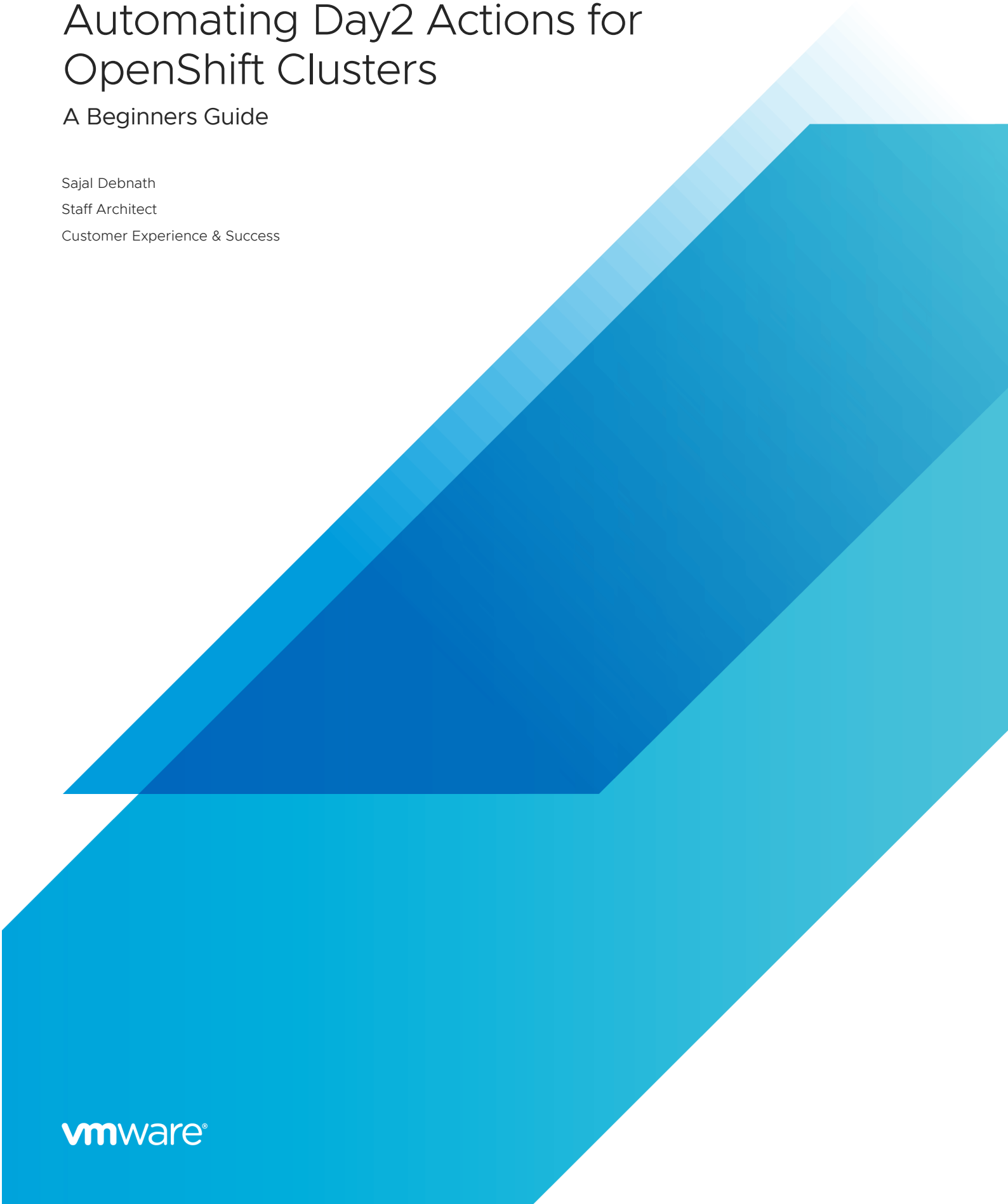
Automating Day2 Actions for OpenShift Clusters

A Beginners Guide

Sajal Debnath
Staff Architect
Customer Experience & Success



vmware®

Table of contents

1. Purpose

Recently I was asked to explore and possibly automate the various Day2 actions of an OpenShift cluster. This document provides guidance on how to automate any Day2 actions for OpenShift Cluster using vRealize Orchestrator and ultimately offer them as XaaS services in vRealize Automation.

While this document covers only a single use case, it provides the framework, following which any and every task for OpenShift Clusters can be automated.

2. Environment

The following environments were used to build and test out the solution.

Table 1: Environment Details

Product	Version
Red Hat OpenShift Container Platform	4.5.x
vRealize Orchestrator	8.1
vRealize Automation	8.1

3. Automation Options

Before we select the automation options, we need to decide how to automate. For any task that you want to automate, it has two parts:

1. Finding out how the task is manually done in OpenShift and how to automate it in there
2. How to automate the same task through vRealize Orchestrator

Looking at the above, we can see that in vRealize Orchestrator we can automate the tasks in two different ways.

1. Running 'kubectl' or 'oc' commands inside a guest VM (using the amazing "Guest Script Manager Package" created by Christophe Decanini (*https://code.vmware.com/samples/1317/guest-script-manager-package*).
2. Using the REST APIs directly

Choosing the right process

Below table provides a comparison of the above-mentioned process:

Table 2: Automation Options

AUTOMATION OPTIONS	
RUNNING 'KUBECTL' OR 'OC' COMMANDS INSIDE A GUEST VM AS REMOTE EXECUTION COMMANDS IN VRO	
PROS	CONS
• Easier to implement at OpenShift level (well documented and straightforward to implement) • Tested way of doing the work done through kubectl or oc	• No official VMware support • Dependency on an external VM which at scale may become a bottleneck • Not scalable or reliable
USING DIRECT REST API	
PROS	CONS
• Easier to implement at vRO level • HTTP REST plugins in vRO are officially supported	• Lack of REST API documentation from OpenShift

Looking at the options we see both the options have positive and negative sides. If we look at HTTP REST way of doing the implementation, initially, the task is much harder as there is nearly no documentation. I had to struggle a lot to find the exact URI and request body for a task. But once you figure out the details, it is much easier to implement and track. Besides it will make the solution future proof.

Choice: Looking at both the options I chose HTTP REST as the preferred way to automate the tasks.

4. How to Find Information

As mentioned above, the next hurdle is to find out the exact details of the HTTP REST request. What should be the URI, what is the body of the request?

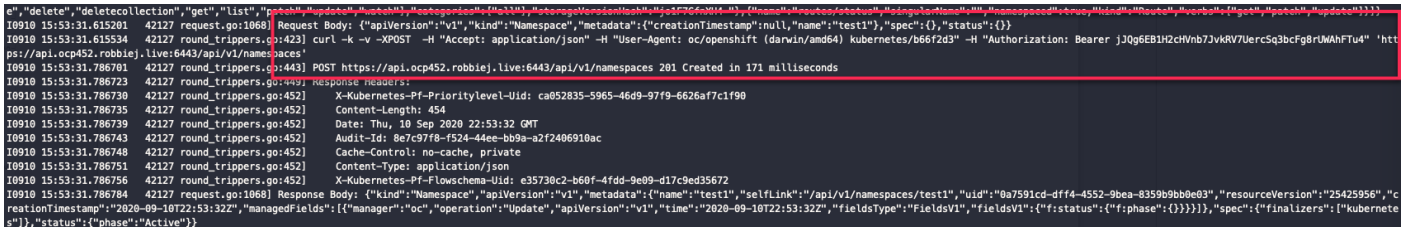
While Openshift 4.5 has API References, they are what they say references only and not extensive. For a beginner with OpenShift API's they don't help much (*https://docs.openshift.com/container-platform/4.5/rest_api/authorization_apis/localresourceaccessreview-authorization-openshift-io-v1.html*).

My colleague Rafael Brito provided his amazing insight and guided me through this. The trick is very simple, just run the same task using OpenShift CLI (oc) with –loglevel=10 option. This will list out all the tasks it is doing. OC is basically a wrapper over REST API calls. So, it lists out all the calls it is making. By tracing the details, I was able to get all the information I needed.

For example, look at the truncated output below. I want to create a namespace and want to know the API URI and the body of the request along with other details.

The command I am going to run is "`oc create namespace test1 --loglevel=10`". Look at the truncated information, specially at the very end.

Figure 1: Truncated Output



It basically lists out the following:

Request Body: {"apiVersion":"v1","kind":"Namespace","metadata":{"creationTimestamp":null,"name":"test1"},"spec":{},"status":{}}

Which is:

```
{
  "apiVersion": "v1",
  "kind": "Namespace",
  "metadata": {
    "creationTimestamp": null,
    "name": "test1"
  },
  "spec": {},
  "status": {}
}
```

From the above we can easily build the body of the request as:

```
{
  "apiVersion": "v1",
  "kind": "Namespace",
  "metadata": {
    "name": "test1"
  },
  "spec": {},
  "status": {}
}
```

Next line is

```
curl -k -v -XPOST  -H "Accept: application/json" -H "User-Agent: oc/openshift (darwin/amd64)
kubernetes/b66f2d3" -H "Authorization: Bearer xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
'https://api.ocp452.xxxxxxx.live:6443/api/v1/namespaces'
```

So, we get the following fields for Header:

```
"Accept: application/json"
```

```
"Authorization: Bearer xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx "
```

And finally, the API URI as `/api/v1/namespaces'.`

Using all these, now I can automate the task of creation of namespace.

By following this method, you can get details of pretty much any task you want to automate.

5. The Package

The whole vRO package is available at *https://github.com/sajaldebnath/openshift-day2-action* . Import the package in vRO and you would be able to use it as is. The package contains three workflows. They are explained below:

- Install vRLI Logging in OpenShift – The workflow to enable logging to vRLI in OpenShift Cluster
- Delete vRLI Logging in OpenShift - The workflow to delete logging to vRLI in OpenShift Cluster
- Run a Task on OpenShift Cluster – The magical workflow – Runs anything you want on OpenShift Cluster

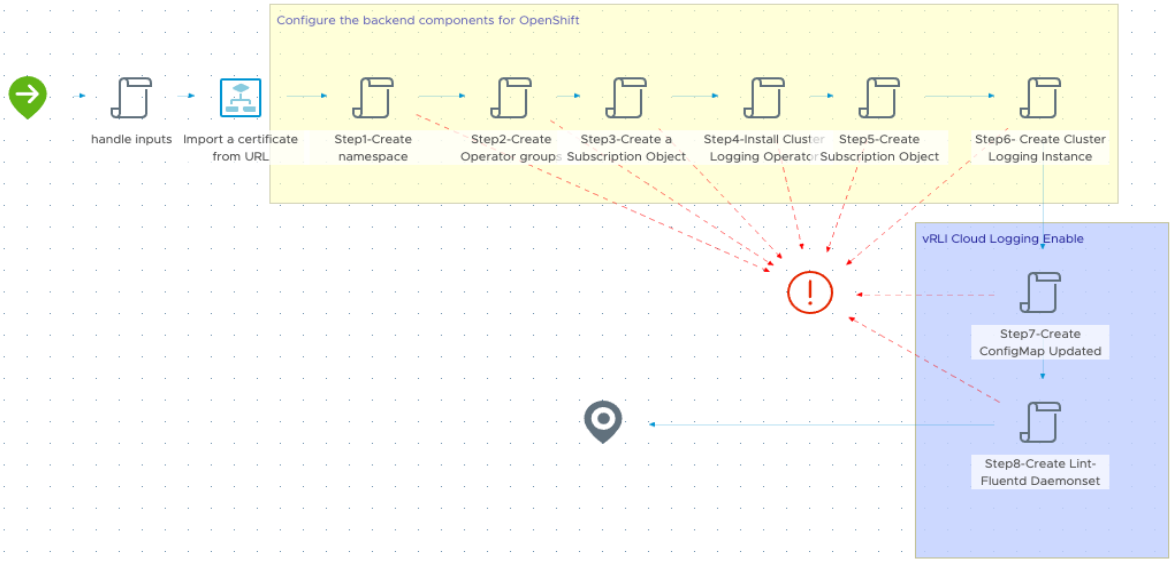
6. The Task - Example

In this section I will explain a sample task that I automated using the above method. The example is to automate the task of configuring OpenShift Log Forwarding to vRealize Log Insight Cloud. I have created the following vRO code to do just that.

6.1. Workflow – Install

The workflow to do the task is “Install vRLI Logging in OpenShift”.

Figure 2: vRO Workflow to Automatically Install Log Forwarding in OpenShift to vRLI Cloud



For these tasks I am creating transient Rest Hosts in vRO as I do not want to store the information in vRO.

6.1.1. Detailed Steps Information

Detailed steps for the installation are provided in the following two places:

- OpenShift official documentation on how to Install Cluster Logging: *https://docs.openshift.com/container-platform/4.5/logging/cluster-logging-deploying.html*
- VMware Log Insight Log Source configuration document for OpenShift: *https://www.mgmt.cloud.vmware.com/li/sources/details?id=openshift_kubernetes_application*

The steps I followed are listed in detail above.

Basically, to do the task you first install cluster logging in OpenShift Cluster and then configure log forwarding to vRealize Log Insight Cloud.

1. Create Namespace
2. Create OperatorGroup Object
3. Create Subscription Object – elasticsearch-operator
4. Install Cluster Logging Operation
5. Create Subscription Object
6. Create Cluster logging instance
7. Create ConfigMap
8. Create Daemonset

6.1.2. Inputs for the Workflow

The workload takes the following inputs:

- **K8sFqdn**: K8s cluster fqdn including the port. For example, “api.ocp452.xxxxx.live:6443”.
- **accessToken**: Token to authenticate with K8s cluster, you can get that from OpenShift cluster UI. Below screenshots shows an example.

Figure 3: Step1

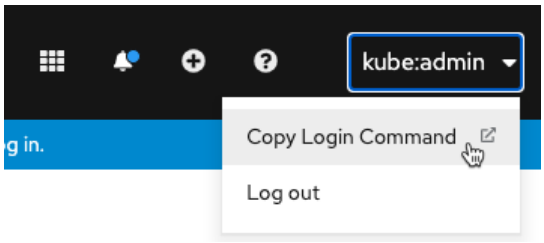


Figure 4: Step 2

Your API token is

dU395FEX2XBGGSPDxY9FDiUN5uY9eI98RUTbDeBlI_k

- **logUri:** The vRealize Log Insight Cloud log ingestion endpoint. Typically, it is *https://data.mgmt.cloud.vmware.com/le-mans/v1/streams/ingestion-pipeline-stream*
- **logApi:** Access Key to forward logs to Log Insight Cloud instance. For details check the documentation above.

6.1.3. Workflow Details – Create Namespace

The required steps are provided in detail in the above two document link. For the actual workflows, they follow the same structure.

```
if(fqdn)
{
    if(token)
    {
        try
        {
            var restHost = RESTHostManager.createHost(fqdn);
            var url = "https://" + fqdn;
            restHost.url = url;
            System.debug ("Created RESTHost object name : " + restHost.name);
            System.debug ("Created RESTHost object URL : " + restHost.url);

            // Create transient REST host
            var host = RESTHostManager.createTransientHostFrom(restHost);
            System.debug ("Created Transient RESTHost object : " + host.name);


            // Define JSON body to create namespace

            var contentObj    =  {
                "apiVersion" : "v1",
                "kind" : "Namespace",
                "metadata" : {
                  "name" : "openshift-operators-redhat",
                  "labels" : {
                      "openshift.io/cluster-monitoring": "true"
                  }
```

```
            }
          };

        System.log ("Content as object : " + JSON.stringify(contentObj));


        var content = JSON.stringify(contentObj);
        System.debug ("Content as string : " + content);

        // Define REST method is POST
        var method = "POST";
        System.debug ("REST method : " + method);

        // Create request object
        var request = host.createRequest(method,url + "/api/v1/namespaces",content);
        request.contentType = "application/json";
        request.setHeader("Accept","application/json");
        request.setHeader("Authorization","Bearer " + token);
        System.debug ("Full request URL : " + request.fullUrl);
        System.debug ("Content Type : " + request.contentType);


        // Execute REST method
        var response = request.execute();
        System.log ("Response status code : " + response.statusCode);
        System.debug ("Response content as string : " + response.contentAsString);

        if(response.statusCode === 200 || response.statusCode === 201 )
        {
            System.log("Request was Successful");
        }
        else
        {
            System.log("Request has returned an error status code: " + response.statusCode);
            throw "Error: " + response.statusCode;
        }
    }
    catch (e)
    {
        System.error("An un-handled exception was caught: " + e);
        errorCode = e;
        throw errorCode;
    }
}
else
{
    throw "No Token was provided";
}
}
```

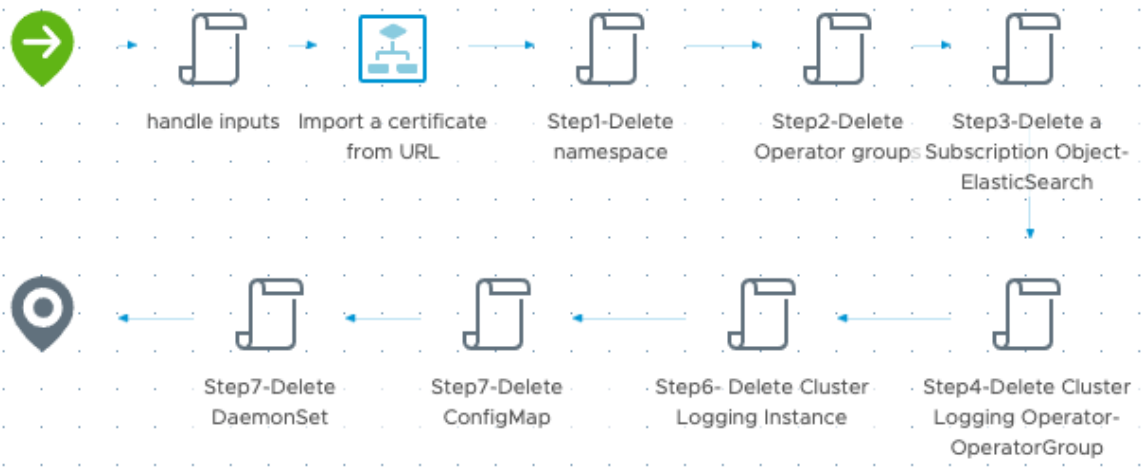
```
else
{
    throw "No K8s Cluster FQDN was provided";
}
```

** The same structure is followed to make all the other requests.

6.2.Workflow – Delete

The workflow to clean the environment for the configurations you did in the other workflow. This will delete and undo all the above steps.

Figure 5: Delete the logging



Sample workflow for delete:

```
var restHost = RESTHostManager.createHost(fqdn);
var url = "https://" + fqdn;
restHost.url = url;
System.debug ("Created RESTHost object name : " + restHost.name);
System.debug ("Created RESTHost object URL : " + restHost.url);

// Create transient REST host
var host = RESTHostManager.createTransientHostFrom(restHost);
System.debug ("Created Transient RESTHost object : " + host.name);
var content = "";
// Define JSON body to create namespace
// Define REST method is POST
var method = "DELETE";
System.debug ("REST method : " + method);

// Create request object
var request = host.createRequest(method,url + "/api/v1/namespaces/openshift-operators-
redhat",content);
```

```
request.contentType = "application/json";
request.setHeader("Accept","application/json");
request.setHeader("Authorization","Bearer " + token);
System.debug ("Full request URL : " + request.fullUrl);
System.debug ("Content Type : " + request.contentType);


// Execute REST method
var response = request.execute();
System.log ("Response status code : " + response.statusCode);
System.debug ("Response content as string : " + response.contentAsString);
```

The above code sample is for removing the namespace. The other steps follows the same structure.

7. The Magical Workflow

I have included a workflow which, when used, can do pretty much whatever task you want to run on OpenShift.

The workflow is "Run a Task on OpenShift Cluster". The inputs are

- **K8sFqdn**: K8s cluster fqdn including the port. For example, "api.ocp452.xxxxx.live:6443".
- **accessToken**: Token to authenticate with K8s cluster, you can get that from OpenShift cluster UI. Below screenshots shows an example.
- **apiUri:** The URI of the API in which to performa the task
- **method:** HTTP REST method to perform
- **requestBody:** Body of the Request. Put Data in JSON format. Keep blank for no body.

Figure 6: Run a generic task

7.1. Workflow details:

```
if(fqdn)
{
    if(token)
    {
        try
        {
            var restHost = RESTHostManager.createHost(fqdn);
            var url = "https://" + fqdn;
```

```
        restHost.url = url;
        System.debug ("Created RESTHost object name : " + restHost.name);
        System.debug ("Created RESTHost object URL : " + restHost.url);

        // Create transient REST host
        var host = RESTHostManager.createTransientHostFrom(restHost);
        System.debug ("Created Transient RESTHost object : " + host.name);


        // Define JSON body to create namespace

        //var contentObj    =  JSON.parse(requestBody);

        //System.log ("Content as object : " + JSON.stringify(contentObj));


        //var content = JSON.stringify(contentObj);
        var content = requestBody ;
        System.debug ("Content as string : " + content);

        // Define REST method is POST
        //var method1 = method;
        System.debug ("REST method : " + method);

        // Create request object
        var request = host.createRequest(method,url + apiUri,content);
        request.contentType = "application/json";
        request.setHeader("Accept","application/json");
        request.setHeader("Authorization","Bearer " + token);
        System.debug ("Full request URL : " + request.fullUrl);
        System.debug ("Content Type : " + request.contentType);


        // Execute REST method
        var response = request.execute();
        System.log ("Response status code : " + response.statusCode);
        System.debug ("Response content as string : " + response.contentAsString);
        requestData = response.contentAsString ;
        if(response.statusCode === 200 || response.statusCode === 201 )
        {
            System.log("Request was Successful");
        }
        else
        {
            System.log("Request has returned an error status code: " + response.statusCode);
            throw "Error: " + response.statusCode;
        }
    }
    catch (e)
```

```
        {
            System.error("An un-handled exception was caught: " + e);
            errorCode = e;
            throw errorCode;
        }
    }
    else
    {
        throw "No Token was provided";
    }
}
else
{
    throw "No K8s Cluster FQDN was provided";
}
```

7.2. Explanation

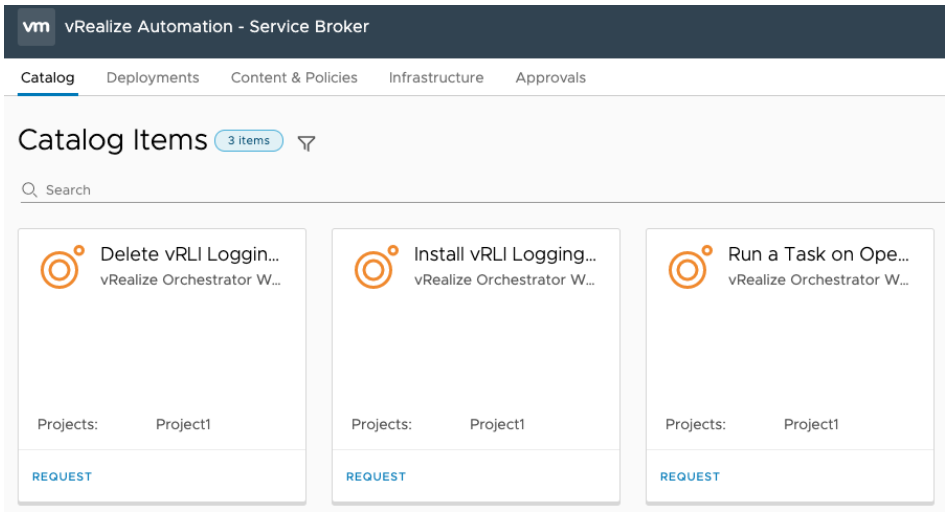
The workflow is simple and straightforward. It takes the inputs and calls out the HTTP REST API to carry out the task.

The reason this workflow is important is this provides us the framework to build a larger workflow to automate any Day2 tasks we want to do.

8. Building a XaaS Service

Once you have the package imported into vRO, you have the workflows ready to be used. Head over to vRealize Automation and import them into Service Broker. Follow the guides provided in the VMware official document at *https://docs.vmware.com/en/vRealize-Automation/8.1/Using-and-Managing-Service-Broker/GUID-416D6130-96AD-4912-9AD2-D52CE30CAA31.html* to achieve this. The below picture shows how it would look once the workflows are published in Service Broker.

Figure 7: Catalog Items in Service Broker



9. Known Limitations

Please note, these workflows are the bare minimum framework and should be improved upon. Provided below is a list of the known limitations:

- The workflows are written as proof of concept. These does not include best practices or enough error checking.
- These scripts should be modified to make them production grade.

10. Future Work

Provided below is a list of future work that can be done to make these workflows more acceptable.

- Create customer resources in vRA for OpenShift clusters.
- All the custom Day-2 Actions would automatically attach to the OpenShift Clusters and be available as Day2 Action (Out of the Box).

11. Acknowledgements

Mandy Botsko-Wilson: For trusting me and giving me the task in the first place.

Robbie Jerrom: Providing me with the OpenShift Cluster environment.

Rafael Brito: For guiding me how to and where to find the required information.

Glossary

Term	Details
vRO	vRealize Orchestrator
vRLI	vRealize Log Insight

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 vmware.com