

Kubernetes Operators, in Java?

Hold my mouse and look



Contents



1. **whoami**
2. what can you **expect**
3. what is the **problem** to solve
4. **what** are Operators
5. **create** an Operator
6. **test** the Operator
7. **publish** an Operator
8. **execution** process
9. references
10. thank you

whoami





@vilojona



jvilalop@redhat.com



aytartana.wordpress.com



github.com/jonathanvila



Java Champion



JUG co-leader at BarcelonaJUG



Co-founder of JBCNConf



Red Hat

Software Engineer at Red Hat
App. Modernisation and Migration team

What can you expect ?



my experience



introductory knowledge



magistral lecture



This is **<NOT>** the way



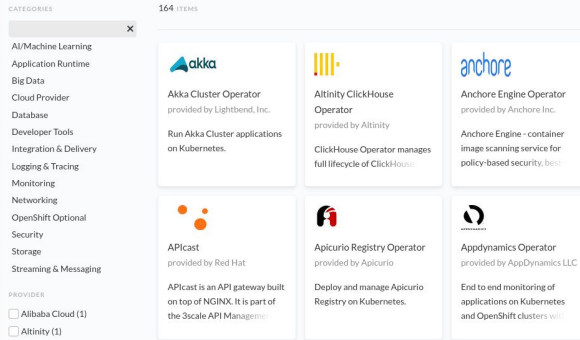
WHAT TO
EXPECT

What's the problem
to solve



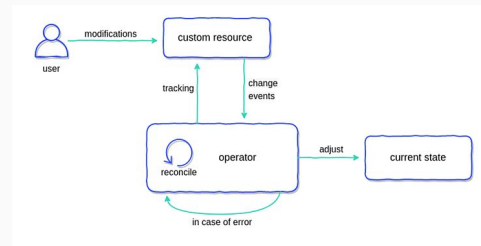
What is the problem to solve ?

1. You need an easy to use market to publish your app



2. You want something to help you to automate tasks :

- scaling
- upgrading
- complex tasks



3. You want a “robot” to take care of issues

- Pods falling
- health checks failing
- avoid night shifts

4. You want to have more control

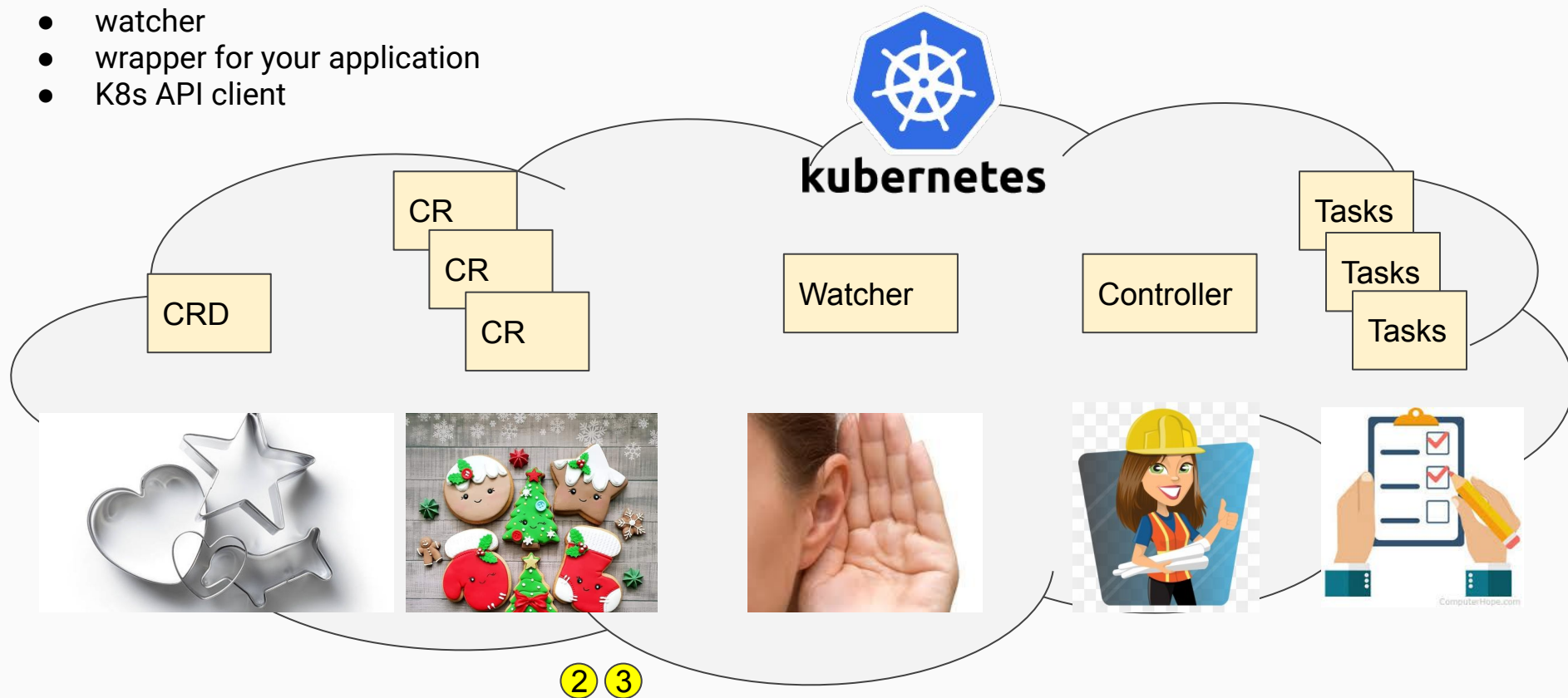
- metrics
- encapsulate your app
- auto config tuning

What's an Operator



What's an Operator ?

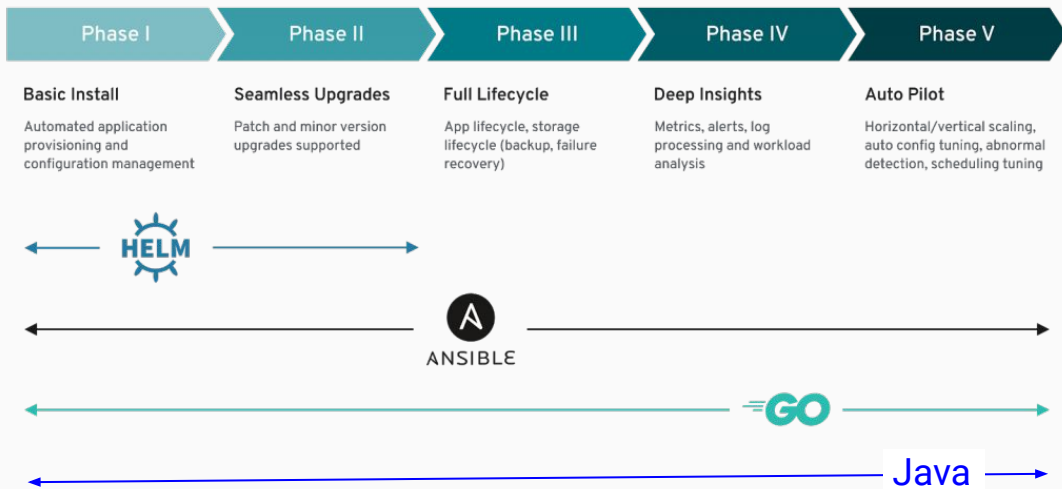
- extension of the Kubernetes API
- watcher
- wrapper for your application
- K8s API client




Create an Operator



Create an Operator



 **Migration Toolkit for Applications Operator**
0.0.1 provided by Red Hat

[Install](#)

Operator Version 0.0.1

The Migration Toolkit for Applications (MTA) is an application modernization and migration project

Java
OpenJDK 11

Java Tools
Fabric8 K8s Client
JIB
Quarkus

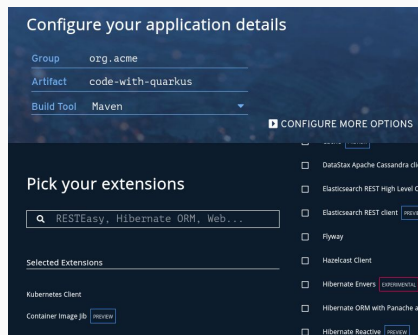
Tools
kubectl / oc
dive
podman

Registry
quay.io

Local Cluster
minikube
Kind

Create an Operator

① Java app skeleton



④ Watcher on CRs

```
crClient.watch(controller);
```

② RBAC & CRD yaml

```
Cluster wide  
CRD  
Namespace
```

```
Namespace wide  
ServiceAccount  
Role  
RoleBinding
```

⑤ Watcher on Deployments

```
k8Client  
.apps()  
.deployments()  
.inNamespace(namespace)  
.watch(depController);
```

③ K8s client Beans

```
namespace  
kubernetesClient  
crClient
```

⑥ Build & Push & Deploy

```
./mvnw clean package -Pnative  
-Dquarkus.native.container-build=true  
-Dquarkus.container-image.push=true  
  
kubectl apply -f deployment.yaml
```

Java & Yaml

.... YES

..... Yaml



Java skeleton

```
└─ windup-operator
  └─ .github
  └─ .history
  └─ src
    └─ main
      └─ java / org / jboss / windup / operator
        └─ controller
          └─ WindupController.java
          └─ WindupDeploymentController.java
        └─ model
          └─ WindupResource.java
          └─ WindupResourceDoneable.java
          └─ WindupResourceList.java
          └─ WindupResourceSpec.java
          └─ WindupResourceStatus.java
          └─ WindupResourceStatusCondition.java
        └─ util
          └─ WindupDeployment.java
          └─ KubernetesClientProducer.java
          └─ WindupOperator.java
```

Controllers

Model definition

Beans & Clients

K8s scripts for manual deployment

Files for OperatorHub deployment

```
└─ resources
  └─ k8s
    └─ def
      └─ script.create.all.sh
      └─ script.delete.all.sh
      └─ windup.crd.yaml
      └─ windup.deployment.yaml
      └─ windup.namespace.yaml
      └─ windup.role.yaml
      └─ windup.rolebinding.yaml
      └─ windup.serviceaccount.yaml
    └─ examples
    └─ META-INF
    └─ operatorhub
      └─ mta-operator
      └─ catalog-source.yaml
      └─ create-catalog.sh
      └─ delete-operator.sh
      └─ application.properties
```

CRD

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: windups.windup.jboss.org
  labels:
    application: windup
spec:
  group: windup.jboss.org
  scope: Namespaced
  preserveUnknownFields: false
  names:
    plural: windups
    singular: windup
    kind: Windup
  versions:
  - name: v1
    served: true
    storage: true
  subresources:
    status: {}
  schema:
    openAPIV3Schema:
      type: object
      properties:
        spec:
          type: object
          properties:
            hostname_http:
              type: string
              default: ""
            volumeCapacity:
              type: string
              default: "20G"
```

Role

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: windup-operator-role
  namespace: mta
  labels:
    application: windup
rules:
- apiGroups:
  - apps
  - extensions
  resources:
  - deployments
  verbs:
  - create
  - delete
  - get
  - list
  - watch
```

RoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: windup-operator-role-binding
  namespace: mta
  labels:
    application: windup
subjects:
- kind: ServiceAccount
  name: windup-operator
  namespace: mta
roleRef:
  kind: Role
  name: windup-operator-role
apiGroup: rbac.authorization.k8s.io
```

ServiceAccount

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: windup-operator
  namespace: mta
  labels:
    application: windup
```

Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: windup-operator
  namespace: mta
  labels:
    application: windup
spec:
  selector:
    matchLabels:
      app: windup-operator
  replicas: 1
  template:
    metadata:
      labels:
        app: windup-operator
    spec:
      serviceAccountName: windup-operator
      containers:
      - image: quay.io/windupeng/windup-operator-native:latest
        name: windup-operator
        imagePullPolicy: Always
        env:
        - name: WATCH_NAMESPACE
          valueFrom:
            fieldRef:
              filePath: metadata.namespace
```

Global K8s client

```
@Produces
@Singleton
@ifBuildProfile("prod")
NamespacedKubernetesClient makeDefaultClient() {
    return new DefaultKubernetesClient();
}
```

CRD specific client

```
@Produces
@Singleton
MixedOperation<WindupResource, ...
makeWindupCustomResource (NamespacedKubernetesClient defaultClient,
                          CustomResourceDefinitionContext
                          crdContext,
                          @Named("namespace") String namespace) {
    KubernetesDeserializer .registerCustomKind ("windup.jboss.org/v1" ,
                                                "Windup" ,
                                                WindupResource .class);

    return defaultClient .inNamespace (namespace)
        .customResources (crdContext, WindupResource .class,
                          WindupResourceList .class,
                          WindupResourceDoneable .class);
}
```

CRD context for the CRD client

```
@Produces
@Singleton
CustomResourceDefinitionContext makeCRDContext (NamespacedKubernetesClient defaultClient,
                                                @Named("namespace") String namespace) {

    InputStream fileStream =
        KubernetesClientProducer.class.getResourceAsStream ("/k8s/def/windup.crd.yaml");
    Resource<CustomResourceDefinition, DoneableCustomResourceDefinition> resource = defaultClient
        .inNamespace (namespace) .apiextensions () .v1 () .customResourceDefinitions () .load (fileStream
        );
    CustomResourceDefinition windupCRD = resource.get ();

    return new CustomResourceDefinitionContext.Builder ()
        .withGroup (windupCRD.getSpec ().getGroup ())
        .withVersion (windupCRD.getSpec ().getVersions ().stream ().findFirst ()
            .map (CustomResourceDefinitionVersion::getName) .orElse (""))
        .withScope (windupCRD.getSpec ().getScope ())
        .withName (windupCRD.getMetadata ().getName ())
        .withPlural (windupCRD.getSpec ().getNames ().getPlural ())
        .withKind (windupCRD.getSpec ().getNames ().getKind ())
        .build ();
}
```

Namespace from where the operator is running

```
@Produces
@Singleton
@Named("namespace")
String findMyCurrentNamespace (NamespacedKubernetesClient client) {
    return client.getConfiguration ().getNamespace ();
}
```


Watchers

THE Operator

```
@ApplicationScoped
@Log
public class WindupOperator {
    @Inject
    WindupController windupController;
    @Inject
    WindupDeploymentController windupDeploymentController;
    @Inject
    KubernetesClient k8sClient;
    @Inject
    MixedOperation<WindupResource, ...> crClient;

    @Named("namespace")
    String namespace;

    public void onStart(@Observes StartupEvent event) {
        log.info("Startup");

        log.info("Adding Windup watcher ...");
        crClient.watch(windupController);

        log.info("Adding Windup Deployments watcher ...");
        k8sClient.apps().deployments().inNamespace(namespace)
            .watch(windupDeploymentController);
    }
}
```

CR Watcher

```
@Log
@ApplicationScoped
public class WindupController implements Watcher<WindupResource> {
    @Inject
    KubernetesClient k8sClient;

    private void watch(WindupResource resource) {
        if (!resource.isDeploying() && !resource.isReady()) {
            new WindupDeployment(resource, crClient,
                k8sClient, namespace,
                serviceAccount, ssoPublicKey)
                .deploy();
        }
    }

    private void updateStatus(WindupResource newResource) {
        // Consolidate status of the CR
        if (newResource.deploymentsReady() == 3 &&
            !newResource.isReady()) {
            newResource.setReady(true);
            crClient.inNamespace(namespace)
                .updateStatus(newResource);
        }
    }

    @Override
    public void eventReceived(Action action,
        WindupResource resource) {
        if (action == Action.ADDED) onAdd(resource);
        if (action == Action.MODIFIED) onUpdate(resource);
        if (action == Action.DELETED) onDelete(resource);
    }
}
```

Deployments Watcher

```
@Log
@ApplicationScoped
public class WindupDeploymentController implements
    Watcher<Deployment> {
    @Inject
    MixedOperation<WindupRe...> crClient;

    @Named("namespace")
    String namespace;

    @Override
    public void eventReceived(Action action,
        Deployment deployment) {
        updateCRStatus(deployment);
    }
}
```

Reconciliation

IF expected-status != current-status DO

- add more pods
- ...

IF expected-status == current-status DO

- set status = READY

on every Deployment UPDATE event

on every CR UPDATE event - caused by

```
public class WindupController implements Watcher<WindupResource> {  
  
    private void onUpdate(WindupResource newResource) {  
        // Consolidate status of the CR  
        if (newResource.deploymentsReady() == 3 && !newResource.isReady()) {  
            newResource.setReady(true);  
            newResource.getOrCreateConditionByType("Deploy")  
                .setStatus(Boolean.FALSE.toString());  
  
            log.info("Setting this CR as Ready");  
            crClient.inNamespace(namespace).updateStatus(newResource);  
        }  
    }  
}
```

```
public class WindupDeploymentController implements Watcher<Deployment> {  
    public void eventReceived(Action action, Deployment deployment) {  
        updateCRStatus(deployment);  
    }  
  
    private void updateCRStatus(Deployment obj) {  
        // Get the CR name where this deployment belongs to  
        String crName = obj.getMetadata().getOwnerReferences().get(0).getName();  
  
        if (!StringUtil.isBlank(crName)) {  
            WindupResource cr = crClient.inNamespace(namespace).withName(crName).get();  
  
            // We want 1 replica per deployment, so checking if there is 1 replica Ready  
            cr.getOrCreateConditionByType(obj.getMetadata().getName())  
                .setStatus(Boolean.toString(obj.getStatus().getReadyReplicas() == 1))  
                .setReason(WindupResource.DEPLOYMENT);  
  
            // Sending the new status to K8s  
            crClient.inNamespace(namespace).updateStatus(cr);  
        }  
    }  
}
```

Test the Operator



Test the operator

Unit Tests

.. dah....tell me something I don't know



Test the operator

Integration test

```
@QuarkusTest
public class WindupControllerTest {
    @Inject
    KubernetesMockServer server;

    @Inject
    MixedOperation<WindupResource...crClient;

    @Test
    public void onAddCR_shouldServerReceiveExactCalls() throws InterruptedException {
        InputStream fileStream = WindupControllerTest.class.getResourceAsStream("/windup.resource.yaml");
        WindupResource windupResource = Serialization.unmarshal(fileStream, WindupResource.class);
        crClient.inNamespace("test").create(windupResource);
        dispatcher.getRequests().clear();
        Awaitility
            .await()
            .atMost(5, TimeUnit.SECONDS)
            .untilAsserted(() -> assertEquals(2,
                dispatcher.getRequests().stream()
                    .filter(e-> "POST".equalsIgnoreCase(e.getMethod()) &&
                        e.getPath().indexOf("ingress") >= 0).count()));
        assertEquals(4, dispatcher.getRequests()
            .stream().filter(e-> "PUT".equalsIgnoreCase(e.getMethod()) &&
                e.getPath().indexOf("status") >= 0).count());
        assertEquals(3, dispatcher.getRequests().stream()
            .filter(e-> "POST".equalsIgnoreCase(e.getMethod()) &&
                e.getPath().indexOf("deployments") >= 0).count());
    }
}
```

```
public class KubernetesCrudRecorderDispatcher extends
KubernetesCrudDispatcher {
    private List<RecordedRequest> requests = new ArrayList<>();
    @Override
    public MockResponse dispatch(RecordedRequest request) {
        requests.add(request);
        return super.dispatch(request);
    }
}
```

```
@Produces
@Singleton
@ifBuildProfile("test")
KubernetesMockServer makeKubernetesServer
    (KubernetesCrudRecorderDispatcher dispatcher) {
    MockWebServer webServer = new MockWebServer();
    KubernetesMockServer kubernetesServer = new
        KubernetesMockServer(new MockWebServer(),
            webServer, new ArrayList<>(),
            dispatcher, false);
    return kubernetesServer;
}
```

Test the operator

Local test (using Kind) : local-test.sh

```
mvn clean package -Pnative -Dquarkus.container-image.build=true -DskipTests "-Dquarkus.container-image.tag=${containertag}" # Build the operator
kind load docker-image quay.io/windupeng/windup-operator-native:${containertag} # Load new image into the Kind cluster

# Execute all objects
kubectl apply -f windup.namespace.yaml
kubectl apply -f windup.serviceaccount.yaml
kubectl apply -f windup.role.yaml
kubectl apply -f windup.rolebinding.yaml
kubectl apply -f windup.crd.yaml

# Deploy Operator image just create above
sed "s/windup-operator-native:latest/windup-operator-native:${containertag}/g" ../../../../test/resources/windup.deployment.yaml | kubectl apply -f -

# Create the CR to trigger the Operator
kubectl apply -f ../../../../test/resources/windup-test.yaml

sleep 20
num=`kubectl get all,ing,pvc -n mta -o name | wc -l` # Request all objects in the namespace
# 4 deployments (including operator), 2 ingresses, 3 services, 2 pvc, 4 pods (including operator), 4 replicaset
echo "num $num"
if [ "$num" -gt "19" ]; # Checking the number of objectes created by the Operator
then echo "Test not passed";
fi
if [ "$num" -eq "19" ];
then echo "Test OK";
fi
```

CI test (using Github Actions and Minikube)

```
name: MTA Operator E2E Tests

on: [pull_request,push]

jobs:
  minikube:
    name: K8S
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v2.0.0

      - name: Setup Minikube-Kubernetes
        uses: manusa/actions-setup-minikube@v2.0.1
        with:
          minikube version: v1.13.1
          kubernetes version: 'v1.19.2'
          github token: ${ secrets.GITHUB_TOKEN }
          start args: ' --force'

      - name: setup-graalvm-ce
        uses: rinx/setup-graalvm-ce@v0.0.5
        with:
          graalvm-version: "20.1.0"
          java-version: "javall"
          native-image: "true"
```

```
- name: Build operator image and push it to the minikube
  run: cd ${github.workspace} &&
    eval $(minikube -p minikube docker-env) &&
    mvn clean package -Pnative
    -Dquarkus.container-image.build=true -DskipTests
    -Dquarkus.native.container-runtime=docker

- name: Deploy Objects and Operator
  run: cd ${github.workspace} &&
    cd src/main/resources/k8s/def &&
    kubectl apply -f windup.namespace.yaml &&
    kubectl apply -f windup.serviceaccount.yaml &&
    kubectl apply -f windup.role.yaml &&
    kubectl apply -f windup.rolebinding.yaml &&
    kubectl apply -f windup.crd.yaml &&
    kubectl apply -f windup.deployment.yaml &&
    # Deploy CR to trigger Operator
    kubectl apply -f windup-test.yaml

- name: Sleep for 30 seconds to allow objects to be created
  uses: jakejarvis/wait-action@master
  with:
    time: '30s'

- name: Get and store number of k8s objects on mta namespace
  id: getobjects
  run: echo "::set-output name=value::$(kubectl get
all,ing,pvc -n mta -o name | wc -l)"
  shell: bash
```

```
- name: Operator Test
  uses: therussiankid92/gat@v1.3
  with:
    assertion: should.equal
    expected: 19
    actual: ${steps.getobjects.outputs.value}
```

Publish Operator



Publish the Operator

Operatorhub.io

The screenshot shows the OperatorHub.io website. At the top, there's a navigation bar with "OperatorHub.io", a search bar, and a "Contribute" button. Below the navigation bar is a large blue banner with the text "Welcome to OperatorHub.io" and a sub-header "OperatorHub.io is a new home for the Kubernetes community to share Operators. Find an existing Operator or list your own today." Below the banner is a grid of operators, each represented by a card with a logo, name, provider, and a brief description. The grid is filtered to show 168 items. On the left side, there is a sidebar with categories and filters. The categories include AI/Machine Learning, Application Runtime, Big Data, Cloud Provider, Database, Developer Tools, Integration & Delivery, Logging & Tracing, Monitoring, Networking, OpenShift Optional, Security, Storage, and Streaming & Messaging. The filters include PROVIDER (Allibaba Cloud (1), Atinity (1), Anchore (1), Apicurio (1), AppDynamics (1), Show 108 more) and CAPABILITY LEVEL (Basic Install (80), Seamless Upgrades (35), Full Lifecycle (26), Show 108 more).

Openshift internal Operatorhub

The screenshot shows the Red Hat OpenShift Container Platform internal OperatorHub interface. The top navigation bar includes the Red Hat logo, "OpenShift Container Platform", and a user profile "kube/admin". Below the navigation bar is a blue banner with the text "You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in." Below the banner is a sidebar with navigation options: Administrator, Home, Overview, Projects, Search, Explore, Events, Operators, OperatorHub, Installed Operators, Workloads, Networking, Storage, Builds, Pipelines, Monitoring, Compute, User Management, and Administration. The main content area shows the "OperatorHub" section. It includes a "Project: user1-mta" dropdown, a "6 Items" count, and a list of operators. The operators are displayed in a grid, each with a logo, name, provider, and a brief description. The operators include: AI/Machine Learning, Application Runtime, Big Data, Cloud Provider, Database, Developer Tools, Integration & Delivery, Logging & Tracing, Migrations, Monitoring, Networking, OpenShift Optional, Security, Storage, and Streaming & Messaging. The operators are filtered to show 6 items. The operators are: Konveyor Operator, Konveyor Operator for Containers, Konveyor Operator for VMs, Migration, Migration Toolkit for Containers Operator, and MTA Operator. The Migration operator is highlighted with a blue border. The Migration Toolkit for Containers Operator and MTA Operator are marked as "Installed".

Files to publish the operator

Submit PR to <https://github.com/operator-framework/community-operators.git>

Folder

- `community-operators` : Appear in Openshift OperatorHub
- `upstream-community-operators` : Appear in operatorhub.io web site
100% compatible vanilla Kubernetes

```
▼ mta-operator / 0.0.1
  ▼ manifests
    ! windup-operator.v0.0.1.clusterserviceversion.yaml
    ! windup.crd.yaml
  ▼ metadata
    ! annotations.yaml
  Dockerfile
```

CSV : cluster service version
contains visual info, rbac, deployment

CRD : crds being used

annotations : for the bundle

Dockerfile : to create the bundle image

Cluster Service Version

```
apiVersion: operators.coreos.com/v1alpha1
kind: ClusterServiceVersion
metadata:
  name: windup-operator.0.0.1
  namespace: placeholder
  annotations:
    alm-examples: >-
      [ # Yaml with the CR we used before , for each CRD ]
  capabilities: Basic Install
  categories: Modernization & Migration
  certified: "false"
  containerImage: quay.io/windupeng/myoperator:0.0.1
  ...
spec:
  description: > # Multiline description
  version: 0.0.1
  maturity: alpha
  icon: # Base64 of the image
  customresourcedefinitions:
    owned:
      - description: > # CRD multiline description
        displayName: Migration Toolkit for Applications
        kind: Windup
        name: windups.windup.jboss.org
        version: v1
  displayName: Migration Toolkit for Applications Operator
  provider:
    name: Red Hat
```

```
deployments:
  - name: windup-operator.0.0.1
    spec:
      replicas: 1
      selector:
        matchLabels:
          name: windup-operator
      template:
        metadata:
          labels:
            name: windup-operator
        spec:
          serviceAccountName: windup-operator
          containers:
            - name: windup-operator
              image: quay.io/windupeng/myoperator:0.0.1
              imagePullPolicy: Always
              env:
                - name: WATCH_NAMESPACE
                  valueFrom:
                    fieldRef:
                      fieldPath: metadata.namespace
              resources:
                limits:
                  cpu: 1
                  memory: 200Mi
                requests:
                  cpu: 100m
                  memory: 50Mi
      strategy:
        type: Recreate
```

```
strategy: deployment
installModes:
  - supported: true
    type: OwnNamespace
  - supported: true
    type: SingleNamespace
  - supported: false
    type: MultiNamespace
  - supported: false
    type: AllNamespaces
keywords:
  - monitoring
  - security
  - alerting
  - metric
  - troubleshooting
  - run-time
  - migration
  - modernization
```

Test the operator on Openshift Operatorhub

Using project <https://github.com/operator-framework/operator-registry>

```
#!/bin/sh -x

# everytime you test you need to increase this version number
# it only affects your test, has no other purposes
version=0.0.1

# Create operator bundle image
podman build -f mta-operator/0.0.1/Dockerfile -t mta-operator-bundle:$version
                    mta-operator/0.0.1/
podman tag mta-operator-bundle:$version
                    quay.io/windupeng/mta-operator-bundle:$version
podman push quay.io/windupeng/mta-operator-bundle:$version

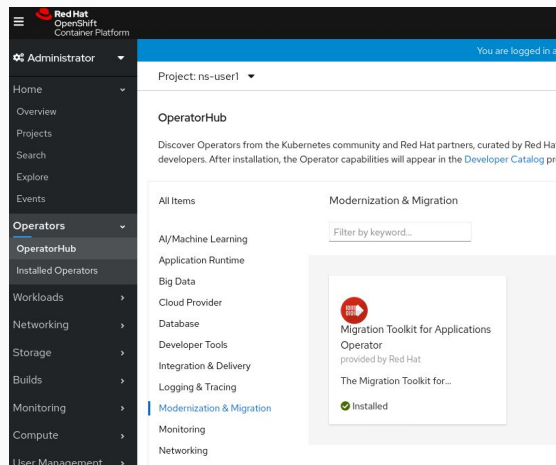
# Install operator-registry
# git clone https://github.com/operator-framework/operator-registry
# cd operator-registry
# make build

# Build operator catalog
operator-registry/bin/opm index add --bundles
                    quay.io/windupeng/mta-operator-bundle:$version \
                    --from-index quay.io/openshift-community-operators/catalog:latest \
                    --tag quay.io/windupeng/mta-operator-test-catalog:$version
podman push quay.io/windupeng/mta-operator-test-catalog:$version

sleep 20

oc apply -f catalog-source.yaml
```

1. build **bundle** image
2. tag image
3. push image
4. build **catalog** image
5. push image
6. deploy **catalog** into OCP



Execution flow



Execution Summary : Operator installation by cluster wide admin

Red Hat OpenShift Container Platform

You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: ns-user1

OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. You can purchase commercial software from the [Red Hat Marketplace](#). You can install Operators on your clusters to provide optional add-ons and shared services to your developers. When you install an Operator, the Operator capabilities will appear in the [Developer Catalog](#) providing a self-service experience.

All Items

AI/Machine Learning

Application Runtime

Big Data

Cloud Provider

Database

Developer Tools

Integration & Delivery

Logging & Tracing

Modernization & Migration

Monitoring

Networking

OpenShift Optional

Security

Storage

Streaming & Messaging

Migration Toolkit for Applications Operator
provided by Red Hat
The Migration Toolkit for Applications (MTA) is a web console application that support large-scale...

Transformation Advisor Operator
provided by IBM
IBM Cloud Transformation Advisor helps you plan, prioritize, and package your on-premise workload...

vFunction Operator
provided by vFunction
vFunction is a cutting-edge code analysis, machine learning, and...

vFunction Operator
provided by vFunction
vFunction is a cutting-edge code analysis, machine learning, and...



Migration Toolkit for Applications Operator

0.0.1 provided by Red Hat

Install

Operator Version
0.0.1
The Migration Toolkit for Applications (MTA) is a web console application modernization and migration projects across...

Capability Level
 Basic Install
 Seamless Upgrades
 Full Lifecycle
 Deep Insights
 Auto Pilot
In order to get more information about the tool, please proceed to [Getting Started](#)

Provider Type
test-catalog

Provider
Red Hat

Repository



OperatorHub > Operator Installation

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date.

Update Channel *
 alpha

Installation Mode *
 All namespaces on the cluster (default)
This mode is not supported by this Operator
 A specific namespace on the cluster
Operator will be available in a single namespace only.

Installed Namespace *
ns-user1

Approval Strategy *
 Automatic
 Manual

Install Cancel

Project: ns-user1

Installed Operators

Installed Operators are represented by Cluster Service Versions within this namespace. For more information, see the [Operator Lifecycle Manager documentation](#) or create an Operator and Cluster Service Version using the [OperatorHub](#).


| Name | Managed Namespaces | Status |
|--|--------------------|-------------------------|
| Migration Toolkit for Applications Operator 0.0.1 provided by Red Hat | ns-user1 | Succeeded Up to date |



Execution Summary : CR creation by user


Project: ns-user1

Installed Operators > Operator Details

 Migration Toolkit for Applications Operator
0.0.1 provided by Red Hat

Details | YAML | Subscription | Events | Migration

Provided APIs

 Migration Toolkit for Applications

It creates an MTA Web Console application. To access the MTA Web Console application from : * Developer Perspective. Please go to Topology and click on the small arrow icon on web-console pod. * Administrator Perspective. Please go to Networking->Routes and click on location field in the route element. In order to connect with default login credentials , please use "mta" as username and "password" as password. * Known issue If you want to customize the MTA Web Console instance's parameters and you can not see any in the `Form view`, please switch to the `YAML view` and change them as needed.

[Create Instance](#)



Project: ns-user1

Migration Toolkit for Applications Operator > Create Windup

Create Windup

Create by completing the form. Default values may be provided by the Operator authors.

Configure via: Form View YAML View

Note: Some fields may not be represented in this form. Please select "YAML View" for full control of object creation.

Name *
mta

Labels
application=windup x

max_post_size *
4294967296


executor_cpu_request *
0.5

messaging_serializer *
http.post.serializer



Project: ns-user1



Installed Operators > Operator Details

 Migration Toolkit for Applications Operator
0.0.1 provided by Red Hat

Details | YAML | Subscription | Events | Migration To

Windups


Name Search by name...

| Name ↑ | Kind ↓ |
|--|--------|
|  mta | Windup |
|  mta2 | Windup |









Project: ns-user1

Installed Operators > windup-operator:0.0.1 > Windup Details


 mta

Details | YAML | Resources

Filter Name Search by name...

| Name ↑ | Kind ↓ |
|--|------------|
|  mta | Deployment |
|  mta | Service |
|  mta-76644bcd4f | ReplicaSet |
|  mta-76644bcd4f-62r9k | Pod |
|  mta-amq | Service |
|  mta-executor | Deployment |





Migration Toolkit for Applications

Web Console

Username or email:

Password:

[Log In](#)

Welcome to the Migration Toolkit for Applications Web Console. Learn more about Migration Toolkit for Applications from the documentation.

References



References

Windup Operator Github project : github.com/windup/windup-operator

Kubernetes API reference : <https://kubernetes.io/docs/reference/>

Fabric8 Kubernetes client : <https://github.com/fabric8io/kubernetes-client> // <https://quarkus.io/guides/kubernetes>

Fabric8 Kubernetes mock-server : <https://itnext.io/mock-kubernetes-api-server-in-java-using-fabric8-kuber.....>

Quarkus <https://quarkus.io>

OperatorHub project <https://github.com/operator-framework/community-operators>
<https://github.com/operator-framework/operator-registry>

JIB <https://quarkus.io/guides/container-image>

dive <https://github.com/wagoodman/dive>

podman <https://podman.io/>

Minikube <https://minikube.sigs.k8s.io/docs>

Quay.io <https://quay.io>

Kind <https://kind.sigs.k8s.io/docs/>

Thank you



Thank you :)



@vilojona



jvilalop@redhat.com



aytartana.wordpress.com



github.com/jonathanvila



The End