O'REILLY®

Programming Kubernetes

Developing Cloud Native Applications

Michael Hausenblas &
Stefan Schimanski

Table of Contents

Preface. vii

1. Introduction. 1
What does Programming Kubernetes Mean? 1
A Motivational Example 3
Extension Patterns 4

2. Kubernetes API basics. 7
The API Server 7
  The HTTP Interface of the API Server 8
  How The API Server Processes Requests 9
API Server Terminology 11
The Kubernetes API 13
  Declarative State Management 13
  Using the API from the Command Line 14

3. Basics of `client-go`. 19
The Repositories 19
  The Client Library 19
  Kubernetes API Types 21
  API Machinery 21
  Creating and Using a Client 22
  Versioning and Compatibility 24
  API Versions and Compatibility Guarantees 26
Kubernetes Objects in Go 27
  TypeMeta 29
  ObjectMeta 31
  Spec and Status 32

Client sets 32
Status Subresources 34
Listings And Deletions 34
Watches 35
Events 35
Deployment Expansion 36
Client options 36
Informers and Caching 38
API Machinery in Depth 41
Kinds 41
Resources 41
REST Mapping 42
Scheme 43
Vendoring 43
Glide 44
Dep 44
Go Modules 45

4. Using Custom Resources. 47
Discovery Information 49
Type definitions 50
Advanced Features of Custom Resources 52
Validating Custom Resources 52
Short Names And Categories 54
Printer Columns 55
Subresources 56
A Developers View on Custom Resources 60
Dynamic Client 60
Typed Clients 61
Controller-runtime Client of Operator SDK and Kubebuilder 66

5. Automating Code Generation. 69
Why Code Generation 69
Calling the Generators 69
Controlling the generators with tags 71
Global Tags 72
Local Tags 73
deepcopy-gen tags 73
runtime.Object and DeepCopyObject 74
client-gen tags 75
informer-gen and lister-gen 76
Further Material 76

6. Controllers and Operators. 77
The Controller Loop 77
Events 79
Edge Versus Level Driven Triggers 80
Changing The Resource State 81
Optimistic Concurrency 82
Operators 84

7. Solutions For Writing Operators. 87
Kubebuilder 87
The Operator SDK 94
Metacontroller 99
Other Approaches 101
Uptake And Future Directions 101

8. Packaging. 103
Packaging: The Challenge 103
Helm 104
Kustomize 106
Other Packaging Options 108
Packaging Good Practices 108

9. Custom API Servers. 111
Use Cases For Custom API Servers 111
The Architecture: Aggregation 113
  API Services 114
  Inner Structure of a Custom API Server 117
  Delegated Authentication and Trust 118
  Delegated Authorization 119
Writing Custom API Servers 121
  Options and Config Pattern and Startup Plumbing 121
  The First Start 128
  Internal Types and Conversion 129
  Writing the API Types 133
  Conversions 134
  Defaulting 137
  Roundtrip Testing 138
  Validation 140
  Registry and Strategy 141
  API Installation 145
  Admission 149
Deploying Custom API Servers 155

Certificates and Trust 156
Sharing Etcd 156

10. Advanced Custom Resources. 159
Custom Resources Versioning 159
Conversion 160
Admission Webhooks 160
Production-ready Deployments Of Custom Controllers 160
Getting The Permissions Right For Custom Resources 161
Performance Considerations For Custom Resources 163
Observability 163

11. Cloud Native Programming Languages. 167
The Approach 167
Ballerina 169
Pulumi 170
Metaparticle 170

A. Resources. 173

Index. 177

Index

A

admission configuration, 153
admission plugin, 149
admission plugin initializer, 153
admission webhook, 112
admission webhooks, 160
aggregation, 111
  API aggregation, 113
alpha version, 26
API
  invoke via command line, 14
  proxy, 14
API group, 24
API Machinery, 21
API server, 7
  API group, 12
  HTTP interface, 8
  kind, 11
  processing requests, 9
  resource, 12
  version, 12
apiextensions-apiserver, 47, 117
APIService, 155
apps group, 24
auditing, 118, 151

B

Ballerina, 169
bearer token, 119
beta version, 26
builder pattern, 23
burst, 37

C

CA bundle, 156
category, 55
client set, 32
client-gen, 43, 60, 64
client-set, 23
clientset, 64
CNPL (cloud native programming language), 167
cobra command, 126
code examples from this book, x
cohabitation, 112
control plane, 7
  API server, 7
  controller manager, 7
  etcd, 7
  scheduler, 7
control vs. convenience, 168
controller
  changing resource state, 81
  events, 79
  loop, 77
  optimistic concurrency, 82
  triggers, 80
controller-runtime, 60
controllers
  definition, 77
conversion, 27, 134
conversion function
  conversion function naming pattern, 135
conversion-gen, 134
core group
  legacy group, 30
CRDs, 133

custom API server
  aggregated API server, 111
Custom Resource Definition
  CRD, 48
custom resources
  conversion, 160
  versioning, 159
CustomResources
  CRDs, 23
  CRs, 47

D
deep-copy, 64
defaulter-gen, 137
defaulting, 132
defaultning function naming pattern, 137
DeferredDiscoveryRESTMapper
  RESTMapper, 42
delegated authentication, 119
delegated authorization, 119
discovery endpoint, 128
dynamic client, 60

E
edge-driven trigger, 80
etcd, 121
etcd operator, 156
etcdproxy-controller, 157
event, 35
external version, 131

F
field selector, 34
fuzzer, 138

G
GA, 26
generic registry, 141
Git, viii
Go (build system), viii
go modules, 45
go.mod, 46
graceful termination, 127
GroupVersion, 33
GroupVersionKind
  GVK, 28, 41
GroupVersionResource
  GVR, 41
GVR
  GroupVersionResource, 61

H
Helm, 104
  chart, 104
HTTP/2, 112

I
informers, 38
internal clients, 33
internal version
  hub version, 129

J
JSON, 13

K
kind, 41
kube-aggregator, 113
kube-apiserver, 52, 113
kubebuilder, 66
Kubebuilder, 87
kubeconfig, 22
kubectl, 23, 101
Kubernetes
  releases, viii
Kubernetes API, 13
Kustomize, 106

L
label selector, 34
level-driven trigger, 80

M
man-in-the-middle, 115
manifests
  GitHub repository, x
master node, 7
meta/v1, 22
Metacontroller, 99
Metaparticle, 170
mutating admission webhooks, 149

N
NameGenerator, 144

O
ObjectMeta, 141
ObjectTyper, 144
OpenAPI, 126
OpenAPI Schema, 52
OpenAPI), 55
OpenShift, 112, 112
operator
    advanced, 159
    alternatives, 101
    KUDO, 101
    kutil, 101
    logging, 164
    monitoring, 164
    observability, 163
    performance, 163
    permissions, 161
    production ready, 160
    Rook Operator kit, 101
Operator SDK, 66, 94
operators
    basics, 84
    definition, 77
optimistic concurrency, 58
options and config pattern, 121

P
package management, viii
packaging, 103
    Ansible, 108
    challenge, 103
    Chef, 108
    good practices, 108
    Helm, 104
    Ksonnet, 108
    Kustomize, 106
    Puppet, 108
    sed, 108
    YAML, 103
post start hook, 126
Protobuf
    protocol buffers, 23
protobuf
    protocol buffers, 30
Protocol buffer
    protobuf, 36
Protocol Buffer
    protobuf, 111
Protocol Buffers, 13
pseudo version, 46
Pulumi, 170

Q
QPS, 37

R
RBAC, 108, 161
    role based access control, 57, 119
recommeded options, 122
relist period, 38
request header, 118
request header client CA, 118
resource, 41
REST, 141
rest config, 23
REST mapping, 42
RESTMapper, 33, 42
roundtrippable, 133
runtime.Object, 43

S
scale subresource, 58, 76
scheme, 31, 43
semantic versioning
    semver, 20, 25
server-side printing, 55
shared informer factory, 39
short name, 54
spec-status split, 57
status subresource, 34, 57
    subresource, 75
storage version, 27, 130
strategy, 143
subject access review
    SubjectAccessReview
        SAR, 119
SubjectAccessReview, 118, 141
subresource, 56
subresources, 37

T
throttling
    rate limiting, 36
timeout, 36
TokenAccessReview, 119
TokenReview, 118
typed client, 60

TypeMeta, 29, 62, 62

U

unstructured.Unstructured, 60
UserAgent, 36

V

validating admission webhooks, 149

W

watch, 35
webhook admission plugins
    admission webhook, 149
websocket, 112

About the Authors

Michael Hausenblas is a developer advocate for containers at AWS. His background is in large-scale data processing and container orchestration and he is experienced in advocacy and standardization at W3C and IETF. Before Amazon, Michael worked at Red Hat, Mesosphere, MapR, and two research institutions in Ireland and Austria. He contributes to open source software mainly using Go, blogs, writes books, and hangs out on Twitter too much.

Stefan Schimanski is a principal software engineer for Go, Kubernetes, and OpenShift at Red Hat. His focus is the Kubernetes API server, especially the implementation of CustomResourceDefinitions, API Machinery in general and the publishing of the Kubernetes staging repositories client-go, apimachinery, api, etc. Before Red Hat, Stefan worked at Mesosphere on Marathon, Spark and their Kubernetes offering and before as freelancer and consultant in high availability and distributed systems. In a former life Stefan did research in Mathematical Logic about constructive mathematics, type systems and lambda calculus.