
KubeCon
CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022
TiKV
TiKV
HELM

# Beyond Kubebuilder: Tales from the Kubernetes Factory Floor

KubeCon CloudNativeCon North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022

KubeCon CloudNativeCon North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

**October 24-28, 2021**

**Jay Pipes**
Bob the Builder
*AWS*

**Amine Hilaly**
Spud the Scarecrow
*AWS*

# Background



- This is a tour of a Kubernetes controllers factory
- Sharing experience of years running and generating k8s controllers
- You'll learn what's needed if you want to build your own factory!

# Background



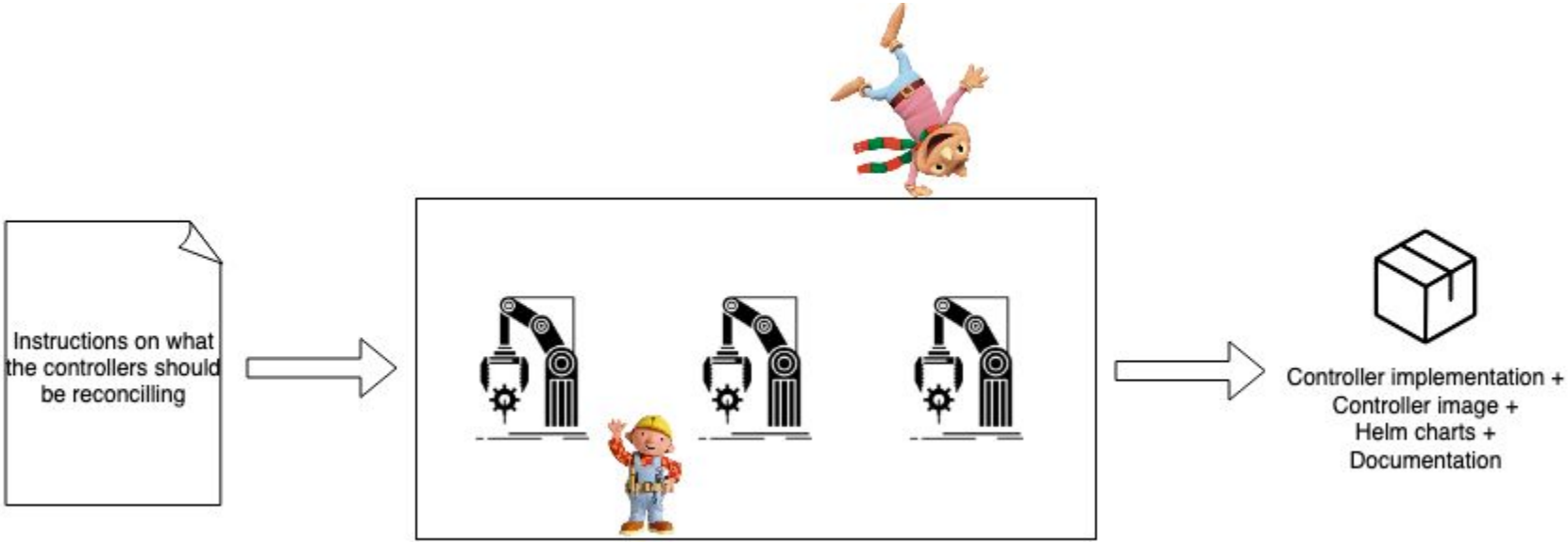
Brace yourself, like in any other factory you need a..

Hard Hat!

# A factory of what?



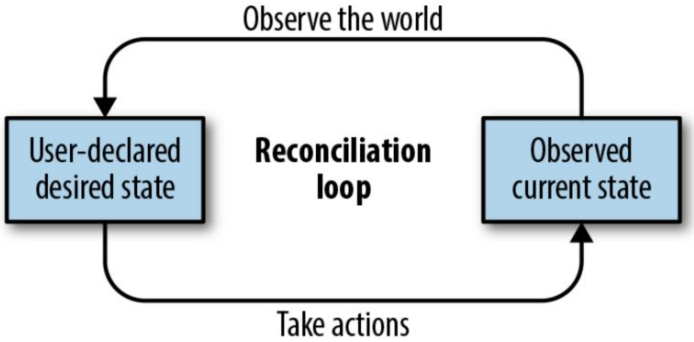
This is a factory that builds Kubernetes controllers…



# First, some background…

# So what is a controller?






```
for {
  desired := getDesiredState()
  current := getCurrentState()
  makeChanges(desired, current)
}
```

# Kubernetes-native controllers



Kubernetes is a smelting pot of controllers, each reconciling different Kinds of resources

- Deployment
- Endpoint[Slice]
- Pod
- Job
- Node
- … even Namespaces

# Custom Controllers



Then you can build your own controllers…

- Controlling your Home smart bulbs
- Ordering Pizza (shout out to @mhausenblas)
- Managing cloud resources
  - AWS S3, Azure blob storage, Google Cloud Storage, etc…

Even managing Kubecon CFP (see the tutorial at 4.30pm how to build your own controllers…)

# Your toolbox



- Kubebuilder
- controller-gen/controller-tools
- operator-framework
- controller-runtime
- Or for the bravest (old fashioned) you can hand-build your own with client-go Informers, Listers, SharedInformer, WorkQueue etc…

# What do developers still need to do?



Even though there are tools and libraries that:

- Generate Go types and CRDs
- Handle CRD versioning
- Generate Go clients
- Custom kubectl get tables etc…
- Handle logging
- Simple reconciliation calls
- Handle leader election
- Rate limiting
- Generate Webhooks

Developers still need to handle:

- Write the reconciliation logic
- Manage and handle conditions
- Write validation/mutation webhooks logic
- Write unit and e2e tests
- Fight with prow bot
- Maintain controller images
- Maintain helm charts and documentation
- Fight with prow bot
- /retest

KubeCon CloudNativeCon North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022

# Scaling controller development

# 200+ Controllers problem



- One day you will have to manage hundreds of controllers
- We could not manually craft them
- 200 Services at AWS = 200 Controllers to write
- We needed a controller factory!

# 200+ Controllers problem



- APIs aren’t static. They evolve over time.
- New services coming out all the time.
- New resources within an existing API.
- New fields in an existing resource.

# A tour of our factory

# The controller factory tour



- Inputs: API model, generate.yaml config file
- Output: the whole darn controller

# The controller factory tour



## Step 1: Finding resources (CRDs)

First thing we do is detect the resources that the controllers manage.

- Resource name often mentioned in the API operation name
- E.g `CreateTable` means we can have a `Table` resource
- For each operation that starts with `Create` we infer that the rest of the operation name is the resource Name

```
100      "CreateTable":{
101        "name":"CreateTable",
102        "http":{
103          "method":"POST",
104          "requestUri":"/"
105        },
106        "input":{"shape":"CreateTableInput"},
107        "output":{"shape":"CreateTableOutput"},
108        "errors":[
109          {"shape":"ResourceInUseException"},
110          {"shape":"LimitExceededException"},
111          {"shape":"InternalServerError"}
112        ],
113        "endpointdiscovery":{
```

# The controller factory tour

KubeCon CloudNativeCon North America 2022

## Step 2: Finding the fields

We find fields in the Create operation input and output :)

- Fields mentioned in the input and output goes to Spec
- Fields mentioned only in output goes to Status

```
1489    "CreateTableInput":{
1490      "type":"structure",
1491      "required":[
1492        "AttributeDefinitions",
1493        "TableName",
1494        "KeySchema"
1495      ],
1496      "members":{
1497        "AttributeDefinitions":{"shape":"AttributeDefinitions"},
1498        "TableName":{"shape":"TableName"},
1499        "KeySchema":{"shape":"KeySchema"},
1500        "LocalSecondaryIndexes":{"shape":"LocalSecondaryIndexList"},
```

```
1510    "CreateTableOutput":{
1511      "type":"structure",
1512      "members":{
1513        "TableDescription":{"shape":"TableDescription"}
1514      }
1515    },
```

# The controller factory tour

KubeCon CloudNativeCon North America 2022

## Step 3: Writing CRD Go types into apis/v1alpha1

```
14  // Code generated by ack-generate. DO NOT EDIT.
15
16  package v1alpha1
17
18  import (
19      ackv1alpha1 "github.com/aws-controllers-k8s/runtime/apis/core/v1alpha1"
20      metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
21  )
22
23  // TableSpec defines the desired state of Table.
24  type TableSpec struct {…
158 }
159
160 // TableStatus defines the observed state of Table
161 type TableStatus struct {…
251 }
252
253 // Table is the Schema for the Tables API
254 // +kubebuilder:object:root=true
255 // +kubebuilder:subresource:status
256 type Table struct {
257     metav1.TypeMeta   `json:",inline"`
258     metav1.ObjectMeta `json:"metadata,omitempty"`
259     Spec              TableSpec   `json:"spec,omitempty"`
260     Status            TableStatus `json:"status,omitempty"`
261 }
262
263 // TableList contains a list of Table
264 // +kubebuilder:object:root=true
265 type TableList struct {
266     metav1.TypeMeta `json:",inline"`
267     metav1.ListMeta `json:"metadata,omitempty"`
268     Items           []Table `json:"items"`
269 }
```

# The controller factory tour



## Step 4: A quick round of controller-gen ft. kubebuilder :)

During the resource file generation we make sure to user kubebuilder comment markers to specify Printer Columns, Validation patterns, role scoping etc…

We leverage controller-gen to generate the complimentary files like zz_deepcopy.go, the CRD and Role/ClusterRole yaml files

# The controller factory tour



# The controller factory tour



**Here's where the fun starts!**

What's needed in controller implementation?

- Transform API responses to Custom Resource objects
- Transform Custom Resource objects to API requests
- Compute the difference between two Custom Resource objects
- Transform API errors into conditions

# The controller factory tour

KubeCon CloudNativeCon North America 2022

```go
// sdkFind returns SDK-specific information about a supplied resource
func (rm *resourceManager) sdkFind(
        ctx context.Context,
        r *resource,
) (latest *resource, err error) {
        rlog := ackrtlog.FromContext(ctx)
        exit := rlog.Trace("rm.sdkFind")
        defer func() {
                exit(err)
        }()
        // If any required fields in the input shape are missing, AWS resource is
        // not created yet. Return NotFound here to indicate to callers that the
        // resource isn't yet created.
        if rm.requiredFieldsMissingFromReadOneInput(r) {
                return nil, ackerr.NotFound
        }

        input, err := rm.newDescribeRequestPayload(r)
        if err != nil {
                return nil, err
        }

        var resp *svcsdk.GetFunctionOutput
        resp, err = rm.sdkapi.GetFunctionWithContext(ctx, input)
        rm.metrics.RecordAPICall("READ_ONE", "GetFunction", err)
        if err != nil {
                if awsErr, ok := ackerr.AWSError(err); ok && awsErr.Code() == "ResourceNotFoundException"
                        return nil, ackerr.NotFound
                }
                return nil, err
        }

        // Merge in the information we read from the API call above to the copy of
        // the original Kubernetes object we passed to the function
        ko := r.ko.DeepCopy()

        if resp.Tags != nil {
                f3 := map[string]*string{}
```

```go
        if resp.Configuration.MemorySize != nil {
                ko.Spec.MemorySize = resp.Configuration.MemorySize
        } else {
                ko.Spec.MemorySize = nil
        }
        if resp.Configuration.PackageType != nil {
                ko.Spec.PackageType = resp.Configuration.PackageType
        } else {
                ko.Spec.PackageType = nil
        }
        if resp.Configuration.RevisionId != nil {
                ko.Status.RevisionID = resp.Configuration.RevisionId
        } else {
                ko.Status.RevisionID = nil
        }
        if resp.Configuration.Role != nil {
                ko.Spec.Role = resp.Configuration.Role
        } else {
                ko.Spec.Role = nil
        }
        if resp.Configuration.Runtime != nil {
                ko.Spec.Runtime = resp.Configuration.Runtime
        } else {
                ko.Spec.Runtime = nil
        }
        if resp.Configuration.SigningJobArn != nil {
                ko.Status.SigningJobARN = resp.Configuration.SigningJobArn
        } else {
                ko.Status.SigningJobARN = nil
        }
        if resp.Configuration.SigningProfileVersionArn != nil {
                ko.Status.SigningProfileVersionARN = resp.Configuration.SigningProfileVersionArn
        } else {
                ko.Status.SigningProfileVersionARN = nil
        }
        if resp.Configuration.State != nil {
                ko.Status.State = resp.Configuration.State
        } else {
```

# The controller factory tour



```
33  func newResourceDelta(
34          a *resource,
35          b *resource,
36  ) *ackcompare.Delta {
37          delta := ackcompare.NewDelta()
38          if (a == nil && b != nil) ||
39                  (a != nil && b == nil) {
40                  delta.Add("", a, b)
41                  return delta
42          }
43          customPreCompare(delta, a, b)
44
45          if !ackcompare.SliceStringPEqual(a.ko.Spec.Architectures, b.ko.Spec.Architectures) {
46                  delta.Add("Spec.Architectures", a.ko.Spec.Architectures, b.ko.Spec.Architectures)
47          }
48          if ackcompare.HasNilDifference(a.ko.Spec.CodeSigningConfigARN, b.ko.Spec.CodeSigningConfigARN) {
49                  delta.Add("Spec.CodeSigningConfigARN", a.ko.Spec.CodeSigningConfigARN, b.ko.Spec.CodeSigningConfigARN)
50          } else if a.ko.Spec.CodeSigningConfigARN != nil && b.ko.Spec.CodeSigningConfigARN != nil {
51                  if *a.ko.Spec.CodeSigningConfigARN != *b.ko.Spec.CodeSigningConfigARN {
52                          delta.Add("Spec.CodeSigningConfigARN", a.ko.Spec.CodeSigningConfigARN, b.ko.Spec.CodeSigningConfigARN)
53                  }
54          }
55          if ackcompare.HasNilDifference(a.ko.Spec.DeadLetterConfig, b.ko.Spec.DeadLetterConfig) {
56                  delta.Add("Spec.DeadLetterConfig", a.ko.Spec.DeadLetterConfig, b.ko.Spec.DeadLetterConfig)
57          } else if a.ko.Spec.DeadLetterConfig != nil && b.ko.Spec.DeadLetterConfig != nil {
58                  if ackcompare.HasNilDifference(a.ko.Spec.DeadLetterConfig.TargetARN, b.ko.Spec.DeadLetterConfig.TargetARN) {
59                          delta.Add("Spec.DeadLetterConfig.TargetARN", a.ko.Spec.DeadLetterConfig.TargetARN, b.ko.Spec.DeadLetterConfig.TargetARN)
60                  } else if a.ko.Spec.DeadLetterConfig.TargetARN != nil && b.ko.Spec.DeadLetterConfig.TargetARN != nil {
61                          if *a.ko.Spec.DeadLetterConfig.TargetARN != *b.ko.Spec.DeadLetterConfig.TargetARN {
62                                  delta.Add("Spec.DeadLetterConfig.TargetARN", a.ko.Spec.DeadLetterConfig.TargetARN, b.ko.Spec.DeadLetterConfig.TargetARN)
63                          }
64                  }
65          }
66          if ackcompare.HasNilDifference(a.ko.Spec.Description, b.ko.Spec.Description) {
67                  delta.Add("Spec.Description", a.ko.Spec.Description, b.ko.Spec.Description)
68          } else if a.ko.Spec.Description != nil && b.ko.Spec.Description != nil {
69                  if *a.ko.Spec.Description != *b.ko.Spec.Description {
70                          delta.Add("Spec.Description", a.ko.Spec.Description, b.ko.Spec.Description)
```

# The controller factory tour



```
46  // ManagerFor returns a resource manager object that can manage resources for a
47  // supplied AWS account
48  func (f *resourceManagerFactory) ManagerFor(
49          cfg ackcfg.Config,
50          log logr.Logger,
51          metrics *ackmetrics.Metrics,
52          rr acktypes.Reconciler,
53          sess *session.Session,
54          id ackv1alpha1.AWSAccountID,
55          region ackv1alpha1.AWSRegion,
56  ) (acktypes.AWSResourceManager, error) {
57          rmId := fmt.Sprintf("%s/%s", id, region)
58          f.RLock()
59          rm, found := f.rmCache[rmId]
60          f.RUnlock()
61
62          if found {
63                  return rm, nil
64          }
65
66          f.Lock()
67          defer f.Unlock()
68
69          rm, err := newResourceManager(cfg, log, metrics, rr, sess, id, region)
70          if err != nil {
71                  return nil, err
72          }
73          f.rmCache[rmId] = rm
74          return rm, nil
75  }
76
```

# The controller factory tour



```
29  // AWSResourceManager is responsible for providing a consistent way to perform
30  // CRUD+L operations in a backend AWS service API for Kubernetes custom
31  // resources (CR) corresponding to those AWS service API resources.
32  //
33  // Use an AWSResourceManagerFactory to create an AWSResourceManager for a
34  // particular APIResource and AWS account.
35  type AWSResourceManager interface {
36          ReadOne(context.Context, AWSResource) (AWSResource, error)
37          Create(context.Context, AWSResource) (AWSResource, error)
38          Update(context.Context, AWSResource, AWSResource, *ackcompare.Delta) (AWSResource, error)
39          Delete(context.Context, AWSResource) (AWSResource, error)
40          ARNFromName(string) string
41          LateInitialize(context.Context, AWSResource) (AWSResource, error)
42          ResolveReferences(context.Context, client.Reader, AWSResource) (AWSResource, error)
43          IsSynced(context.Context, AWSResource) (bool, error)
44          EnsureTags(context.Context, AWSResource, ServiceControllerMetadata) error
45  }
```

# The controller factory tour



But where is the reconciliation?

# The controller factory tour



github.com/aws-controllers-k8s/runtime

- Generic reconciliation function
- Common comparison utility (string, arrays, maps)
- Metrics
- Error handling
- Condition utilities
- Common CRDs

# Some numbers…

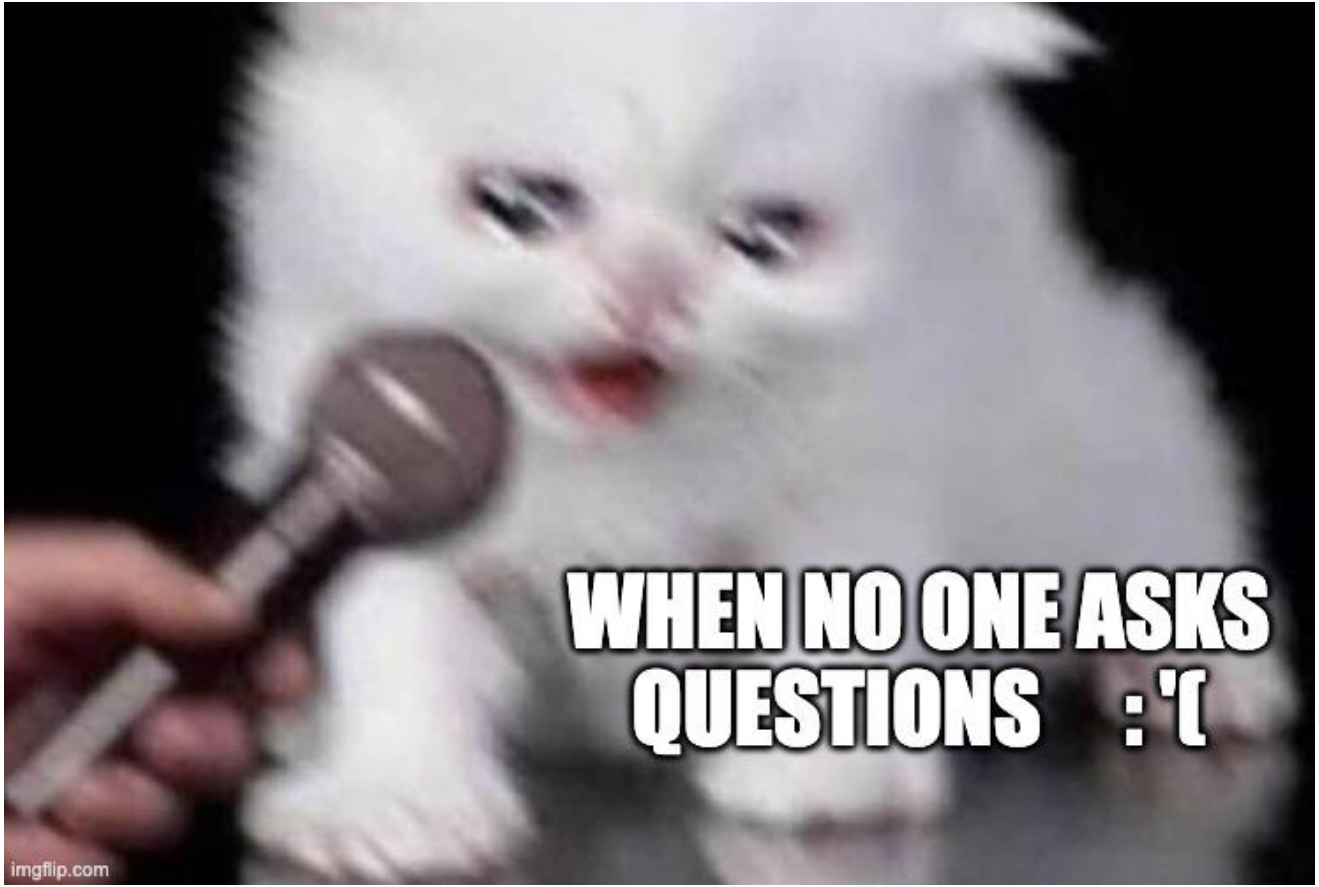
# Some numbers



- 21+ controllers (13 in GA), adding new ones all the time
- 1M+ generated lines of code
- Average generated code represents 98% of the repositories
- Over 1M+ chart/image downloads

# q&a






Q&A

# Resources



- Cloud Native Glossary: glossary.cncf.io
- Kubernetes community developer guide: https://github.com/kubernetes/community/blob/master/contributors/devel/sig-api-machinery/controllers.md
- Kubebuilder book: sigs.k8s.io/kubebuilder
- Kubernetes controller-runtime Project: github.com/kubernetes-sigs/controller-runtime
- ACK Runtime: github.com/aws-controllers-k8s/runtime
- ACK Code Generation: github.com/aws-controllers-k8s/code-generator
- Prow infra and auto generation: github.com/aws-controllers-k8s/test-infra

KubeCon | CloudNativeCon North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Please scan the QR Code above
to leave feedback on this session