Programming in K8s

Frameworks and good practices

Why programming in K8s

- The main reason is to extend Kubernetes functionalities. There are a few main extension points:
  1. Client plugins (e.g. extending kubectl)
  2. API Access extensions
  3. API extensions
  4. Scheduling extensions
  5. CRDs and Controllers (new APIs)
  6. Network Plugins
  7. Device Plugins
  8. Storage Plugins

  More info official documentation

How can I program in Kubernetes?

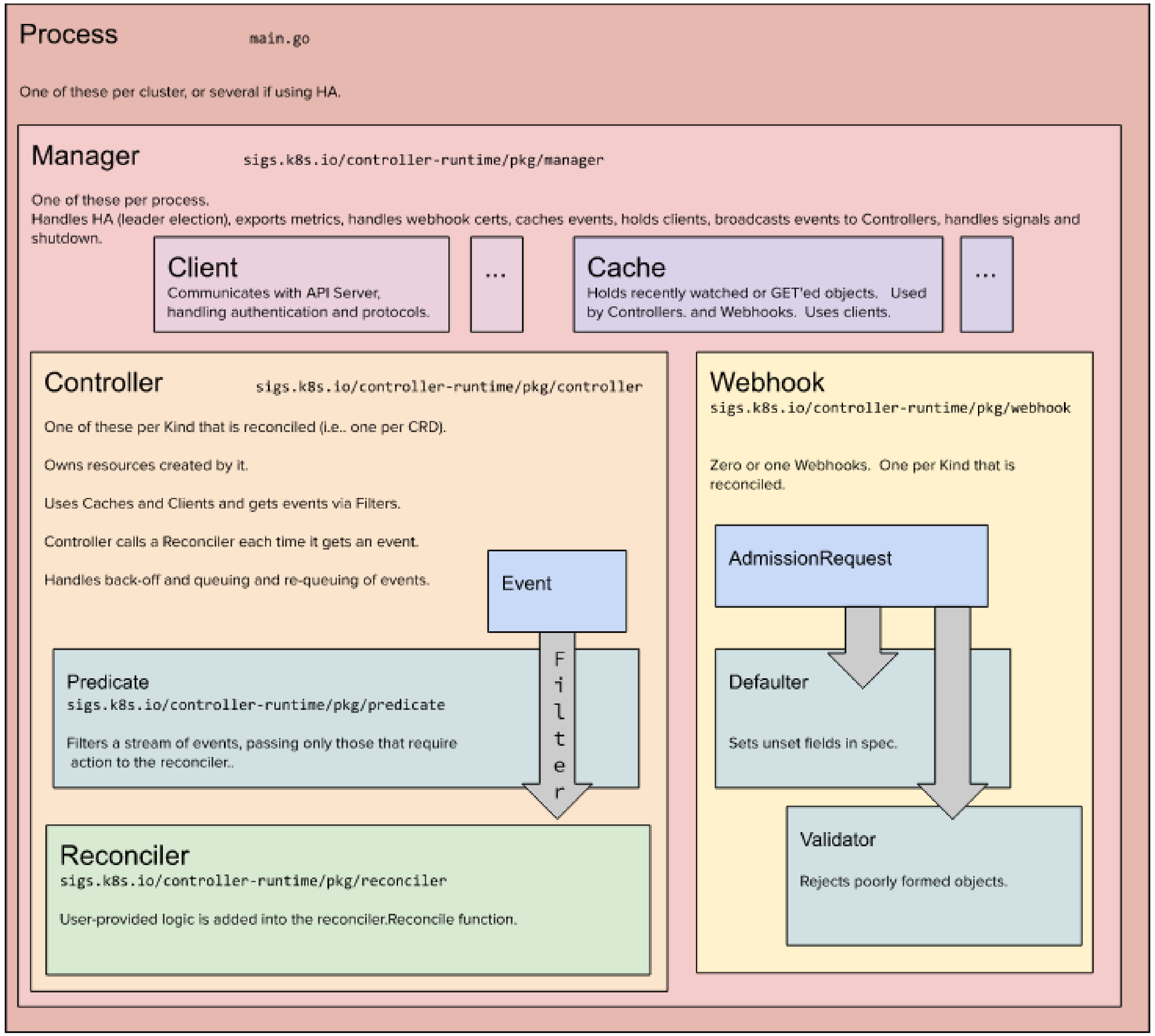
1. Using client libraries
2. Using appropriate framework which can easy and speed up the development process. Among the most common ones there are:
   - Kubebuilder
   - Operator SDK

Kubebuilder

- Kubebuilder is an tool for rapidly building and publishing Kubernetes APIs in Go.
- It builds on top of the canonical techniques used to build the core Kubernetes APIs to provide simple abstractions that reduce boilerplate and toil.
- Who is it for?
  1. Users of Kubernetes
  2. Kubernetes extensions developers
  3. Go developers

More info on the official documentation

Kubebuilder Architecture



Process `main.go`

One of these per cluster, or several if using HA.

Manager `sigs.k8s.io/controller-runtime/pkg/manager`

One of these per process.
Handles HA (leader election), exports metrics, handles webhook certs, caches events, holds clients, broadcasts events to Controllers, handles signals and shutdown.

Client
Communicates with API Server, handling authentication and protocols.

Cache
Holds recently watched or GET'ed objects. Used by Controllers. and Webhooks. Uses clients.

Controller `sigs.k8s.io/controller-runtime/pkg/controller`

One of these per Kind that is reconciled (i.e.. one per CRD).

Owns resources created by it.

Uses Caches and Clients and gets events via Filters.

Controller calls a Reconciler each time it gets an event.

Handles back-off and queuing and re-queuing of events.

Predicate
`sigs.k8s.io/controller-runtime/pkg/predicate`

Filters a stream of events, passing only those that require action to the reconciler..

Reconciler
`sigs.k8s.io/controller-runtime/pkg/reconciler`

User-provided logic is added into the reconciler.Reconcile function.

Webhook
`sigs.k8s.io/controller-runtime/pkg/webhook`

Zero or one Webhooks. One per Kind that is reconciled.

Defaulter

Sets unset fields in spec.

Validator

Rejects poorly formed objects.

Operator SDK

- The Operator SDK is a tool that provides the facilities to build, test, and package Operators.
- Yes but.. What is an Operator?

Operators in K8s

- Software extensions to Kubernetes that make use of custom resources to manage applications and their components.
- The *operator pattern* aims to capture the key aim of a human operator who is managing a service or set of services.
- The operator pattern captures how you can write code to automate a task beyond what Kubernetes itself provides.

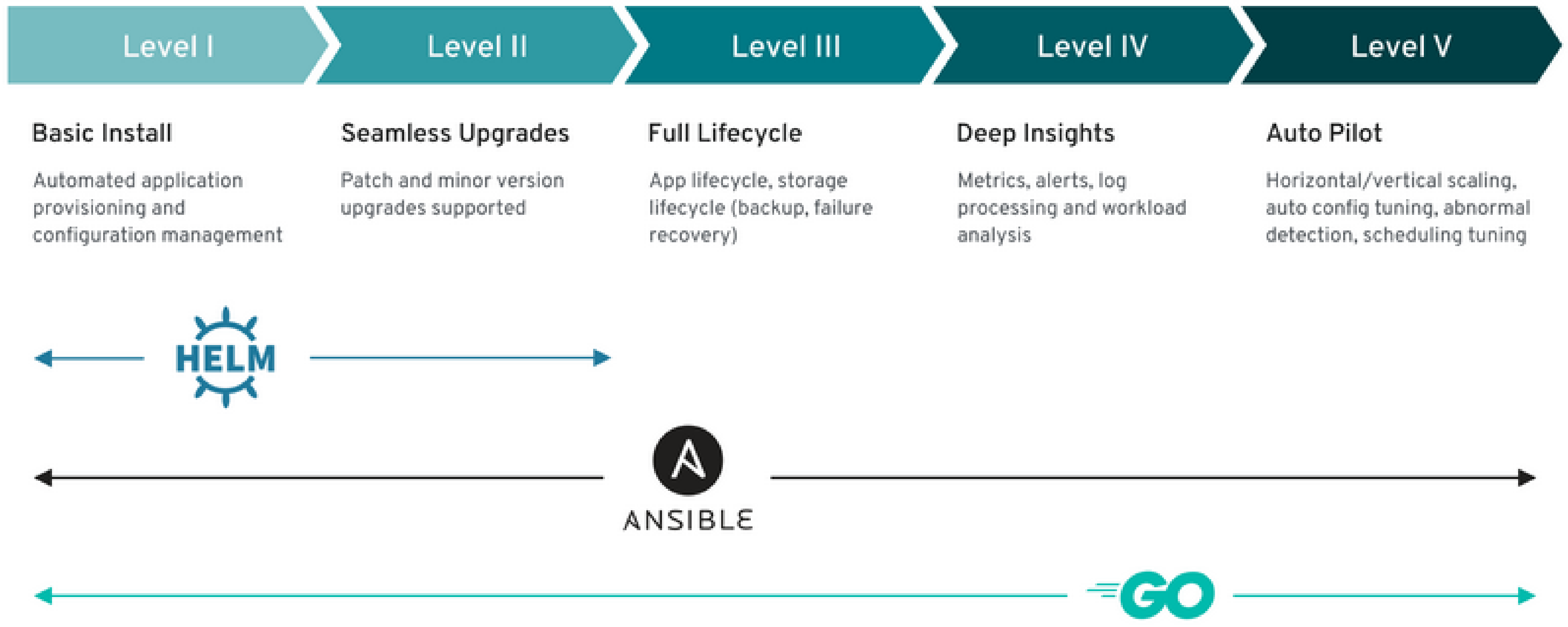
More info on the official documentation

Operator SDK cont'd

- The Operator SDK is a framework that uses the controller-runtime library to make writing operators easier by providing:
    1. High level APIs and abstractions to write the operational logic more intuitively
    2. Tools for scaffolding and code generation to bootstrap a new project fast
    3. Extensions to cover common Operator use cases

- Different Operator "types":
    1. Helm-based
    2. Ansible-based
    3. Go-based

Operator types and capability levels in Operator SDKs


Level I
Level II
Level III
Level IV
Level V
Basic Install
Automated application provisioning and configuration management
Seamless Upgrades
Patch and minor version upgrades supported
Full Lifecycle
App lifecycle, storage lifecycle (backup, failure recovery)
Deep Insights
Metrics, alerts, log processing and workload analysis
Auto Pilot
Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning
HELM
ANSIBLE
GO

- More info on the official documentation

Kubebuilder vs Operator SDK

Kubebuilder

- **Origin:** Kubebuilder was developed by the Kubernetes community and is an open-source project hosted by the Cloud Native Computing Foundation (CNCF).
- **Philosophy:** Kubebuilder takes a more low-level approach, providing a framework for building custom controllers and Operators. It relies heavily on writing Go code to define your custom resources and controllers. This allows for a high degree of customization and flexibility but can also involve more manual coding.

Operator SDK

- **Origin:** Operator SDK is also an open-source project but is developed by Red Hat. It's now part of the Operator Framework, which aims to make it easier to build, test, and package Kubernetes Operators.
- **Philosophy:** Operator SDK takes a higher-level, more opinionated approach by providing a set of pre-built templates, scaffolding, and libraries for developing Operators. It allows developers to define custom resources and controllers using YAML manifests and automates a significant portion of the development process.

Kubebuilder vs Operator SDK cont'd

Kubebuilder

- **Use Cases:** Kubebuilder is well-suited for experienced Kubernetes developers who want fine-grained control over their Operators and are comfortable writing Go code. It's a good choice if you have complex requirements or want to implement unique logic that isn't easily accommodated by higher-level abstractions.
- **When to Use:** Choose Kubebuilder when you need a high level of control, customization, and are comfortable with Go programming. It's a good fit for building Operators for complex or unique applications.

Operator SDK

- **Use Cases:** Operator SDK is a good choice for developers who want to streamline Operator development, reduce boilerplate code, and follow best practices without getting deeply into Go coding. It's particularly useful when building Operators for applications that have common deployment and management patterns.
- **When to Use:** Choose Operator SDK when you want to accelerate Operator development, follow established best practices, and prefer working with YAML manifests for custom resources. It's a good fit for many standard application deployments on Kubernetes.