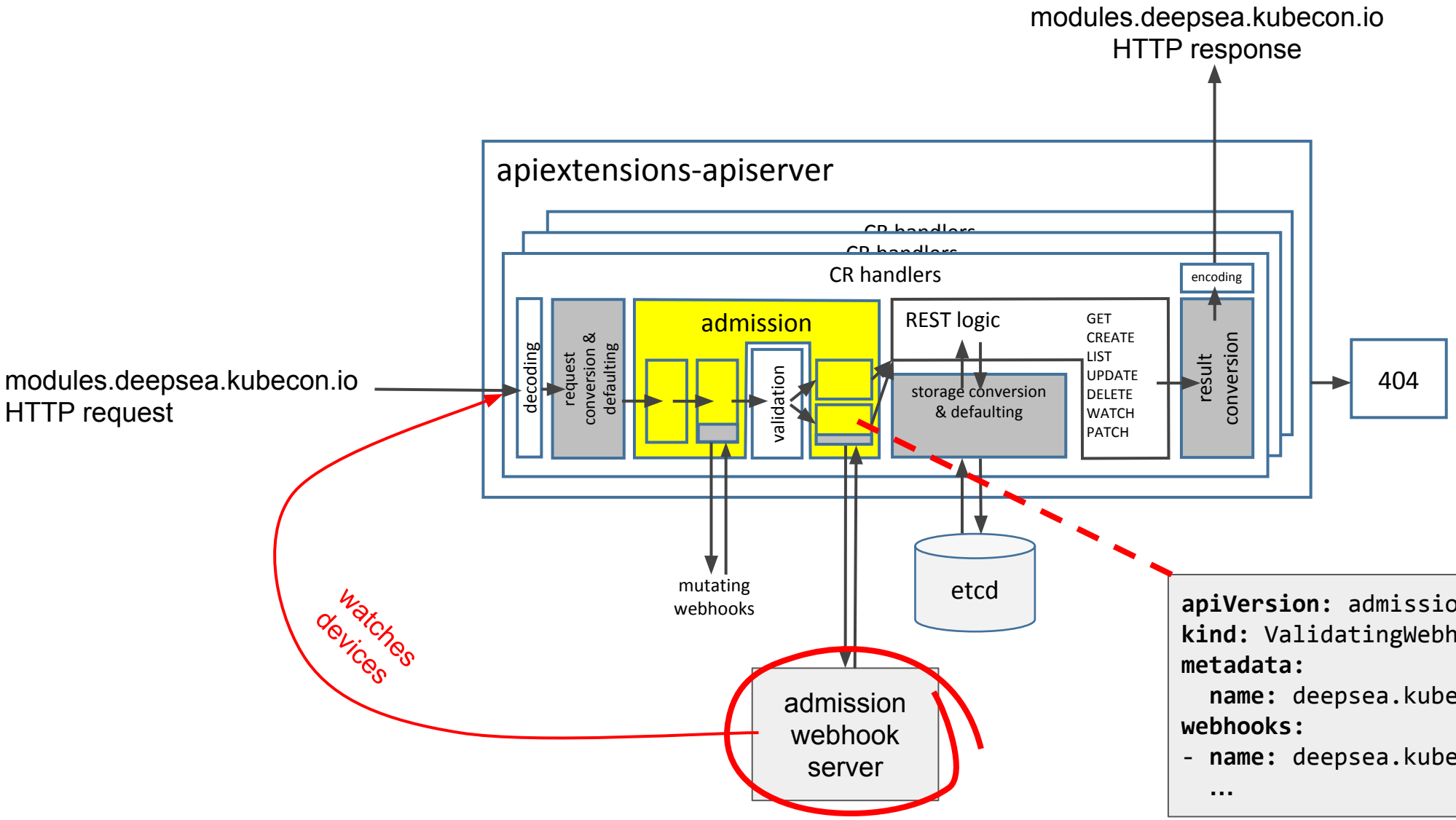

```yaml
apiVersion: things.kubecon.io/v1alpha1
kind: Device
metadata:
  name: pump-1
spec:
  inputs:
  - name: activeCount
    value: "3"
    type: "Integer"
status:
  observedInputs: …
  outputs: …
```

```yaml
apiVersion: deepsea.kubecon.io/v1alpha1
kind: Module
spec:
  devices:
    pump: pump-1
    waterAlarm: alarm-1
    pressureSensor: sensor-1
```

these should exist

Admission:
1. watch devices
2. check modules on
   - CREATE
   - UPDATE

Admission in the request pipeline




```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: deepsea.kubecon.io
webhooks:
- name: deepsea.kubecon.io
  ...
```

Registering a validating admission webhook


```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: deepsea.kubecon.io
webhooks:
- name: deepsea.kubecon.io
  failurePolicy: Fail
  admissionReviewVersions:
  - v1beta1
  rules:
  - apiGroups:
    - deepsea.kubecon.io
    apiVersions:
    - v1alpha1
    operations:
    - CREATE
    - UPDATE
    resources:
    - modules
  clientConfig:
    service:
      namespace: deepsea
      name: admission-webhook
      path: /validate/v1beta1/modules
    caBundle: ...
```

when webhook is down => error

we accept AdmissionReview in version v1beta1

we get called for this API group

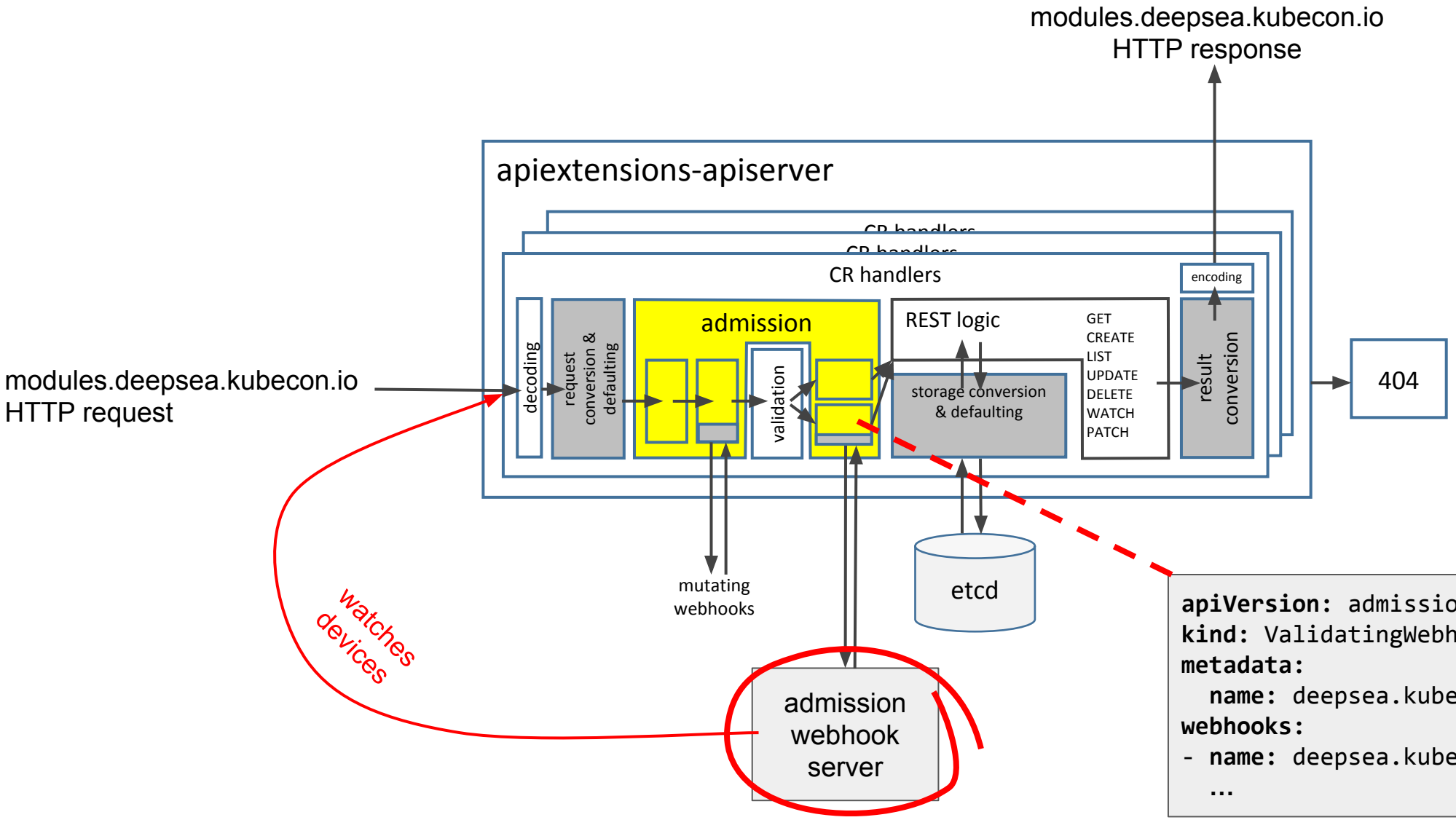
and this version

and these operations (implies also patch)

and this resource

corresponds

Admission in the request pipeline




```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: deepsea.kubecon.io
webhooks:
- name: deepsea.kubecon.io
  ...
```

admission.k8s.io/v1beta1 (we also have v1)

```
type AdmissionReview struct {
    metav1.TypeMeta
    Request *AdmissionRequest
    Response *AdmissionResponse
}
```

```
type AdmissionRequest struct {
    ...
}
```

```
type ConversionResponse struct {
    ...
}
```

Receiving an admission request


```
type AdmissionRequest struct {
    UID types.UID
    Kind metav1.GroupVersionKind
    Resource metav1.GroupVersionResource
    SubResource string
    RequestKind *metav1.GroupVersionKind
    RequestResource *metav1.GroupVersionResource
    Name string
    Namespace string
    Operation Operation
    UserInfo authenticationv1.UserInfo
    Object runtime.RawExtension
    OldObject runtime.RawExtension
    Options runtime.RawExtension
}
```

```yaml
apiVersion: deepsea.kubecon.io/v1alpha1
kind: Module
metadata:
  name: research
spec:
  devices:
    pump: pump-research
    waterAlarm: water-research
    pressureSensor: pressure-research
```

Sending an admission response

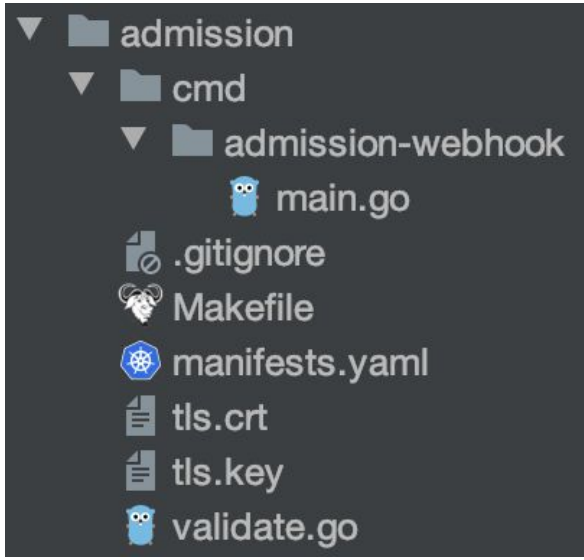

```
type AdmissionResponse struct {

    UID types.UID
    Allowed bool                    false for reject
    Result *metav1.Status           error message sent to the client
    Patch []byte  <------ for mutating admission webhooks
    PatchType *PatchType

}
```

Admission webhook for modules

main.go – webhook main func

validate.go – handler for serving `/validate/v1beta1/modules`

```
func ModuleValidation(informers informers.SharedInformerFactory) func(http.ResponseWriter, *http.Request) {
  devicesInformer := informers.Things().V1alpha1().Devices().Informer()
  devicesLister := informers.Things().V1alpha1().Devices().Lister()

  return func(w http.ResponseWriter, req *http.Request) {
    ... decode AdmissionReview from req ...
    ... decode review.Request.Object ...

    switch module := review.Request.Object.Object.(type) {
    case *deepseev1alpha1.Module:
      ... verify module ...
    default:
      review.Response.Result = &metav1.Status{
        Message: fmt.Sprintf("unexpected type %T", review.Request.Object.Object),
        Status:  metav1.StatusFailure,
      }
    }

    responsewriters.WriteObjectNegotiated(
      codecs, negotiation.DefaultEndpointRestrictions, gvk.GroupVersion(), w, req, http.StatusOK, review,
    )
  }
}
```



Pressure Controller

Deep Sea Research Station

Command Module	Research Module	Crew Module
9.84bar	9.95bar	9.95bar
3 Pumps Running	3 Pumps Running	3 Pumps Running
Simulation Controls	**Simulation Controls**	**Simulation Controls**
Pressure 1 10 15	Pressure 1 10 15	Pressure 1 10 15
Pumps 0 3 6	Pumps 0 3 6	Pumps 0 3 6
Alarm	Alarm	Alarm

Exercise 3

(1)

Local Docker Build/Push

```
$ make build-admission

$ make push-admission
```

Cloud Docker Build/Push

```
$ gcloud builds submit --config
hack/cloudbuild/admission.yaml

(answer with ‘y’ if asked to enable
cloudbuild on the project)

make cloudbuild-admission-set-image
```

(2)

```
$ kubectl apply -f admission/manifests.yaml
```

Exercise 3

WARNING

At the end of this exercise, we will push docker images. They're about 50mb each.

If the wi-fi doesn't cooperate, **don't worry**, you can follow up on this last step at your convenience after the session.

Exercise 3

Admission Webhook:

- implement validation logic, checking that devices referenced in `Module.Spec.Devices` all exist as object (hint: use the lister)
- Test after deploying by creating an invalid module.

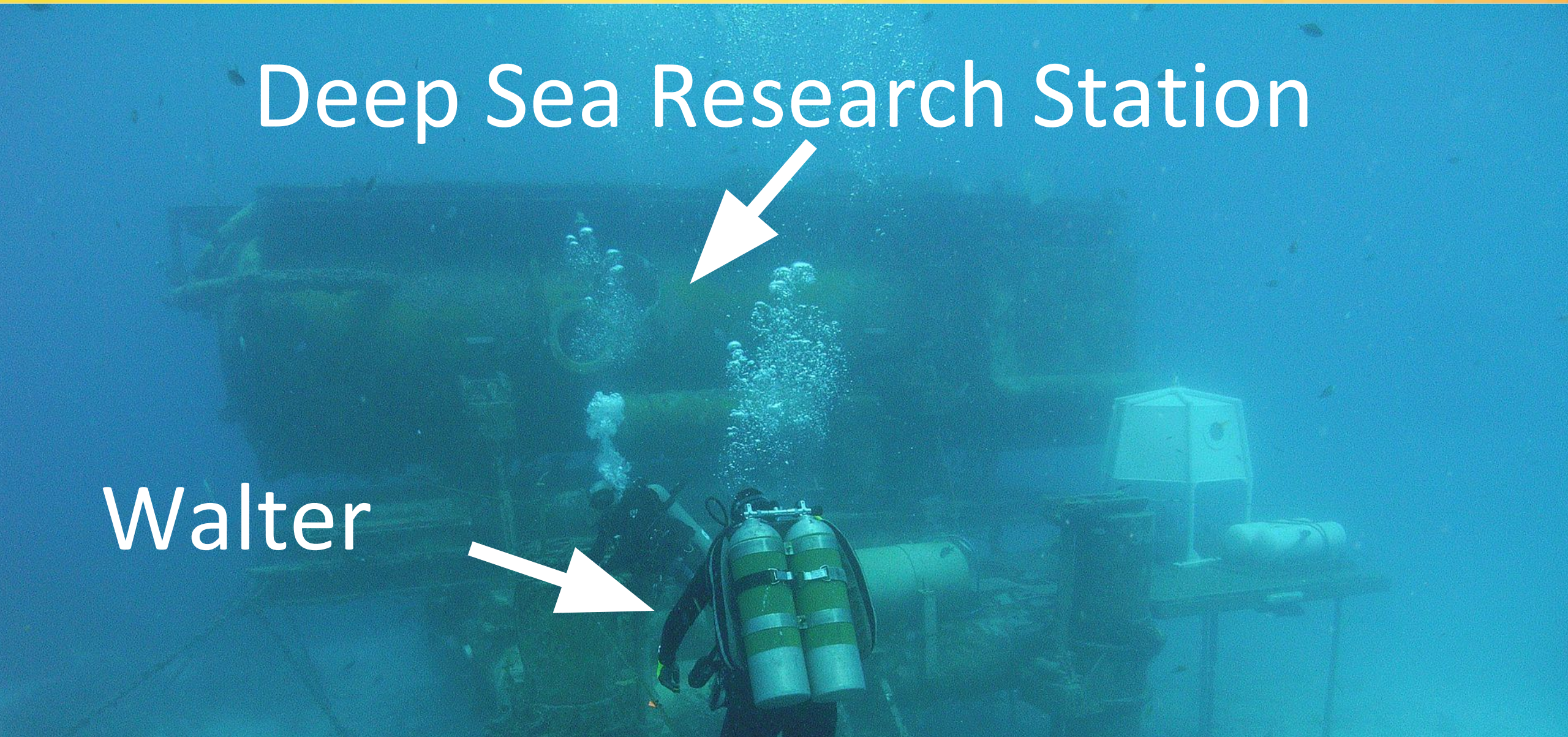
And Finally, Let's complete our Controller:

- Implement controller logic to activate pumps and maintain pressure
- See calculateActivePumps() in controllers/pressurecontroller.go
- Test with: go test -mod vendor ./controllers

Docker build and publish instructions at: https://bit.ly/2JWsbxC

Recap

Our Task Today



What is an advanced CRD?


```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: deepsea.kubecon.io
webhooks:
- name: deepsea.kubecon.io
  sideEffects: None
  failurePolicy: Fail
  ...
```

admission

```
validation:
  openAPIV3Schema:
    type: object
    properties:
      spec:
        type: object
        ...
```

OpenAPI schemas

```
conversion:
  strategy: Webhook
  webhook:
    clientConfig:
      caBundle: LS0tLS1CRUdJTiBDRV...
      service:
        namespace: things
        name: conversion-webhook
        path: /convert/v1/devices
```

conversion

```
crd.yaml:
apiVersion: apiextensions/v1
kind: CustomResourceDefinition
spec:
  group: …
  names: ...
```

code generation

```
controller-tools controller-gen:
  types.go => Go client + OpenAPI schema
```

```
validation:
  openAPIV3Schema:
    …
    replicas:
      type: integer
      minimum: 0
    type:
      type: string
      enum: ["Foo","Bar"]
```

value validation

```
versions:
- name: v1alpha1
  strorage: true
  schema:
    openAPIV3Schema:
      …
- name: v1
  storage: false
```

versioning

```
spec:
  preserveUnknownFields: false
```

pruning

To the next level

		beta since	GA since
(Subresources)	YAML	1.11	1.16
Validation	YAML + OpenAPI	1.9	1.16
Validating admission webhooks		1.9	1.16
Mutating admission webhooks		1.9	1.16
Pruning	YAML + OpenAPI	1.15	1.16
Defaulting	YAML + OpenAPI	1.16	1.17
Multi-version	YAML	1.12	1.16
Conversion webhooks		1.15	1.16

Backup Slides

```
func Serve(w http.ResponseWriter, req *http.Request) {
    // read body
    body, err := ioutil.ReadAll(req.Body)
    if err != nil {
        responsewriters.InternalError(w, req, fmt.Errorf("failed to read
body: %v", err))
        return
    }
```

Simulator Setup