Openshift Operators

Saif Ur Rehman

IBM Cloud

Operators

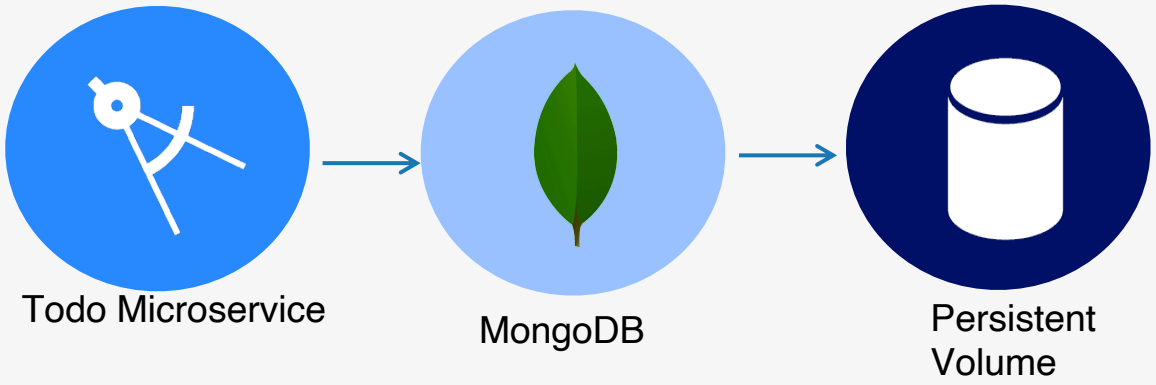
Containers bought simplicity to development world ….

- $ docker pull postgres
- $ docker pull redis
- $ docker run --name redis –d redis
- $ docker run – name postgres –e POSTGRES-PASSWORD=foo –d postgres

Need a Orchestrator

- Need a platform like Kubernetes to deploy production grade containers
- Abstracted complexities such as
  - Deployments
  - Statefulsets
  - Configmaps, Secrets
  - Daemonsets
  - Replication Controller
  - Autoscaling
  - And much more ....

Todo App on Openshift



- Application code

crmController.ts

```ts
public getTodo(req: Request, res: Response) {
  res.setHeader("Content-Type", "application/json");
  Todo.find({}, (err, todo) => {
    if (err) {
      res.status(404).json({ err });
      return;
    } else {
      res.status(200).send(todo);
    }
  });
}
public searchTodo(req: Request, res: Response) {
  res.setHeader("Content-Type", "application/json");
  let query = {
    $and:[{
      'Username': req.body.Username,
    }]
  };
  Todo.find( query , (err, todo) => {
    if (err) {
      res.status(404).json({ err });
      return;
    } else {
      res.status(200).send(todo);
    }
  });
}
public getTodoById(req: Request, res: Response) {
  res.setHeader("Content-Type", "application/json");
  Todo.findById(req.params.ID, (err, todo) => {
    if (err) {
      res.status(404).json({ err });
      return;
    } else {
      res.status(200).send(todo);
    }
  });
}
public updateTodo(req: Request, res: Response) {
  res.setHeader("Content-Type", "application/json");
  Todo.findOneAndUpdate(
    { _id: req.params.ID },
    req.body,
    { new: true },
```

Untitle.json ● | imagestream.yaml | buildconfig.yaml | configmap.yaml | deployment.yaml

```ts
import bodyParser from 'body-parser';
import session from 'express-session';
import ListingMongoController from './controllers/listing/index';
var mongoose = require('mongoose');
mongoose.Promise = Promise;
class App {
  public express: express.Application;
  public mongoUrl: string = 'mongodb://'+process.env.MONGOUSERNAME+':'+process.env.MONGOPASSWORD+'@

  constructor() {
    this.express = express();
    this.middleware();
    this.routes();
    this.mongoSetup();
  }
  private mongoSetup(): void{
    mongoose.Promise = global.Promise;
    mongoose.connect(this.mongoUrl);
}
  private middleware(): void {
    this.express.use(function(req, res, next) {
      res.header("Access-Control-Allow-Origin", "*");
      res.header("Access-Control-Allow-Headers", "X-Requested-With,content-type");
      res.header("Access-Control-Allow-Methods", "GET, POST, OPTIONS, PUT, PATCH, DELETE");
      next();
    });
    this.express.use(bodyParser.json());
    this.express.use(bodyParser.urlencoded({ extended: false }));
    this.express.use(session({secret: 'test123', saveUninitialized: false, resave: true}))
  }
  private routes(): void {
    this.express.get('/', function(_, res) {
      res.send('hi');
    });
    this.express.use('/', ListingMongoController);
  }
}
export default new App().express;
```

IBM

- Setting build environment (webpack, tsconfig.json etc)

Dockerfile | webpack.config.js × | values.yaml | service.yaml | crmCon

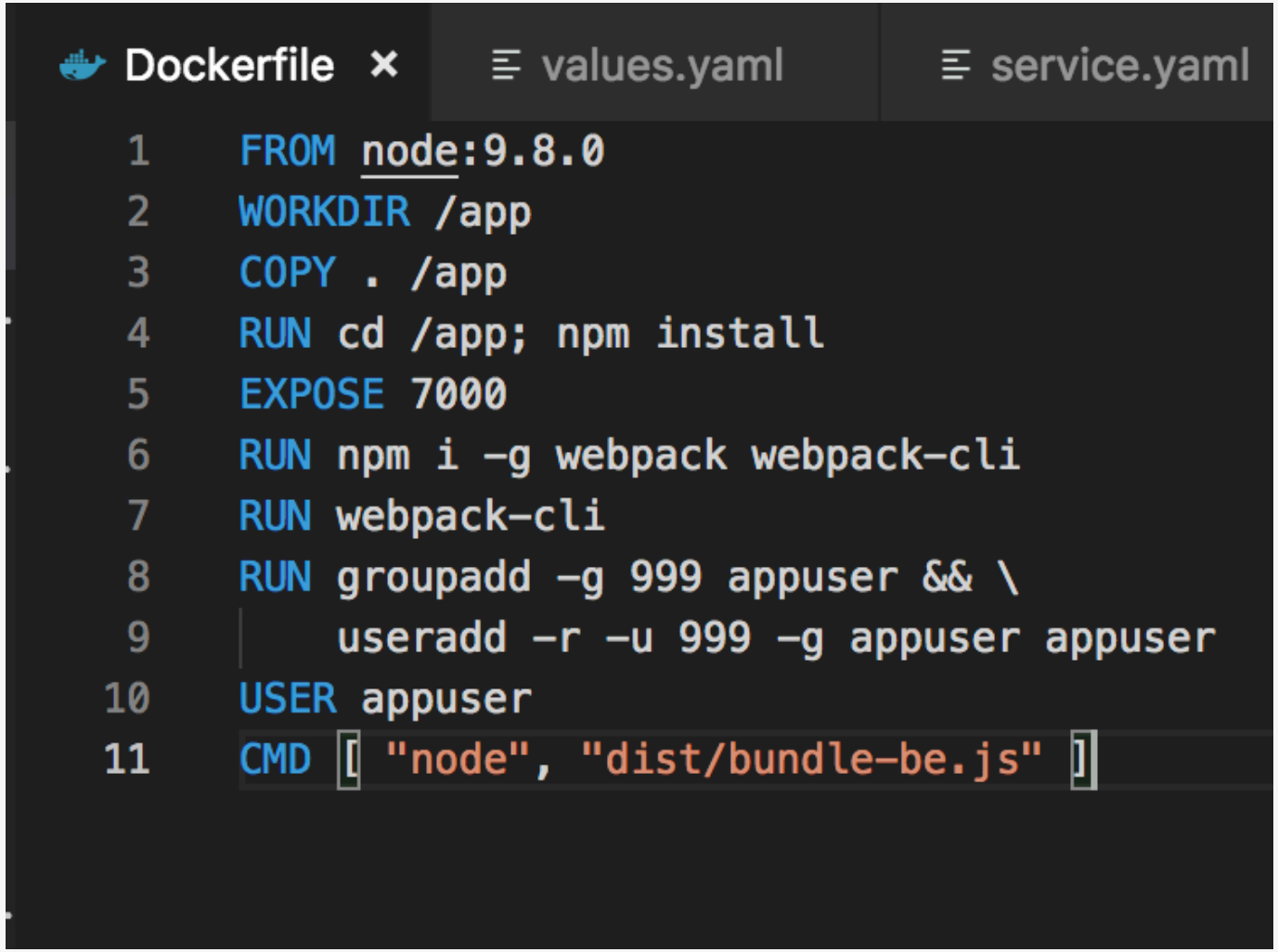
```
const path = require('path');
const CleanWebpack = require('clean-webpack-plugin');
const nodeExternals = require('webpack-node-externals');
const webpack = require('webpack');

const BACKENDSRC = path.resolve(__dirname, 'src');

module.exports = {
  target: 'node',
  // @babel/polyfill is needed to use modern js functionalities in old browsers.
  entry: ['@babel/polyfill', path.resolve(BACKENDSRC, 'index.ts')],
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'bundle-be.js',
    publicPath: path.join(__dirname, 'dist')
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        use: {
          loader: 'babel-loader',
          options: {
            // To process async functions.
            plugins: ['@babel/plugin-transform-async-to-generator']
          }
        },
        exclude: /(node_modules|bower_components)/
      },
      {
        test: /.ts$/,
        loaders: ['ts-loader'],
        exclude: /(node_modules|bower_components)/
      }
    ]
  },
  resolve: {
    modules: ['node_modules', BACKENDSRC, path.resolve(__dirname, 'helpers')],
    extensions: ['.js', 'web.js', 'webpack.js', '.ts', '.tsx']
  },
  plugins: [
    new CleanWebpack([path.resolve('dist', 'bundle-be.js')]),
    new webpack.HotModuleReplacementPlugin()
  ],
```

Dockerfile | tsconfig.json × | values.yaml | service.yaml

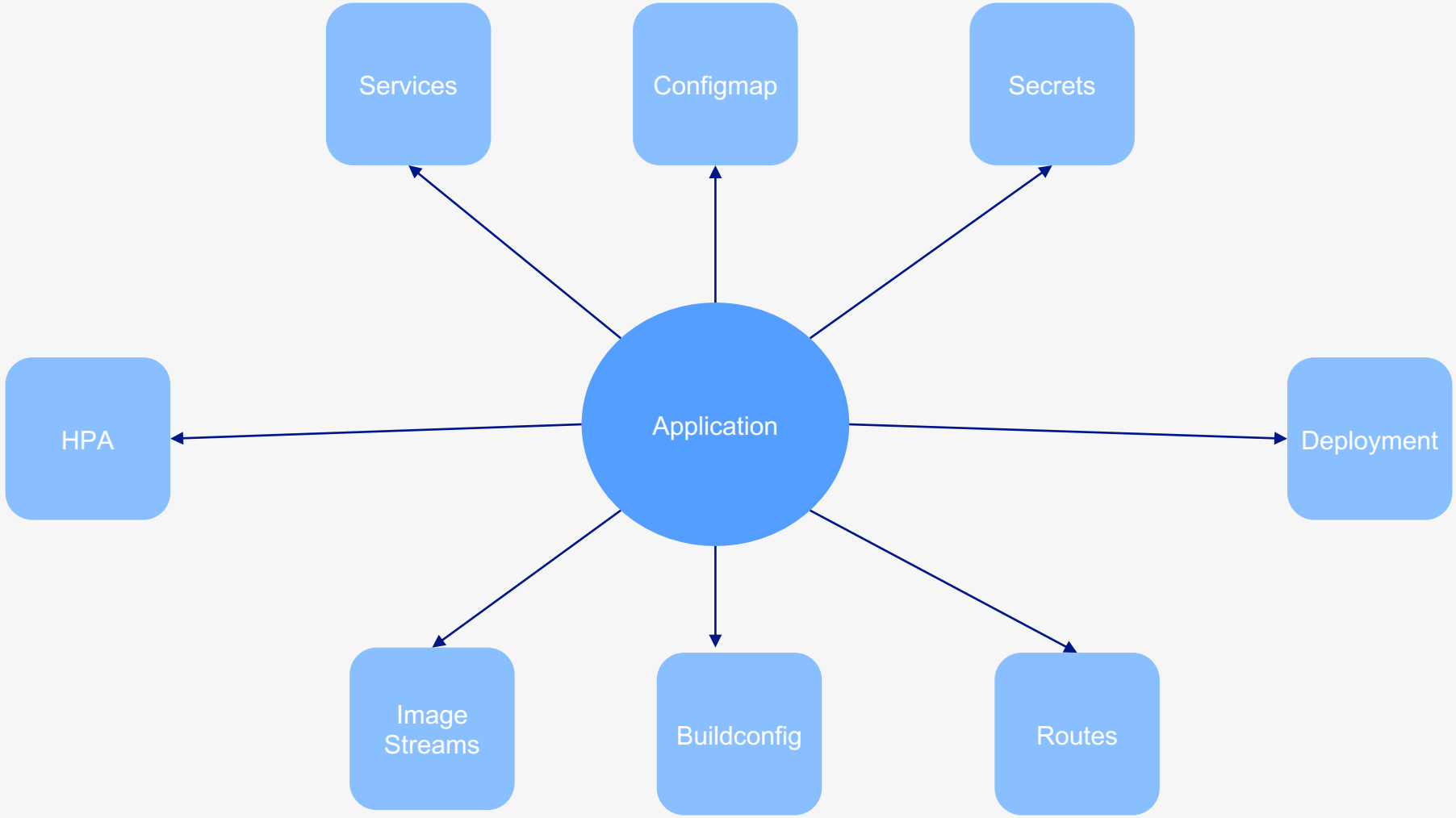
```
{
  "compilerOptions": {
    "types": ["reflect-metadata"],
    "moduleResolution": "node",
    "experimentalDecorators": true,
    "emitDecoratorMetadata": true,
    /* Basic Options */
    // "strict": true /* Enable all strict type-checking options.
    // "noImplicitAny": true /* Raise error on expressions and dec
    "outDir": "./dist/" /* Redirect output structure to the direct
    "module": "es6" /* Specify module code generation: 'none', 'co
    "target": "es5" /* Specify ECMAScript target version: 'ES3' (d
    "jsx": "react" /* Specify JSX code generation: 'preserve', 're
    "allowJs": true /* Allow javascript files to be compiled. */,
    "moduleResolution": "node" /* Specify module resolution strate
    "typeRoots": [
      "node_modules/@types/*"
    ] /* List of folders to include type definitions from. */,
    "lib": [
      "es2015"
    ] /* Specify library files to be included in the compilation.
    // "checkJs": true,                       /* Report errors in
    // "declaration": true,                   /* Generates corresp
    // "declarationMap": true,                /* Generates a sourc
    // "sourceMap": true,                     /* Generates corresp
    // "outFile": "./",                       /* Concatenate and e
    // "rootDir": "./",                       /* Specify the root
    // "composite": true,                     /* Enable project co
    // "removeComments": true,                /* Do not emit comme
    // "noEmit": true,                        /* Do not emit outpu
    // "importHelpers": true,                 /* Import emit helpe
    // "downlevelIteration": true,            /* Provide full supp
    // "isolatedModules": true,               /* Transpile each fi

    /* Strict Type-Checking Options */
    // "strictNullChecks": true,              /* Enable strict nul
    // "strictFunctionTypes": true,           /* Enable strict che
    // "strictPropertyInitialization": true,  /* Enable strict che
    // "noImplicitThis": true,                /* Raise error on 't
    // "alwaysStrict": true,                  /* Parse in strict m

    /* Additional Checks */
    // "noUnusedLocals": true,                /* Report errors on
    // "noUnusedParameters": true,            /* Report errors on
```

IBM

- Dockerfile


Dockerfile
values.yaml
service.yaml
1 FROM node:9.8.0
2 WORKDIR /app
3 COPY . /app
4 RUN cd /app; npm install
5 EXPOSE 7000
6 RUN npm i -g webpack webpack-cli
7 RUN webpack-cli
8 RUN groupadd -g 999 appuser && \
9 useradd -r -u 999 -g appuser appuser
10 USER appuser
11 CMD ["node", "dist/bundle-be.js"]

- Kubernetes


Services
Configmap
Secrets
HPA
Application
Deployment
Image Streams
Buildconfig
Routes

- Kubernetes

alues.yaml | service.yaml × | crmController.ts | index.ts | mongores

```yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ .Values.metadata.name }}
  labels:
    app: {{ .Values.metadata.name }}
spec:
  type: {{ .Values.service.servicePortConfiguration.type}}
  selector:
    app: {{ .Values.metadata.name }}
  ports:
  - protocol: {{ .Values.service.servicePortConfiguration.protocol}}
    name: {{ .Values.service.servicePortConfiguration.name}}
    port: {{ .Values.service.servicePortConfiguration.port}}
    targetPort: {{ .Values.service.servicePortConfiguration.targetPort}}
```

Go Debug Terminal Window Help

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: {{ .Values.metadata.name }}
  labels:
    app: "{{ .Values.metadata.name }}"
  namespace: "{{ .Values.namespace }}"
type: Opaque
data:
  SECRET: "{{ .Values.mongodb.secret }}"
  MONGOUSERNAME: "{{ .Values.mongodb.username }}"
  MONGOPASSWORD: "{{ .Values.mongodb.password }}"
```

IBM

• Kubernetes

t_v1alpha1_mongorest_cr.yaml ● {} Untitle.json ● imagestream.y

```yaml
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  annotations:
    openshift.io/generated-by: OpenShiftWebConsole
    openshift.io/host.generated: "true"
  labels:
    app: "{{ .Values.metadata.name }}"
  name: "{{ .Values.metadata.name }}"
  namespace: "{{ .Values.namespace }}"
spec:
  host: "{{ .Values.routes.host }}"
  port:
    targetPort: {{ .Values.routes.targetPort }}
  to:
    kind: Service
    name: {{ .Values.metadata.name }}
    weight: {{ .Values.routes.weight }}
  wildcardPolicy: None
```

t_v1alpha1_mongorest_cr.yaml ● {} Untitle.json ● imagestream.yaml

```yaml
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
  name: {{ .Values.metadata.name }}
spec:
  replicas: {{ .Values.replicaCount }}
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: {{ .Values.image.maxSurge }}
      maxUnavailable: {{ .Values.image.maxUnavailable }}
  triggers:
    - type: "ConfigChange"
    - type: "ImageChange"
      imageChangeParams:
        automatic: true
        containerNames:
          - {{ .Values.metadata.name }}
        from:
          kind: "ImageStreamTag"
          name: {{ .Values.buildconfig.imagename }}
  strategy:
    type: "Rolling"
  paused: false
  template:
    metadata:
      name: {{ .Values.metadata.name }}
      labels:
        app: {{ .Values.metadata.name }}
    spec:
      containers:
        - name: {{ .Values.metadata.name }}
          image: {{ .Values.metadata.name }}
          readinessProbe:
            httpGet:
              path: {{ .Values.readinessProbe.path }}
              port: {{ .Values.readinessProbe.port }}
              scheme: HTTP
            initialDelaySeconds: {{ .Values.readinessProbe.ini
            timeoutSeconds: {{ .Values.readinessProbe.timeoutS
            periodSeconds: {{ .Values.readinessProbe.periodSec
          livenessProbe:
            httpGet:
              path: {{ .Values.livenessProbe.path }}
```

- Kubernetes

t_v1alpha1_mongorest_cr.yaml ● {} Untitle.json ● image

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: "{{ .Values.metadata.name }}"
  name: {{ .Values.metadata.name }}
  namespace: "{{ .Values.namespace }}"
data:
  MONGOPORT: "{{ .Values.mongodb.port }}"
  HOSTNAMEMONGODB: "{{ .Values.mongodb.host }}"
  ENDPOINT: "{{ .Values.mongodb.endpoint }}"
  DB: "{{ .Values.mongodb.db }}"
  SCHEMA: "{{ .Values.mongodb.schema }}"
  MODEL: "{{ .Values.mongodb.model }}"
```

t_v1alpha1_mongorest_cr.yaml ● {} Untitle.json ● imagestrea

```yaml
apiVersion: build.openshift.io/v1
kind: BuildConfig
metadata:
  name: {{ .Values.metadata.name }}
  labels:
    app: "{{ .Values.metadata.name }}"
  annotations:
    openshift.io/generated-by: OpenShiftWebConsole
spec:
  output:
    to:
      kind: ImageStreamTag
      name: {{ .Values.buildconfig.imagename }}
  runPolicy: Serial
  source:
    git:
      ref: {{ .Values.buildconfig.branch }}
      uri: {{ .Values.buildconfig.uri }}
    type: Git
  strategy:
    dockerStrategy:
    type: Docker
  triggers:
    - imageChange:
      type: ImageChange
    - type: ConfigChange
    - generic:
        secret: {{ .Values.buildconfig.secret1 }}
      type: Generic
    - github:
        secret: {{ .Values.buildconfig.secret2 }}
      type: GitHub
```

IBM

Operator SDK

Operator Pattern

- A pattern use to deploy native Kubernetes application.
- Purposely built for specific application
- Handling complex failures scenarios
- Handling complex failovers

- Is this better?

mongorest_v1alpha1_mongorest_cr.yaml ● | {} Untitle.json ● | imagestream.y

```yaml
apiVersion: mongorest.ibm.com/v1alpha1
kind: Mongorest
metadata:
  name: todomicroservice
spec:
  replicaCount: 3
  namespace: "kubeapp"
  metadata:
    name: todo-microservice
  mongodb:
    host: "mongodb.kubeapp"
    username: "YWRtaW4="
    password: "YWRtaW4="
    endpoint: "license"
    db: "sampledb"
    schema: "Todo"
    model: "{'task':'String','completed':'Boolean','uid':'String'}"
  buildconfig:
    imagename: "todo-microservice:latest"
  routes:
    host: "todo-microservice-kubeapp.apps.192.168.64.20.nip.io"
```

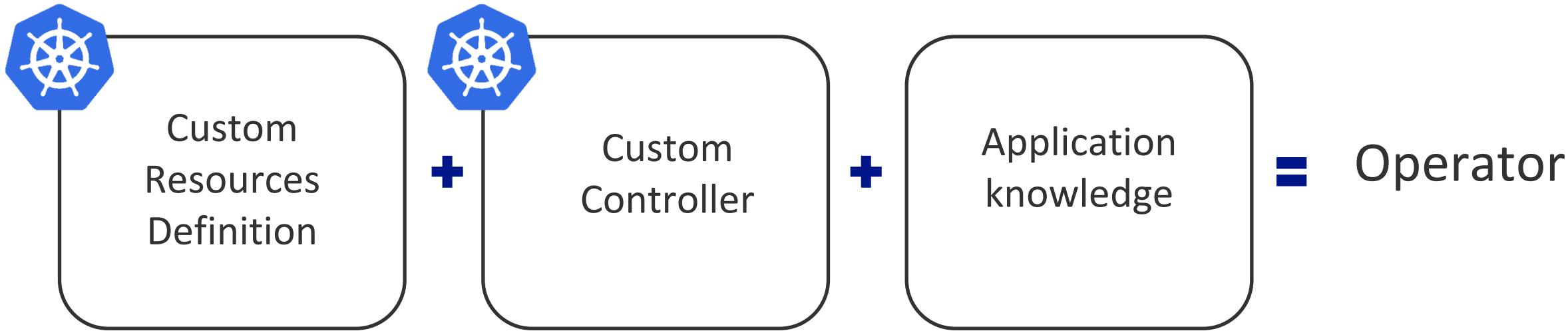
Kubernetes Doesn’t and Operators Does

- This is where Operators Come in, all the cons of Stateful can be achieved by Operators.
- Operators fill the gap of the application specific things that Kubernetes can’t do.
- Operators extend Kubernetes functionality.
- Human experience as code.
- Focus on desired state.
- Complex, Manual Operational tasks become a single line of Config.

What is Operator?

“An Operator is a method of packaging, deploying and managing a Kubernetes application. A Kubernetes application is an application that is both deployed on Kubernetes and managed using the Kubernetes APIs and kubectl tooling.”

IBM


Custom
Resources
Definition
+
Custom
Controller
+
Application
knowledge
=
Operator

Application Knowledge

- Deploy
- Upgrades
- Scale
- Backup
- Self-Heal/Repair

Operator Interaction With Kubernetes

- Operators take advantage of Custom Resource Definition(CRD).
- CRD’s are extensions of the Kubernetes API to register new Resource.
- Creating a Custom Resource(CR) from CRD’s.
- Operator monitors for new CR request, acknowledges and creates the CR.
- It Can be used like any other native Kubernetes Resource.

```
summit@instance-31:~$ kubectl get cloner
NAME     AGE
cloner   71s
```

IBM

When to Choose Creating A Operator?

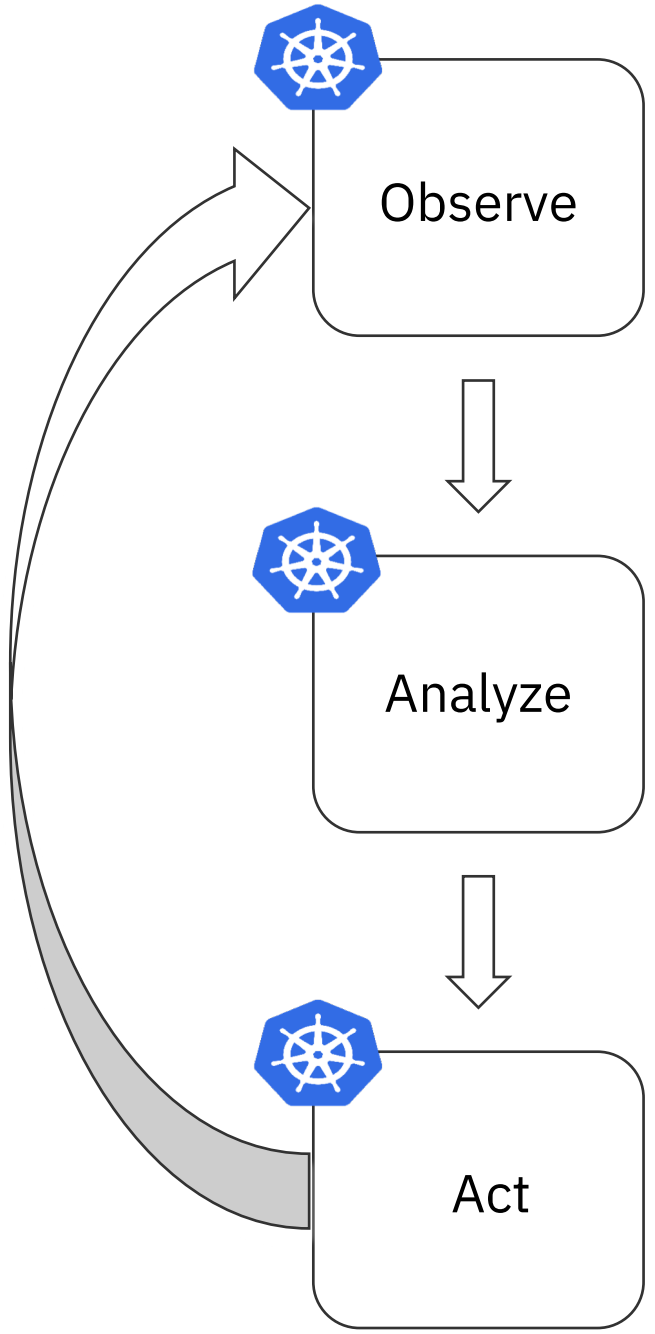
- Business logic is required.
- Application uses a declarative API.
- Automation that watches for updates of Kubernetes object.
- Create or update resources using Kubernetes API.

Etcd Operator

- Cluster A has 3 Running Pods
  Name: summit-etcd-0 Version 2.9.1
  Name: summit-etcd-1 Version 2.9.2
  Name: summit-etcd-1 Version 2.9.1

- Desired = False
  Difference in Configuration
  Version should be 2.9.2

- Cluster
  Clean, Backup Cluster
  Upgrade to 2.9.2



IBM

Operator Framework

- Operator SDK

  Supports developers in bootstrapping and building an Operator based on their expertise without requiring knowledge of Kubernetes API complexities

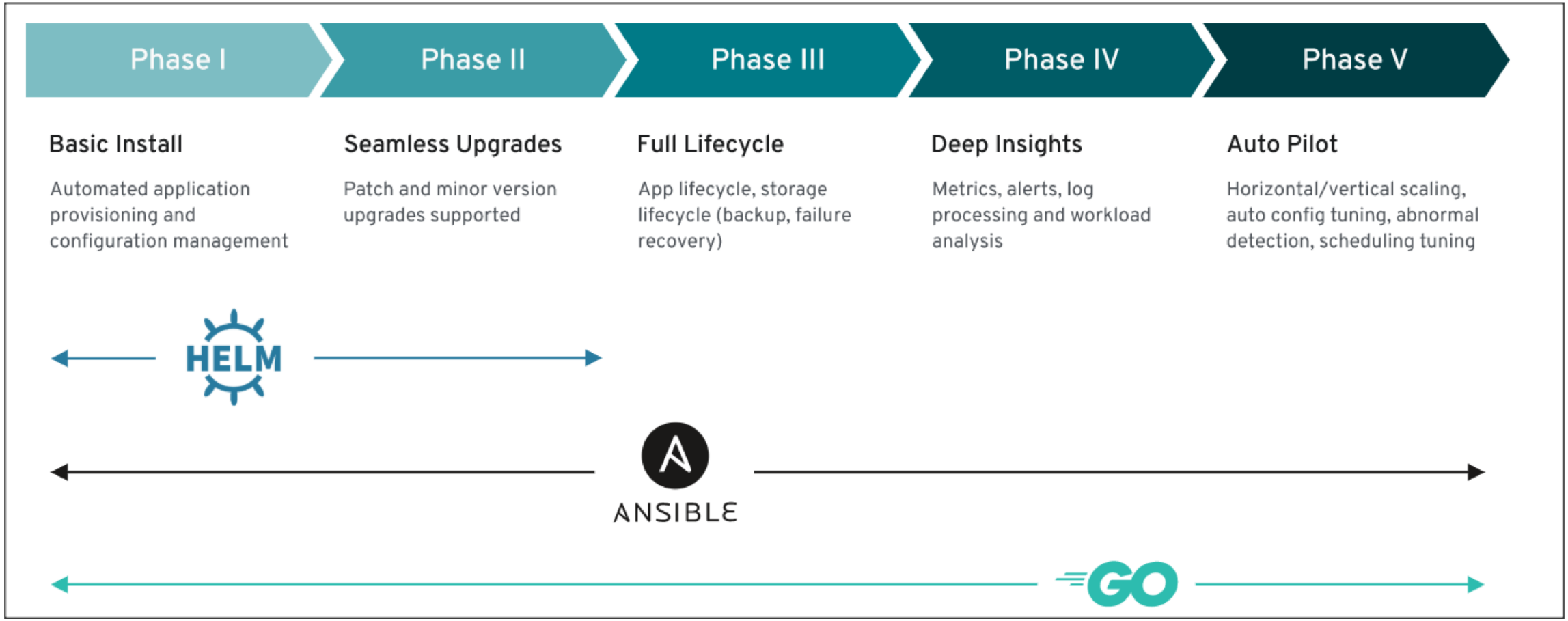
- Operator Lifecycle Manager

  Helps you to install, update, and generally manage the lifecycle of all of the operators (and their associated services) running across your clusters

- Operator Metering

  Metering records historical cluster usage, and can generate usage reports showing usage breakdowns by pod or namespace over arbitrary time periods

Maturity Models of Operators



Cloud Pak for Applications uses Kubernetes Operators for deploy and lifecycle of:

1. Cloud Pak for Applications product

2. Deployment of apps onto OpenShift

For more information see here:
https://www.openshift.com/learn/topics/operators

CP4Apps Product Operator

Leverages Kabanero operator from open source community

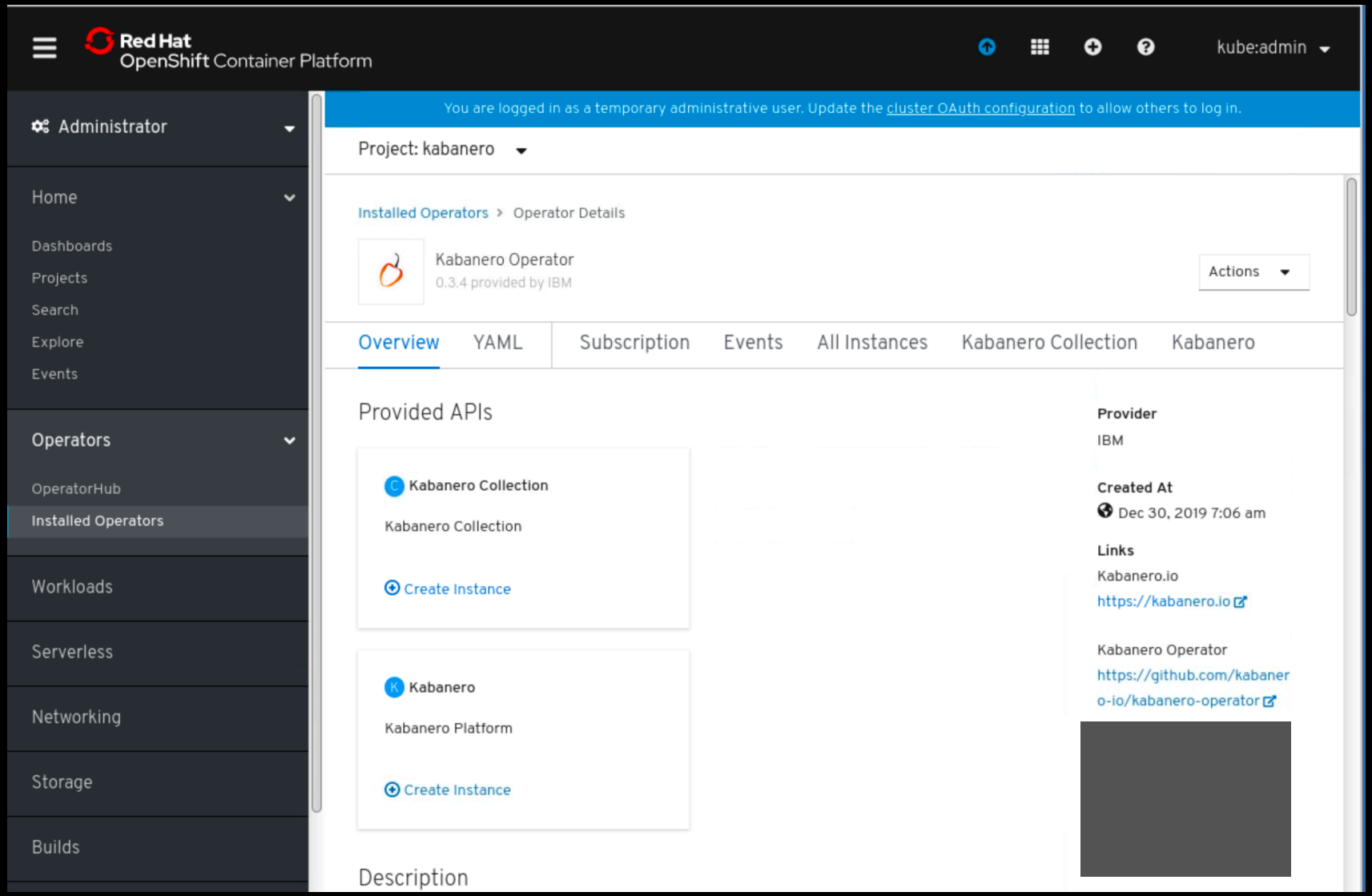
Works with OpenShift OLM (Operator Lifecycle Manager)

Declares dependencies using Operator Subscriptions:

- OpenShift Service Mesh operator
- OpenShift Serverless operator
- Appsody operator
- kAppNav operator
- Tekton operator

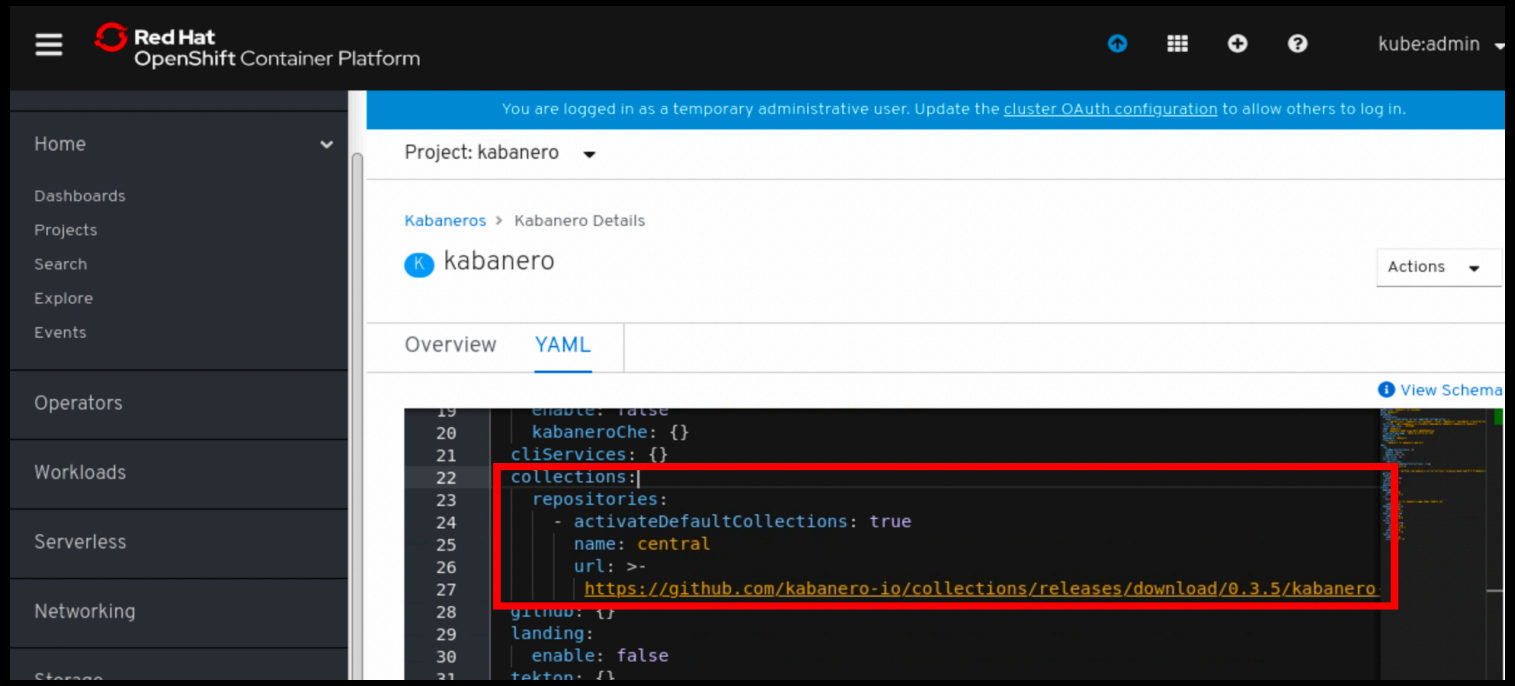
“Product Operator”

(Kabanero Operator today)



Activating your customized stack

Update ***Kabanero CR*** to point at your collection hub release URL



Kabanero operator:

Activates collections

Installs pipeline resources

Governs stack builds

Activate: oc apply kabanero.yaml