報道各位

2020年3月2日
株式会社インプレスR&D
https://nextpublishing.jp/

Kubernetesを自分で拡張できる！

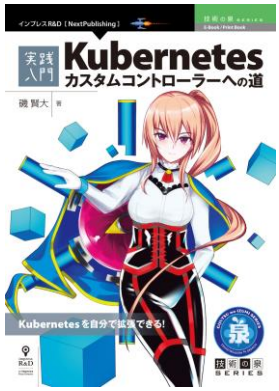
『実践入門 Kubernetesカスタムコントローラーへの道』発行

技術の泉シリーズ、2月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレスR&Dは、『実践入門 Kubernetesカスタムコントローラーへの道』(著者:磯 賢大)を発行いたします。

最新の知見を発信する『技術の泉シリーズ』は、「技術書典」や「技術書同人誌博覧会」をはじめとした各種即売会や、勉強会・LT会などで頒布された技術同人誌を底本とした商業書籍を刊行し、技術同人誌の普及と発展に貢献することを目指します。

『実践入門 Kubernetesカスタムコントローラーへの道』
https://nextpublishing.jp/isbn/9784844378549



著者:磯 賢大
小売希望価格:電子書籍版 1600円(税別)／印刷書籍版 2000円(税別)
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5判／カラー／本文174ページ
ISBN:978-4-8443-7854-9
発行:インプレスR&D

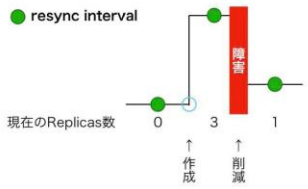
＜＜発行主旨・内容紹介＞＞

本書は、Kubernetesの拡張機能であるCustom Resource DefinitionとCustom Controllerを自作するための概要・方法を解説します。KubernetesのCustom Controllerを実装したいけれど、ツールの使い方や実装方法が分からないという方、Kubernetes自体の実装に興味がある方にもオススメいたします。本書の内容が理解できれば、Kubernetesの実装をある程度、自力で読み解くこともできるようになるでしょう。

（本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。）

Controller や CRD とはなにかを解説

図 1.6: resync interval ごとに Reconcile



こうしたedge-triggerとlevel-triggerの違いとKubernetesにおけるReconcileの詳細について、詳しく知りたい方は脚注の記事も参考にしてください[13]。

本節で、ControllerがControl Loopを実行すること、なぜControl Loopを実行するのかについてご理解いただけたでしょうか。

Control Loopは、Kubernetesの基本骨子ともいえます。Custom Controllerであっても Control Loopを実装していくことに変わりはありません。

本書では、Controller開発ツールの整理・必要なコンポーネントの理解・サンプルの実装を通して、Controllerの実装に踏み出すためのサポートをしていきます。

1.2 Kubernetesの拡張機能

Kubernetesは、それ自体に標準のResourceを多く揃えています。たとえば、ひとつ以上のコンテナを動かすPodや複数のPodを制御するReplicaSet、ReplicaSetを管理してデプロイ戦略を実現するDeploymentなどのResourceを備えています。Kubernetesではそれらの標準Resourceに加えて、ユーザーが独自のResourceを定義し、そのResourceをクラスタ上で利用できます。それがCustom Resource Definition（以降、CRD）です。

Kubernetesが提供しているAPI機能を拡張するための方法として、CRDも含めた次の方法が主に提供されています。

1．Admission Webhook: APIリクエストを変更・検証するためのWebhookを追加する方法
2．CRD: 独自のResourceを定義するAPI拡張方法
3．API Aggregation: 追加のAPIを実装し、Aggregation Layerに登録することで拡張する方法

13 Level Triggering and Reconciliation in Kubernetes　https://bit.ly/2IETovI

1.2.1 Admission Webhook

1のAdmission Webhookは、プラグイン型のWebhookです。Admission Webhookには、**Mutating Admission Webhook**と**Validating Admission Webhook**の2種類があります。`Mutating Webhook`と`Validating Webhook`は、それぞれ次の目的で利用されます。

- `Mutating`: APIリクエストの変更
- `Validating`: APIリクエストの検証

`Mutating`は、APIリクエストの変更に使われています。

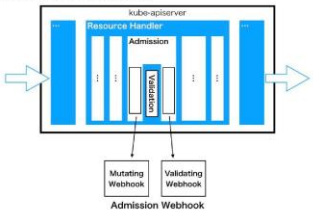
たとえばPodのObjectを作る際に、`pod.spec.imagePullPolicy`を記載しなくても、作成後のObjectのspecには`pod.spec.imagePullPolicy: Always`がデフォルトで設定されます。これはAlwaysPullImages[14]というAdmission Controllerによる効果です。

同じように、APIリクエストにデフォルト値を設定したり値を変更したりするような独自実装をしたい場合は、`Mutating Webhook`で実現できます。

`Validating`は、APIリクエストの検証を行います。`Validating`では、labelやspecなどのフィールドのチェックを行い、チェックエラーの場合はエラーを返却します。

標準で備わっているAdmission Webhookと同様に、独自にAPIリクエストの変更やバリデーションチェックをカスタマイズして行いたい場合は、プラグイン用のAPIを実装し、Admission Webhookとしてkube-api-serverに追加で登録して利用します。

図 1.7: Admission Webhook の処理の流れ



14 AlwaysPullImages　https://bit.ly/2PTIuiH

Kubernetes 標準 Controller の実装を学びます

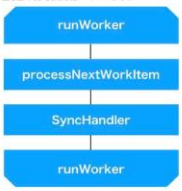
リスト 3.14: runWorker: sample-controller/controller.go

```
// runWorker is a long-running function that will continually call the
// processNextWorkItem function in order to read and process a message on the
// workqueue.
func (c *Controller) runWorker() {
  for c.processNextWorkItem() {
  }
}
```

リスト3.14で見て分かるように、`processNextWorkItem`はループで処理が繰り返されています。

図3.2は、メインロジックの処理の流れをフローチャートで表したものです。

図 3.2: Foo Controller メインロジック

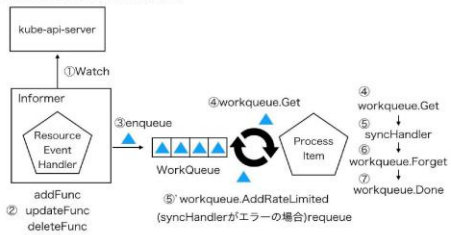


3.9 processNextWorkItem関数

`processNextWorkItem`は、WorkQueueのキューからアイテムを取り出して、処理する関数です。

EventHandlerから呼ばれる`handleObject`関数や`enqueueFoo`関数によってWorkQueueに追加されたアイテムを、`processNextWorkItem`が処理していきます。

図 3.3: 再掲: WorkQueue と processNextWorkItem



リスト3.15が、`processNextWorkItem`の実装[20]です。

リスト 3.15: processNextWorkItem: sample-controller/controller.go

```
func (c *Controller) processNextWorkItem() bool {
  obj, shutdown := c.workqueue.Get()

  if shutdown {
    return false
  }

  err := func(obj interface{}) error {
    defer c.workqueue.Done(obj)
    var key string
    var ok bool

    if key, ok = obj.(string); !ok {
      c.workqueue.Forget(obj)
      utilruntime.HandleError( ⇨
        fmt.Errorf("expected string in workqueue but got %#v", obj))
      return nil
    }

    if err := c.syncHandler(key); err != nil {
```

20 func processNextWorkItem　https://bit.ly/36Bdhw

Webhook、API Version 互換など応用機能を学びます

図 7.1: 再掲: Admission Webhook の処理の流れ

Admission Webhookには、**Mutating Admission Webhook**と**Validating Admission Webhook**の2種類があります。それぞれの目的を簡単にまとめると、次のようになります。

- Mutating: APIリクエストの変更
- Validating: APIリクエストの検証

Mutatingは、APIリクエストの変更に利用します。たとえば、特定のObjectのSpecが設定されていない時にデフォルト値を設定するDefaultingにも使われます。
本章では、後ほど例としてReplicasが設定されていないときに、デフォルト値としてReplicas: 1を設定するMutating Webhookを実装します。
Validatingは、APIリクエストの検証に利用します。Validatingでは、Mutatingの実行結果が確実に実行されていることを保障したり、禁止したいResourceの作成を禁止したりすることができます。
本書のFoo Controllerの例では、SpecがDeploymentNameとReplicasのふたつしかないため特に設定すべきValidationはありませんが、簡単な例を後ほど実装していきます。
Kubebuilderを使って、Admission Webhookを実装するときは、第5章で開発したOperatorにWebhook用のコードを追加していきます。

7.1.2 Mutating, Validation Webhookの実装

第5章で開発した、Foo Controllerを追加開発する形で、Admission Webhookを実装します。
ソースの全文は、次のリンクのブランチから確認してください。

https://github.com/govargo/foo-controller-kubebuilder/tree/admission_webhook

7.1.3 Kubebuilder でのAdmission Webhook開発の順番

Kubebuilderを使って、Admission Webhookを作成する際は、次の流れで開発をしていきます。
1．Admission Webhook用のコード初期化
2．Mutating、Validatingの実装
3．実行フェーズとして、Operatorを実際に動かす

7.1.4 1. Admission Webhook用のコード初期化

Admission Webhook用のコード初期化をするには、再びkubebuilderコマンドを実行します。
kubebuilder create webhookを実行することで、Admission Webhookに必要なコードを初期生成します。その際、引数を与えることで、MutatingやValidating, Conversion用などのWebhookの用途を選択できます。
それでは、次のコマンドを実行します[1]。

```
$ kubebuilder create webhook --group samplecontroller \
  --version v1alpha1 --kind Foo --defaulting --programmatic-validation
Writing scaffold for you to edit...
api/v1alpha1/foo_webhook.go
```

kubebuilderコマンドに、--defaultingと--programmatic-validationを引数に指定することで、それぞれDefaulting（Mutating）とValidating用のコードが自動生成されます。
api/v1alpha1/foo_webhook.goファイルが自動生成され、main.goにもコードが追記されます。
リスト7.1が、main.goのmain関数に追記されたコードです。

リスト7.1: main.go

```
if err = (&samplecontrollerv1alpha1.Foo{}). ⇨
  SetupWebhookWithManager(mgr); err != nil {
    setupLog.Error(err, "unable to create webhook", "webhook", "Foo")
    os.Exit(1)
}
```

リスト7.1でSetupWebhookWithManagerが追加されたことで、Controller Managerの起動前に、Admission Webhookが登録されます。
続けて、リスト7.2が自動生成されたapi/v1alpha1/foo_webhook.goファイルです。

リスト7.2: api/v1alpha1/foo_webhook.go

```
package v1alpha1

import (
```

1.Commit22 https://bit.ly/2SGmMka

＜＜目次＞＞

第 1 章　CRD と Controller
1.1　Kubernetes のハイレベルアーキテクチャ
1.2　Kubernetes の拡張機能
1.3　CRD と CR
1.4　CRD・CR を作ってみよう
1.5　CRD の応用機能
1.6　Controller と CRD を自作するために必要なもの
1.7　第一章のまとめ
第 2 章　client-go と知っておくべき周辺知識
2.1　Kind、Resource、Object
2.2　client-go と apimachinery
2.3　client-go で Pod の情報を取得してみる
2.4　知っておくべき周辺知識
2.5　Informer
2.6　Workqueue
2.7　runtime.Object
2.8　Scheme
2.9　第二章のまとめ
第 3 章　Sample Controller 解説
3.1　Sample Controller は何をするためのものか？
3.2　Sample Controller ディレクトリー構成
3.3　Sample Controller の CRD
3.4　types.go
3.5　main 関数

3.6 Foo Controller
3.7 Controller の定義
3.8 メインロジック（Reconcile）
3.9 processNextWorkItem 関数
3.10 syncHandler 関数
3.11 第三章のまとめ
第 4 章 controller-runtime と controller-tools
4.1 controller-tools
4.2 controller-runtime
4.3 第四章のまとめ
第 5 章 Kubebuilder で Sample Controller を実装しよう
5.1 環境構築
5.2 Kubebuilder での開発の順番
5.3 1. Kuberbuilder PROJECT の初期化
5.4 2. Kuberbuilder で、API Object と Controller のテンプレートを作成
5.5 3. types.go を編集して、API Object を定義
5.6 4. controller.go を編集して、Reconcile を実装
5.7 5. main.go を編集して、main 関数を修正
5.8 6. 実行フェーズとして、Operator を実際に動かす
5.9 第五章のまとめ
第 6 章 OperatorSDK で Sample Controller を実装しよう
6.1 環境構築
6.2 Operator SDK での開発の順番
6.3 1. Operator SDK で、Operator 用のディレクトリーの初期化
6.4 2. Operator SDK で、API Object のテンプレートを作成
6.5 3. Operator SDK で、Controller のテンプレートを作成
6.6 4. types.go を編集して、API Object を定義
6.7 5. controller.go を編集して、Reconcile を実装
6.8 6. 実行フェーズとして、Operator を実際に動かす
6.9 第六章のまとめ
第 7 章 Custom Resource の応用機能を実装しよう
7.1 Admission Webhook
7.2 Conversion Webhook
7.3 第七章のまとめ

＜＜著者紹介＞＞

磯 賢大

システムインテグレーターで 4 年間業務システムの構築に従事したのち、2019 年 5 月よりインフラエンジニアに転職。業務で Cloud Native な技術を使った基盤構築に携わる。Cloud Native が好きで、2019 年より Kubernetes コミュニティにも貢献中。著書に「実践 Helm」（インプレス R&D 刊）がある。

＜＜販売ストア＞＞

電子書籍:

Amazon Kindle ストア、楽天 kobo イーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER

印刷書籍：

Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス

※　各ストアでの販売は準備が整いしだい開始されます。

※　全国の一般書店からもご注文いただけます。

【インプレス R&D】 https://nextpublishing.jp/

株式会社インプレス R&D（本社：東京都千代田区、代表取締役社長：井芹昌信）は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishing を使った「インターネット白書」の出版など IT 関連メディア事業を展開しています。

※NextPublishing は、インプレス R&D が開発した電子出版プラットフォーム（またはメソッド）の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド（POD）による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 https://www.impressholdings.com/

株式会社インプレスホールディングス（本社：東京都千代田区、代表取締役：唐島夏生、証券コード：東証 1 部 9479）を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「旅・鉄道」「学術・理工学」を主要テーマに専門性の高いメデイア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】

株式会社インプレス R&D NextPublishing センター

TEL 03-6837-4820

電子メール: np-info@impress.co.jp