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

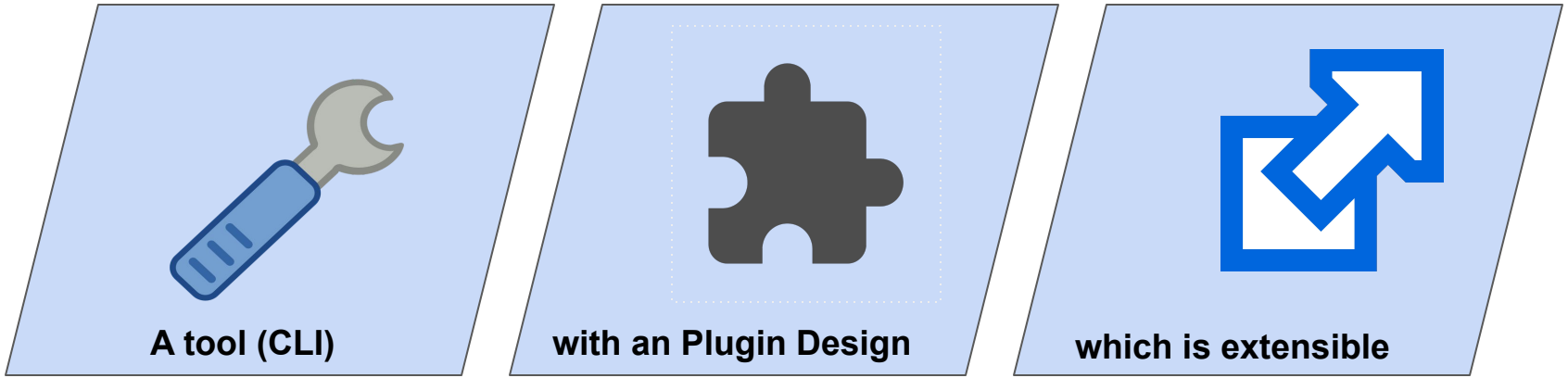
Reference: https://sched.co/1HyUw

TOPICS

- 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?

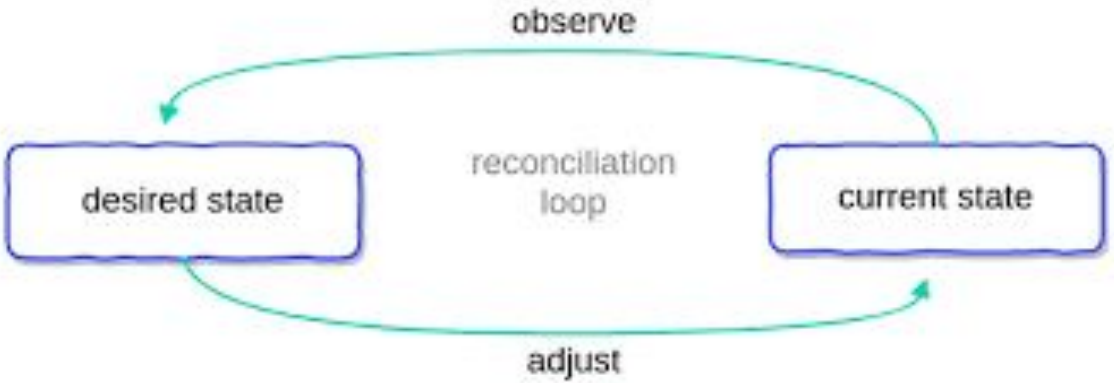
Kubebuilder is a framework for building solutions that extend K8s API(s). It provides:



Reference: https://github.com/kubernetes-sigs/kubebuilder

OPERATOR PATTERN

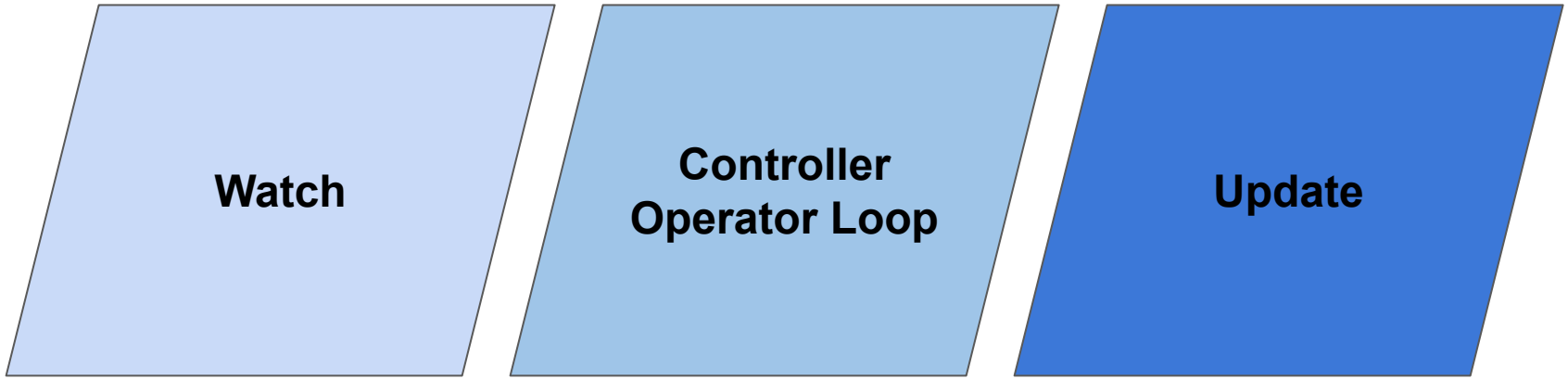
"Operators are software extensions to Kubernetes that make use of custom resources to manage applications and their components. Operators follow Kubernetes principles, notably the control loop."



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

ADVANTAGES

"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"



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

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

BEST PRACTICES

Golden Rules

- Develop idempotent reconciliation solutions to ensure that the desired state is the current state.
- Understand k8s API extructure 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/

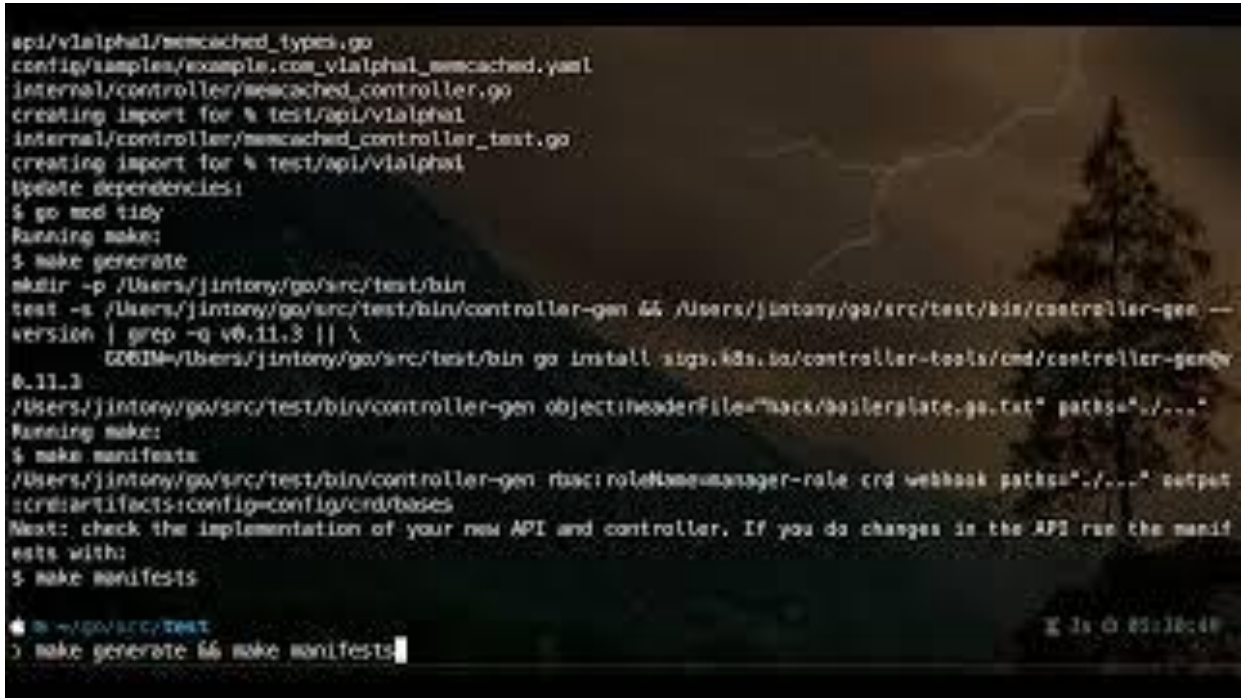
DEPLOY IMAGE PLUGIN *(deploy-image/v1-alpha)*


```
$ 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)*



GRAFANA PLUGIN *(grafana/v1-alpha)*


```
$ 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

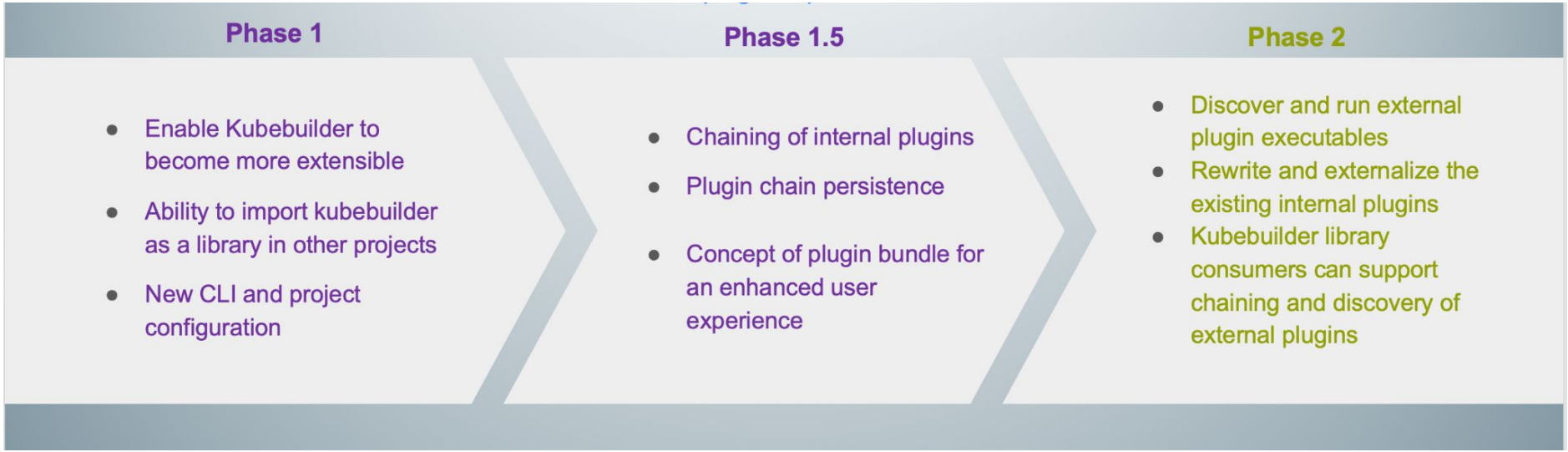
EXTENSIBLE CLI AND SCAFFOLDING

- 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.

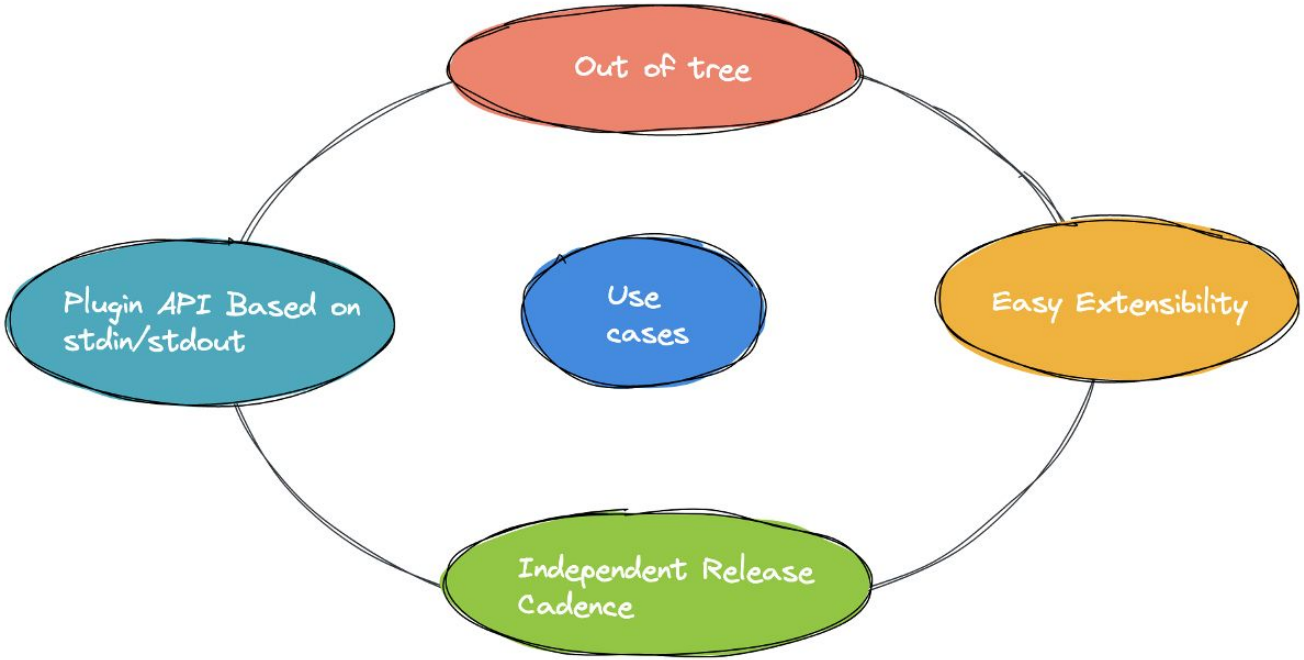
KUBEBUILDER PLUGINS

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

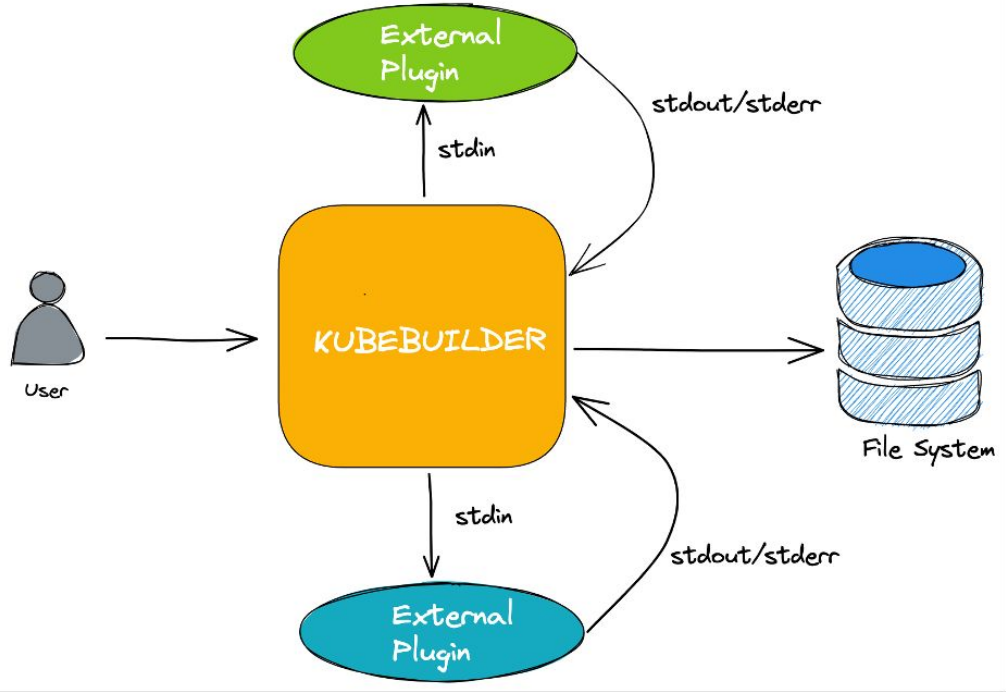
EXTENSIBLE CLI AND SCAFFOLDING PLUGINS



PHASE 2 PLUGINS: WHAT THEY PROVIDE



PHASE 2 PLUGINS WORKFLOW



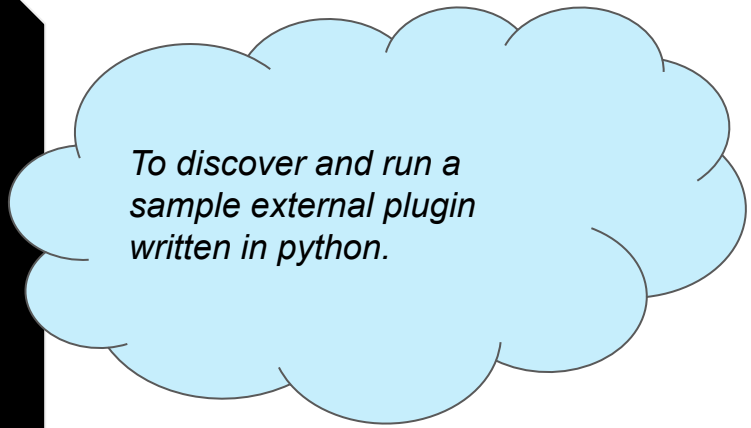
Reference: https://github.com/kubernetes-sigs/kubebuilder/blob/master/designs/extensible-cli-and-scaffolding-plugins-phase-2.md

PHASE 2 PLUGINS DEMO


```
$ kubebuilder init --plugins
myexternalplugin

$ kubebuilder create api --plugins
myexternalplugin

$ kubebuilder create webhook
--plugins myexternalplugin
```



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).

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

CONTROLLER-GEN EXAMPLE

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

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

CUSTOM GENERATORS

- 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.

COMMUNITY

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

REFERENCES

- 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