aws DEV DAY
ONLINE JAPAN

20-22.10.2020

B - 9

CDK for Kubernetes

- Kubernetes の YAML 運用からの解放 -

Shinichi Hama
Solutions Architect
Amazon Web Services Japan/ Solutions Architecture

In Partnership with intel

Shinichi Hama

Twitter/@track3jyo

Solutions Architect

Amazon Web Service Japan

西日本のお客様の支援 & Container のあれこれ

<好きな AWS サービス>

Amazon ECS Amazon EKS AWS Fargate Amazon DynamoDB

In Partnership with intel

本セッションは…

想定聴講者

- Kubernetes マニフェストファイルを触ったことがある・書いたことがある
- もう Kubernetes の マニフェストを YAML で書くのに疲れた・・・
- CDK for Kubernetes (cdk8s) の存在は知っているが、どういう時につかうものかを知りたい

ゴール

- cdk8s が解決する課題について理解する
- cdk8s を今すぐに触ってみたい・試してみたい気持ちになる！

アジェンダ

Kuberenets におけるマニフェスト作成・管理運用

CDK for Kubernetes (cdk8s) について

cdk8s+ について

Demo

まとめ

cdk8s now in alpha!

and cdk8s+ is joining 

In Partnership with

aws DEV DAY
ONLINE JAPAN

Kuberenets における マニフェスト作成・管理運用

In Partnership with intel

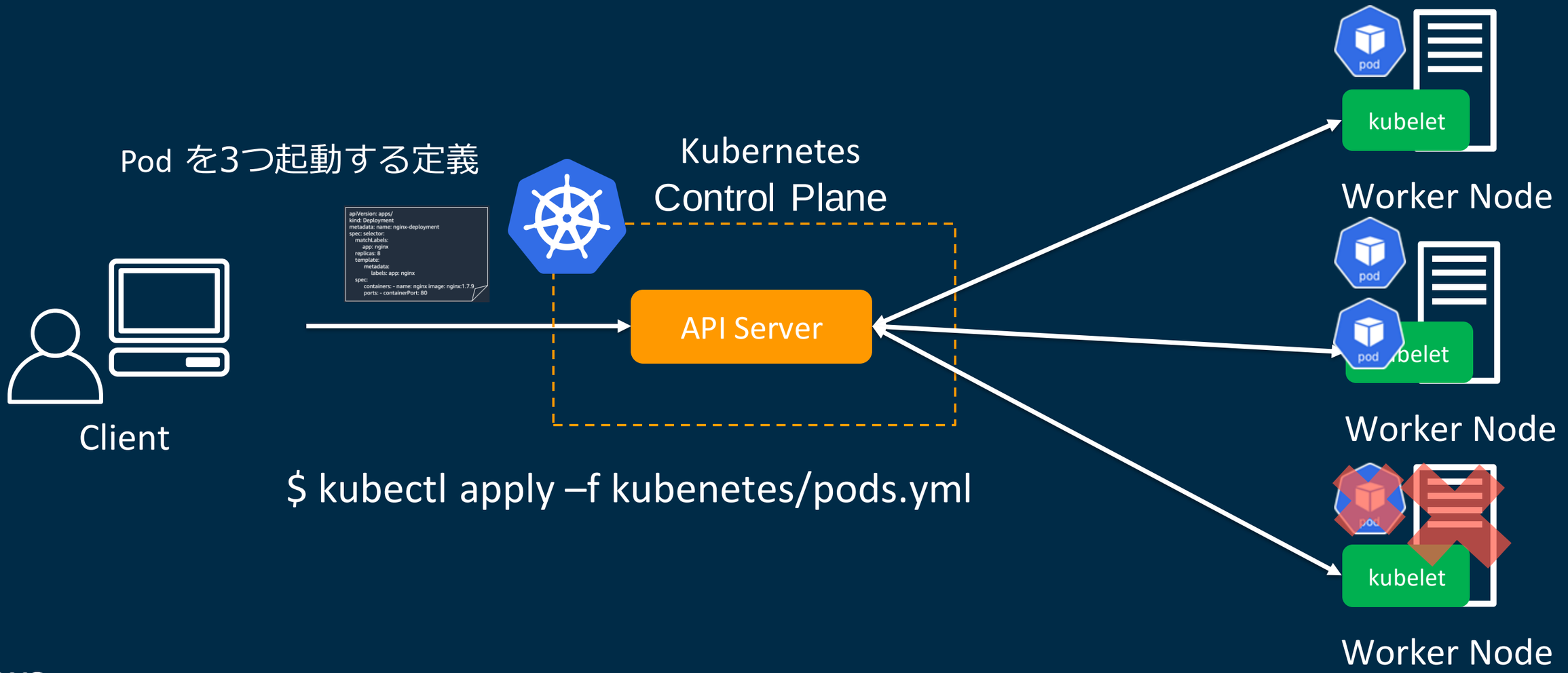
Kubernetes の概要

- 複数のホスト間でコンテナ化されたアプリケーションを管理するオープンソースシステム
- Cloud Native Computing Foundation (CNCF) によって管理、推進
- “Pod”、”Deployment”、“Service”、“Job” などのリソースに代表される高い表現力
- オーケストレーションツールとして拡張性が非常に高い

In Partnership with intel

YAML による宣言的デプロイメント

Kubernetes はマニフェストの定義からクラスタ全体をあるべき状態に収束させる



In Partnership with intel

Example - Kubernetes YAML

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-k8s
  labels:
    app: hello-k8s
  namespace: hamaan
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-k8s
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: hello-k8s
    spec:
      containers:
      - image: paulbouwer/hello-kubernetes:1.7
        imagePullPolicy: Always
        name: hello-kubernetes
        ports:
        - containerPort: 8080

---
apiVersion: v1
kind: Service
apiVersion: v1
metadata:
  name: hello-k8s
spec:
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: hello-k8s
  type: LoadBalancer
```

YAML as an easy DSL

読みやすい

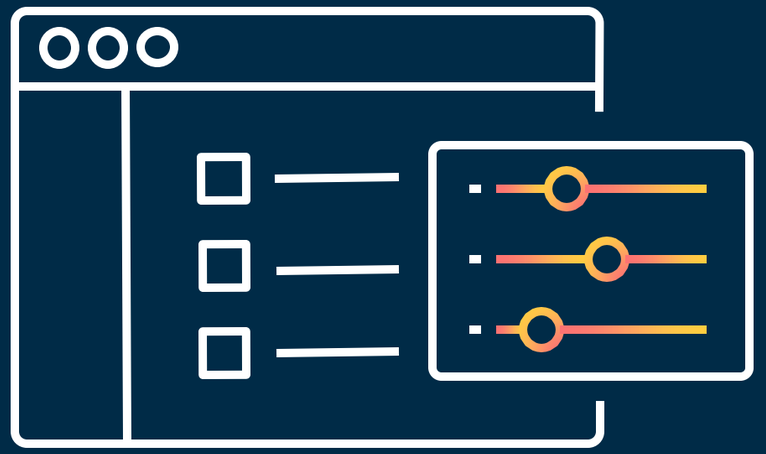
- 静的に定義が宣言されている
- インデントを使いデータの階層構造を定義

書きやすい

- 配列・スカラー・ハッシュ等表現力が豊富
- タグ等が不要

わかりやすい

- コメントを書くことができる



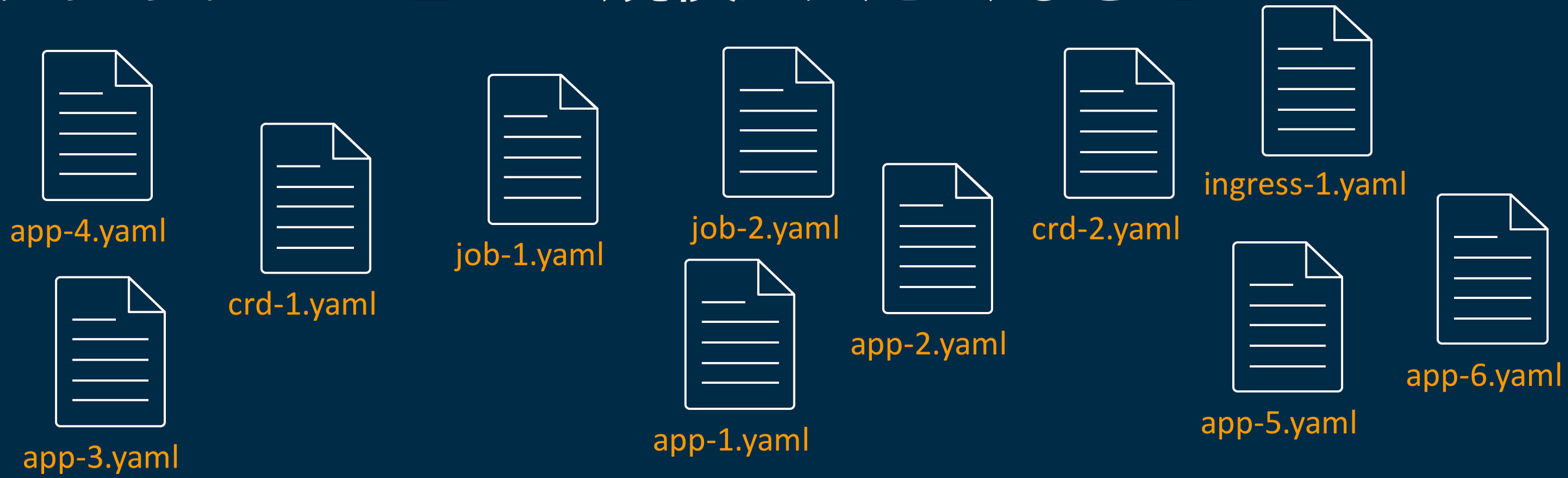
Kubernetes の YAML 管理における課題

- 必要な定義全てを宣言する必要があり、効率的に書けない
  - selector/label
  - containerPort/targetPort
- 表現力豊かな Kubernetes オブジェクトの定義を理解していないと書くのが難しい
- ユニットテストが難しいため、メンテナンスが辛くなる

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-k8s
  labels:
    app: hello-k8s
  namespace: hamaan
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-k8s
template:
    metadata:
      labels:
        app: hello-k8s
    spec:
      containers:
<pod-spec-ommitted-for-brevity>

---
```

```
apiVersion: v1
kind: Service
apiVersion: v1
metadata:
  name: hello-k8s
spec:
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: hello-k8s
  type: LoadBalancer
```

アプリケーションの規模が大きくなると…

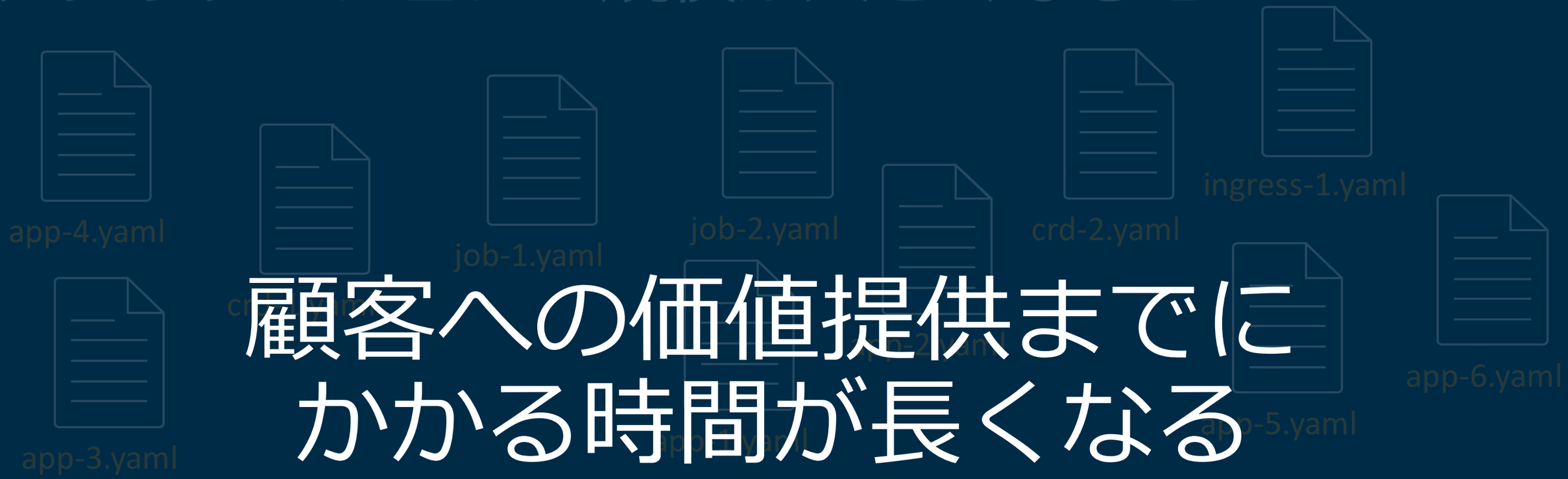


- 新しい app の構築にはボイラープレートを用意し、定義をコピーして手動調整
- ベストプラクティスの共有と更新を行うには、既存マニフェストの手作業での修正や作り直しが伴う

  -> 開発者にとってどんどん重荷になる

In Partnership with intel

アプリケーションの規模が大きくなると…



顧客への価値提供までに かかる時間が長くなる

- 新しい app の構築にはボイラープレートを用意し、定義をコピーして手動調整
- ベストプラクティスの共有と更新を行うには、既存マニフェストの手作業での修正や作り直しが伴う

  -> 開発者にとってどんどん重荷になる

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

CDK for Kubernetes (cdk8s) について

In Partnership with intel

AWS Cloud Development Kit (AWS CDK)

AWS の環境を一般のプログラミング言語で記述できるツールキット

Familiar
Your language
Just classes and methods

Tool Support
AutoComplete
Inline documentation

Abstraction
Sane defaults
Reusable classes

```
class UrlShortener extends Stack {
  constructor(scope: App, id: string, props?: UrlShortenerProps) {
    super(scope, id, props);

    const vpc = new ec2.Vpc(this, 'vpc', { maxAzs: 2 });
    const cluster = new ecs.Cluster(this, 'cluster', { vpc: vpc });
    const service = new patterns.NetworkLoadBalancedFargateService(this, 'sample-app', {
      cluster,
      taskImageOptions: {
        image: ecs.ContainerImage.fromAsset('ping'),
      },
      dom
    });
    // Setup AutoScaling policy
    const scaling = service.service.autoScaleTas
    scaling.scaleOnCpuUtilization('CpuScaling',
      targetUtilizationPercent: 50,
      scaleInCooldown: Duration.seconds(60),
      scaleOutCooldown: Duration.seconds(60)
    });
  }
}
```

domainName
domainZone

(property) patterns.NetworkLoadBalancedServiceBaseProps.domainName?: string | undefined

The domain name for the service, e.g. "api.example.com."

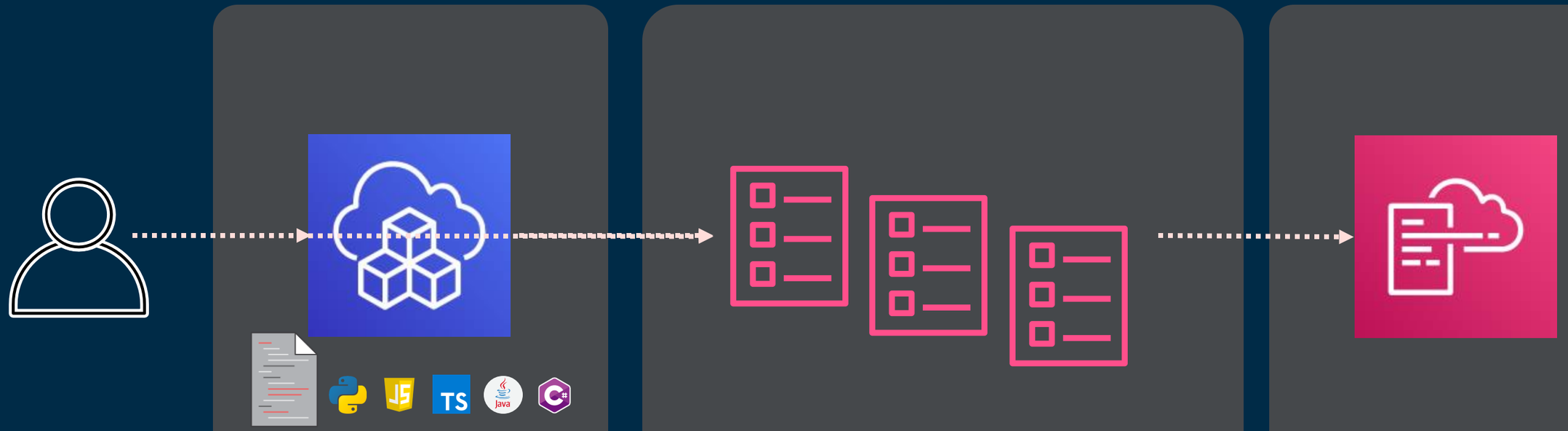
@default

- No domain name.

https://aws.amazon.com/cdk/

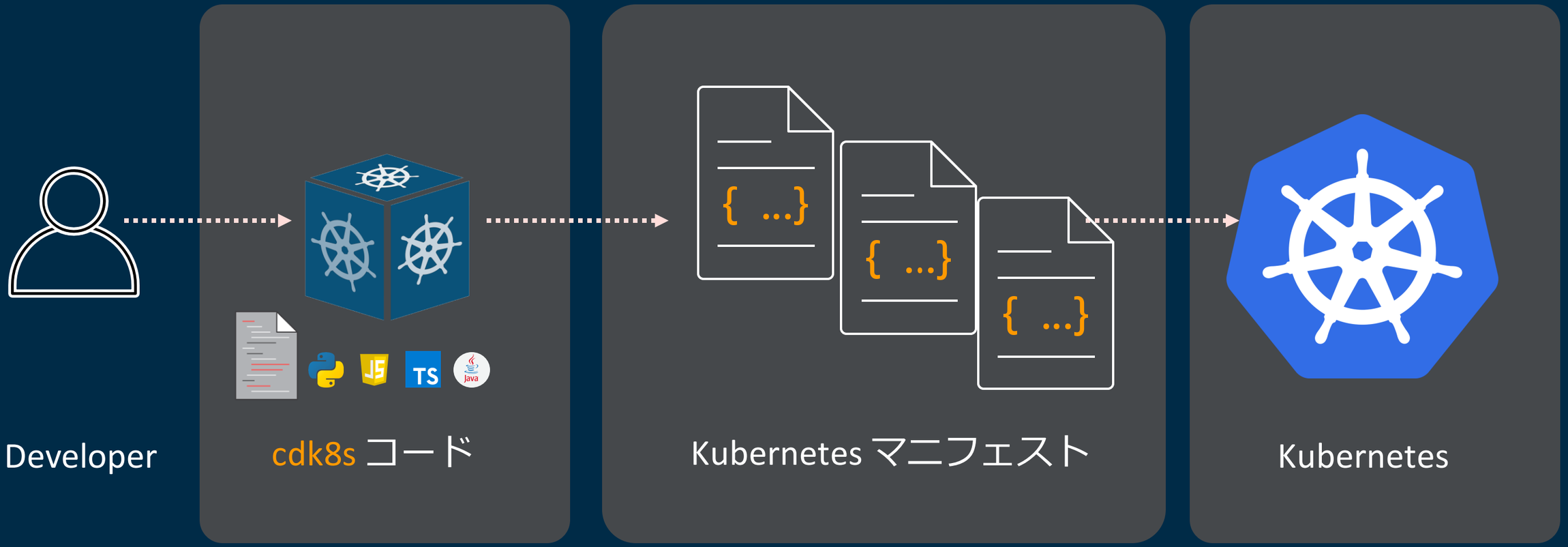
In Partnership with intel

AWS Cloud Development Kit (AWS CDK)



In Partnership with

Cloud Development Kit (CDK) for Kubernetes = cdk8s



AWS CDK と同様の設計概念ですべての Kubernetes ユーザーを支援する

In Partnership with intel

Cloud Development Kit (CDK) for Kubernetes = cdk8s

一般のプログラミング言語で Kubernetes のマニフェストファイルを生成できるツールキット

- ソースコードから Kubernetes のマニフェスト YAML を生成
  - 制御構文やクラス・継承などの概念で効率的に書くことが可能
  - エディタによる型チェック、サジェスト、API 仕様の参照
- オープンソースとして開発
  - ベストプラクティスを定義した Construct として拡張・共有
- 任意の Kubernetes クラスターで利用可能
- 言語サポート
  - TypeScript/JavaScript、 Python、 Java

New

- 任意の Kubernetes API バージョンとカスタムリソースを使用可能

cdk8s

https://cdk8s.io/

aws

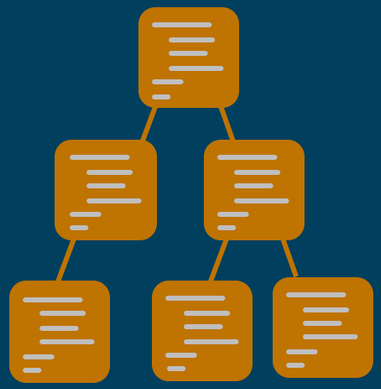
In Partnership with intel

Main Components

App

Charts

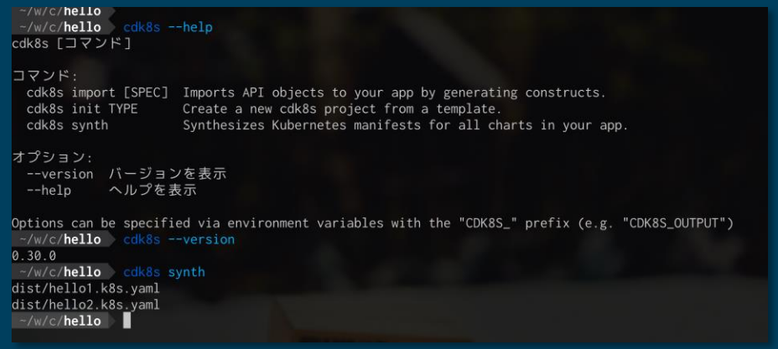
Constructs



Core Framework

...etc

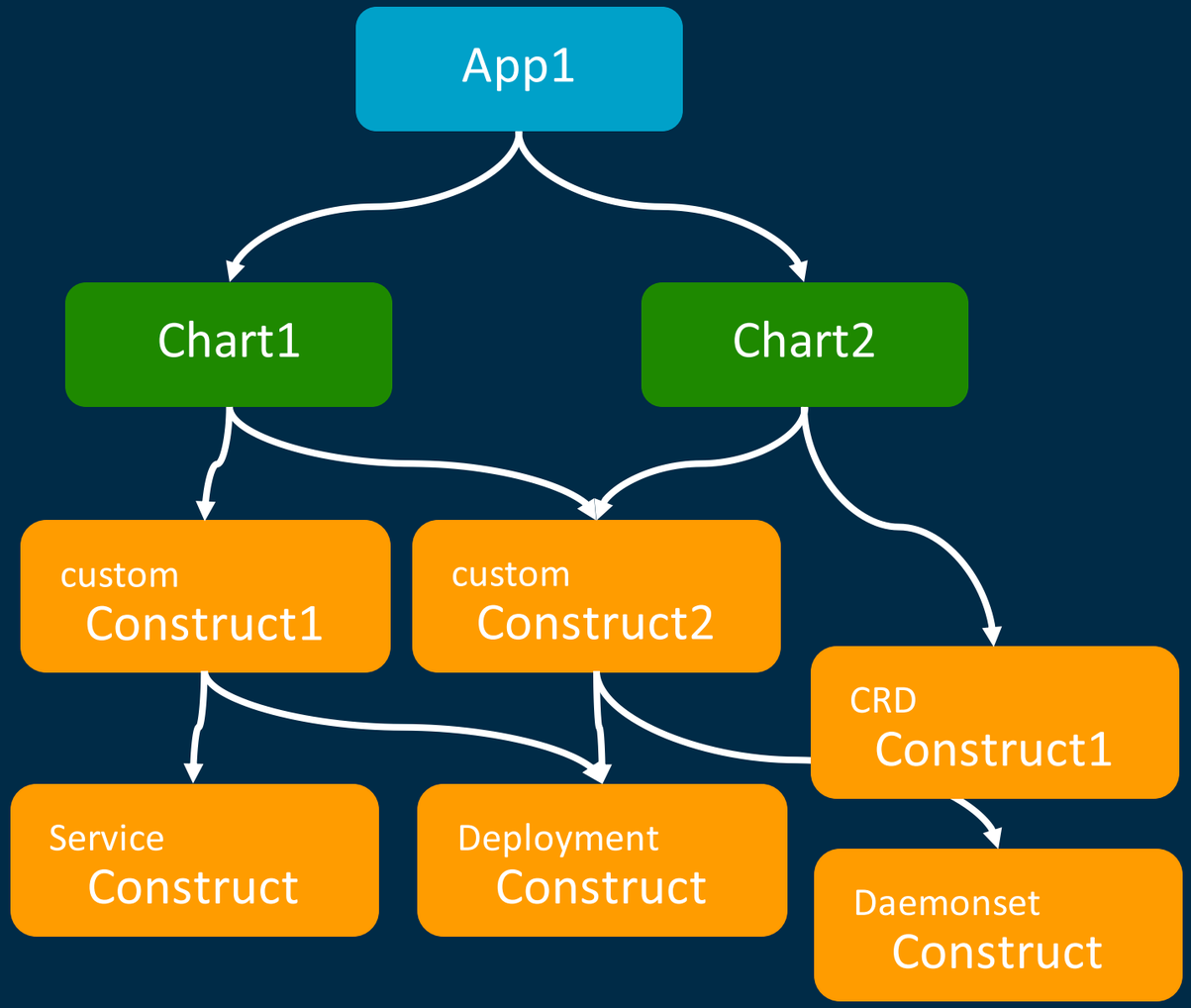
K8s API objects(constructs)



cdk8s-cli

In Partnership with

cdk8s アプリケーションの構成



App

- Kubernetes マニフェスト YAML の生成に利用する最上位要素
- 複数の Chart とその依存関係を定義

Chart

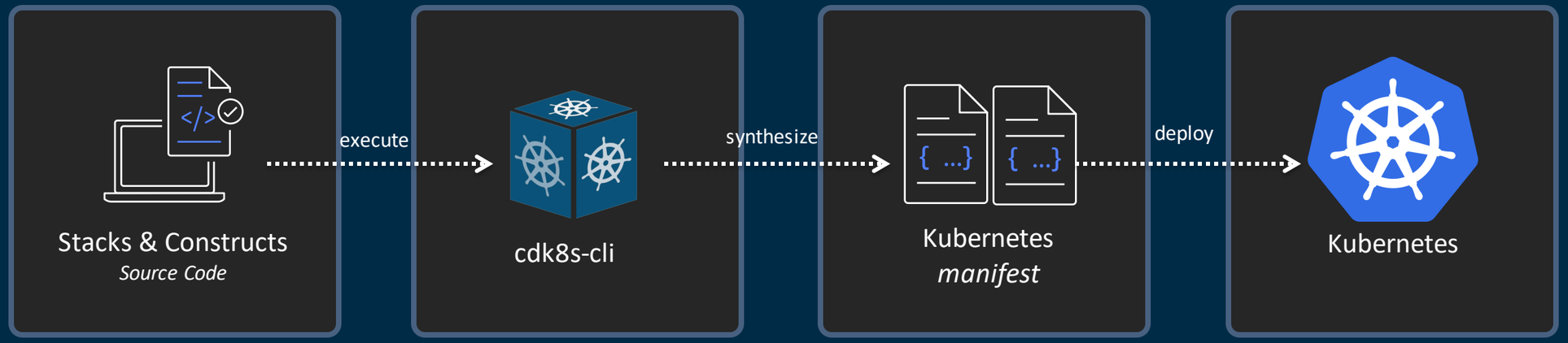
- 1つの Kubernetes マニフェスト YAML を表す要素
- 複数の Construct とその依存関係を定義

Construct

- cdk8s アプリの基本ビルディングブロック
- K8s の単一もしくは複数リソースをカプセル化
- 独自に定義したり配布することも可能

In Partnership with intel

Development Workflow




```
cdk8s init typescript-app          // create new project
npm run compile && npm run test    // compile and test
cdk8s synth                        // create manifest yaml
kubectl apply –f dist -R           // push changes to K8s
```


In Partnership with

cdk8s のテスト

- 一般のプログラミング言語同様テストコードによるユニットテストが行いやすい

CDK のテストの cdk8s での対応状況

- Snapshot tests (golden master tests) = 利用可能
  - あるべきテンプレート/マニフェスト 全体を用意し、コードで作成したテンプレートが一致することを確認
- Validation tests = 利用可能
  - Construct に与えられたパラメータが正しいことを検証するなど一般的なユニットテスト
  - expect(...).toThrowError() など
- Fine-grained assersions = On Roadmap(https://github.com/awslabs/cdk8s/issues/169)
  - 作成したテンプレート/マニフェストの一部をチェックし、指定したリソースが特定のプロパティを持つことを確認
  - CDK だと、expect(stack).toHaveResource('AWS::SQS::Queue', {プロパティ}) の形で検証

https://github.com/awslabs/cdk8s/tree/master/test

cdk8s のその他便利な機能

- 既存の YAML ファイルを Include して利用可能
  - Include Construct が用意されている
  - https://awscdk.io/packages/cdk8s@0.30.0#/./cdk8s-readme?id=include

```
import { Include } from 'cdk8s’;
・・・・
new Include(this, 'dashboard', {
      url: 'https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml’,
      // or
      url: `${__dirname}/dashboard.yaml`
});
```

- Construct 間や、 Chart 間で依存関係の定義が可能
  - ex) service.addDependency(namespace)

```
const namespace = new k8s.Namespace(chart, 'backend’);
const service = new k8s.Service(chart, 'Service', { metadata: { namespace: namespace.name }});

service.addDependency(namespace);
```

aws DEV DAY
ONLINE JAPAN

cdk8s+ について

In Partnership with intel

cdk8s vs Kubernetes マニフェスト

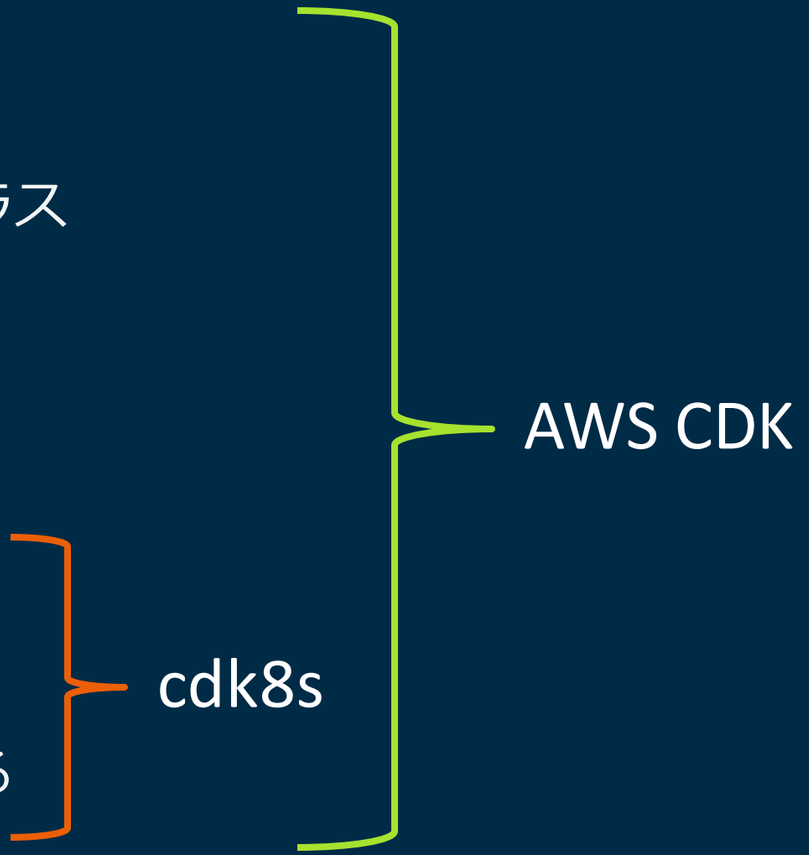
```
const label = { app: 'hello-k8s' };

new Service(this, 'service', {
  spec: {
    type: 'LoadBalancer',
    ports: [ { port: 80, targetPort: IntOrString.fromNumber(8080) } ],
    selector: label
  }
});

new Deployment(this, 'deployment', {
  spec: {
    replicas: 2,
    selector: {
      matchLabels: label
    },
    template: {
      metadata: { labels: label },
      spec: {
        containers: [
          {
            name: 'hello-kubernetes',
            image: 'paulbouwer/hello-kubernetes:1.7'
            ports: [ { containerPort: 8080 } ]
          }
        ]
      }
    }
  }
});
```

YAML で書いていたのと同様の内容をコードで記述しており、あくまでプロパティと 1:1 の関係のまま

In Partnership with intel

Constructs Library

- High-level constructs (L2)
  - デフォルト値や便利なメソッドを定義したリソースを表すクラス
  - ex） AWS: クラス s3 は メソッド s3.Bucket.addLifeCycleRule() を持つ
- Low-level constructs (L1)
  - YAML 上のリソースおよびプロパティと 1:1 で対応する
  - 実装の中ですべてのプロパティを明示的に指定する必要がある



https://docs.aws.amazon.com/cdk/latest/guide/constructs.html

cdk8s+

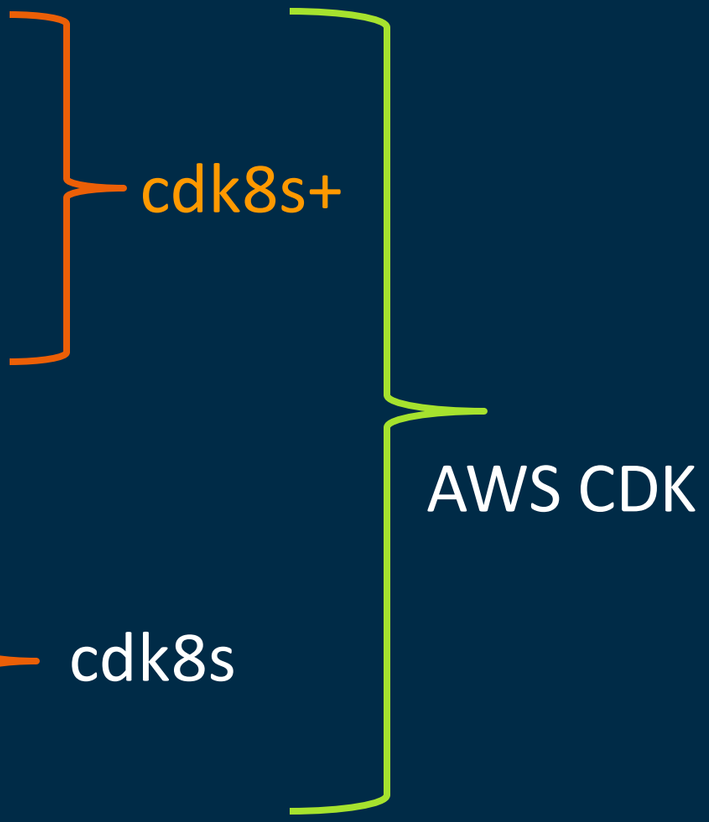
- cdk8s における High-level constructs (L2) なライブラリ
- Kubernetes v1.17 以上に対応
- 現在は Experimental ステージ

https://github.com/awslabs/cdk8s/tree/master/packages/cdk8s-plus

In Partnership with intel

Constructs Library

- High-level constructs (L2)
  - デフォルト値や便利なメソッドを定義したリソースを表すクラス
  - ex） AWS: クラス s3 は メソッド s3.Bucket.addLifeCycleRule() を持つ
    Kubernetes: クラス deployment は メソッド deployment.expose() を持つ
- Low-level constructs (L1)
  - YAML 上のリソースおよびプロパティと 1:1 で対応する
  - 実装の中ですべてのプロパティを明示的に指定する必要がある



https://docs.aws.amazon.com/cdk/latest/guide/constructs.html

In Partnership with intel

Example - cdk8s+ code

```typescript
const deployment = new kplus.Deployment(this, 'MyApp', {
  replicas: 3,
  containers: [new kplus.Container({
    image: 'node',
    port: 9000
  })],
});

// this will internally create a service
deployment.expose(8000);
```

$ cdk8s synth

```yaml
kind: Deployment
apiVersion: apps/v1
spec:
  replicas: 3
  selector:
    matchLabels:
      cdk8s.deployment: MyAppC6A88652
  template:
    metadata:
      labels:
        cdk8s.deployment: MyAppC6A88652
    spec: <pod-spec-ommitted-for-brevity>
---
kind: Service
apiVersion: v1
spec:
  type: ClusterIP
  ports:
    - port: 8000
      targetPort: 9000 # this is the port exposed by container.
  selector:
    cdk8s.deployment: MyAppC6A88652
```

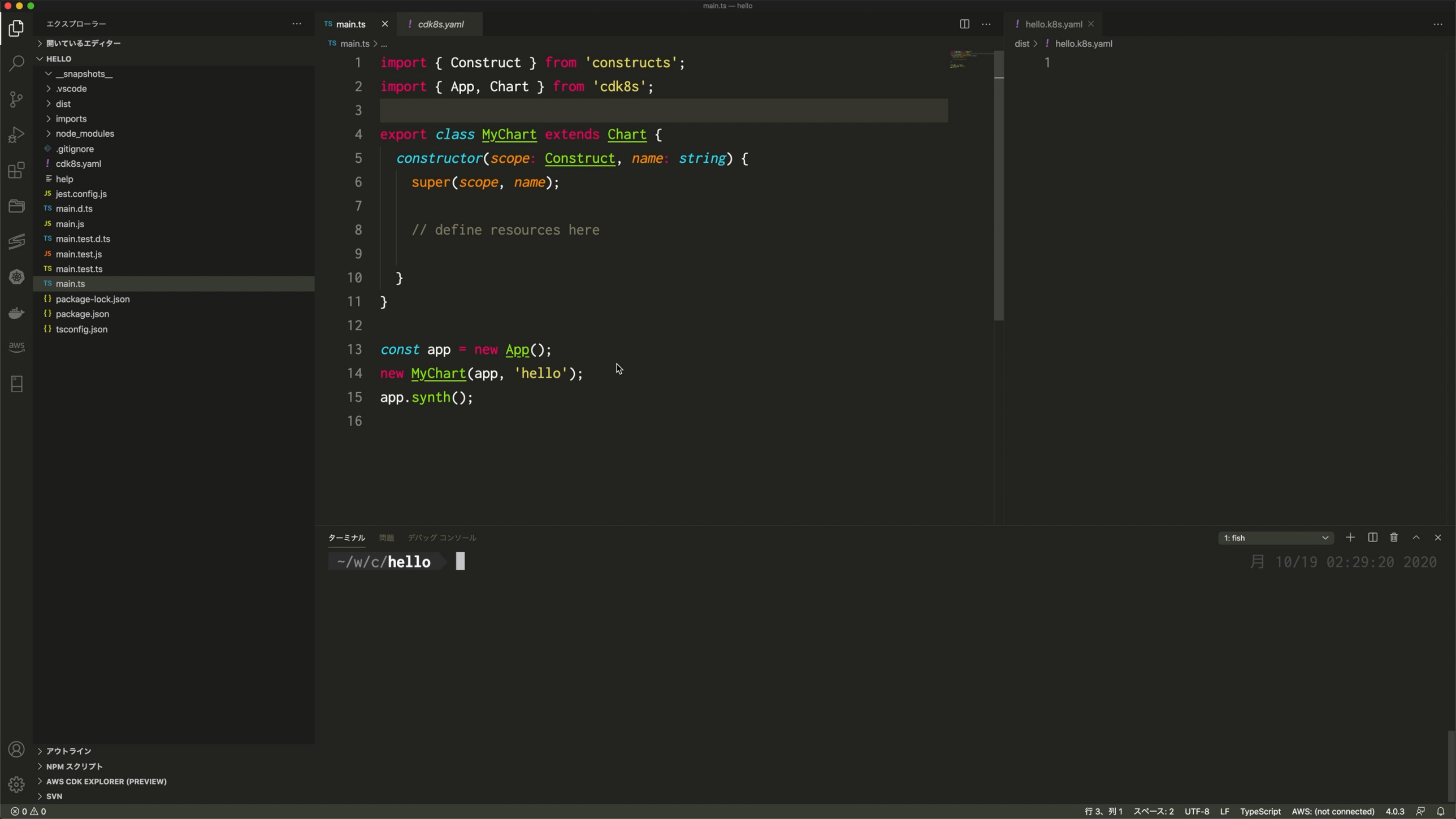
- 9000 ポートで接続可能な Pod (Deployment) を 8000 ポートで公開する (expose) という直感的な実装
- Selector/label の明示的な定義を抽象化

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

Demo

In Partnership with intel


main.ts — hello
エクスプローラー
開いているエディター
HELLO
__snapshots__
.vscode
dist
imports
node_modules
.gitignore
cdk8s.yaml
help
jest.config.js
main.d.ts
main.js
main.test.d.ts
main.test.js
main.test.ts
main.ts
package-lock.json
package.json
tsconfig.json
main.ts
cdk8s.yaml
main.ts > ...
1 import { Construct } from 'constructs';
2 import { App, Chart } from 'cdk8s';
3
4 export class MyChart extends Chart {
5 constructor(scope: Construct, name: string) {
6 super(scope, name);
7
8 // define resources here
9
10 }
11 }
12
13 const app = new App();
14 new MyChart(app, 'hello');
15 app.synth();
16
hello.k8s.yaml
dist > hello.k8s.yaml
1
ターミナル
問題
デバッグ コンソール
1: fish
~/w/c/hello
月 10/19 02:29:20 2020
アウトライン
NPM スクリプト
AWS CDK EXPLORER (PREVIEW)
SVN
0 0
行 3、列 1
スペース: 2
UTF-8
LF
TypeScript
AWS: (not connected)
4.0.3

(2020/10/19) cdk8s v0.31.0 がリリース

New!

- cdk8s+: すべての cdk8s+ Construct から spec オブジェクトが削除され、フラットな構造に

```
// before

const deployment = new kplus.Deployment(chart, 'Deployment', {
  spec: {
    replicas: 3,
    podSpecTemplate: {
      containers: [ container ]
    }
  },
});
```

```
// now

const deployment = new kplus.Deployment(chart, 'Deployment', {
  replicas: 3,
  containers: [ container ]
});
```

- cdk8s+: deployment.expose() メソッドで port が位置引数に
  - before: deployment.expose({port:8080,…})
  - now: deployment.expose(8080, {…})

https://github.com/awslabs/cdk8s/releases/tag/v0.31.0

In Partnership with intel

(2020/10/19) cdk8s v0.31.0 がリリース

New!

- cdk8s+: service.addDeployment() が利用可能に
- cdk8s+: Ingress を定義することが可能に
- cdk8s-lib: Helm Construct が利用可能に (Helm Chart の Include)
- ・・・その他機能追加や bug Fixes など

https://github.com/awslabs/cdk8s/releases/tag/v0.31.0

まとめ

- Kubernetes マニフェストを YAML で管理する上での課題
  - 開発者にとって冗長な書き方も多く、もっと効率的に書きたい
  - アプリケーションの成長とともに YAML ファイルの管理が複雑化する
- cdk8s は一般のプログラミング言語をインターフェースとして Kubernete マニフェスト YAML を生成管理できるツールキット
- cdk8s+ を使うことで Intent-driven な実装で Kubernetes マニフェスト YAML を生成することが可能
- cdk8s は現在 alpha stage な OSS。今後 GA に向けて乞うご期待
  - ぜひ一緒に良いものをつくっていきましょう！

In Partnership with intel

ドキュメントと参考資料

公式サイト

- https://cdk8s.io/

GitHub

- cdk8s：https://github.com/awslabs/cdk8s
- cdk8s+(cdk8s-plus)：https://github.com/awslabs/cdk8s/tree/master/packages/cdk8s-plus

ドキュメント

- cdk8s-Library：https://awscdk.io/packages/cdk8s@0.30.0#/
- cdk8s-cli：https://github.com/awslabs/cdk8s/blob/v0.30.0/packages/cdk8s-cli/README.md
- cdk8s+：https://awscdk.io/packages/cdk8s-plus@0.30.0/#/

Example code (Typescript, Python, Java)

- https://github.com/awslabs/cdk8s/tree/master/docs/getting-started

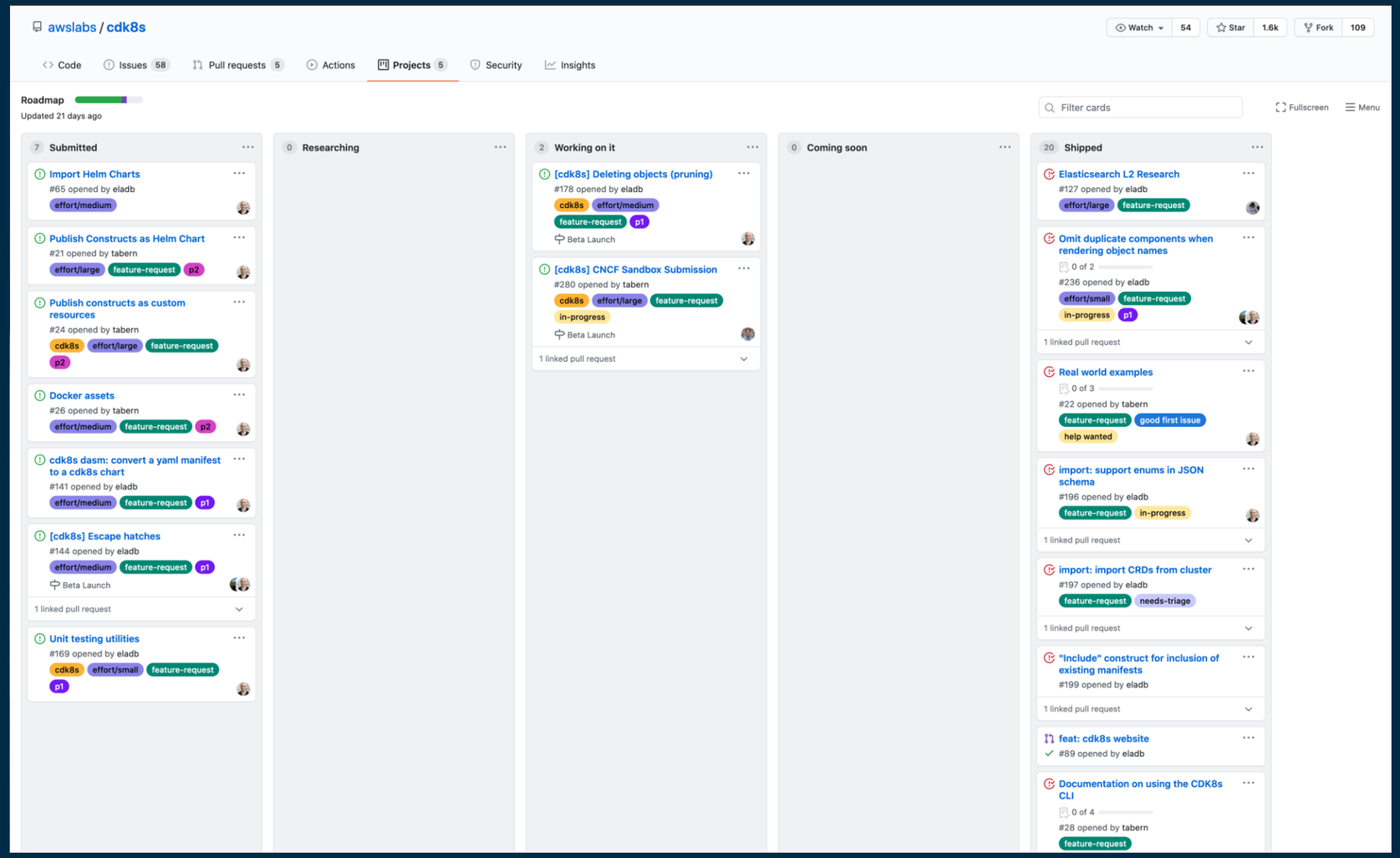
動画

- End YAML engineering with cdk8s! ： https://youtu.be/QcF_6ZSEd5k
- CDK for Kubernetes - No more YAML engineering with cdk8s： https://youtu.be/PdR3SlkwjLM
- Saying Goodbye to YAML Engineering with the CDK for Kubernetes： https://youtu.be/1PJqAYqxHio

blog 等

- CDK for Kubernetes のご紹介：
  https://aws.amazon.com/jp/blogs/news/introducing-cdk-for-kubernetes/
- Introducing cdk8s+: Intent-driven APIs for Kubernetes objects ：
  https://aws.amazon.com/jp/blogs/containers/introducing-cdk8s-intent-driven-apis-for-kubernetes-objects/
- cdk8s, the future of Kubernetes application deployments? ：
  https://brennerm.github.io/posts/cdk8s-the-future-of-k8s-application-deployments.html
- Integrating cdk8s with Argo CD ：
  https://brennerm.github.io/posts/integrating-cdk8s-with-argocd.html
- Integrating cdk8s with Flux ：
  https://brennerm.github.io/posts/integrating-cdk8s-with-flux.html
- Awesome cdk8s ：
  https://github.com/dungahk/awesome-cdk8s

cdk8s Roadmap



https://github.com/awslabs/cdk8s/projects/1

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

アンケートへの回答を
お願いいたします。

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

Thank you!

Shinichi Hama
track3jyo

In Partnership with intel