



KubeCon



CloudNativeCon

Europe 2023

# Updates and Best practices in Kubebuilder and Controller-Tools

## Presenters:

Bryce Palmer, Red Hat

Camila Macedo, Replicated

Rashmi Gottipati, Red Hat

Tony Jin, Microsoft

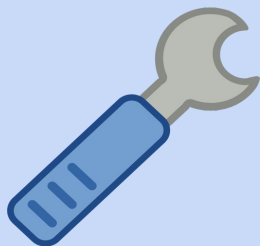
Varsha Prasad Narsing, Red Hat

Reference: <https://sched.co/1HyUw>

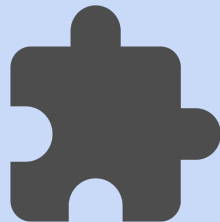
- Introduction
- Best Practices
- Kubebuilder Updates
  - (New Optional Plugins) Deploy Image and Grafana
  - (Plugins Phase 2) Extensible CLI and Scaffolding Plugins
- Usage of Controller-Tools and Controller-Gen
- Custom Generators
- Community
- References

# WHAT IS KUBEBUILDER?

Kubebuilder is a framework for building solutions that extend K8s API(s). It provides:



**A tool (CLI)**



**with an Plugin Design**



**which is extensible**

# OPERATOR PATTERN



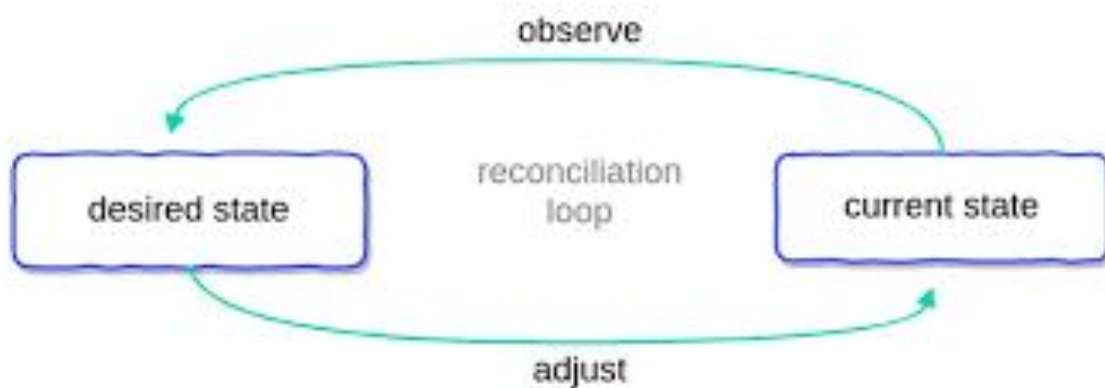
KubeCon



CloudNativeCon

Europe 2023

*"Operators are software extensions to Kubernetes that make use of custom resources to manage applications and their components. Operators follow Kubernetes principles, notably the control loop."*



Reference: <https://kubernetes.io/docs/concepts/extend-kubernetes/operator/>

*"People who run workloads on Kubernetes often like to use automation to take care of repeatable tasks. The operator pattern captures how you can write code to automate a task beyond what Kubernetes itself provides"*



The diagram illustrates the Operator Loop as a sequence of three steps, each represented by a parallelogram. The first parallelogram is light blue and labeled 'Watch'. The second is a medium blue and labeled 'Controller Operator Loop'. The third is a dark blue and labeled 'Update'. The parallelograms are arranged horizontally and slightly overlap, suggesting a continuous process.

**Watch**

**Controller  
Operator Loop**

**Update**

# WHAT DOES KUBEBUILDER IMPORT ?

- Kubebuilder is majorly dependent on:
  - Controller - Runtime
    - Library that provides helpful abstractions to watch resources, perform CRUD operations and manage controllers.
  - Controller - Tools
    - Used to generate utility code and K8s yaml manifests (i.e. CRDs, RBAC)

*Reference:*

Controller-runtime: <https://github.com/kubernetes-sigs/controller-runtime>

Controller-tools: <https://github.com/kubernetes-sigs/controller-tools>

## Golden Rules

- Develop idempotent reconciliation solutions to ensure that the desired state is the current state.
- Understand k8s API exstructure and follow up its [API Conventions](#) (i.e use [Status Conditionals](#) )



- You can use:  
<https://github.com/kubernetes/apimachinery>
- Avoid a design solution where more than one Kind is reconciled by the same controller
- You can find good content in:  
<https://sdk.operatorframework.io/docs/best-practices/>

# DEPLOY IMAGE PLUGIN (*deploy-image/v1-alpha*)



KubeCon



CloudNativeCon

Europe 2023

```
$ kubebuilder init
$ kubebuilder create api \
  --group example.com \
  --version v1alpha1 \
  --kind MyApp \
  --image=myregistry/myapp:v1.0.0 \
  --plugins="deploy-image/v1-alpha"
```

*To create controllers  
and API(s) (CRD)  
which will deploy and  
manage an image on  
the cluster.*

Reference: <https://book.kubebuilder.io/plugins/deploy-image-plugin-v1-alpha.html>



# DEPLOY IMAGE PLUGIN (*deploy-image/v1-alpha*)



KubeCon



CloudNativeCon

Europe 2023

```
api/v1alpha1/memcached_types.go
config/samples/example.com_v1alpha1_memcached.yaml
internal/controller/memcached_controller.go
creating import for % test/api/v1alpha1
internal/controller/memcached_controller_test.go
creating import for % test/api/v1alpha1
Update dependencies:
$ go mod tidy
Running make:
$ make generate
mkdir -p /Users/jintony/go/src/test/bin
test -s /Users/jintony/go/src/test/bin/controller-gen && /Users/jintony/go/src/test/bin/controller-gen --
version | grep -q v0.11.3 || \
    GOBIN=/Users/jintony/go/src/test/bin go install sigs.k8s.io/controller-tools/cmd/controller-gen@
0.11.3
/Users/jintony/go/src/test/bin/controller-gen object:headerFile="hack/bailierrate.go.txt" paths="./..."
Running make:
$ make manifests
/Users/jintony/go/src/test/bin/controller-gen rbac:roleName=manager-role crd webhook paths="./..." output
:stdout:artifacts:config=config/crd/bases
Next: check the implementation of your new API and controller. If you do changes in the API run the manif
ests with:
$ make manifests

$ cd ~/go/src/test
$ make generate && make manifests
```

# GRAFANA PLUGIN (*grafana/v1-alpha*)



KubeCon



CloudNativeCon

Europe 2023

```
$ kubebuilder init|edit \  
--plugins=grafana.kubebuilder.io/v1-alpha
```

*To scaffold Grafana Dashboards that allow you to check out the default metrics which are exported by projects using controller-runtime.*

Reference: <https://book.kubebuilder.io/plugins/deploy-image-plugin-v1-alpha.html>



- Create your own scaffolding and use them as external plugins.
- Enables the community to:
  - Create plugins that work with other programming languages.
  - Create helpers and integrations on top of existing scaffolds.
  - Generate customized layouts.



- What is a Plugin?
- Extends the scaffolding of Kubebuilder commands
  - **init**: project initialization
  - **create api**: scaffold kubernetes API definitions
  - **create webhook**: scaffold Kubernetes webhooks

# EXTENSIBLE CLI AND SCAFFOLDING PLUGINS



KubeCon



CloudNativeCon

Europe 2023

## Phase 1

- Enable Kubebuilder to become more extensible
- Ability to import kubebuilder as a library in other projects
- New CLI and project configuration

## Phase 1.5

- Chaining of internal plugins
- Plugin chain persistence
- Concept of plugin bundle for an enhanced user experience

## Phase 2

- Discover and run external plugin executables
- Rewrite and externalize the existing internal plugins
- Kubebuilder library consumers can support chaining and discovery of external plugins

# PHASE 2 PLUGINS: WHAT THEY PROVIDE

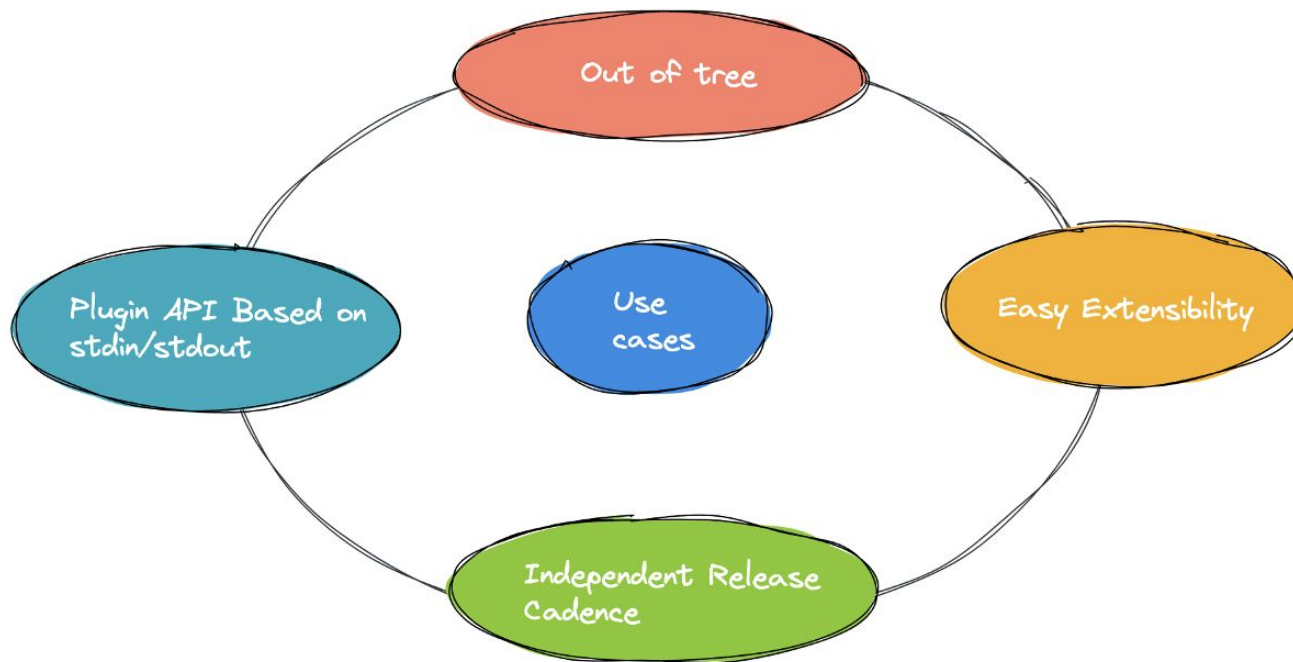


KubeCon



CloudNativeCon

Europe 2023



# PHASE 2 PLUGINS WORKFLOW

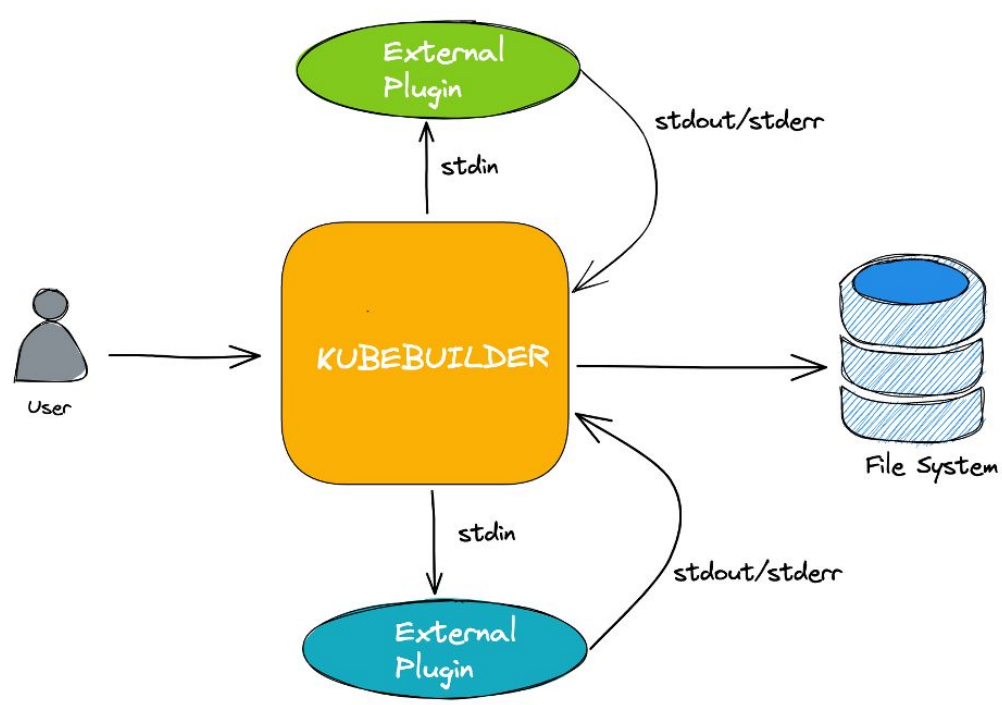


KubeCon



CloudNativeCon

Europe 2023



Reference: <https://github.com/kubernetes-sigs/kubebuilder/blob/master/designs/extensible-cli-and-scaffolding-plugins-phase-2.md>

# PHASE 2 PLUGINS DEMO



KubeCon



CloudNativeCon

Europe 2023

```
$ kubebuilder init --plugins  
myexternalplugin
```

```
$ kubebuilder create api --plugins  
myexternalplugin
```

```
$ kubebuilder create webhook  
--plugins myexternalplugin
```

*To discover and run a  
sample external plugin  
written in python.*

Reference: <https://github.com/rashmigottipati/POC-Phase2-Plugins>



# WHAT IS THE USE CONTROLLER-TOOLS?

- Kubebuilder uses Controller-gen which is a tool provide by Controller-tools to generate:
  - K8s YAML manifests, like CRDs and RBAC templates.
  - Go code required to extend the K8s API (runtime.Object/DeepCopy implementations).

# CONTROLLER-GEN EXAMPLE



KubeCon



CloudNativeCon

Europe 2023

To define some validation rules for the Busybox Kind:

```
25
26 // BusyboxSpec defines the desired state of Busybox
27 type BusyboxSpec struct {
28     // INSERT ADDITIONAL SPEC FIELDS - desired state of cluster
29     // Important: Run "make" to regenerate code after modifying this file
30
31     // Size defines the number of Busybox instances
32     // The following markers will use OpenAPI v3 schema to validate the value
33     // More info: https://book.kubebuilder.io/reference/markers/crd-validation.html
34     // +kubebuilder:validation:Minimum=1
35     // +kubebuilder:validation:Maximum=3
36     // +kubebuilder:validation:ExclusiveMaximum=false
37     Size int32 `json:"size,omitempty"`
38 }
```

Reference: <https://book.kubebuilder.io/reference/controller-gen.html>

# CONTROLLER-GEN EXAMPLE



KubeCon



CloudNativeCon

Europe 2023

After running `$make generate`:

`spec:`

`description:` BusyboxSpec defines the desired state of Busybox

`properties:`

`size:`

`description:` 'Size defines the number of Busybox instances The following markers will use OpenAPI v3 schema to validate the value More info: <https://book.kubebuilder.io/reference/markers/crd-validation.html>'

`format:` int32

`maximum:` 3

`minimum:` 1

`type:` integer

`type:` object

Reference: <https://book.kubebuilder.io/reference/controller-gen.html>



- Ability to define custom “marker” formats and write generators for them.
- Steps for writing custom generators:
  - a. Declaring “marker” format - [ref](#).
  - b. Parsing “markers” - [ref](#).
  - c. Generating output - [ref](#).



You are welcome to be part of the Kubebuilder project and contribute to it.

- Slack Channel:
  - *(Under kubernetes org)*
  - **Channel: #kubebuilder**
- Community Meetings:
  - join to [kubebuilder@googlegroups.com](mailto:kubebuilder@googlegroups.com)

# REFERENCES



KubeCon



CloudNativeCon

Europe 2023

- Kubebuilder Documentation: <https://book.kubebuilder.io/>
  - To Quick Start: <https://book.kubebuilder.io/quick-start.html>
  - Deploy-Image Plugin: <https://book.kubebuilder.io/plugins/deploy-image-plugin-v1-alpha.html>
  - Grafana-Plugin: <https://book.kubebuilder.io/plugins/grafana-v1-alpha.html>
  - Plugins: <https://book.kubebuilder.io/plugins/plugins.html>
  - Phase 2 Plugins: <https://github.com/kubernetes-sigs/kubebuilder/blob/master/designs/extensible-cli-and-scaffolding-plugins-phase-2.md>
  - To create your plugin: <https://book.kubebuilder.io/plugins/creating-plugins.html>
  - To use KB as a Library: <https://book.kubebuilder.io/plugins/extending-cli.html>
- Controller-Runtime: <https://github.com/kubernetes-sigs/controller-runtime>
- Controller-Tools: <https://github.com/kubernetes-sigs/controller-tools>
- Kubernetes:
  - API conventions: <https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md>
  - Operators: <https://kubernetes.io/docs/concepts/extend-kubernetes/operator/>

# QUESTIONS

