**Enforcing Container Image Security (beta) Using Custom Policy**

**Abstract:**

The focus of this blog post is to secure Docker images before entering a namespace or a cluster using custom IBM Container Image Security Enforcement custom policies. This article introduces an overview, considerations, and recommendations using custom policies for identifying and preventing vulnerable Docker containers from being deployed in the IBM Cloud environment.

**About IBM Container Image Security Enforcement:**

IBM Container Image Security Enforcement can be configured to verify container images before they are deployed to a cluster on IBM® Cloud Kubernetes Service. Images are controlled where they are deployed from, enforce Vulnerability Advisor policies, and certify that content trust is suitably applied to an image before deployment.

**Container Image Security Enforcement in a Cluster:**

IBM Container Image Security Enforcement automatically installs policies to provide a starting point for building security policy in a cluster. Additional information and how to install default policies can be found [here](https://console.bluemix.net/docs/services/Registry/registry_security_enforce.html#security_enforce). These policies will apply the default security policies onto all Kubernetes namespaces in the cluster. The default policies that will install are:

1. **Default Cluster-wide policy** – Enforces that all images in all registries are from a trusted source and Vulnerability Advisor has no security reported issues on the image.
2. **Default Kube-system policy** – A namespace-wide policy installed specifically for the kube-system namespace. It allows all images for any container registry to be deployed into the kube-system namespace without restriction.
3. **Default IBM-system policy** – A namespace-wide policy installed by default. This policy permits all images from any container registry to be deployed into the ibm-system without restrictions. Additionally, repositories that are used will be permitted to configure the cluster and to install or upgrade Image Security Enforcement.

**What will I explore in this article?**

This blog will focus on the default **cluster-wide policy** and its two important flag components: “**va:**” and “**trust:**” as shown in Figure 1. Before moving forward, it is important to explain that “va:” stands for **Vulnerability Advisor** and its primary function is to scan images for any security vulnerabilities. The main function of the “trust:” flag is to ensure the image’s integrity, to confirm that it’s signed, and be sure that it originated from a trusted source. Signing an image and verifying its integrity is explained in detail [here](https://console.bluemix.net/docs/services/Registry/registry_trusted_content.html#registry_trustedcontent).

**Figure 1 – An Example of a Default Cluster-Wide Policy.yaml File**

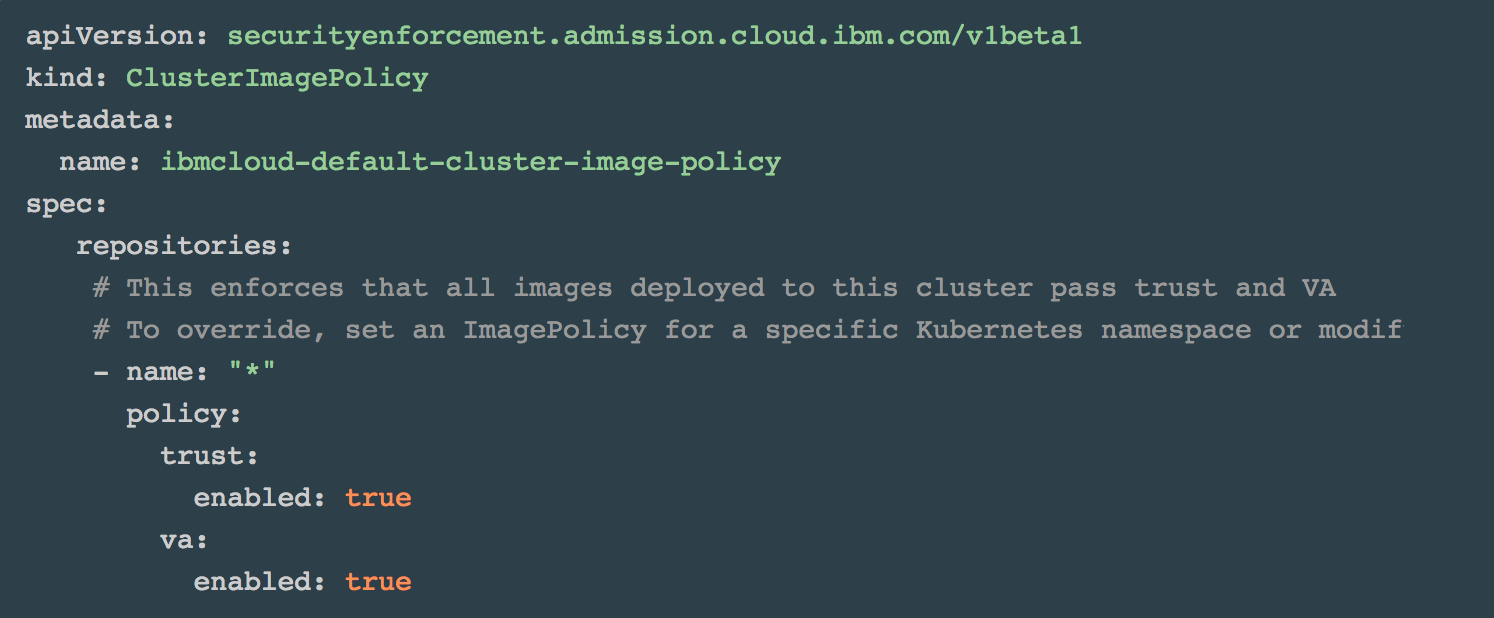
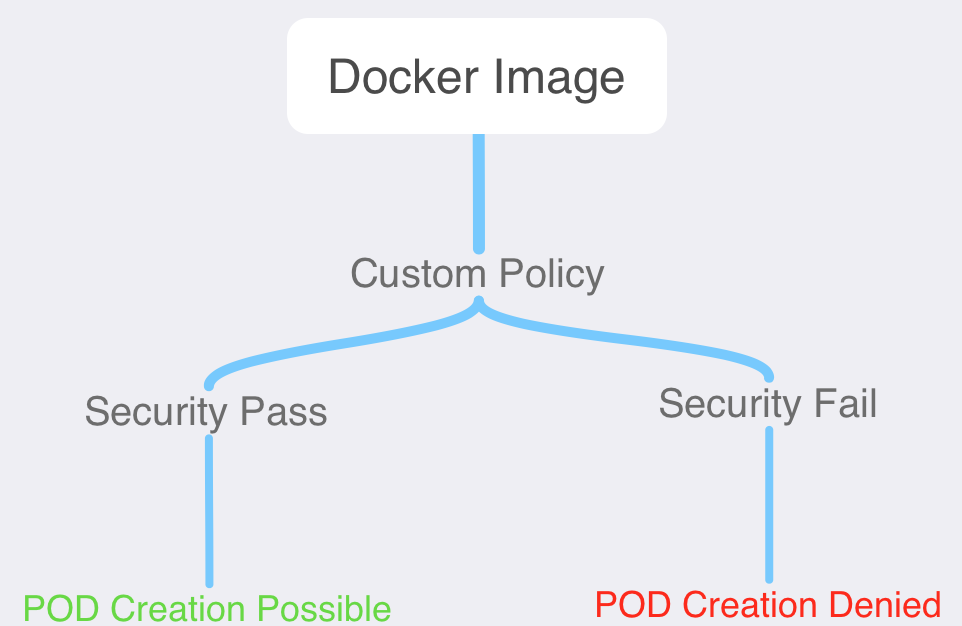


Figure 2 features a high-level overview of an image that’s been screened by the default cluster-wide policy. Depending on the policy’s flag configuration in a cluster, the image will be validated for the next step in its deployment process differently.

**Figure 2 – High-Level Overview of a Custom Policy**

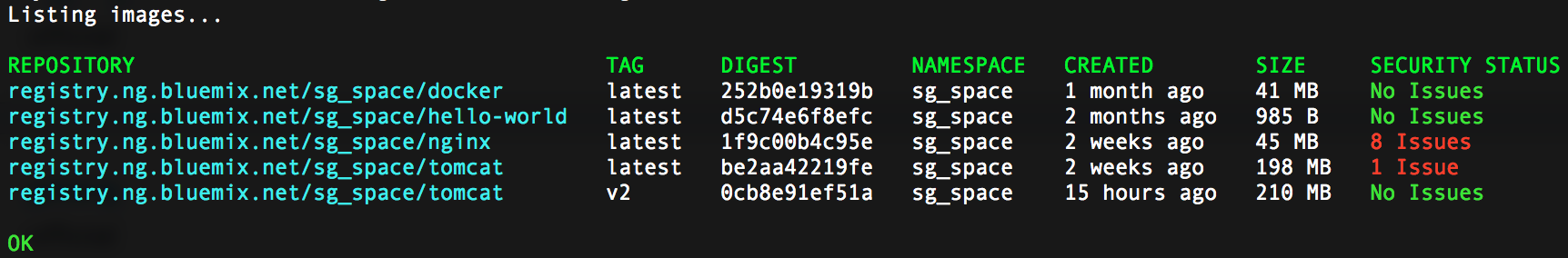


**Discovering the outcome of a tailored default cluster-wide policy against an image:**

The image list in Figure 3 displays information that can differ when the default cluster-wide policy flags “**trust:**” and “**va:**” are modified.

**$** bx cr images

**Figure 3 – IBM Cloud Image Repository**



For example, when both the “**trust:**” and “**va:**” components in the default cluster-wide policy, exhibited in the [*DefaultCluster-wide.yaml*](https://github.com/ibm-client-success/iks-container-registry/blob/master/image-security/vulnerability-scanning-bluemix/Contents/defaultclusterwidepolicy_ff.yaml) file below, are set to “**false**” the policy will not prevent a POD deployment regardless of the integrity or any potential vulnerabilities in an image. This means that image deployment and POD creation is allowed without any issues, such as issues reported by Vulnerability Advisor and/or the trustworthiness of an image.

**$** vi DefaultCluster-wide.yaml

apiVersion: securityenforcement.admission.cloud.ibm.com/v1beta1

kind: ClusterImagePolicy

metadata:

name: ibmcloud-default-cluster-image-policy

spec:

repositories:

# This enforces that all images deployed to this cluster pass trust and VA

# To override, set an ImagePolicy for a specific Kubernetes namespace or # modify this policy

- name: "\*"

policy:

trust:

enabled: false

va:

enabled: false

**$** kubectl apply -f DefaultCluster-wide.yaml

Results:



Attempting to deploy the **tomcat:latest** image using the [*tomcat.yaml*](https://github.com/ibm-client-success/iks-container-registry/blob/master/image-security/vulnerability-scanning-bluemix/Contents/tomcatdeployment.yaml) file example:

**Example of *tomcat.yaml* File**

apiVersion: apps/v1

kind: Deployment

metadata:

name: tomcat-deployment

labels:

app: tomcat

spec:

replicas: 2

selector:

matchLabels:

app: tomcat

template:

metadata:

labels:

app: tomcat

spec:

containers:

- name: tomcat

image: registry.ng.bluemix.net/sg\_space/tomcat:latest

ports:

- containerPort: 80

**$** kubectl apply -f tomcat.yaml

Results:

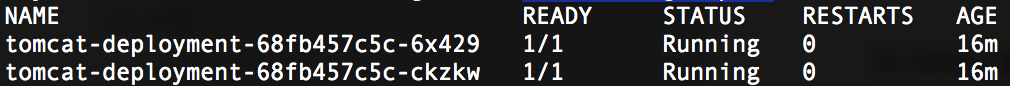


**$** kubectl get deployments

Results:

**$** kubectl get pods

Results:



**$** kubectl delete deployment tomcat-deployment

Results:



Now, let’s leave the default cluster-wide policy “**trust:**” to “**false**” and change the “**va:**” to “**true**” and see the results:

**$** vi DefaultCluster-wide.yaml

apiVersion: securityenforcement.admission.cloud.ibm.com/v1beta1

kind: ClusterImagePolicy

metadata:

name: ibmcloud-default-cluster-image-policy

spec:

repositories:

# This enforces that all images deployed to this cluster pass trust and VA

# To override, set an ImagePolicy for a specific Kubernetes namespace or # modify this policy

- name: "\*"

policy:

trust:

enabled: false

va:

enabled: true

**$** kubectl apply -f DefaultCluster-wide.yaml

Results:



**$** kubectl apply -f tomcat.yaml

Results:

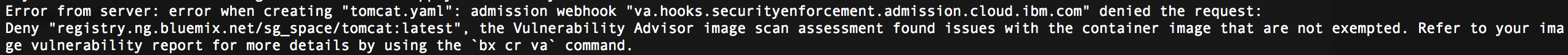
**

Image **tomcat:latest** has one issue reported by the Vulnerability Advisor so POD deployment for this particular image will be prevented.

By having a secure image in the IBM Cloud image repositorywithout any vulnerability issues, we can attempt to deploy the **tomcat:v2** image. Let’s modify the default cluster-wide policy “**trust:**” to “**true**” and “**va:**” to “**false**”, then execute the *tomcat-v2.yaml* file and analyze the outcome:

**Example of *tomcat-v2.yaml* File**

apiVersion: apps/v1

kind: Deployment

metadata:

name: tomcatv2-deployment

labels:

app: tomcat

spec:

replicas: 3

selector:

matchLabels:

app: tomcat

template:

metadata:

labels:

app: tomcat

spec:

containers:

- name: tomcat

image: registry.ng.bluemix.net/sg\_space/tomcat:v2

ports:

- containerPort: 80

**$** vi DefaultCluster-wide.yaml

apiVersion: securityenforcement.admission.cloud.ibm.com/v1beta1

kind: ClusterImagePolicy

metadata:

name: ibmcloud-default-cluster-image-policy

spec:

repositories:

# This enforces that all images deployed to this cluster pass trust and VA

# To override, set an ImagePolicy for a specific Kubernetes namespace or modify this policy

- name: "\*"

policy:

trust:

enabled: true

va:

enabled: false

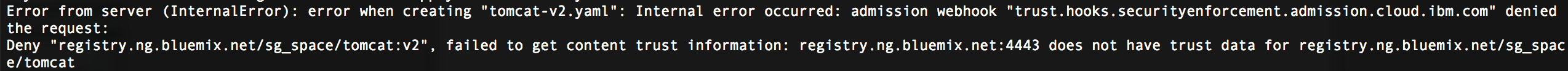
**$** kubectl apply -f DefaultCluster-wide.yaml

Results:



**$** kubectl apply -f tomcat-v2.yaml

Results:

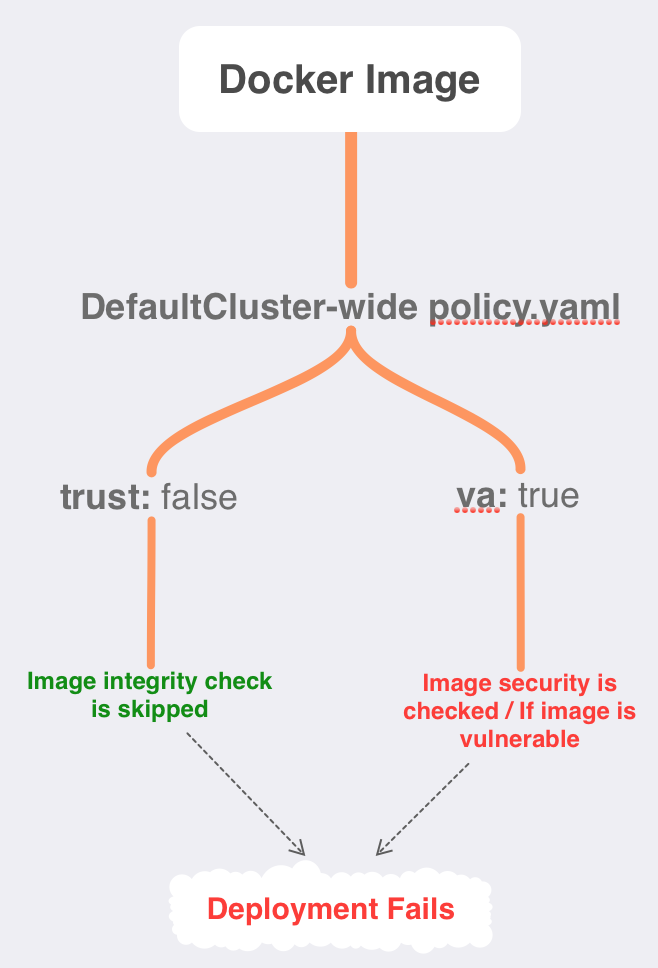
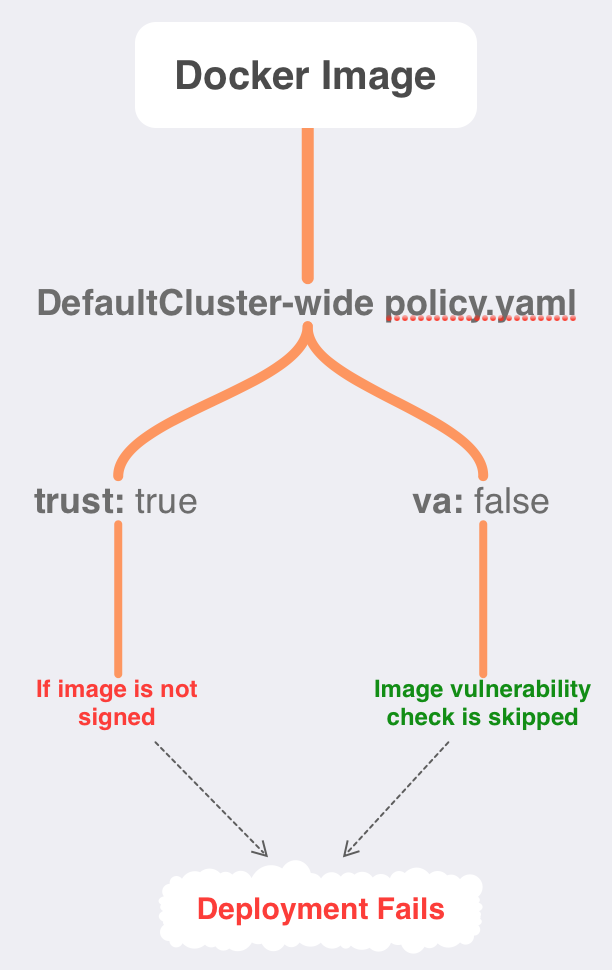


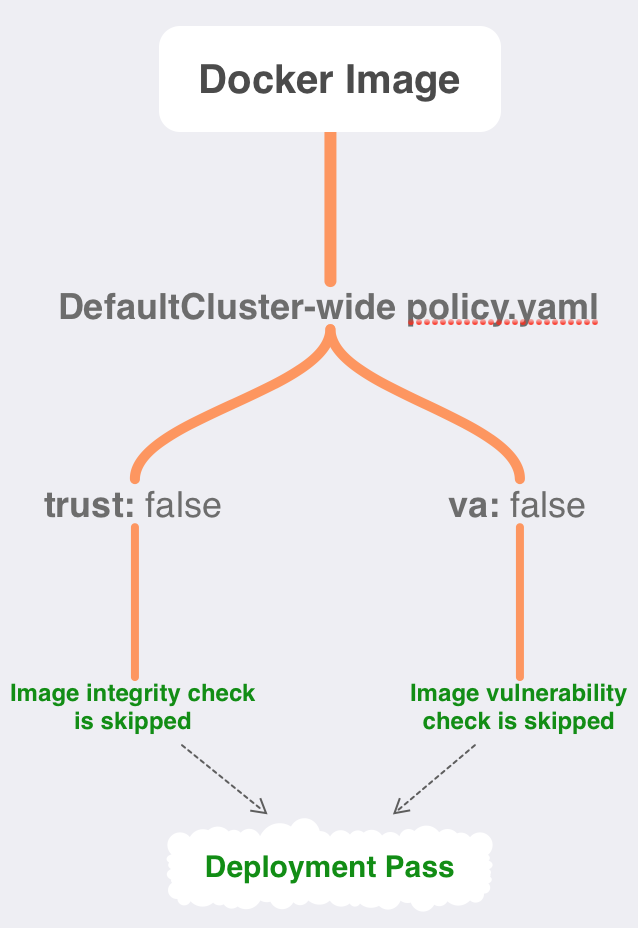
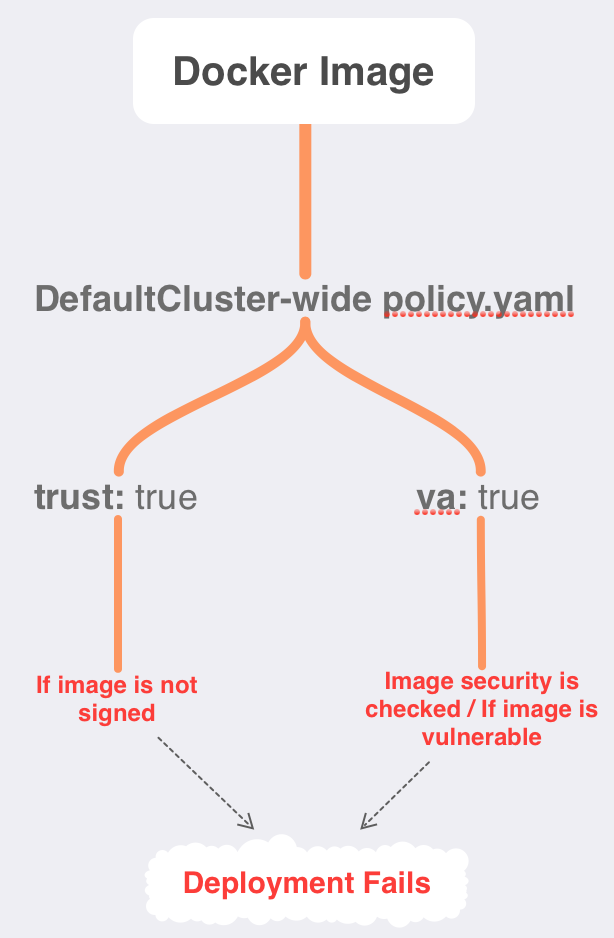
Deployment of the **tomcat:v2** image has been denied because the system has determined that this particular image was not signed when it was pulled from the source and is untrustworthy.

As shown in the discoveries thus far, the first flag of the default **cluster-wide policy** is applied, and the image is scanned to ensure its integrity before it is staged for deployment. Therefore, if the image is not signed and it is not from a trusted source, then the deployment will be denied, and the second flag of the policy will not be applied. Deployment will cease at this point.

The figure below shows the four possible combinations of the component flags in a cluster-wide policy and the corresponding result of attempting to deploy an untrustworthy image with vulnerabilities.

**Figure 4 – High-Level Overview of the Custom Policy Possibilities Against an Image**

**Rectifying Image Issues to Pass POD Deployment**

Depending on the settings of the default cluster-wide policy components, “**trust:**” and “**va:**”, there are several ways to remedy image issues in order to successfully deploy in a cluster:

* Pull a more up-to-date image and verify that it is signed from a trusted source. This is the most common method.
* The user can manually update an existing image and sign an image in a repository to ensure the integrity of images in your registry namespace.
* Create an exception in a repository to bypass an image’s Vulnerability Advisor scan report for a particular reported issue(s).

**Summary**

IBM Container Image Security Enforcement custom policies are easy to be configured and most importantly contribute in pushing secure images into a namespace before the image is deployed in a cluster. Ultimately, these custom policies are put in place to safeguard a cloud environment by validating the images **before** allowing them to be deployed into a kubernetes cluster.