

F-2

Kubernetes の カスタムコントローラーを読み解く - AWS Controllers for Kubernetes Deep Dive -

Shinichi Hama

Solutions Architect

Amazon Web Services Japan/ Solutions Architecture

自己紹介

Shinichi Hama

Twitter/@track3jyo

Solutions Architect

Amazon Web Service Japan

西日本のお客様の支援 & コンテナ技術のあれこれ



<最近好きな AWS サービス>

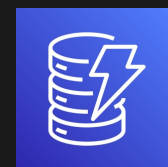
AWS App Runner



AWS Lambda



Amazon DynamoDB



アジェンダ

AWS Controllers for Kubernetes (ACK) について

ACK の導入手順とその仕組み

カスタムリソースとカスタムコントローラーによる Kubernetes API の拡張

ACK のカスタムコントローラー実装を読み解く

AWS Controllers for Kubernetes (ACK) について

AWS Controllers for Kubernetes (ACK) now in preview!



AWS Controllers for Kubernetes (ACK)

KUBERNETES から直接 AWS のサービスリソースを定義して使用できるようにする拡張ツール



AWS Controllers for Kubernetes

AWS サービスリソースを定義・管理する
Kubernetes カスタムリソース/コントローラー

AWS の Kubernetes チームが
オープンソースとして開発・メンテナンス

CloudFormation 経由ではなく、AWS の
API を介して直接管理

- Kubernetes のマニフェストの状態を "single source of truth" に

任意の Kubernetes クラスタで動作可能

現在サポートされているサービス

RELEASED(プレビュー)

- Amazon API Gateway
- Amazon Application Auto Scaling
- Amazon DynamoDB
- Amazon ECR
- Amazon ElastiCache
- Amazon MQ
- Amazon RDS
- Amazon SageMaker
- Amazon SNS
- AWS Step Functions
- Amazon S3

IN PROGRESS

- Amazon EKS
- Amazon EC2 / Amazon VPC
- AWS Lambda
- Amazon KMS
- Amazon SQS

PLANNED, PROPOSED

- more...

<https://aws-controllers-k8s.github.io/community/docs/community/services/>



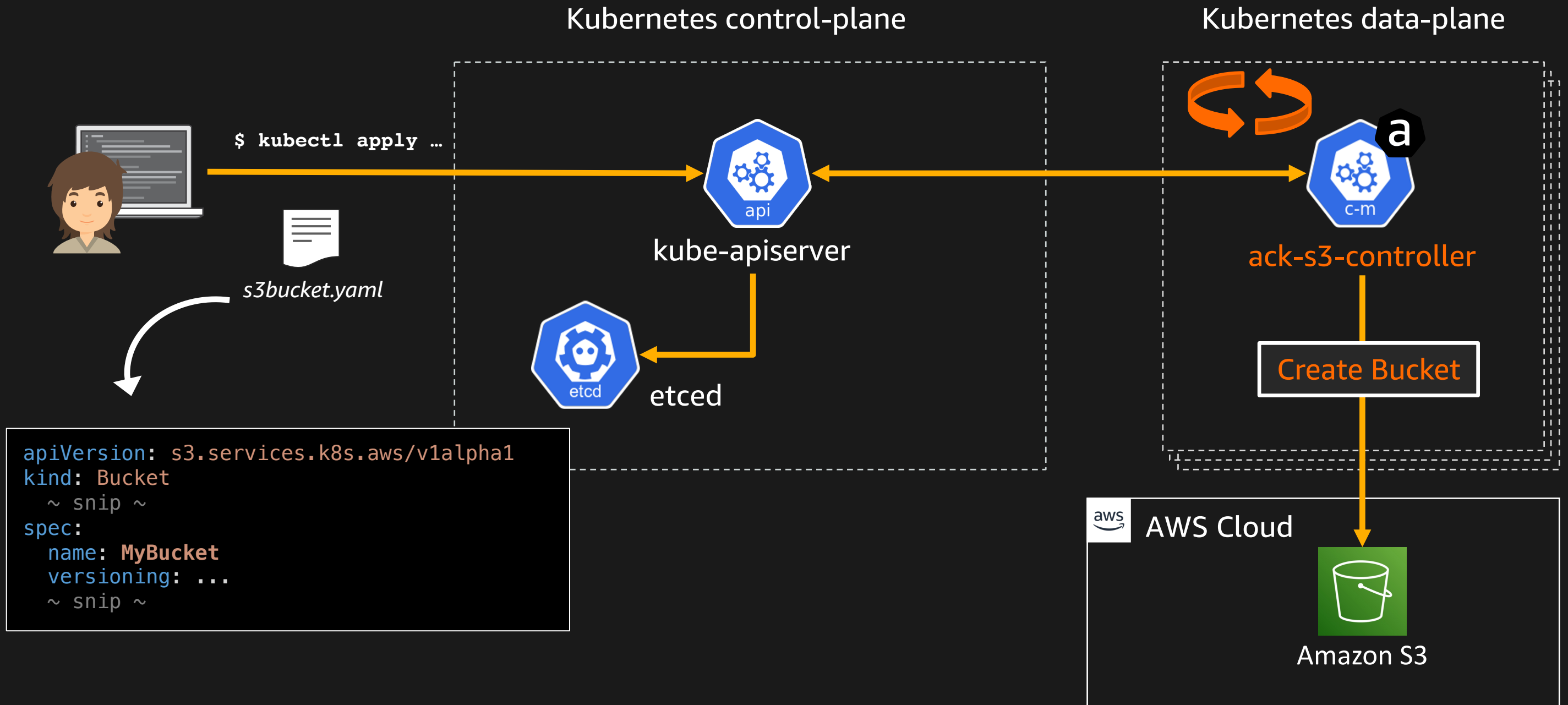
Example - S3 Manifest YAML

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: hama-test-bucket
  namespace: default
spec:
  name: hama-test-bucket
  versioning:
    status: Enabled
  lifecycle:
    rules:
      - id: Move to Glacier after sixty days
        prefix: /logs/
        status: Enabled
        transitions:
          - days: 60
            storageClass: GLACIER
  encryption:
    rules:
      - bucketKeyEnabled: false
        applyServerSideEncryptionByDefault:
          sseAlgorithm: AES256
  tagging:
    tagSet:
      - key: first
        value: 1
      - key: second
        value: 2
```

<https://aws-controllers-k8s.github.io/community/reference/>



ACK のデプロイワークフロー (概略)

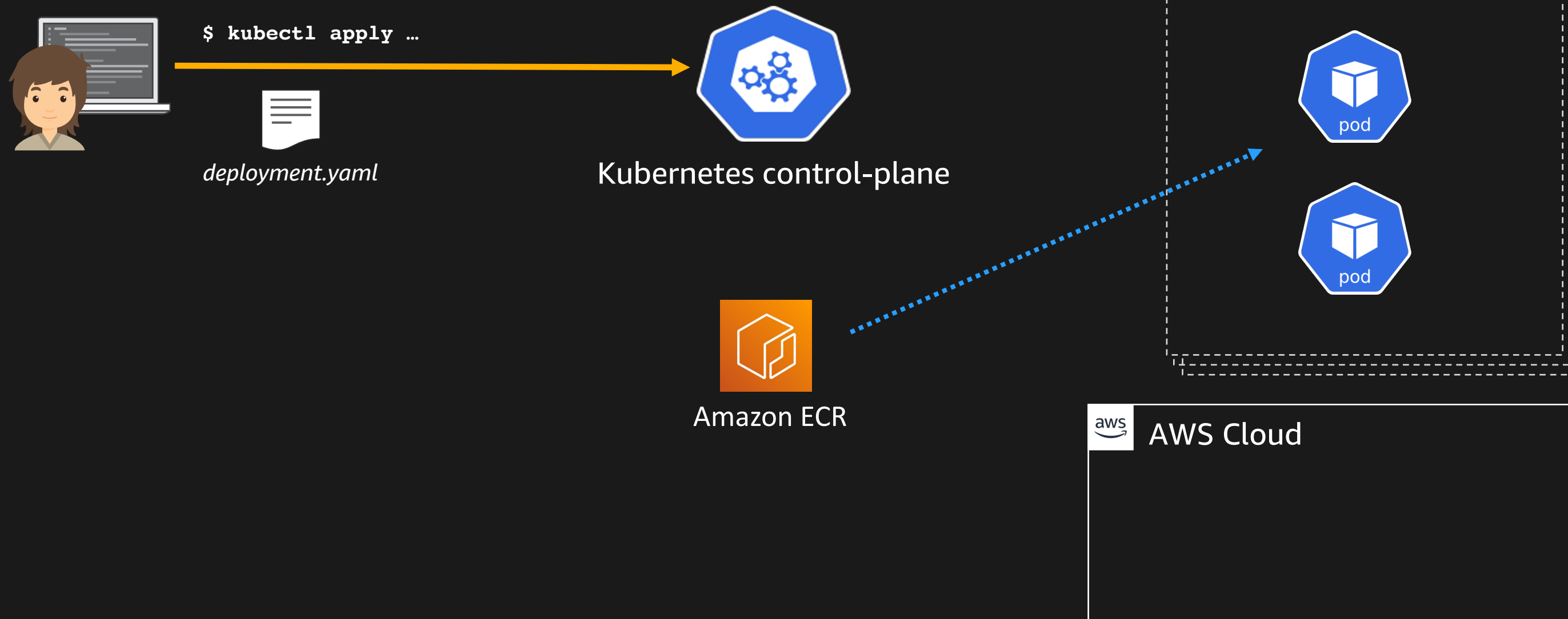


ACK で実現できる体験

- 多くのアプリケーションは「**コンテナだけで完結しない**」
- 開発者は Kubernetes という単一の API を理解するだけで、AWS のストレージやキューといった依存リソースを使用可能
- Kubernetes リソースとして AWS サービスを管理
 - 同様のツール (HELM, ArgoCD, Flux, cdk8s など) で統一された体験
- 考慮事項：他の Kubernetes リソース同様 **クラスターのライフサイクルに依存する**
 - クラスターを削除するタイミングで ACK で管理しているリソースも削除される
 - ライフサイクルの短いステートレスなサービス(Lambda など)を管理するか、In-Place Upgradeを実施するか

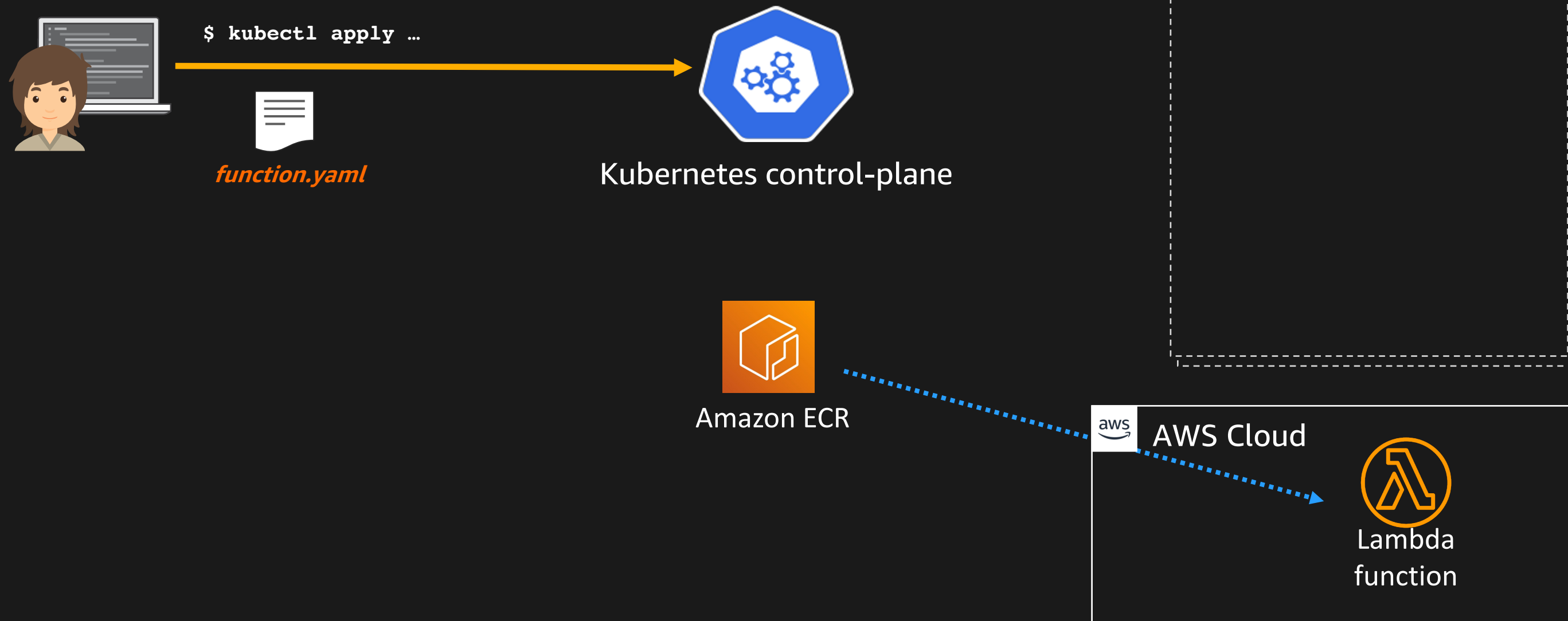
ACK で実現できる体験

※ Lambda は現在ACK未サポートです



ACK で実現できる体験

※ Lambda は現在ACK未サポートです



ACK の導入手順とその仕組み

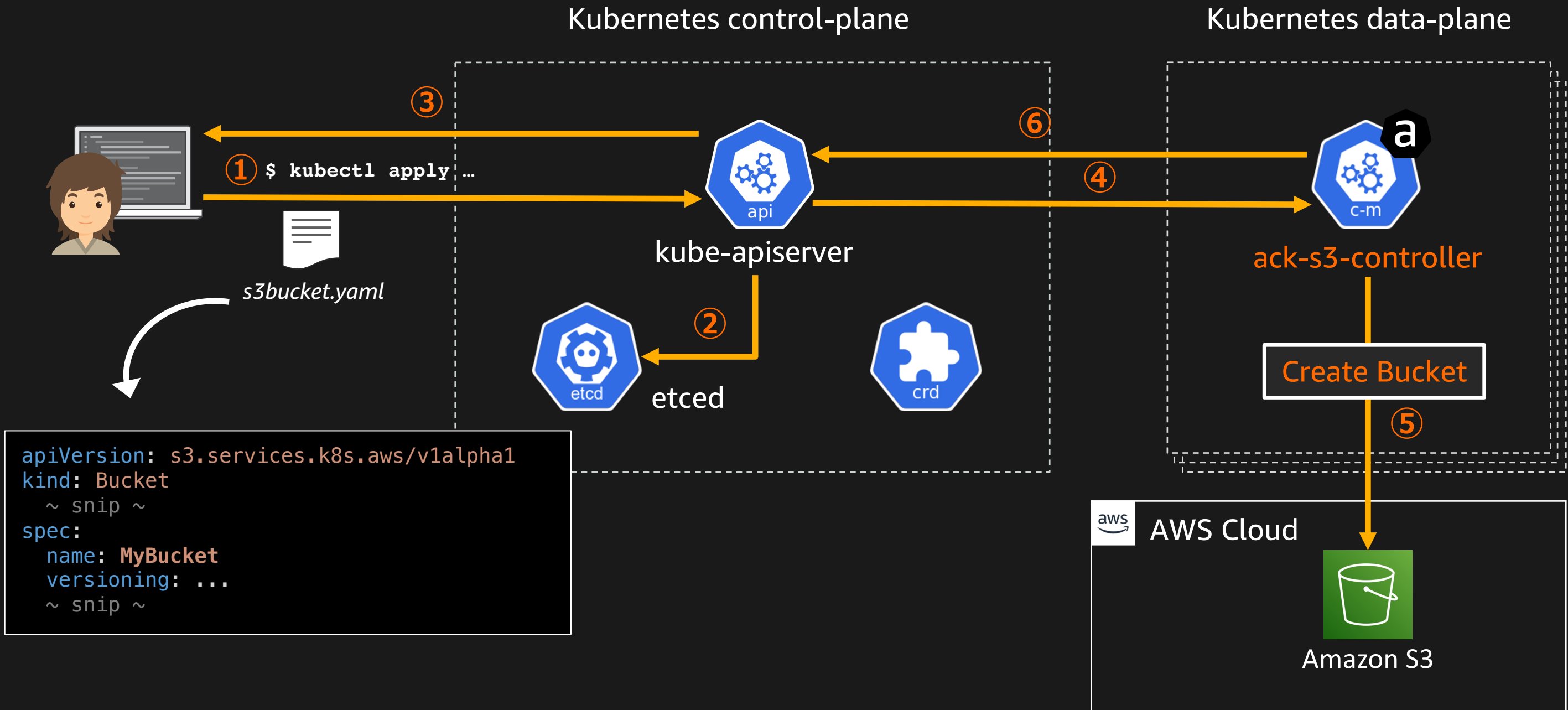
ACK を構成する要素

各 AWS サービスごとに以下のリソースを提供

- **ACK CustomResourceDefinition (CRD)** :
 - e.g. "buckets.s3.services.k8s.aws"
- **ACK サービスコントローラー** :
 - CRD と紐づくカスタムコントローラーとして、AWS リソースの作成・更新・削除を API で操作。Deployment リソースとしてデータプレーンにインストール
 - e.g. "ack-s3-controller"
- etc...(RBAC, ServiceAccount)



ACK のデプロイワークフロー (概略)



ACKの導入手順 (1/2)

- Helm3.7 以降の利用を推奨
- ACK サービスコントローラーの Helm チャート (Deployment, CRD, RBAC) が ECR Public Gallery [1] で提供

```
$ export HELM_EXPERIMENTAL_OCI=1
$ export SERVICE=s3
$ export RELEASE_VERSION=v0.0.1
$ export CHART_EXPORT_PATH=/tmp/chart
$ export CHART_REF=$SERVICE-chart
$ export CHART_REPO=public.ecr.aws/aws-controllers-k8s/$CHART_REF
$ export CHART_PACKAGE=$CHART_REF-$RELEASE_VERSION.tgz

$ mkdir -p $CHART_EXPORT_PATH

$ helm pull oci://$CHART_REPO --version $RELEASE_VERSION -d $CHART_EXPORT_PATH
$ tar xvf $CHART_EXPORT_PATH/$CHART_PACKAGE -C $CHART_EXPORT_PATH
```

[1] <https://gallery.ecr.aws/aws-controllers-k8s>

ACKの導入手順 (2/2)

```
$ export ACK_K8S_NAMESPACE=ack-system
```

```
$ helm install --create-namespace --namespace $ACK_K8S_NAMESPACE ack-$SERVICE-controller  
--set aws.account_id="$AWS_ACCOUNT_ID" --set aws.region="$AWS_REGION" $CHART_EXPORT_PATH/ack-$SERVICE-controller
```

```
NAME: s3-chart  
LAST DEPLOYED: Thu Dec 17 13:09:17 2020  
NAMESPACE: ack-system  
STATUS: deployed  
REVISION: 1  
TEST SUITE: None
```

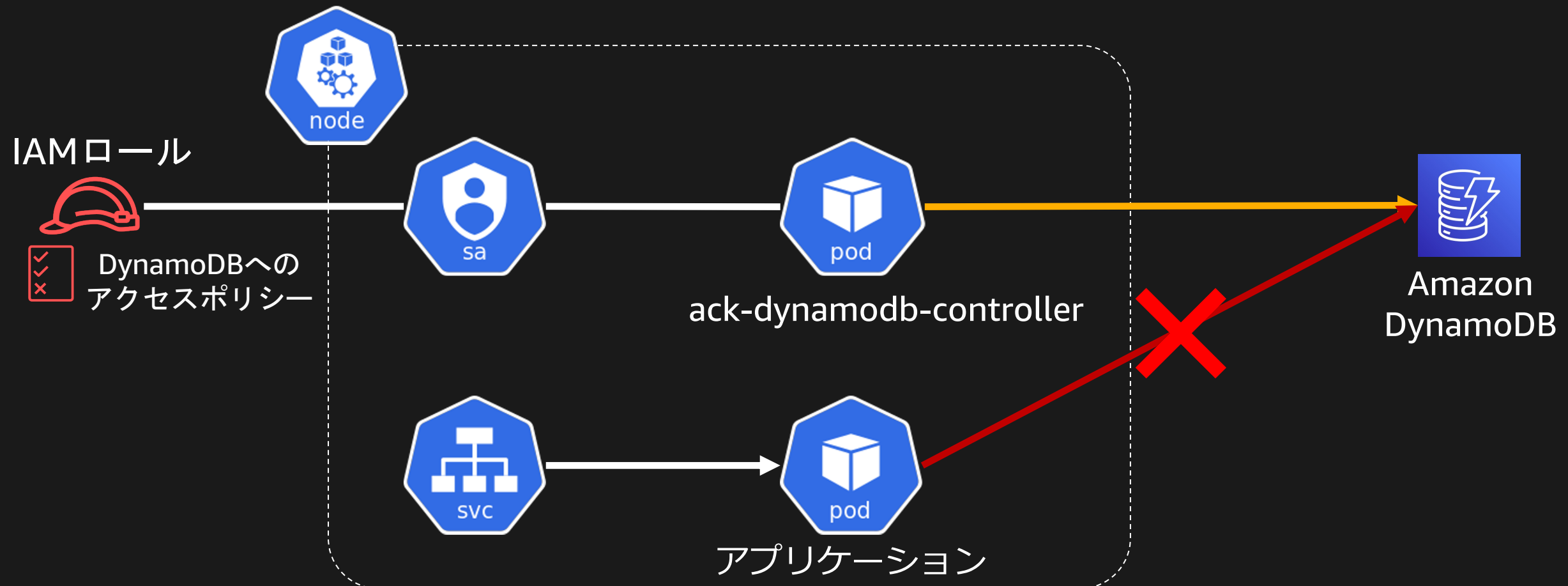
```
$ kubectl get deployment -n ack-system
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/ack-s3-controller	1/1	1	1	30s



サービスコントローラーのアクセスコントロール

- IAM Roles for Service Accounts (IRSA) で Pod レベルのアクセスコントロールを付与することを推奨
- Namespace も別途用意し、開発者から隔離する

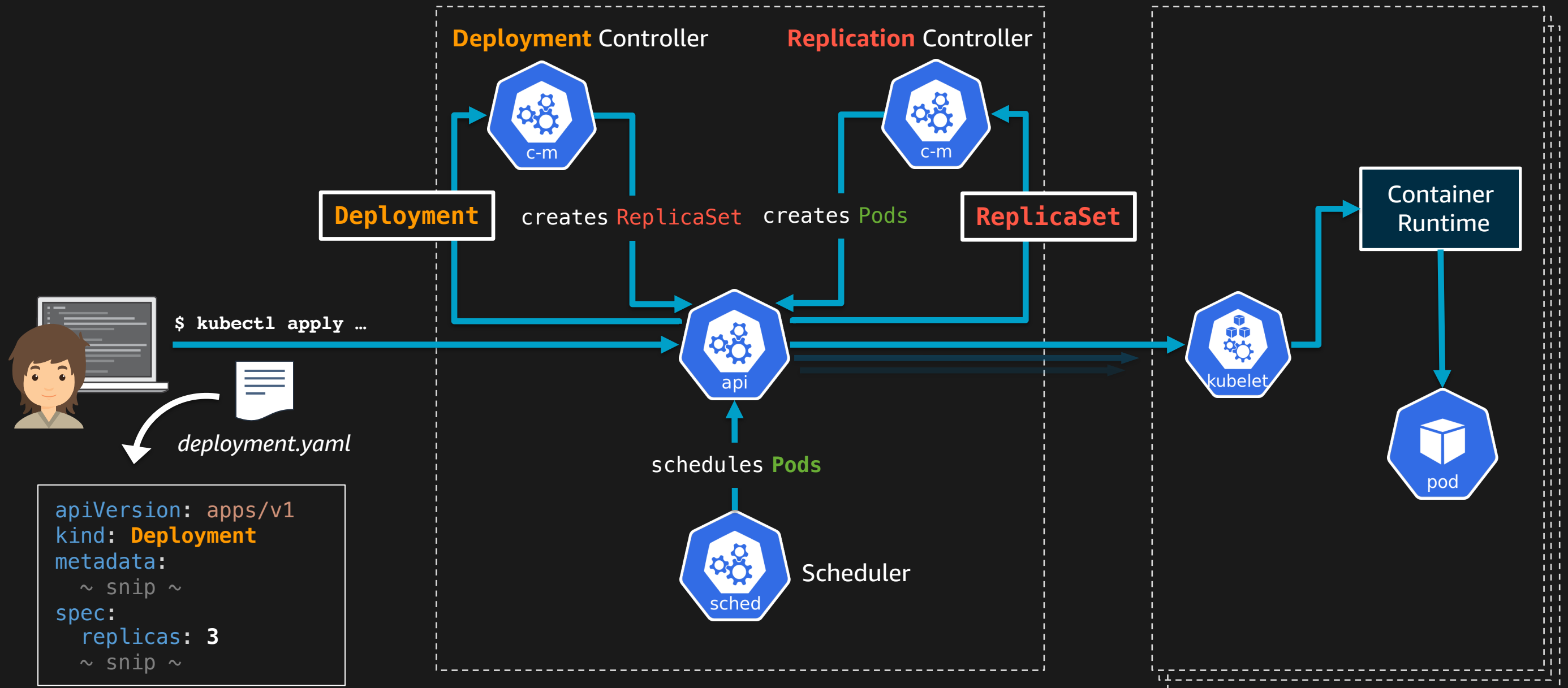


カスタムリソースと カスタムコントローラーによる **Kubernetes API** の拡張

Kubernetes の宣言的デプロイメントモデル (概略)

Kubernetes control-plane

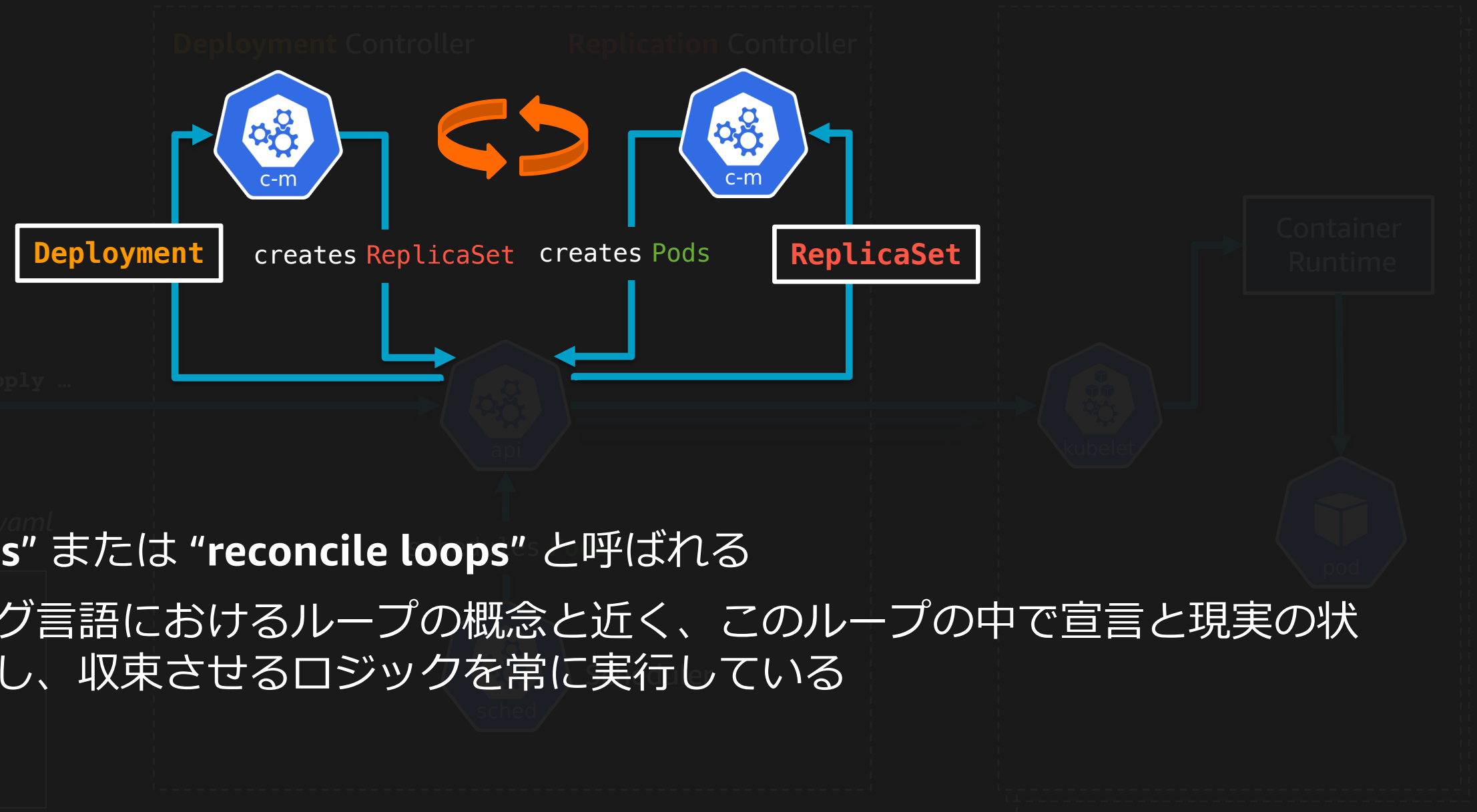
Kubernetes data-plane



Kubernetes の宣言的デプロイメントモデル (概略)

Kubernetes control-plane

Kubernetes data-plane



- "control loops" または "reconcile loops" と呼ばれる
- プログラミング言語におけるループの概念と近く、このループの中で宣言と現実の状態差異を検出し、収束させるロジックを常に実行している

Kubernetes の拡張性

- コンテナレベルの拡張仕様のサポート
 - Container Network Interface (CNI), Container Storage Interface(CSI)
- コンテナランタイムのプラグインインターフェース
 - Container Runtime Interface (CRI)
- Validating/Mutating Admission Webhook
- カスタムスケジューラの実装と利用
- ...
- **カスタム実装による Kubernetes API のプラグブルな拡張** (Today's topic)

Kubernetes API の代表的な拡張実装方法

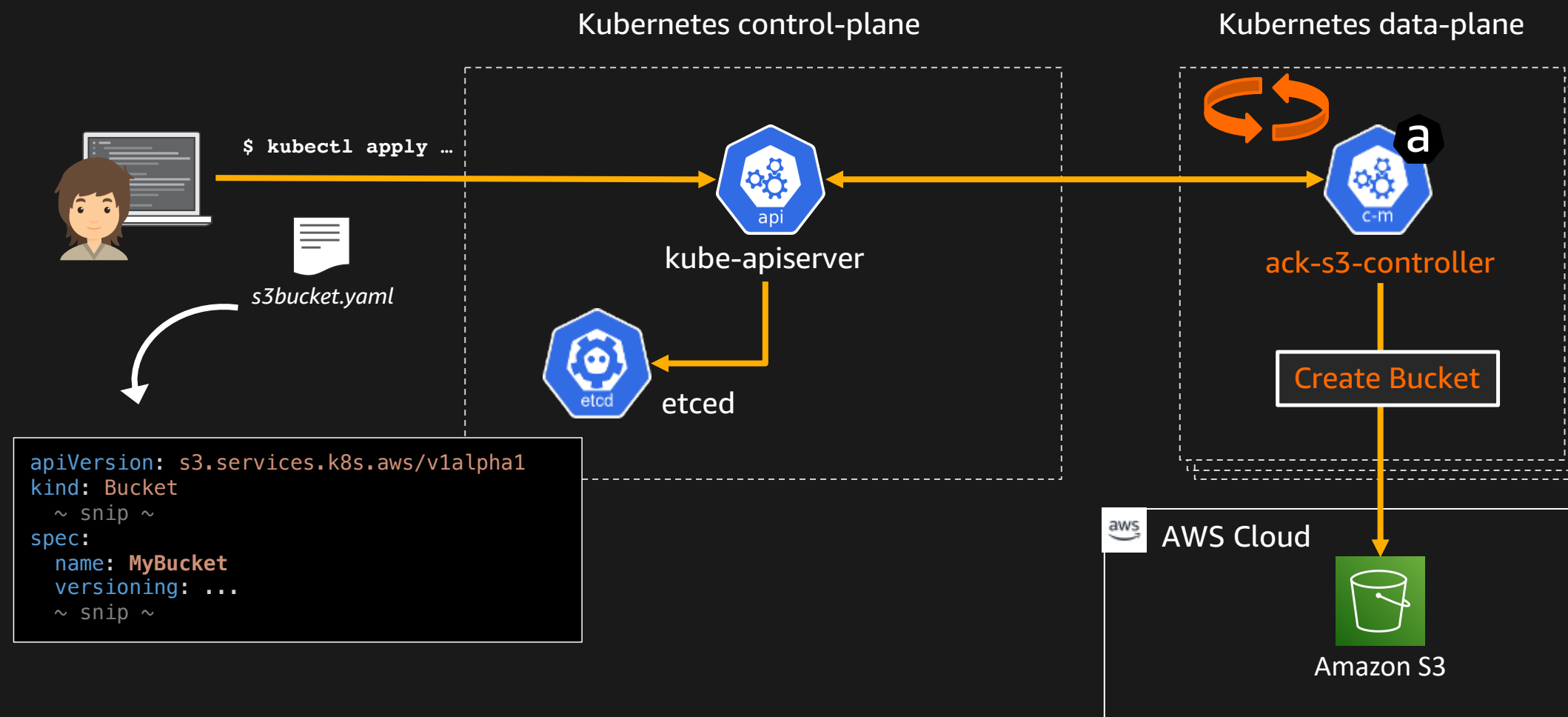
1. Aggregation Layer (API aggregation / AA)

- Kubernetes の **RESTful API としてのリソース** を任意に追加可能
- とにかく自由度が高い

2. カスタムリソース (CustomResourceDefinition, CRD)

- **Kubernetes としてのリソース** を任意に追加可能
e.g. "kind: Bucket"
- Kubernetes リソースは Kubernetes RESTful API のリソースとして表現されるため、結果として API リソースが追加される

カスタムリソースとカスタムコントローラーによる拡張



- Custom resource そのものはいわば ConfigMap のようなもので、ビジネスロジックは持たず、Kubernetes API の永続化ストレージである etcd に保存される
- Custom resource の宣言とリソースの実体の状態差を収束させるビジネスロジックを Custom controller として実装することが一般的

“Operator” パターン

- CoreOS 社が提唱した Kubernetes 拡張パターン [1]
 - カスタムリソース + カスタムコントローラーによる Kubernetes 拡張方法の中でも特にアプリケーション/ドメイン固有の運用知識をコード実装して自動化するパターン
 - 複数のカスタムリソース + カスタムコントローラー群によって構成されることも
 - e.g. あるデータベースのセットアップ、バックアップ、リストアをそれぞれ個別の CR + Custom Controller で実装
- Operator の例
 - アプリケーションコードの更新と同時に、例えばデータベーススキーマ、追加の設定修正など必要な変更の対応
 - クラスターの回復力をテストするために、全てまたは一部分の障害をシミュレート
 - 内部のリーダー選出プロセス無しに、分散アプリケーションのリーダー選択

Operator Hub

- Kubernetes Operator のパブリックレジストリ
- 203件のOperatorが登録されている (2021/09/29 時点)
- Launched by Red Hat "in collaboration with AWS, Google Cloud, Microsoft" (Feb. 28, 2019)

<https://operatorhub.io/>

The screenshot displays the OperatorHub.io website interface. At the top, there is a navigation bar with the site name, a search bar, and a 'Contribute' button. Below the header, a large blue banner reads 'Welcome to OperatorHub.io' and provides a brief description of the platform. The main content area is a grid of 203 operator items, each represented by a card with a logo, name, provider, and a short description. The grid is filtered by categories and providers. On the left side, there is a sidebar with filters for categories (e.g., AI/Machine Learning, Application Runtime, Big Data, Cloud Provider, Database, Developer Tools, Drivers and plugins, Integration & Delivery, Logging & Tracing, Modernization & Migration, Monitoring, Networking, OpenShift Optional, Security, Storage, Streaming & Messaging) and providers (e.g., aiven, alauda, Alibaba Cloud, Altinity, Alvearie). The operator cards shown include Aiven Operator, Akka Cluster Operator, Altinity ClickHouse Operator, Alvearie Imaging Ingestion Operator, Anchore Engine Operator, Apache Spark Operator, API Operator for Kubernetes, APIcast, Apicurio Registry Operator, APIMatic Operator, Appdynamics Operator, Appratrix CPS Operator, Appsoody Operator, Aqua Security Operator, ArangoDB, Argo CD, Argo CD Operator (Helm), AtlasMap Operator, AWS Auth Operator, and AWS S3 Operator.

Kubebuilder を利用して開発をはじめめる

<https://github.com/kubernetes-sigs/kubebuilder>

```
# プロジェクト用ディレクトリの作成
$ mkdir my-new-controller && cd $_

# ボイラープレートからプロジェクトを生成
$ kubebuilder init --domain track3jyo.com --license apache2 --owner hama

# ボイラープレートから API 実装の生成
$ kubebuilder create api --group webapp --version v1 --kind Guestbook

# CRD の元となる実装
$ vi api/v1/guestbook_types.go

# カスタムコントローラの実装
$ vi controllers/guestbook_controller.go

# Kubernetes クラスタへの CRD インストールとカスタムコントローラの実行
$ make install && make run

# あとは YAML を書いてクラスタに適用するだけ
$ cat config/samples/webapp_v1_guestbook.yaml | kubectl apply -f -
```

※ See also the Kubebuilder's Quick Start <https://book.kubebuilder.io/quick-start.html>



カスタムコントローラーを実装・運用していく難しさ

- Kubebuilder, Operator SDK ^[1]
 - Kubebuilder、Operator SDK が CRD + CC 実装によく利用される
 - 両者とも controller-runtime と controller-tools ^[2, 3] を利用しており、それぞれの最新バージョンは v0.10.1 と v0.7.0 (2021/09/29 時点)
 - 今後も破壊的変更が入る可能性がある
- Kubernetes クラスタ外リソースを触るコントローラ実装の難しさ
 - e.g. S3 バケット を AWS MC で削除してしまっていたらどういう挙動を取るべき？
AWS API 側のスロットリングを受けたらどうする？
 - e.g. 冪等な更新に対応していないものを扱うときは？

[1] <https://github.com/operator-framework/operator-sdk>

[2,3] <https://github.com/kubernetes-sigs/controller-runtime>, <https://github.com/kubernetes-sigs/controller-tools>

ACK のカスタムコントローラー実装を 読み解く

ACK service controller for Amazon Elastic Container Registry (ECR)

<https://bit.ly/3ETNbzP>

(<https://github.com/aws-controllers-k8s/ecr-controller>)

ack-bot Update ACK runtime to v0.14.1 (#17) ...		6c190af 3 days ago	🕒 37 commits
📁 .github/workflows	Create create-release.yml		5 days ago
📁 apis/v1alpha1	Update ACK runtime to v0.14.1 (#17)		3 days ago
📁 cmd/controller	bring ecr-controller to ACK runtime v0.11.0		last month
📁 config	Update ACK runtime to v0.14.1 (#17)		3 days ago
📁 helm	Update ACK runtime to v0.14.1 (#17)		3 days ago
📁 pkg	Update to ACK runtime v0.14.0 (#16)		5 days ago
📁 test/e2e	release artifacts for release v0.0.5		4 months ago
📄 .gitignore	Separate ECR service controller		8 months ago
📄 CONTRIBUTING.md	Separate ECR service controller		8 months ago
📄 LICENSE	Separate ECR service controller		8 months ago
📄 README.md	Separate ECR service controller		8 months ago
📄 generator.yaml	Update to ACK runtime v0.14.0 (#16)		5 days ago
📄 go.mod	Update ACK runtime to v0.14.1 (#17)		3 days ago
📄 go.sum	Update ACK runtime to v0.14.1 (#17)		3 days ago
📄 metadata.yaml	Add service metadata file		2 months ago



ACK service controller for Amazon Elastic Container Registry (ECR)

当日は ACK service controller for ECR のソースコードを見ながら、ACK をカスタムコントローラーレベルで掘り下げて解説し、カスタムコントローラー実装について話しました。

まとめ

- ACK は Kubernetes から直接 AWS のサービスリソースを定義して使用できるようにするツール
 - 開発者は Kubernetes の API を理解するだけで、コンテナ以外の AWS リソースも管理可能
 - CloudFormation ではなく AWS の API を介して直接管理するような実装
 - 現在はデベロッパープレビューな OSS 今後 GA に向けて乞うご期待
- Kubernetes はソフトウェアにおけるオペレーションの自動化や効率化のために、カスタムリソースやカスタムコントローラーで拡張可能
- 自分たちの運用や業務をカスタムコントローラーで実装してステキな Kubernetes ライフを送ってください！

ドキュメントや参考になるコンテンツ

- ACK 関連

- Github : <https://github.com/aws-controllers-k8s>
- ドキュメント : <https://aws-controllers-k8s.github.io/community/docs/community/overview/>
- API リファレンス : <https://aws-controllers-k8s.github.io/community/reference/>
- ブログ : <https://aws.amazon.com/blogs/containers/aws-controllers-for-kubernetes-ack/>
- 動画 :
 - <https://www.youtube.com/watch?v=j9GlnqUL7UU>
 - <https://www.youtube.com/watch?v=6TWIoGkWEIc>

- カスタムコントローラー関連

- ドキュメント : <https://kubernetes.io/ja/docs/concepts/extend-kubernetes/api-extension/custom-resources/>
- Kubebuilder : <https://github.com/kubernetes-sigs/kubebuilder>
- オペレーターパターン : <https://kubernetes.io/ja/docs/concepts/extend-kubernetes/operator/>
- OperatorHub.io : <https://operatorhub.io/>
- 実践入門 Kubernetes カスタムコントローラーへの道 : <https://www.amazon.co.jp/dp/B0851QCR81/>

ACK パブリックロードマップ

Service Controller Release Roadmap Updated on Aug 11

Filter cards Fullscreen Menu

8 Proposed	2 Planned	4 In Progress	11 Released
<ul style="list-style-type: none">EFS service controller #328 opened by wreed4 EFS lifecycle/frozen Service ControllerIAM service controller #222 opened by max-lobur IAM lifecycle/frozen Service ControllerRAM service controller #492 opened by mdykes-gw lifecycle/frozen RAM Service ControllerRoute53 service controller #480 opened by mdykes-gw Route53 Service ControllerACM service controller #482 opened by mdykes-gw ACM Service ControllerEC2 VPC service controller #489 opened by mdykes-gw EC2 Service ControllerAmazon Kinesis service controller #235 opened by cc4i Kinesis lifecycle/frozen Service ControllerAmazon DocumentDB service controller #844 opened by thefang12 Amazon DocumentDB Service Controller	<ul style="list-style-type: none">CloudFront service controller #249 opened by akshayvr CloudFront Service ControllerMSK service controller #348 opened by yanivpaz MSK Service Controller	<ul style="list-style-type: none">SQS service controller #205 opened by tabern Service Controller SQSAWS Lambda service controller #238 opened by tabern Lambda Service ControllerKMS service controller #491 opened by mdykes-gw KMS Service ControllerAmazon EKS Service Controller #16 opened by jicowan EKS Service Controller	<ul style="list-style-type: none">S3 service controller #204 opened by tabern S3 Service ControllerSageMaker service controller #385 opened by jaypipes lifecycle/frozen SageMaker Service Controller Sagemaker developer preview 1 linked pull requestECR service controller #208 opened by tabern ECR Service ControllerSNS service controller #202 opened by tabern Service Controller SNSDynamoDB service controller #206 opened by tabern DynamoDB Service ControllerAPI Gateway v2 service controller #207 opened by tabern APIGateway Service ControllerAmazon ElastiCache service controller #240 opened by tabern Elasticache lifecycle/frozen Service Controller Elasticache betaAWS Step Functions service controller #239 opened by tabern Service Controller SFN

<https://github.com/aws-controllers-k8s/community/projects/1>



Thank you!

Shinichi Hama

 track3jyo