

Openshift Operators

Saif Ur Rehman



Operators

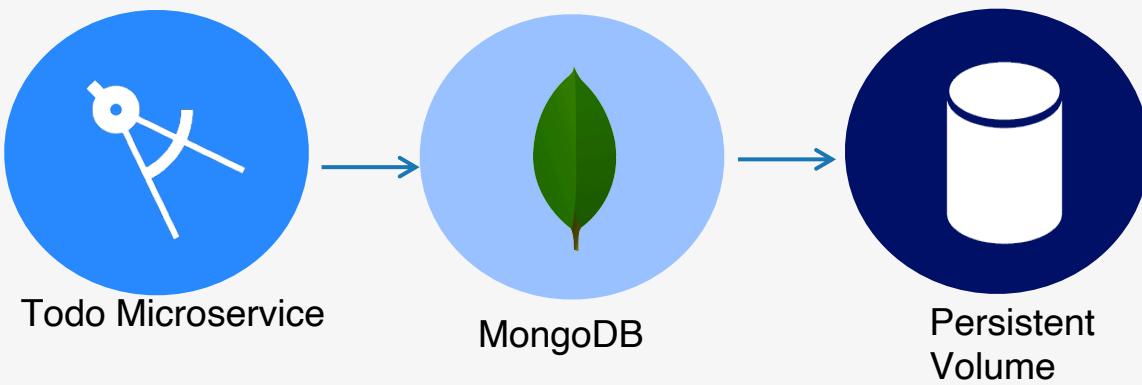
Containers brought simplicity to development world

- \$ 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

```
ts crmController.ts ✘ ts index.ts ! mongorest_v1alpha1_mongorest_cr.
18     public getTodo(req: Request, res: Response) {
19       res.setHeader("Content-Type", "application/json");
20       Todo.find({}, (err, todo) => {
21         if (err) {
22           res.status(404).json({ err });
23           return;
24         } else {
25           res.status(200).send(todo);
26         }
27       });
28     }
29     public searchTodo(req: Request, res: Response) {
30       res.setHeader("Content-Type", "application/json");
31       let query = [
32         $and:[
33           { 'Username': req.body.Username,
34         }]
35       ];
36       Todo.find( query , (err, todo) => {
37         if (err) {
38           res.status(404).json({ err });
39           return;
40         } else {
41           res.status(200).send(todo);
42         }
43       });
44     }
45     public getTodoById(req: Request, res: Response) {
46       res.setHeader("Content-Type", "application/json");
47       Todo.findById(req.params.ID, (err, todo) => {
48         if (err) {
49           res.status(404).json({ err });
50           return;
51         } else {
52           res.status(200).send(todo);
53         }
54       });
55     }
56     public updateTodo(req: Request, res: Response) {
57       res.setHeader("Content-Type", "application/json");
58       Todo.findOneAndUpdate(
59         { _id: req.params.ID },
60         req.body,
61         { new: true }
62       );
63     }
64   }
65 }
```

```
Intitle.json • imagestream.yaml buildconfig.yaml configmap.yaml deployment.yaml
1 import bodyParser from 'body-parser';
2 import session from 'express-session';
3 import ListingMongoController from './controllers/listing/index';
4 var mongoose = require('mongoose');
5 mongoose.Promise = Promise;
6 class App {
7   public express: express.Application;
8   public mongoUrl: string = 'mongodb://'+process.env.MONGOUSERNAME+':'+process.env.MONGOPASSWORD+'@'
9
10  constructor() {
11    this.express = express();
12    this.middleware();
13    this.routes();
14    this.mongoSetup();
15  }
16  private mongoSetup(): void{
17    mongoose.Promise = global.Promise;
18    mongoose.connect(this.mongoUrl);
19  }
20  private middleware(): void {
21    this.express.use(function(req, res, next) {
22      res.header("Access-Control-Allow-Origin", "*");
23      res.header("Access-Control-Allow-Headers", "X-Requested-With,content-type");
24      res.header("Access-Control-Allow-Methods", "GET, POST, OPTIONS, PUT, PATCH, DELETE");
25      next();
26    });
27    this.express.use(bodyParser.json());
28    this.express.use(bodyParser.urlencoded({ extended: false }));
29    this.express.use(session({secret: 'test123', saveUninitialized: false, resave: true}))
30  }
31  private routes(): void {
32    this.express.get('/', function(_, res) {
33      res.send('hi');
34    });
35    this.express.use('/', ListingMongoController);
36  }
37
38 }
39 export default new App().express;
```



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

Dockerfile webpack.config.js values.yaml service.yaml TS crmCon

```

1  const path = require('path');
2  const CleanWebpackPlugin = require('clean-webpack-plugin');
3  const nodeExternals = require('webpack-node-externals');
4  const webpack = require('webpack');
5
6  const BACKENDSRC = path.resolve(__dirname, 'src');
7
8  module.exports = {
9    target: 'node',
10   // @babel/polyfill is needed to use modern js functionalities in old browsers.
11   entry: ['@babel/polyfill', path.resolve(BACKENDSRC, 'index.ts')],
12   output: {
13     path: path.resolve(__dirname, 'dist'),
14     filename: 'bundle-be.js',
15     publicPath: path.join(__dirname, 'dist')
16   },
17   module: {
18     rules: [
19       {
20         test: /\.js$/,
21         use: {
22           loader: 'babel-loader',
23           options: {
24             // To process async functions.
25             plugins: ['@babel/plugin-transform-async-to-generator']
26           }
27         },
28         exclude: /(node_modules|bower_components)/
29       },
30       {
31         test: /\.ts$/,
32         loaders: ['ts-loader'],
33         exclude: /(node_modules|bower_components)/
34       }
35     ]
36   },
37   resolve: {
38     modules: ['node_modules', BACKENDSRC, path.resolve(__dirname, 'helpers')],
39     extensions: ['.js', 'web.js', 'webpack.js', '.ts', '.tsx']
40   },
41   plugins: [
42     new CleanWebpackPlugin([path.resolve('dist', 'bundle-be.js')]),
43     new webpack.HotModuleReplacementPlugin()
44   ],

```

Dockerfile tsconfig.json values.yaml service.yaml

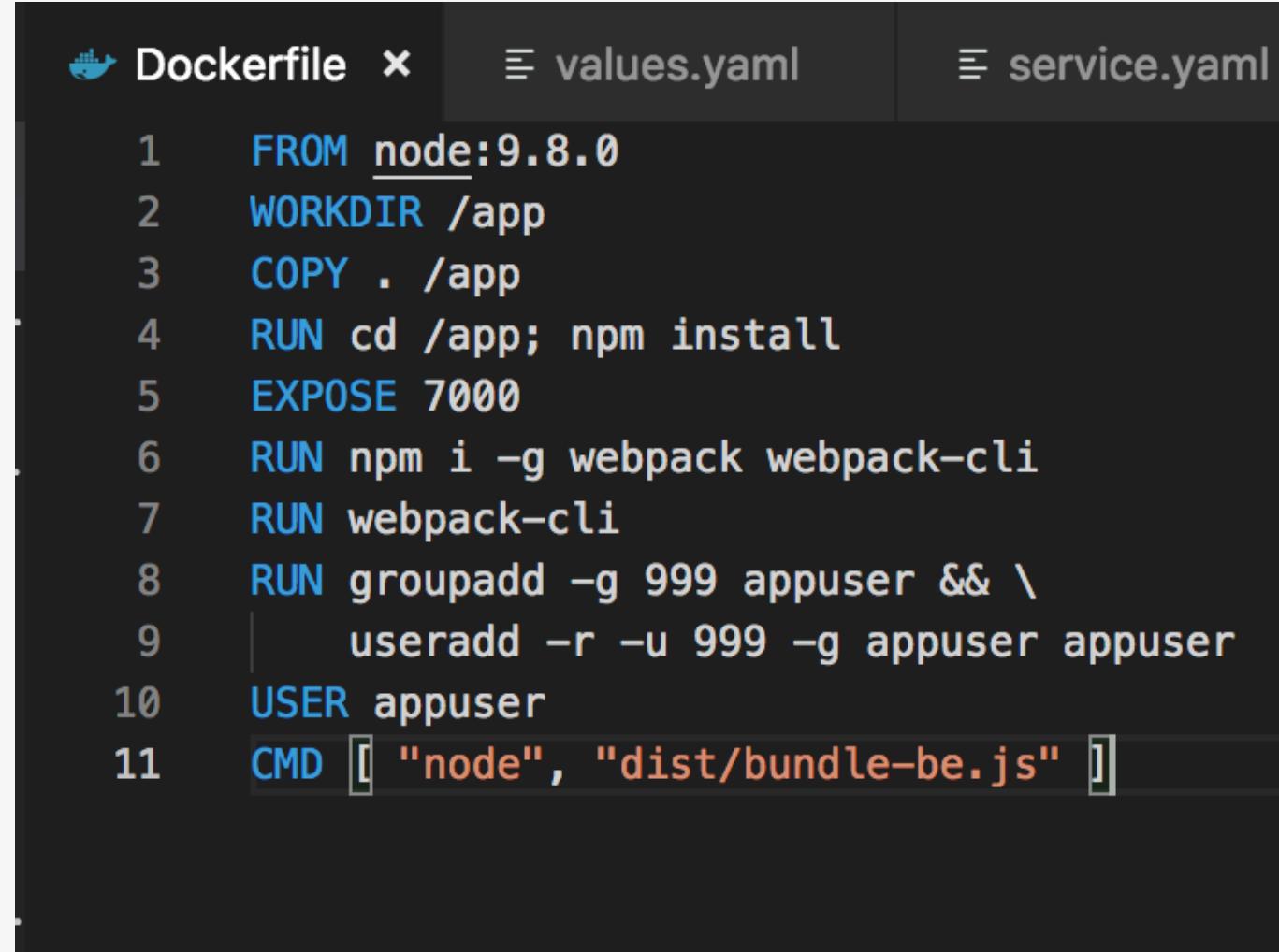
```

1  {
2    "compilerOptions": {
3      "types": ["reflect-metadata"],
4      "moduleResolution": "node",
5      "experimentalDecorators": true,
6      "emitDecoratorMetadata": true,
7      /* Basic Options */
8      // "strict": true /* Enable all strict type-checking options.
9      // "noImplicitAny": true /* Raise error on expressions and dec
10     "outDir": "./dist/" /* Redirect output structure to the direct
11     "module": "es6" /* Specify module code generation: 'none', 'co
12     "target": "es5" /* Specify ECMAScript target version: 'ES3' (d
13     "jsx": "react" /* Specify JSX code generation: 'preserve', 're
14     "allowJs": true /* Allow javascript files to be compiled. */,
15     "moduleResolution": "node" /* Specify module resolution strategy
16     "typeRoots": [
17       "node_modules/@types/*"
18     ] /* List of folders to include type definitions from. */,
19     "lib": [
20       "es2015"
21     ] /* Specify library files to be included in the compilation.
22     // "checkJs": true, /* Report errors in
23     // "declaration": true, /* Generates correspondin
24     // "declarationMap": true, /* Generates a sourc
25     // "sourceMap": true, /* Generates correspondin
26     // "outFile": "./", /* Concatenate and e
27     // "rootDir": "./", /* Specify the root di
28     // "composite": true, /* Enable project composit
29     // "removeComments": true, /* Do not emit commen
30     // "noEmit": true, /* Do not emit output
31     // "importHelpers": true, /* Import emit helper
32     // "downlevelIteration": true, /* Provide full supp
33     // "isolatedModules": true, /* Transpile each file
34
35     /* Strict Type-Checking Options */
36     // "strictNullChecks": true, /* Enable strict null che
37     // "strictFunctionTypes": true, /* Enable strict che
38     // "strictPropertyInitialization": true, /* Enable strict che
39     // "noImplicitThis": true, /* Raise error on 'this' unde
40     // "alwaysStrict": true, /* Parse in strict mode
41
42     /* Additional Checks */
43     // "noUnusedLocals": true, /* Report errors on
44     // "noUnusedParameters": true, /* Report errors on

```



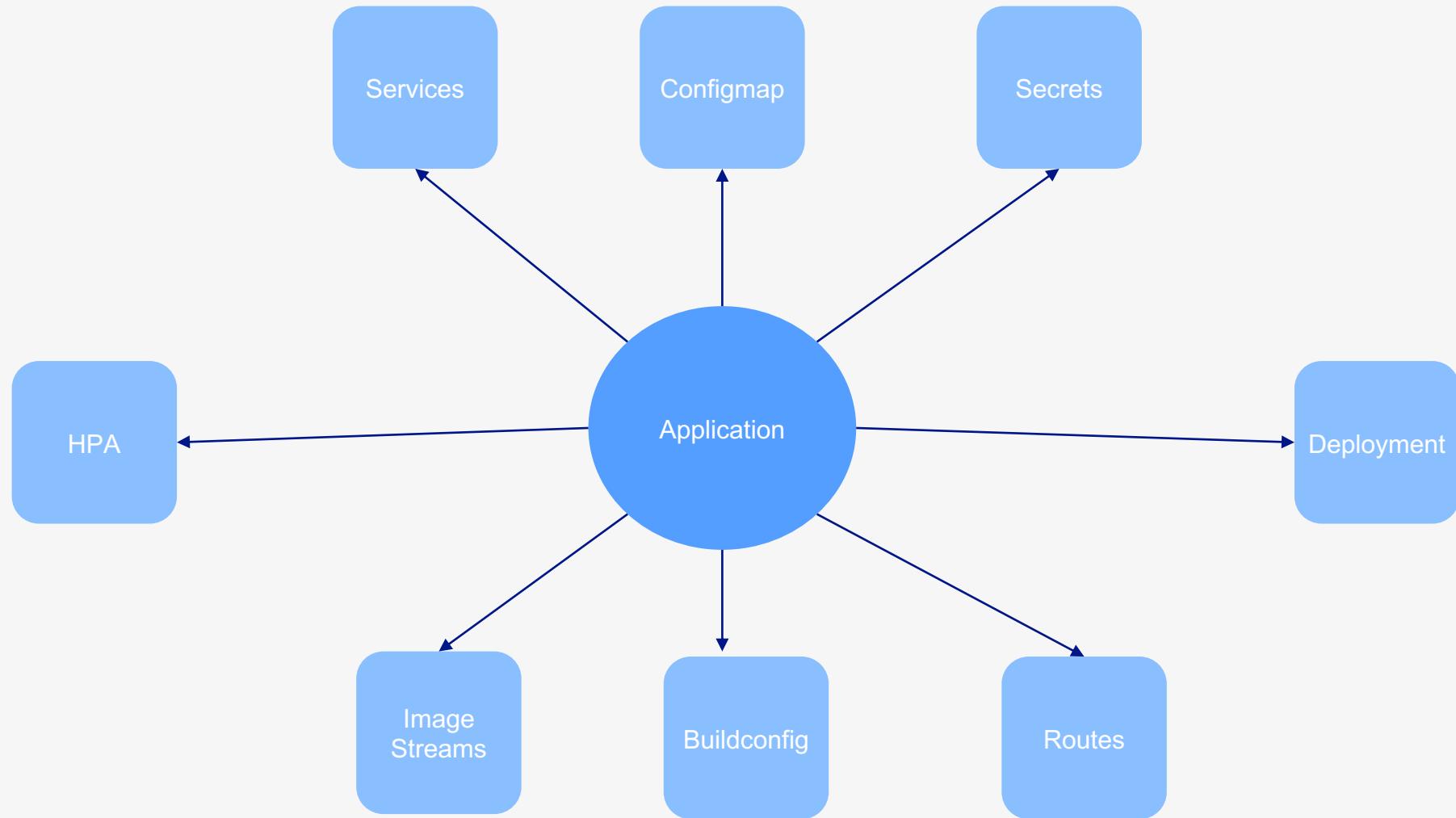
- Dockerfile



```
• 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



- Kubernetes

A screenshot of a code editor showing a `service.yaml` file. The file contains YAML configuration for a Kubernetes Service. It specifies the API version, kind, metadata (name and labels), spec (type, selector, ports, and targetPort), and ports (protocol, name, port, and targetPort). The code uses `Values` placeholder variables.

```
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}}
```

A screenshot of a code editor showing a `mongores.yaml` file. The file contains YAML configuration for a Kubernetes Secret. It specifies the API version, kind, metadata (name and labels), spec (type and data), and data (SECRET, MONGOUSERNAME, and MONGOPASSWORD). The code uses `Values` placeholder variables.

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



- Kubernetes

t_v1alpha1_mongorest_cr.yaml • {} Untitled.json • imagestream.y

```
1 apiVersion: route.openshift.io/v1
2 kind: Route
3 metadata:
4   annotations:
5     openshift.io/generated-by: OpenShiftWebConsole
6     openshift.io/host.generated: "true"
7   labels:
8     app: "{{ .Values.metadata.name }}"
9   name: "{{ .Values.metadata.name }}"
10  namespace: "{{ .Values.namespace }}"
11 spec:
12   host: "{{ .Values.routes.host }}"
13   port:
14     targetPort: {{ .Values.routes.targetPort }}
15   to:
16     kind: Service
17     name: {{ .Values.metadata.name }}
18     weight: {{ .Values.routes.weight }}
19   wildcardPolicy: None
```

t_v1alpha1_mongorest_cr.yaml • {} Untitled.json • imagestream.yaml

```
1 apiVersion: apps.openshift.io/v1
2 kind: DeploymentConfig
3 metadata:
4   name: {{ .Values.metadata.name }}
5 spec:
6   replicas: {{ .Values.replicaCount }}
7   strategy:
8     type: RollingUpdate
9     rollingUpdate:
10       maxSurge: {{ .Values.image.maxSurge }}
11       maxUnavailable: {{ .Values.image.maxUnavailable }}
12   triggers:
13     - type: "ConfigChange"
14     - type: "ImageChange"
15       imageChangeParams:
16         automatic: true
17         containerNames:
18           - {{ .Values.metadata.name }}
19       from:
20         kind: "ImageStreamTag"
21         name: {{ .Values.buildconfig.imagename }}
22   strategy:
23     type: "Rolling"
24   paused: false
25   template:
26     metadata:
27       name: {{ .Values.metadata.name }}
28     labels:
29       app: {{ .Values.metadata.name }}
30   spec:
31     containers:
32       - name: {{ .Values.metadata.name }}
33         image: {{ .Values.metadata.name }}
34         readinessProbe:
35           httpGet:
36             path: {{ .Values.readinessProbe.path }}
37             port: {{ .Values.readinessProbe.port }}
38             scheme: HTTP
39           initialDelaySeconds: {{ .Values.readinessProbe.initialDelaySeconds }}
40           timeoutSeconds: {{ .Values.readinessProbe.timeoutSeconds }}
41           periodSeconds: {{ .Values.readinessProbe.periodSeconds }}
42         livenessProbe:
43           httpGet:
44             path: {{ .Values.livenessProbe.path }}
```

• Kubernetes

```
t_v1alpha1_mongorest_cr.yaml • {} Untitled.json • image
1  apiVersion: v1
2  kind: ConfigMap
3  metadata:
4    labels:
5      app: "{{ .Values.metadata.name }}"
6      name: {{ .Values.metadata.name }}
7      namespace: "{{ .Values.namespace }}"
8  data:
9    MONGOPORT: "{{ .Values.mongodb.port }}"
10   HOSTNAMEMONGODB: "{{ .Values.mongodb.host }}"
11   ENDPOINT: "{{ .Values.mongodb.endpoint }}"
12   DB: "{{ .Values.mongodb.db }}"
13   SCHEMA: "{{ .Values.mongodb.schema }}"
14   MODEL: "{{ .Values.mongodb.model }}"
```

```
t_v1alpha1_mongorest_cr.yaml • {} Untitled.json • imagestream
1  apiVersion: build.openshift.io/v1
2  kind: BuildConfig
3  metadata:
4    name: {{ .Values.metadata.name }}
5    labels:
6      app: "{{ .Values.metadata.name }}"
7      annotations:
8        openshift.io/generated-by: OpenShiftWebConsole
9  spec:
10   output:
11     to:
12       kind: ImageStreamTag
13       name: {{ .Values.buildconfig.imagename }}
14   runPolicy: Serial
15   source:
16     git:
17       ref: {{ .Values.buildconfig.branch }}
18       uri: {{ .Values.buildconfig.uri }}
19       type: Git
20   strategy:
21     dockerStrategy:
22       type: Docker
23   triggers:
24     - imageChange:
25       type: ImageChange
26     - type: ConfigChange
27     - generic:
28       secret: {{ .Values.buildconfig.secret1 }}
29       type: Generic
30     - github:
31       secret: {{ .Values.buildconfig.secret2 }}
32       type: GitHub
```





Operator Pattern

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

- Is this better?

```
! mongorest_v1alpha1_mongorest_cr.yaml ● {} Untitled.json ● ⌂ imagestream.y

1  apiVersion: mongorest.ibm.com/v1alpha1
2  kind: Mongorest
3  metadata:
4    name: todomicroservice
5  spec:
6    replicaCount: 3
7    namespace: "kubeapp"
8    metadata:
9      name: todo-microservice
10   mongodb:
11     host: "mongodb.kubeapp"
12     username: "YWRTaW4="
13     password: "YWRTaW4="
14     endpoint: "license"
15     db: "sampledb"
16     schema: "Todo"
17     model: "{'task':'String','completed':'Boolean','uid':'String'}"
18   buildconfig:
19     imagename: "todo-microservice:latest"
20   routes:
21     host: "todo-microservice-kubeapp.apps.192.168.64.20.nip.io"
```

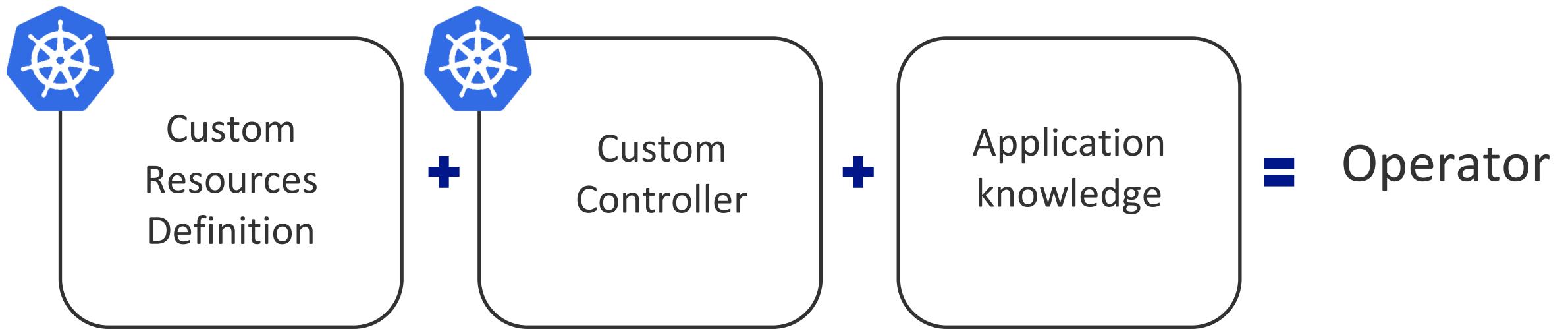


Kubernetes Doesn't and Operators Does

- 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.”



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.

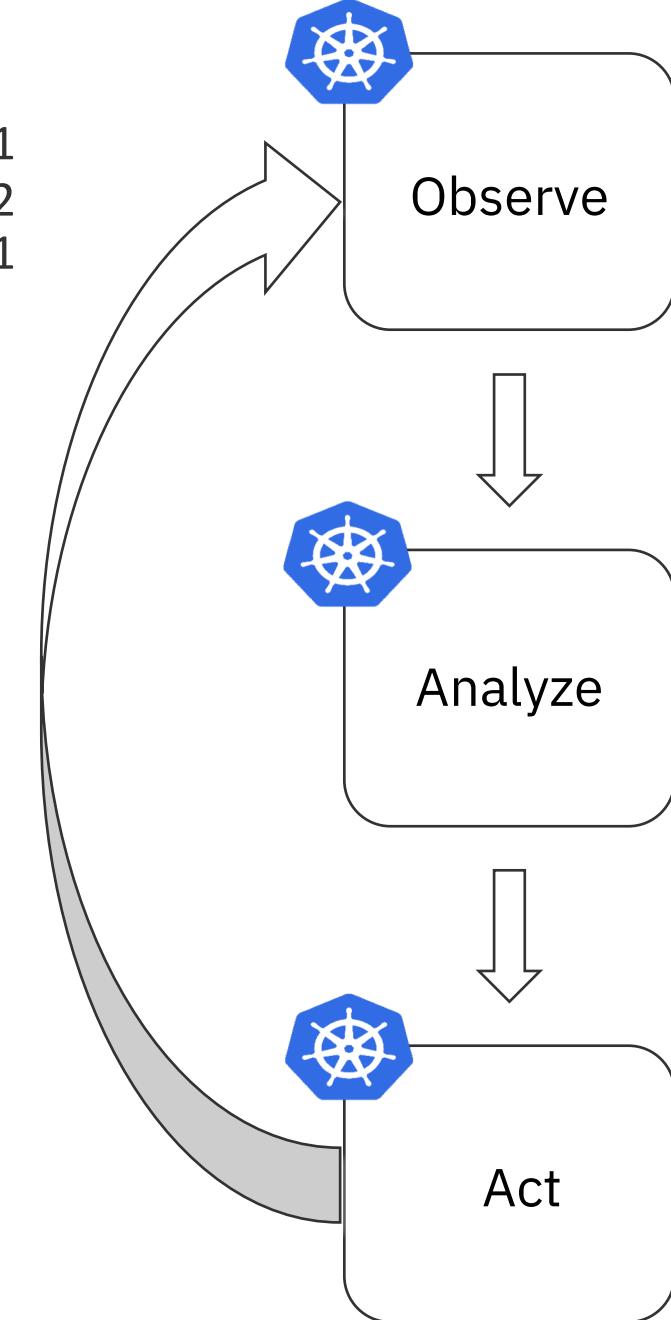
```
summit@instance-31:~$ kubectl get cloner
NAME      AGE
cloner    71s
```

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



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

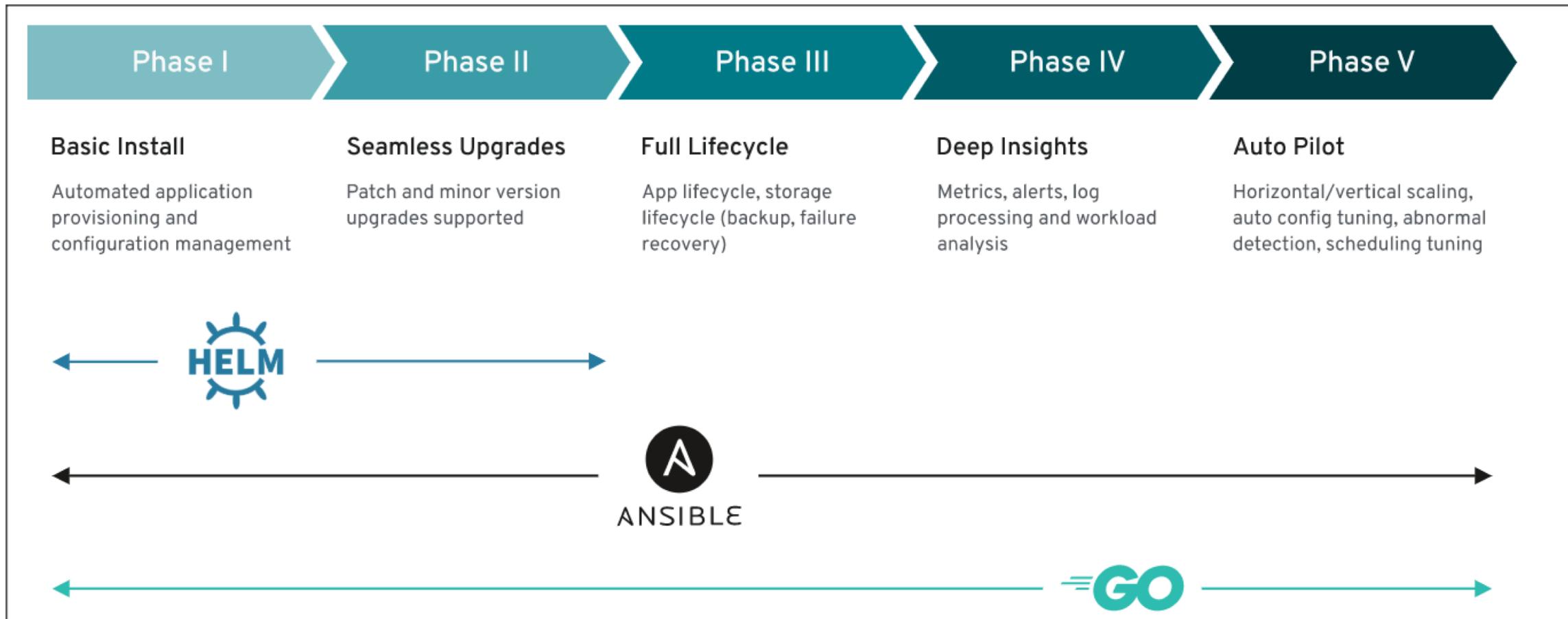
Helps you to install, update, and generally manage the lifecycle of all of the operators (and their associated services) running across your clusters

- 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*)

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar is dark-themed and includes links for Home, Dashboards, Projects, Search, Explore, Events, Operators (with OperatorHub and Installed Operators selected), Workloads, Serverless, Networking, Storage, and Builds. The main content area has a light background. At the top, it says "You are logged in as a temporary administrative user. Update the [cluster OAuth configuration](#) to allow others to log in." Below this, the "Project: kabanero" is selected. The "Installed Operators" section shows the "Kabanero Operator" (0.3.4 provided by IBM). The "Overview" tab is active, showing the "Provided APIs" table. The table has two rows:

	Provider	Created At	Links
Kabanero Collection	IBM	Dec 30, 2019 7:06 am	Kabanero.io https://kabanero.io
Create Instance			
Kabanero	Kabanero Operator		
Create Instance	https://github.com/kabanero-o-io/kabanero-operator		

At the bottom, there is a "Description" section.

Activating your customized stack

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

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar has a 'Home' dropdown, 'Operators' section, and 'Workloads' section. The main area shows a 'Project: kabanero' with a 'Kabaneros' list containing one item: 'kabanero'. Below it is an 'Actions' dropdown. The 'YAML' tab is selected, displaying a YAML configuration file. A red box highlights the 'collections:' section, which contains a 'repositories:' key pointing to a GitHub URL: <https://github.com/kabanero-io/collections/releases/download/0.3.5/kabanero>. The right side of the interface shows a preview of the current state.

```
enable: false
kabaneroChe: {}
cliServices: {}
collections:
  repositories:
    - activateDefaultCollections: true
      name: central
      url: >-
        https://github.com/kabanero-io/collections/releases/download/0.3.5/kabanero
github: {}
landing:
  enable: false
tekton: {}
```

Activate: `oc apply kabanero.yaml`

Kabanero operator:

Activates collections

Installs pipeline resources

Governs stack builds