External Lifecycle Management Guide

Overview

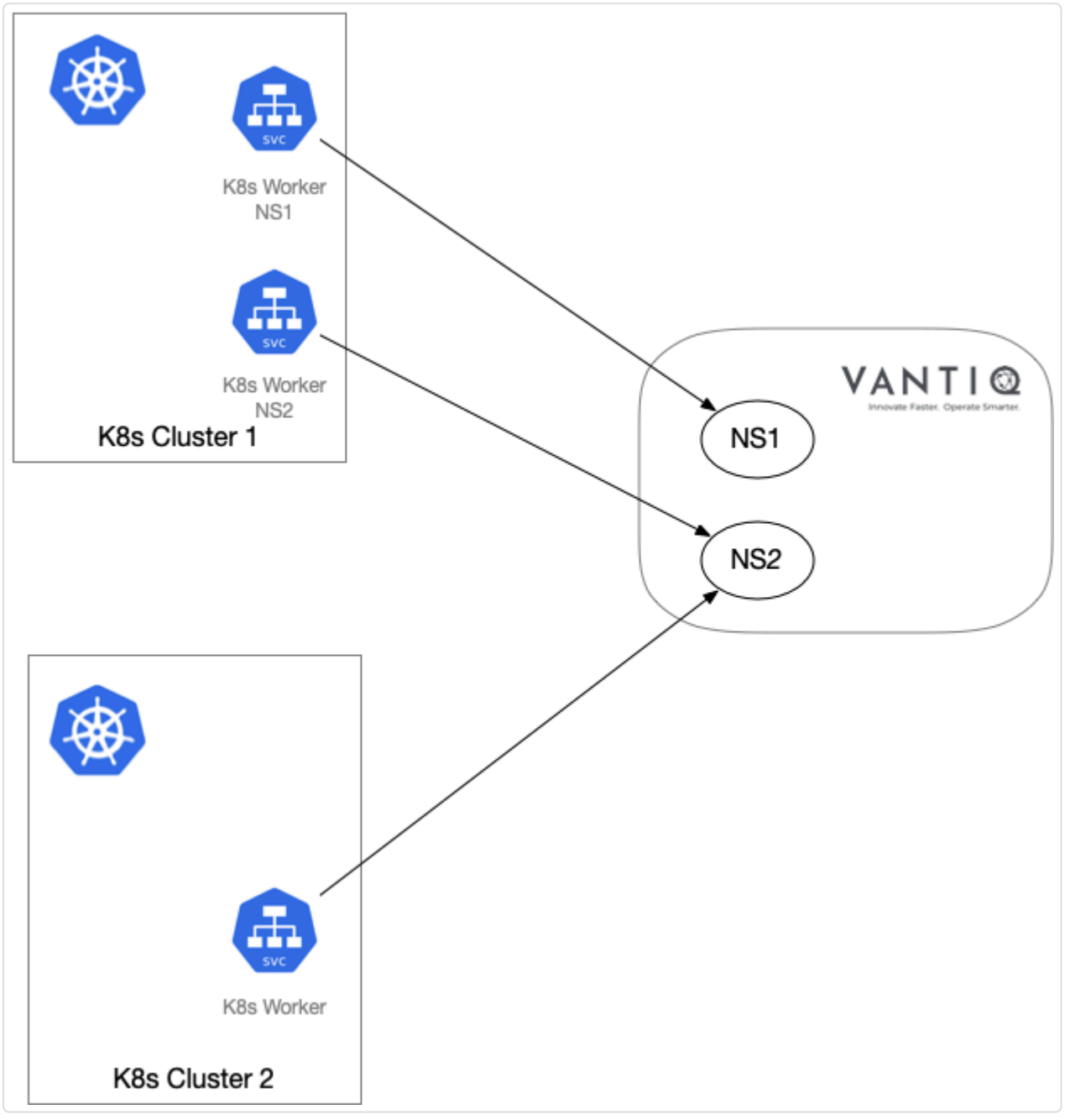
Vantiq applications often involve external components. Some of these, for example, Enterprise Connectors, can benefit from running in a managed environment such as Kubernetes. From the connector perspective, the advantage offered by a managed environment is that the component can be automatically restarted in the event of failure.

This guide describes how the Vantiq system allows you to make use of a Kubernetes environment as part of your application system. This guide assumes that you are familiar with Kubernetes and Kubernetes components.

System View

From the Vantiq perspective, a Kubernetes Cluster is an external operating environment to which Vantiq (more specifically, a Vantiq namespace) has been granted access. Any physical relationship between Vantiq and a particular K8s Cluster (*i.e.* whether the Vantiq installation is deployed in the same Kubernetes environment) is immaterial. Access is granted via the installation of a K8s Worker. This must be performed by a Kubernetes administrator.

A Vantiq installation and, thus, a Vantiq namespace can be connected to 0 or more Kubernetes cluster. When that connection is made, the Vantiq namespace administrator can deploy K8s Installations into that cluster, shut them down, restart them, and, of course, undeploy them.



The figure above shows a Vantiq installation that contains two namespaces (NS1 & NS2) that are of interest. In this case, two K8s Clusters are involved, with K8s Cluster 1 registered with both of these namespaces, and K8s Cluster 2 registered only in NS2. You may also note the existence of the K8s Worker(s) in each K8s Cluster. We will discuss this further below.

Vantiq gains access to a Kubernetes cluster via the following actions:

- Creation of the K8s Cluster, and
- Deploying the K8s Worker

The Vantiq installation does not perform operations directly on the Kubernetes cluster. Instead, it relies on the K8s Worker to pick up K8s Workitems periodically. The K8s Worker is expected to be installed into the Kubernetes cluster by a Kubernetes administrator. This architecture allows the Kubernetes administrator to be aware of Vantiq's access.

Loading Images into a Kubernetes Cluster

Kubernetes loads container images from an image registry such as Docker Hub (hub.docker.com), quay.io, etc. Any image to be used must be available from some image registry.

For connector images, choose the image registry to which your connector image was published. Please see Building Docker Images for information about how to create and publish images for connectors.

The K8s Worker is a Vantiq-provided component running inside your Kubernetes cluster. The K8s Worker is available from the *quay.io* container image registry. To load the K8s Worker, you will need be authorized to access the K8s Worker repository. This access can be obtained by contacting Vantiq support. The access credentials will be made available to the K8s Worker as a Kubernetes secret.

K8s Worker

The K8s Worker is a Cron Job that is installed into the Kubernetes cluster. The K8s Worker will be downloaded from the *quay.io* container image registry (see above). As part of the installation, it is configured (via environment variables) with information allowing it to contact a Vantiq namespace and to specify the K8s Cluster on whose behalf it is running.

Depending upon the roles granted to the K8s worker (see Deploying the K8s Worker), a particular worker may represent an entire Kubernetes cluster OR only a particular Kubernetes namespace within the Kubernetes cluster. The scope of the role granted the worker is controlled by the Kubernetes administrator performing the K8s Worker installation.

As noted above, the K8s Worker is, in Kubernetes terms, a Cron Job, operating only periodically (by default, once per minute). Consequently, requests made to a K8s Cluster (which result in K8s Workitems) may take some time to complete.

Kubernetes Components of a K8s Installation

The result of a *deploy* request on a K8s Cluster is the creation of a K8s Workitem. The work item instructs the K8s Worker to perform the work required to create the appropriate set of components in the Kubernetes cluster. This may involve the following components.

- Stateful Sets
- Ingress Definitions
- Config Maps
- Secrets
- Host Aliases
- References to existing components
    - Config Maps
    - Secrets

A *deploy* operation requested on a K8s Cluster requires that you provide a `name` and `namespace`. Each successful K8s Installation deployment will result in a set of components created in the namespace request requested. With the exception of references to Persistent Volumes (which are not in any namespace), all components created or referenced will be created or must already exist in the same namespace.

All K8s Installations successfully created represent a Kubernetes Stateful Set with the name and namespace provided. Each K8s Installation will also have a *headless service* (of Kubernetes type ClusterIP) created. This is a Kubernetes requirement.

Other Kubernetes components may be created, depending upon the configuration given for the K8s Installation. Components that are generated from the configuration are named in a manner to relate them to the installation/stateful set that caused them to be created. Any such created component will be in the same namespace as the Stateful Set, and have a name like `vantiq-<installation name>-<component type>[-<number>]`. For example, if an installation named *newinstallation* is created that includes a File type in its configuration, we will create a Config Map component named `vantiq-newinstallation-configmap-0`. The number suffix is added when there *could* be more than one of that component created as part of that installation's creation.

Specifically, the following configuration items result in specific, generated, Kubernetes components.

- `file`:
    - If a `File` entry has a `content` property, a new Config Map will be created to provide that content for each such `File` entry. All `File` entries that contain the same `path` component will belong to the same Config Map
    - If the `File` entry has no `content` property, then we create only a reference to an existing Config Map. No new Config Map is created.
- `inboundPort`:
    - An `inboundPort` entry will result in the creation of a Kubernetes Ingress definition. This definition is used to allow external requests for this installation to find their way to the installation.
    - A Kubernetes `NodePort` service will also be created.
- `environmentVariables`:
    - If an `environmentVariable` entry includes an `asSecret` value of `true`, and no `secretName` and `key` is provided, a new Kubernetes Secret will be defined to contain that secret.
    - if a `secretName` and `key` is provided, no new Kubernetes Secret is defined.

Upon completion, the K8s Installation is created in the Vantiq namespace. The definition of the K8s Installation is available here in the Resource Reference Guide.

When the `undeploy` operation on a K8s installation is requested, all of these created components will be removed.

The `shutdown` and `restart` operations neither create nor destroy Kubernetes components.

Access to a Kubernetes Cluster

All access to a Kubernetes cluster is provided via the deploying of a K8s Worker and the creation of a K8s Cluster object in the Vantiq Namespace.

Creation of the K8s Cluster object is described in the K8s Cluster section section. Creation via Modelo is outlined in the Creating a K8s Cluster section.

The following sections discuss the deploying of a K8s Worker.

Namespace Considerations

Kubernetes Namespaces

Each K8s Worker will run under the role defined in the Kubernetes service account within which it is operating. If that role is specific to the namespace in which the K8s Worker is installed (which will often be the case), a separate K8s Worker will need to be installed for each Kubernetes namespace. Moreover, each such Kubernetes namespace will need to have a separate Vantiq K8s Cluster defined. This is because the the K8s Workers identify themselves to Vantiq by K8s Cluster name.

Vantiq Namespaces

Each K8s Worker is provided with a Vantiq access token that allows it to connect to a Vantiq namespace. Consequently, if more than one Vantiq namespace is making use of the same Kubernetes cluster (or the same Kubernetes namespace within the Kubernetes cluster), each Vantiq namespace-Kubernetes Cluster/Kubernetes Namespace pair will require a separate K8s Worker.

K8s Worker Profile

The K8s Worker needs access to various system types. The K8s Worker's access token can be created using the `k8sWorker__system` profile. This profile in a namespace will be prefixed by that namespace's name. So, for a Vantiq namespace named *example*, you can use the `example.k8sWorker__system` profile. The standard namespace admin profile will work as well; the `k8sWorker__system` profile defines the minimum access required by the K8s Worker.

This profile will be present in namespaces created after Vantiq release 1.32. For namespaces created before that release, call the procedure `K8sManager.addK8sWorkerProfile()` (with no parameters) to add that profile to the Vantiq namespace. This should be called by a namespace admin while connected to the Vantiq namespace in question.

Note that this profile is tailored to the requirements of the K8s Worker. It provides access to only those types required for the K8s Worker to operate. It is neither intended nor suitable for the requirements of K8s Installations being created or managed via the K8s Worker.

Deploying the K8s Worker

Deploying the K8s Worker to a Kubernetes cluster must be performed by a Kubernetes administrator. Doing such a deployment involves defining or creating the following Kubernetes components:

- Kubernetes Secret
- Kubernetes Service Account
- Kubernetes CronJob

At a high level, this involves the following tasks. (See Prerequisites below.)

- You will create a Kubernetes Secret, to be used as an *image pull secret* to pull the K8s Worker image. See Loading Images into a Kubernetes Cluster for information about how to obtain access to the K8s Worker quay.io repository. It will ultimately be added to the service account under which the K8s Worker will run via the *kustomize* or plain text templates described below.
- You will create a Kubernetes Secret that provides access to the Vantiq Namespace. The value of this secret is a Vantiq Access Token. See above for information defining this token.

Then, using the *kustomize* or plain text templates described below,

- You will create a Kubernetes Service Account under which the K8s Worker will run.
- You will create a Kubernetes Cron Job to act as the K8s worker.

The K8s Worker Cron job will run under the authority granted to the Service Account under which it runs. It will use the Kubernetes Secret to access Vantiq. The K8s Worker running in the Kubernetes Cluster must be able to connect to the Vantiq namespace (and, thus, the Vantiq installation) on whose behalf it is performing work.

Using Templates to Deploy the K8s Worker

We provide some templates for creating the K8s Worker. These are described below.

The templates are available from the Vantiq installation to which you are planning to connect. Please download the zip file containing the templates from here.

When you unzip this downloaded file, you will find two (2) directories: `kustomize` and `plainText`. These contain templates for using Kubernetes kustomize operations and plain text YAML files, respectively. To deploy the K8s Worker, you can use either the kustomize templates or the plain text templates. Their use is described below.

Kustomize Templates

Kubernetes Kustomize templates allow you to change the base set of configuration files through the use of *overlays* files. In our case, these *overlay* files are found in the `overlays/worker` directory

The kustomize templates contain three (3) directories.

- `base` contains the basic definitions that will be altered by the
- `overlays/worker` files.
- `prerequisites` contains the definition of the access token you must define to provide access to the Vantiq namespace.

To make use of these templates, perform the following steps.

Prerequisites

This assumes the following preconditions:

- A secret named `quay-secret` is defined in the Kubernetes namespace or cluster. This is used as an `imagePullSecret` to obtain the image to run the K8s Worker.
- A secret named `vantiq-access-token` is defined in the Kubernetes namespace or cluster. This contains the value of the access token the K8s Worker will use to connect to Vantiq.

Quay Secret (to pull from the Vantiq repository)

To provide the `quay-secret`, define that secret in your Kubernetes installation. This is done by running a command like the following:

```
kubectl create secret docker-registry quay-secret \
    --docker-server=quay.io \
    --docker-username=<your-name> \
    --docker-password=<your-password> \
    --docker-email=<your-email>
```

where

- `<your-name>` is your quay login name
- `<your-password>` is your quay.io password
- `<your-email>` is your email

If this is to be used in a specific namespace, add the appropriate `--namespace` option for `kubectl`.

vantiq-access-token

Edit a copy of the `prerequisites/vantiq-access-token-secret.yaml` file, replacing the “accessToken” string with your access token. The result should look something like the following:

```
# This defines a Secret that contains the access token for connecting to Vantiq
# This is used by the Vantiq K8sworker job
apiVersion: v1
kind: Secret
metadata:
  name: vantiq-access-token
  annotations:
    relatedTo: "vantiq-k8sworker"
type: Opaque
stringData:
  token: thisShouldBeYourAccessToken # Vantiq Token String
```

Once that’s done, apply that file to define the secret:

```
kubectl apply -f <path to the file you copied and edited above>
```

With these two preliminary tasks completed, you are ready to set up your K8s Worker.

Apply Kustomization

The directory `overlays/worker` contains the kustomize overlay files to apply. These should be updated as described below.

This directory contains a set of files whose names start with `provide`. These files describe the kustomizations required to create a K8s Worker that works with the prerequisites outlined above.

Things are set up assuming that you have completed the prerequisites outlined above.

Once the changes described below are complete, you will run a `kustomize` command to create the definitions on your Kubernetes client.

provideAccessToken.yaml

Usually, no changes are necessary unless you decide to change the name of your access token secret from `vantiq-access-token` or the token key from `token`.

provideClusterName.yaml

You will update this document with the name of the K8sCluster in Vantiq that your K8s worker will represent. Change the string `YOUR_CLUSTER_NAME_HERE` to the name of the K8s Cluster (the Vantiq object) on whose behalf this K8s Worker will work.

provideQuaySecret.yaml

This associates the image pull secret outlined in the prerequisites section with the service account under which the K8s Worker will run. If you changed the name of the secret, update it here. Otherwise, no changes are necessary.

provideServiceAccount.yaml

Assuming you didn't change the name of the service account, no work to do.

provideVantiqIP.yaml

This updates the name of the Vantiq URL.

Update the VANTIQ_URL. This should be the value you usually use to access the Vantiq system. It must be valid from within the Kubernetes cluster.

Applying the Updates

To apply the updates, you are going to use kubectl to run kustomize. To see what is going to be performed, run the following command:

```
kubectl kustomize <path to overlay/worker directory>
```

This will produce output that looks something like this:

```
apiVersion: v1
imagePullSecrets:
- name: quay-secret
kind: ServiceAccount
metadata:
  labels:
    app: vantiq-k8sworker
  name: vantiq-service-account
  namespace: default
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  labels:
    app: vantiq-k8sworker
  name: vantiq-service-account
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: edit
subjects:
- kind: ServiceAccount
  name: vantiq-service-account
  namespace: default
---
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  labels:
    app: vantiq-k8sworker
  name: vantiq-k8sworker
  namespace: default
spec:
  concurrencyPolicy: Forbid
  jobTemplate:
    metadata:
      labels:
        app: vantiq-k8sworker
    spec:
      template:
        metadata:
          labels:
            app: vantiq-k8sworker
            vantiq-k8sworker: vantiq-k8sworker
        spec:
          containers:
          - env:
            - name: CLUSTER_NAME
              value: myClusterName
            - name: VANTIQ_URL
              value: https://dev.vantiq.com
            - name: VANTIQ_ACCESS_TOKEN
              valueFrom:
                secretKeyRef:
                  key: token
                  name: vantiq-access-token
            image: quay.io/vantiq/vantiq-k8sworker:1.31.0
            imagePullPolicy: IfNotPresent
            name: vantiq-k8sworker
          restartPolicy: OnFailure
          serviceAccountName: vantiq-service-account
  schedule: '*/1 * * * *'
```

Once you are satisfied with what you see, you can create these K8s entities by running the following command

```
kubectl apply -k <path to overlay/worker directory>
```

resulting in output something like the following:

```
kubectl apply -k servers/k8sworkerserver/templates/kustomize/vantiq-k8sworker/overlays/vantiqDev
serviceaccount/vantiq-service-account created
rolebinding.rbac.authorization.k8s.io/vantiq-service-account created
cronjob.batch/vantiq-k8sworker created
```

At this point, assuming you have succeeded and your Vantiq server is running, you should see results something like the following (after about a minute):

```
 kubectl get pods
NAME                                READY   STATUS      RESTARTS   AGE
vantiq-k8sworker-1611969960-xl5vx   0/1     Completed   0          20s
```

where `vantiq-k8sworker-...` is the part in which you are interested. You may see other things listed as well, depending upon the state of your Kubernetes cluster. Note that if you deployed your K8s Worker to another namespace, you will need to update the command appropriately.

To validate things, check the logs:

```
kubectl logs vantiq-k8sworker-1611969960-xl5vx
2021-01-30T01:34:04.183 [main] INFO  io.vantiq.k8sworkmgr.K8sWorker - Vantiq Kubernetes Worker 1.31.0 Started.  Running in Kubernet
2021-01-30T01:34:04.193 [main] DEBUG io.vantiq.k8sworkmgr.K8sWorker - Server Config: VANTIQ_URL :: https://vantiq-host for cluster (
Manager Version: 1.31.0
Server Version: 1.31.0
Server Build Date: Thu Jan 28 12:22:52 PST 2021
Server Commit: ...
2021-01-30T01:34:06.232 [main] DEBUG i.v.k8sworkmgr.utils.K8sConnection - testConnection: false
2021-01-30T01:34:06.321 [main] DEBUG io.vantiq.k8sworkmgr.K8sWorker - No workItem available
2021-01-30T01:34:06.322 [main] INFO  io.vantiq.k8sworkmgr.K8sWorker - ProcessWorkItems() 1 completed (0 total work items processed)
```

Once you see things like this, things are working. The Cronjob will run the worker once per minute (unless that schedule has been changed).

PlainText Templates

If you prefer to use the plain text templates in lieu of the kustomize templates, please perform the following steps.

Generally, you are going to be looking for the text `<PLACEHOLDER>`, replacing that with the appropriate values for your situation.

Edit the Service Account

In the `vantiq-service-account.yaml` file, replace the `<PLACEHOLDER>` string with the appropriate namespace value.

Provide the Access Token

In the `vantiq-access-token-secret.yaml` file, replace the `<PLACEHOLDER>` string with the the Vantiq access token to be used.

Define the K8s Worker Cron Job

In the `vantiq-k8sworker.yaml` file, make the following changes:

- for `metadata/namespace`, replace `<PLACEHOLDER>` with the appropriate namespace.
- for `spec/jobTemplate/spec/template/spec/hostAliases/ip`, replace `<PLACEHOLDER>` with the IP address of the Vantiq server.
- for `spec/jobTemplate/spec/template/spec/containers/env`, replace the `value` `<PLACEHOLDER>` for the `VANTIQ_URL` with the Vantiq URL for the K8s Worker to use. Generally, this should be the same value you would use outside the cluster. The Vantiq URL must use the same node name you usually use.
- for `spec/jobTemplate/spec/template/spec/containers/env`, replace values for the `VANTIQ_ACCESS_TOKEN` `secretKeyRef` `name` and `key` `<PLACEHOLDER>` values with the name and key, respectively, for the Kubernetes Secret that contains the Vantiq access token. If you left the structure of the `vantiq-access-token-secret.yaml` file unchanged, these will be `vantiq-access-token` and `token`, respectively.
- for `spec/jobTemplate/spec/template/spec/containers/env`, replace the `CLUSTER_NAME` value `<PLACEHOLDER>` with the name of your K8s Cluster.
- for `spec/jobTemplate/spec/template/spec/serviceAccountName`, replace `<PLACEHOLDER>` with the name of the service account under which the K8s Worker should run. This is the name of the service account defined in the Edit the Service Account section.
- for the ‘spec/jobTemplate/spec/template/spec/imagePullSecrets/name’, replace `<PLACEHOLDER>` with the name of the docker registry secret defined in the Prerequisites section.

Apply the Results

Once you are satisfied with your edit, apply the changes using the `kubectl` command. You will need to apply the `vantiq-access-token.yaml` and `vantiq-service-account.yaml` files before applying the `vantiq-k8sworker.yaml` file.

To apply any file, use the command

```
kubectl apply -f <path to plainText directory>/filename
```

Having completed successfully deploying the K8s Worker, you can now move forward and make use of the Kubernetes Cluster.

Using a Kubernetes Cluster

Once a K8s Worker for a Vantiq namespace is deployed, namespace administrators within that namespace can make use of that Kubernetes cluster.

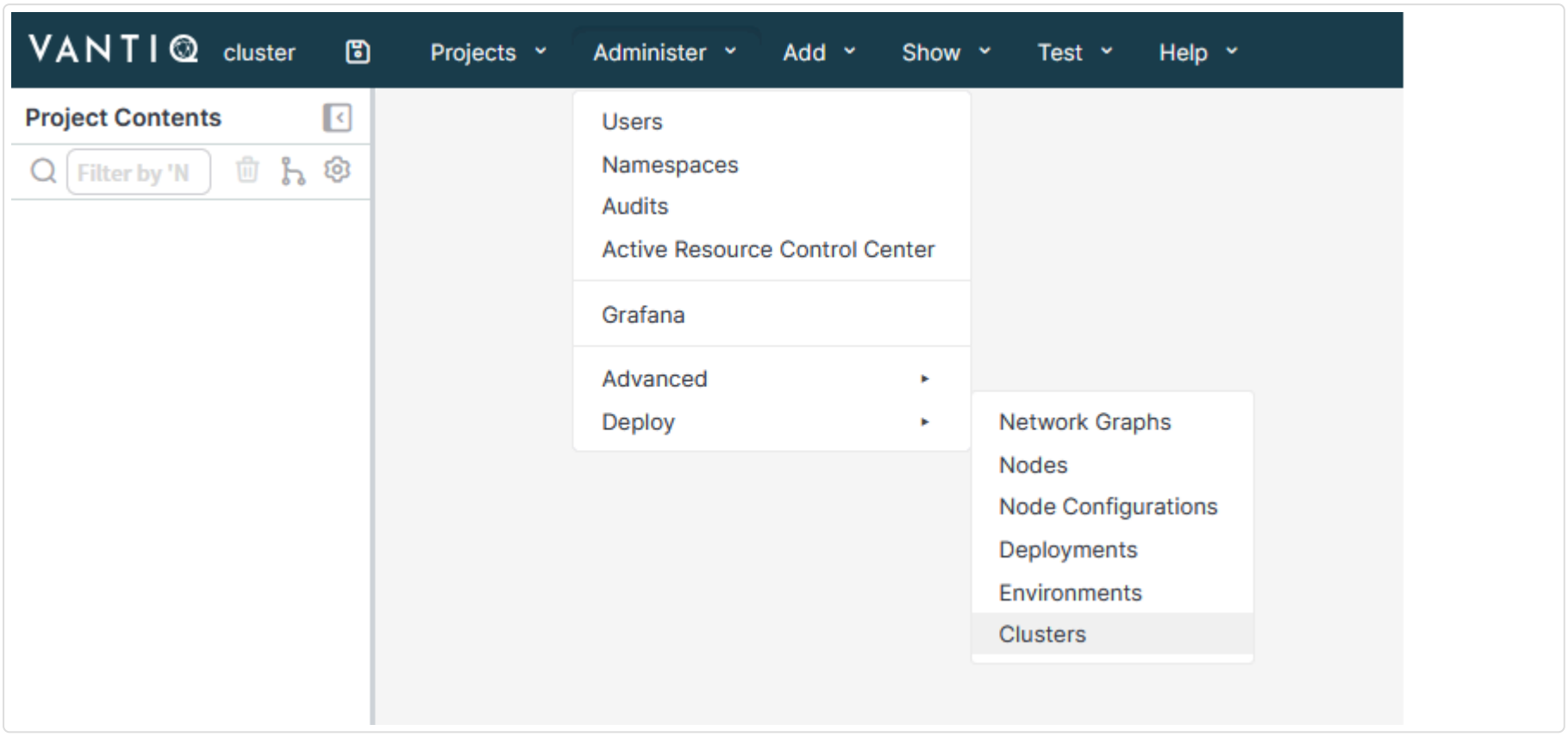
Note: if you (via your organization) have sufficient quota, you may see the `self` cluster in the list of clusters to which you can deploy. The `self` cluster is the cluster in which Vantiq runs. Deployments here are quota controlled, and, because of that, deploying to the `self` cluster must be performed by an admin for the organization to which the Vantiq namespace belongs.

To qualify the quota used, all deployments to the `self` cluster must specify their resource usage. If you are deploying to the `self` K8s Cluster, you must specify the `resourceRequest` or `resourceLimit` to be used for your installation. If you specify both, the maximum value for these resources is charged against your organization's quota before deployment. See deploying a K8s Installation for further details.

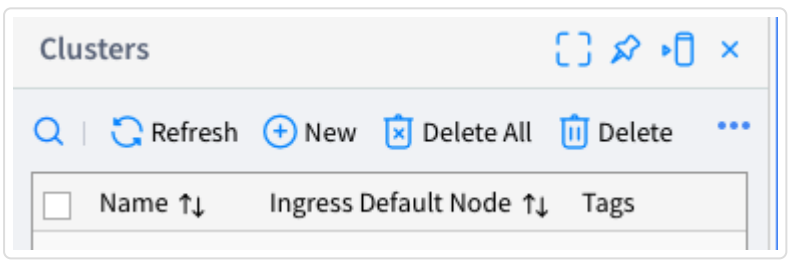
Creating a K8s Cluster

Creating a K8s Cluster via the REST interface is described in the K8s Clusters section of the Resource Reference Guide. Please ensure that the name used for the K8s Cluster matches that defined for the K8s Worker.

To create a K8s Cluster in Modelo, select `Deploy->Clusters`



and select the `New` button.



There, fill in the cluster name and default ingress node name (if any). The default ingress node name is the default name of the node used to access inbound ports for installations deployed on this cluster. Generally, you will want each installation to specify its ingress node name. For details, please see deploying a K8s Installation.

Here, we have created a cluster named `minikube`.

Once the K8s Cluster is defined, operations on that K8s Cluster can be performed. Each such operation results in the creation of a K8s Workitem. The K8s Worker for that K8s cluster will pick up these work items, perform the required work, and complete them, allowing the Vantiq model to be updated. Specifically, the appropriate K8s Installations will be created, shutdown, restarted, or removed.

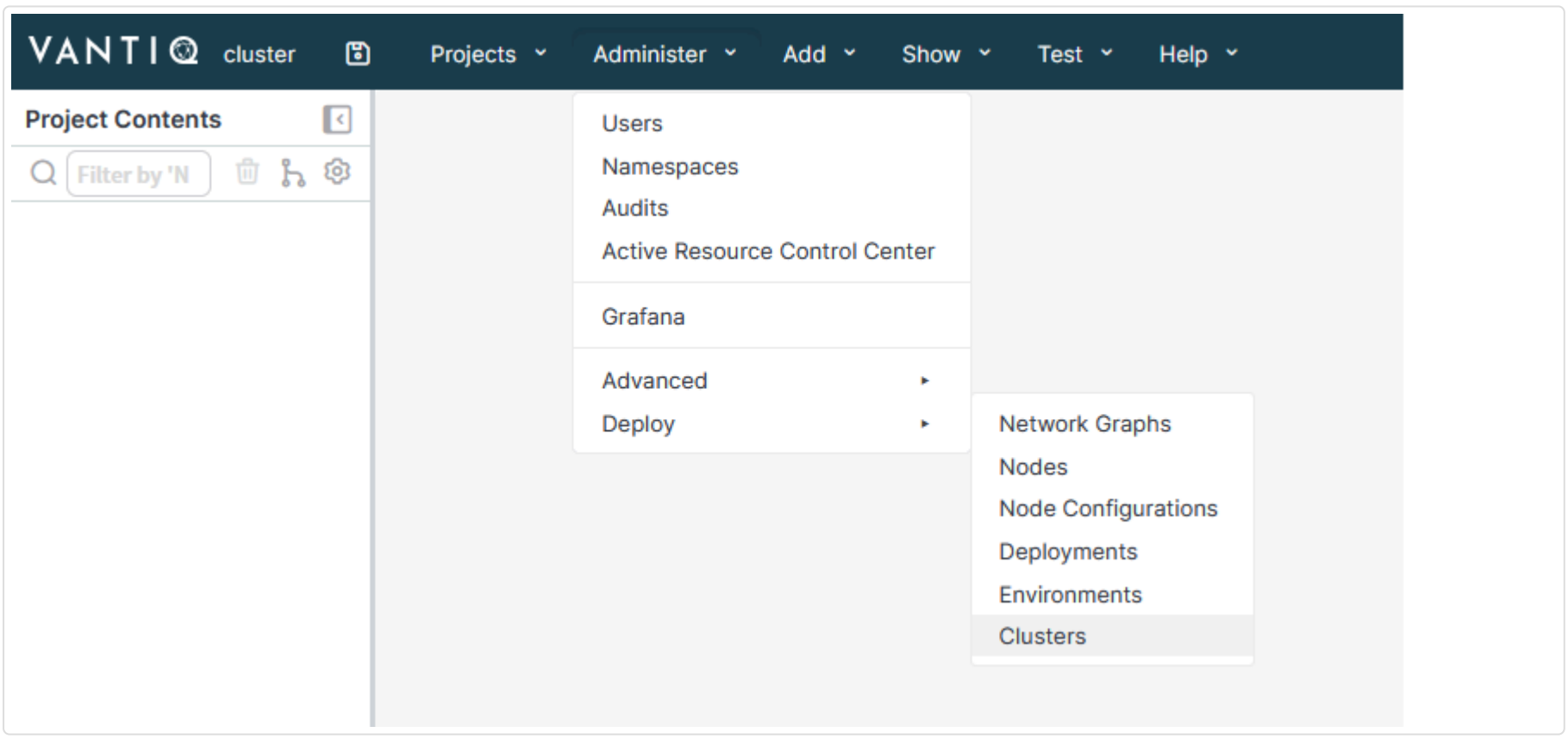
Deploying K8s Installations to a Kubernetes Cluster

To deploy a K8s Installation, you must provide the installation name and the Kubernetes namespace in which it will live. If the associated K8s Worker is namespace specific, the Kubernetes namespace supplied should match that of the K8s Worker.

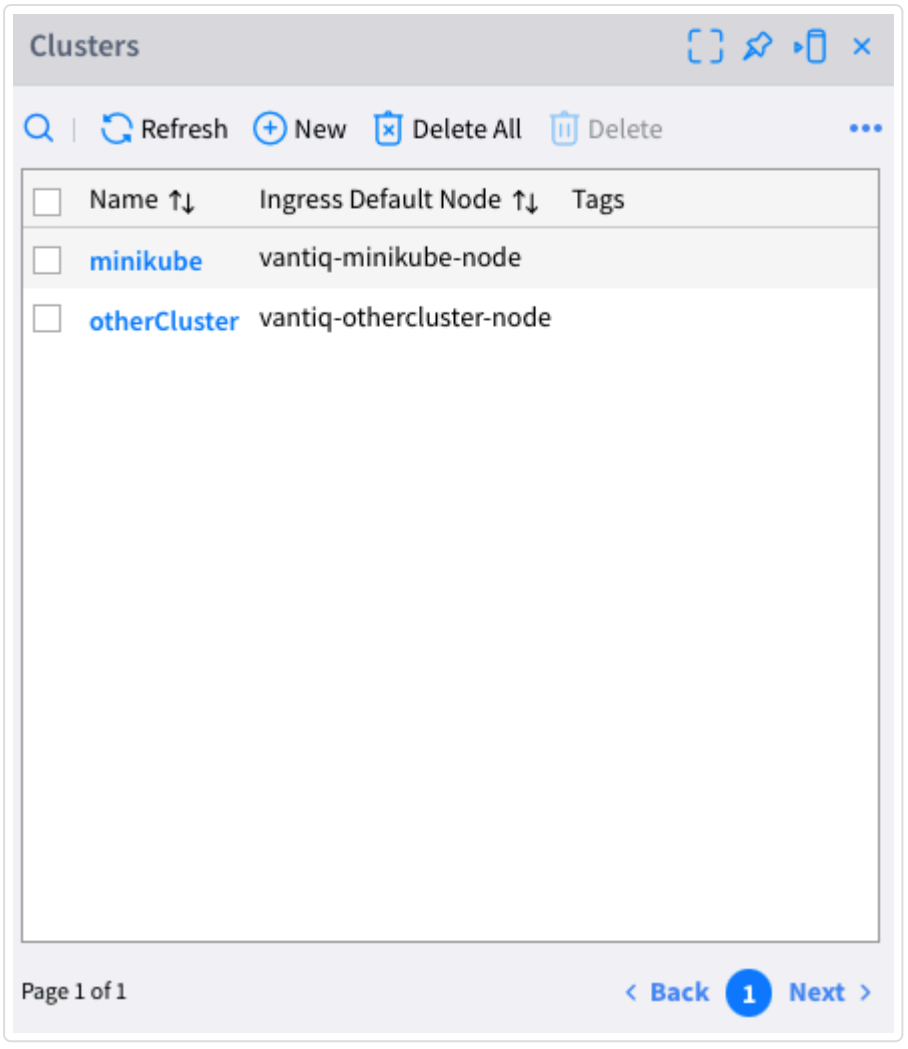
In addition to the name and namespace, you must also provide a configuration for the K8s Installation. The definition of the configuration is provided in more detail in the K8s Installations section of the Resource Reference Guide.

We will illustrate the deployment by deploying a JDBC Connector. For information about the JDBC Enterprise Connector, please see the JDBC Enterprise Connector section of the Extension Sources repository.

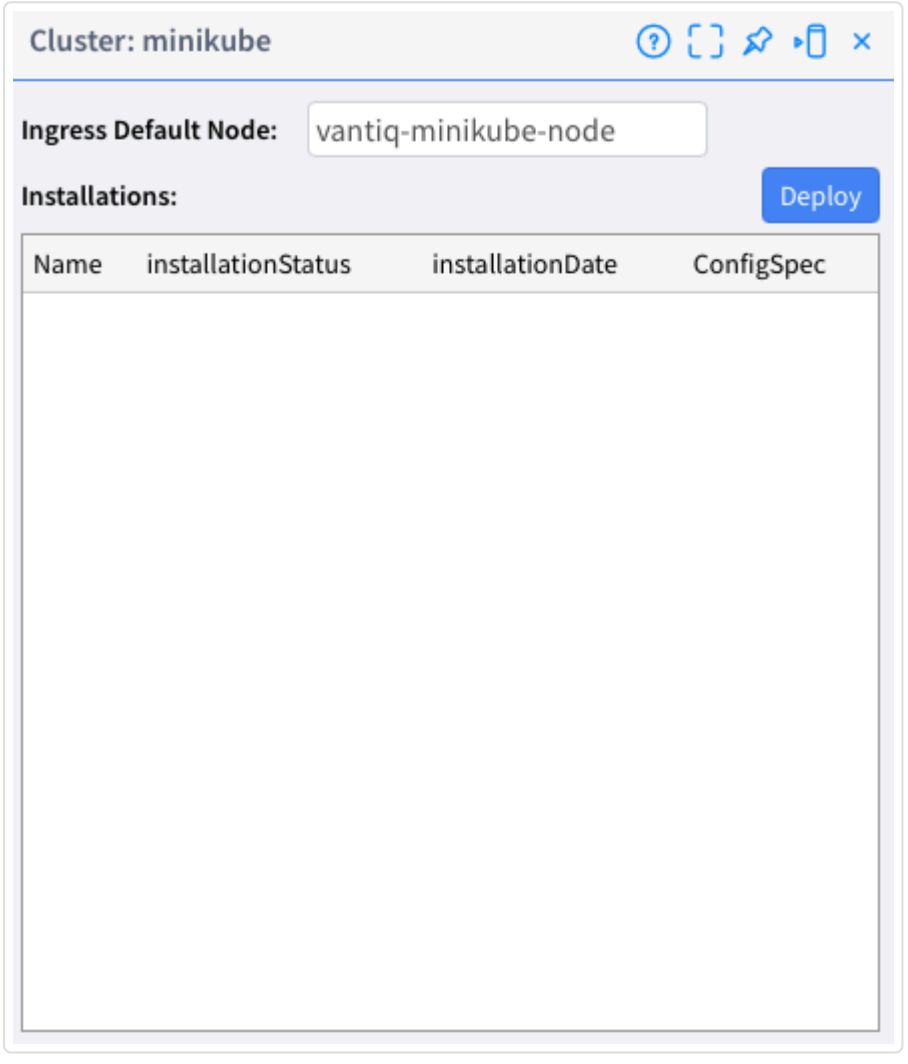
First, we'll look for our cluster list. This is done using the `Deploy->Clusters` control.



Once that is done, you will see the list of K8s Clusters you have. In our case, we see two K8s Clusters.

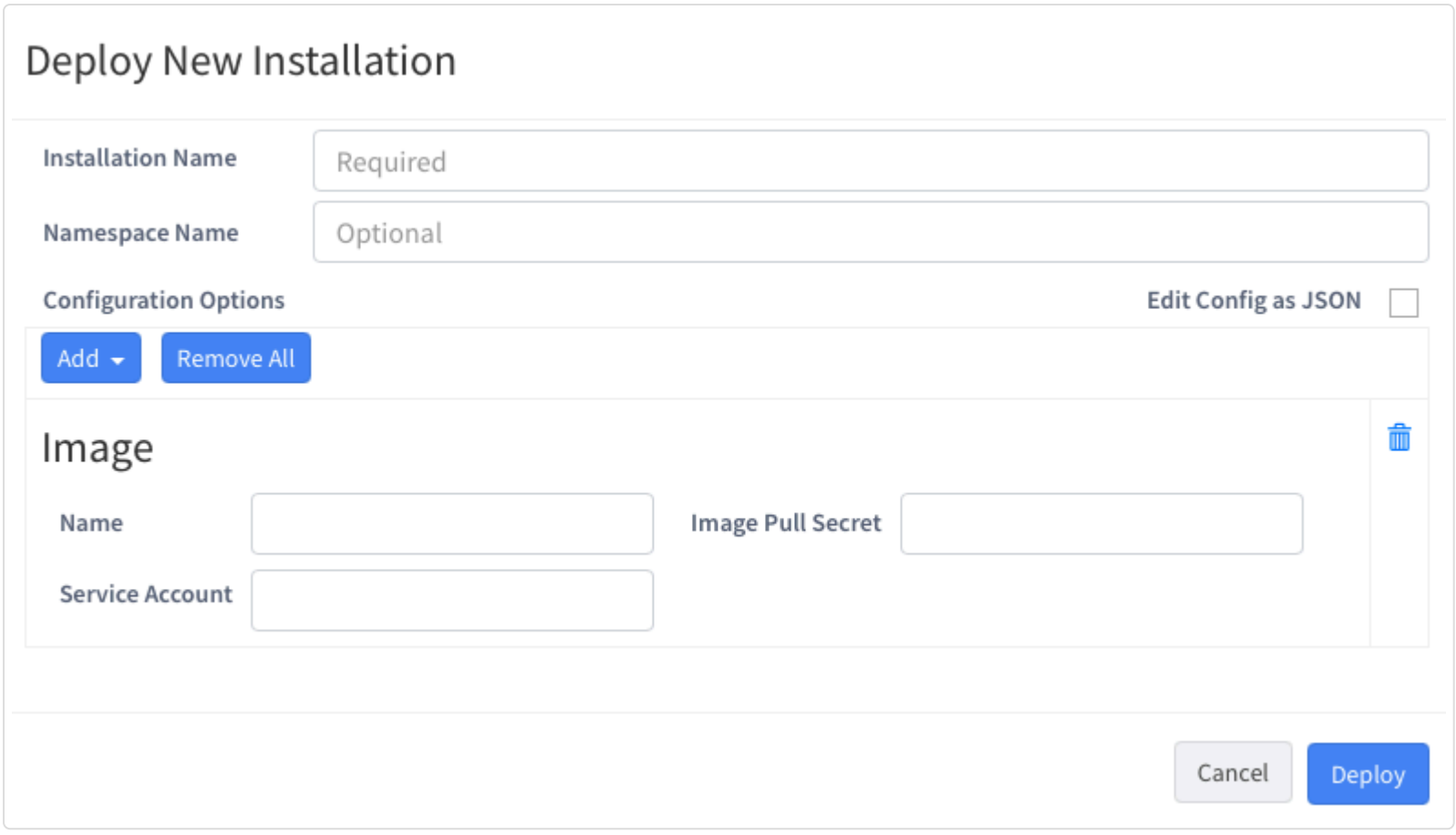


We will select *minikube*.



From here, we see the panel for the selected cluster. To deploy a K8s Installation, select the *Deploy* button.

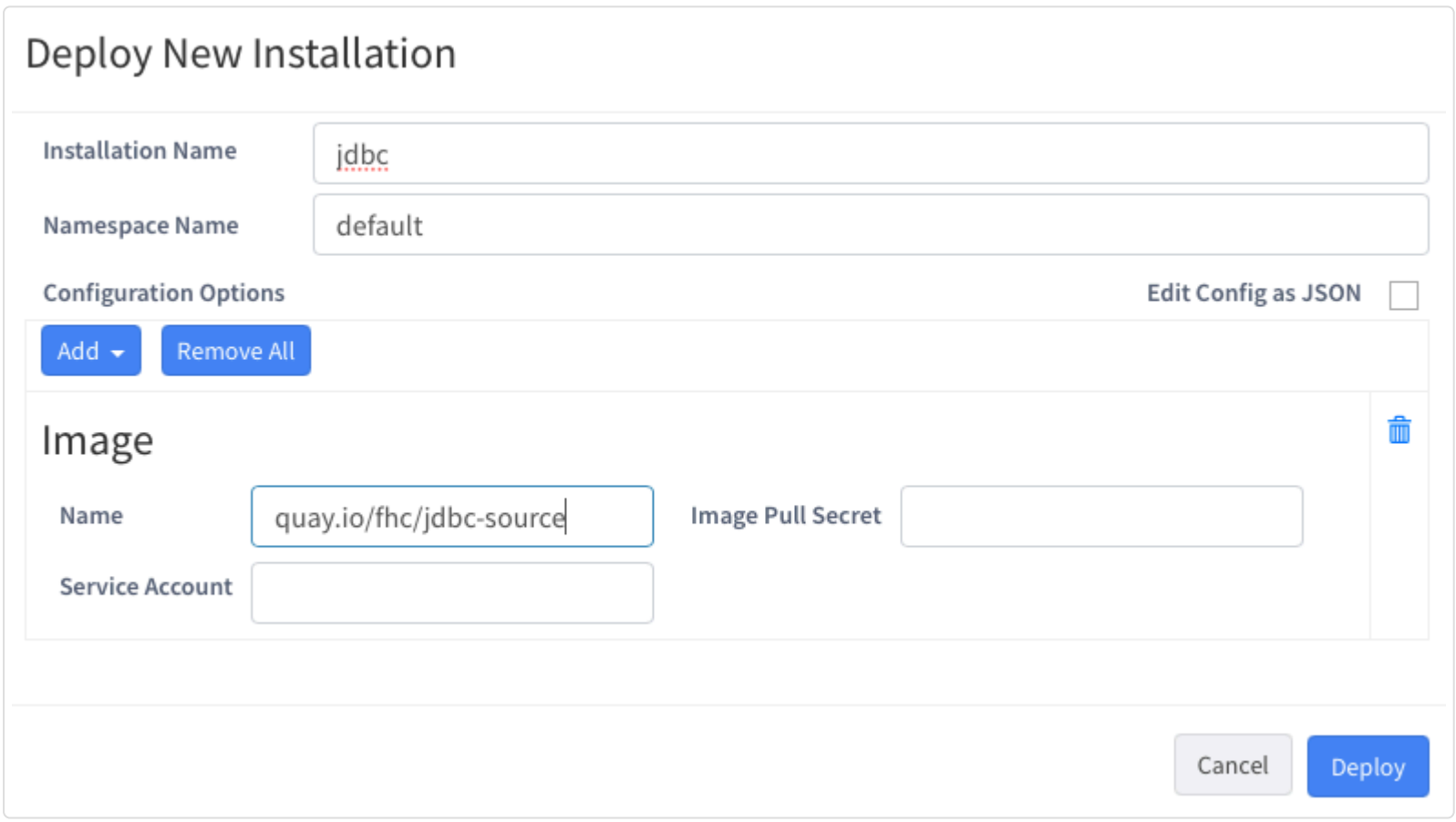
That will bring up the following.



From there, provide the information required for deploying your installation. We will name our K8s Installation `jdbc`, and place it in the `default` namespace. Then, add an image repository name. This is the source for the image that Kubernetes will instantiate and run.

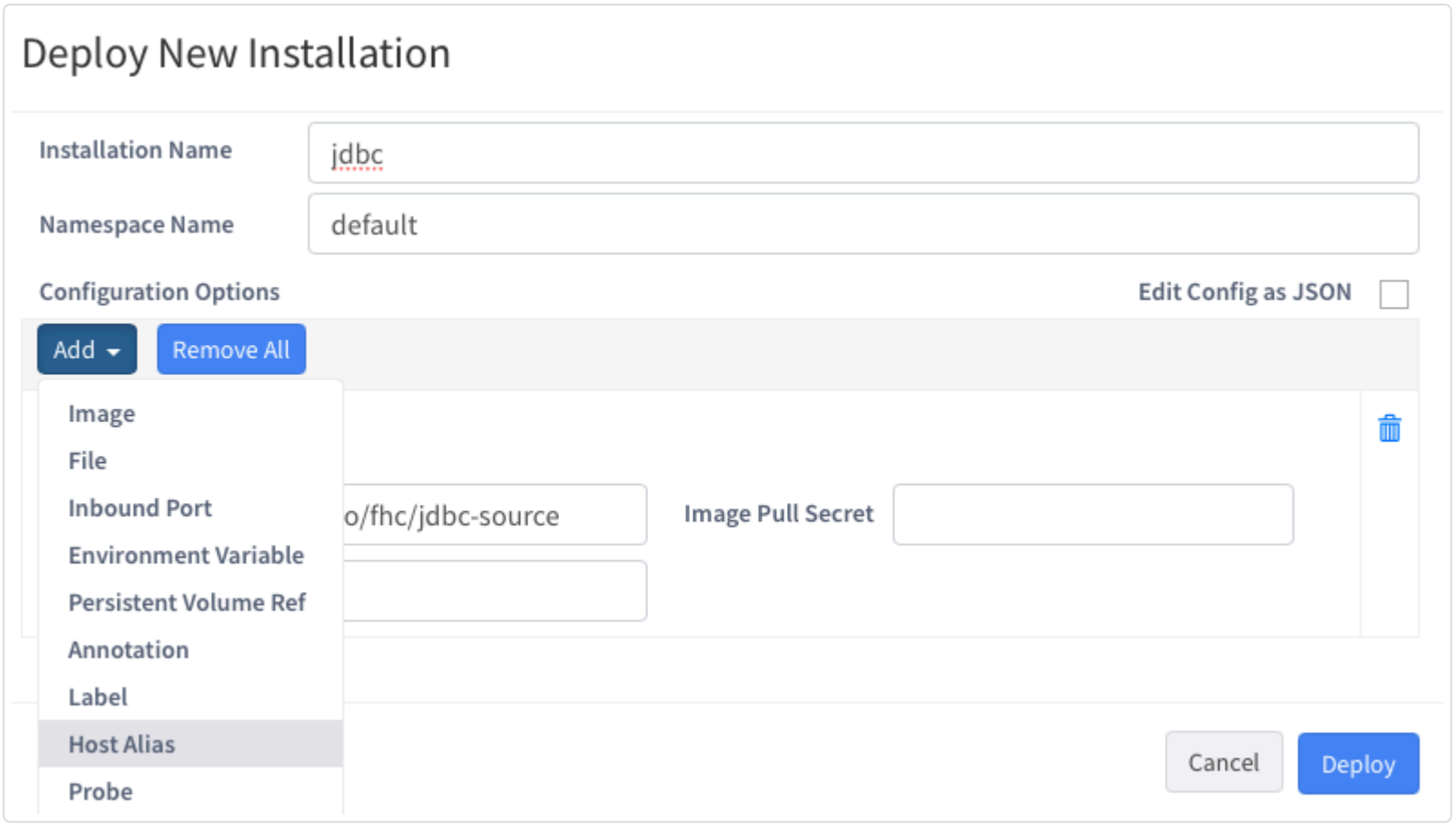
Note that the image (repository) name begins with the name of a container registry (in this case, *quay.io*). It is from this registry that the image will be downloaded. If the particular repository (often called the *repo*) is not public, then you must either provide the appropriate *image pull secret* or have that image pull secret available via the *service account*. The *Image Pull Secret* and *Service Account* can be filled in with the names of the appropriate `docker-registry` secret and Kubernetes service account, respectively.

In the example shown below, no image pull secret is necessary, and the default service account is sufficient.

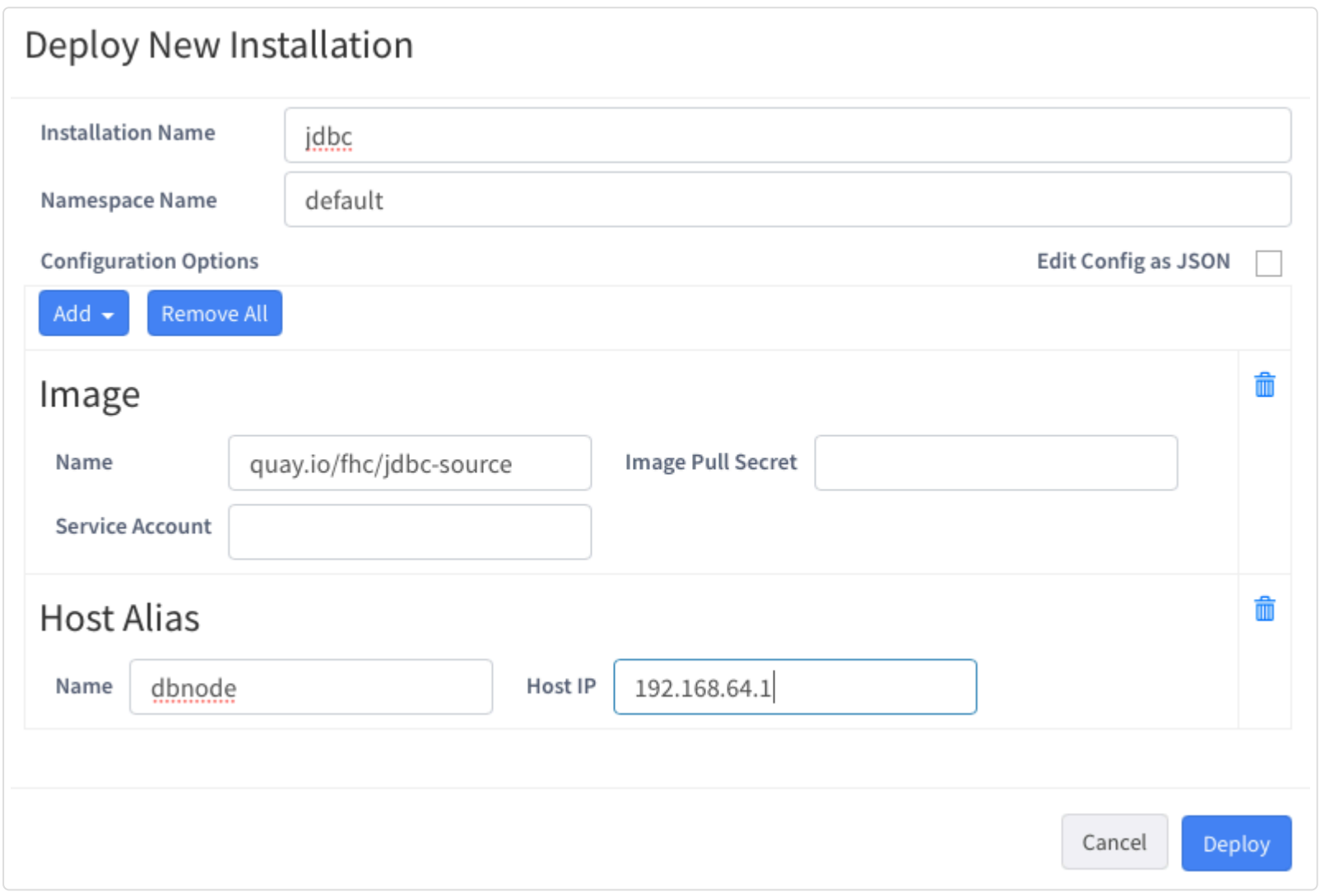


> Note that the image specified is not generally available. If you wish to deploy a connector, please see the Connector/Extension Source repo for general information, and the Building Docker Images section for details about publishing your own images.

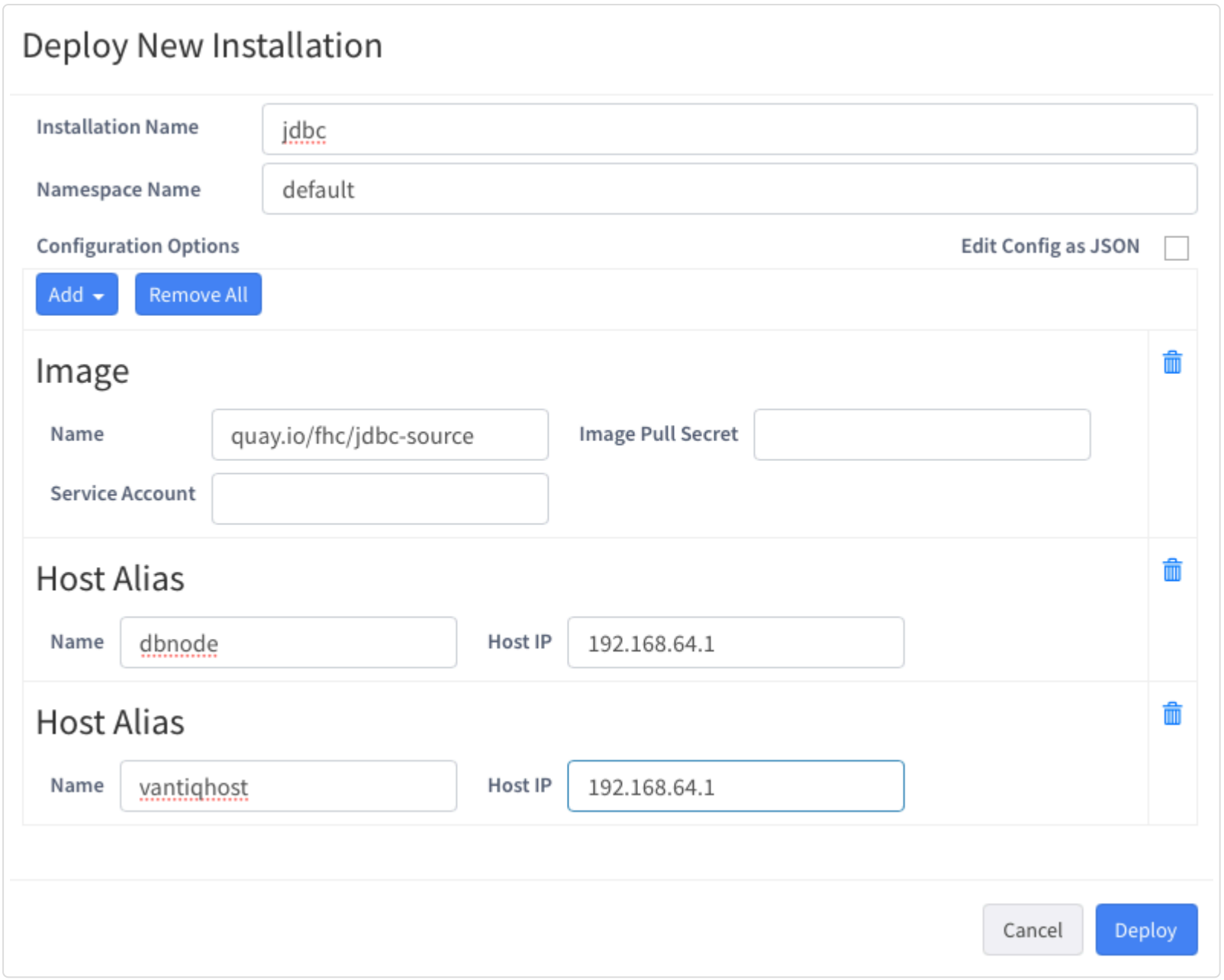
Once this is done, we'll add other configuration options. To do so, press the *Add* button under Configuration Options, and select the corresponding option.



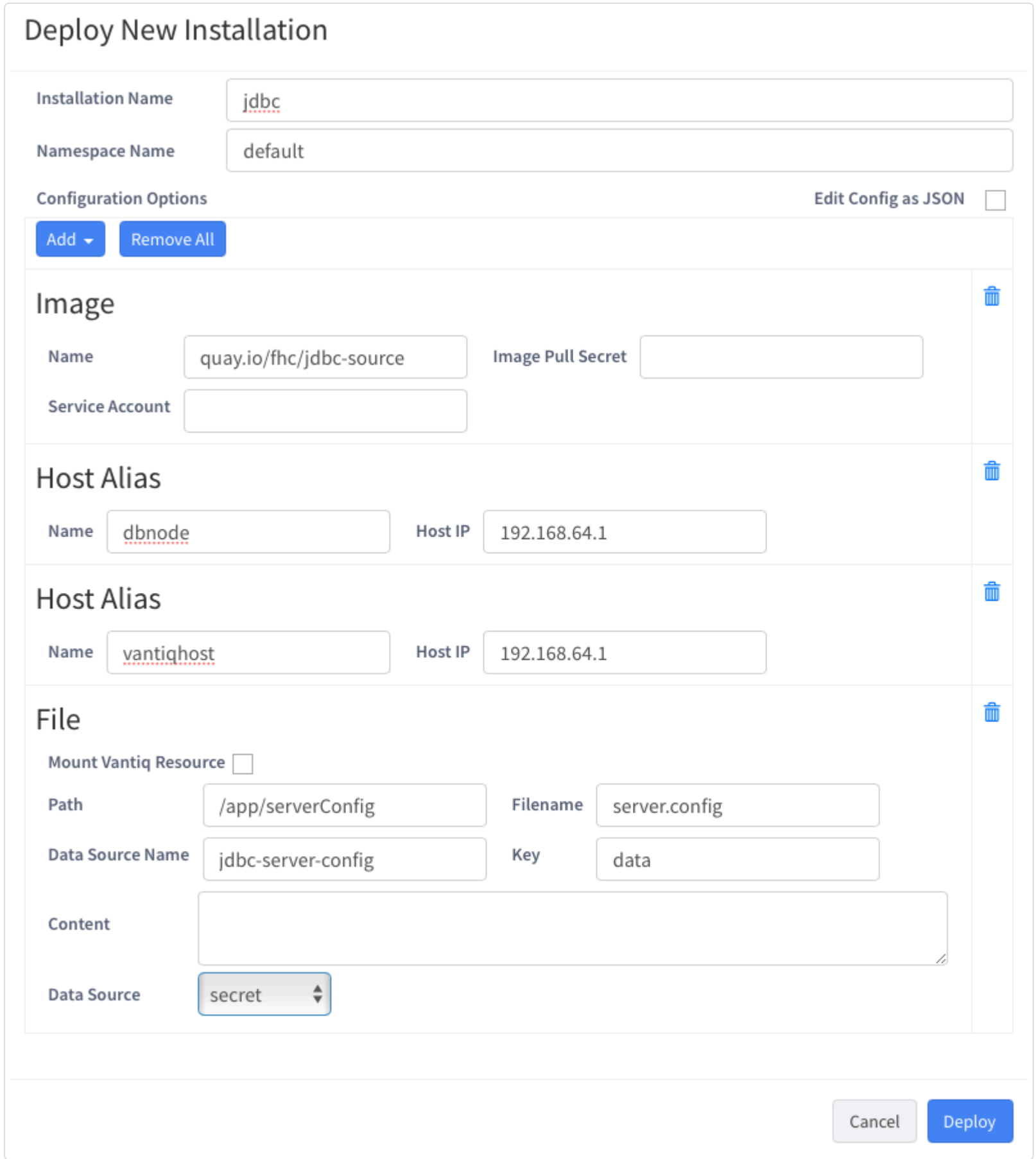
In the case of our example, we will add a host alias for the database node.



We will add a second host alias for the node on which the Vantiq server is running. This host alias will be for `vantiqhost`. In this case, both `dbnode` and `vantiqhost` have the same value, but that will not always be the case.



Finally, we will add the configuration file for our JDBC Enterprise Connector. In this case, we are using an existing Kubernetes secret (`jdbc-server-config`) that contains the server configuration data in a key named `data` . We use a Kubernetes secret here because the information contained in that configuration data contains a Vantiq access token. Note that we had to change the *Data Source* field to **secret**.



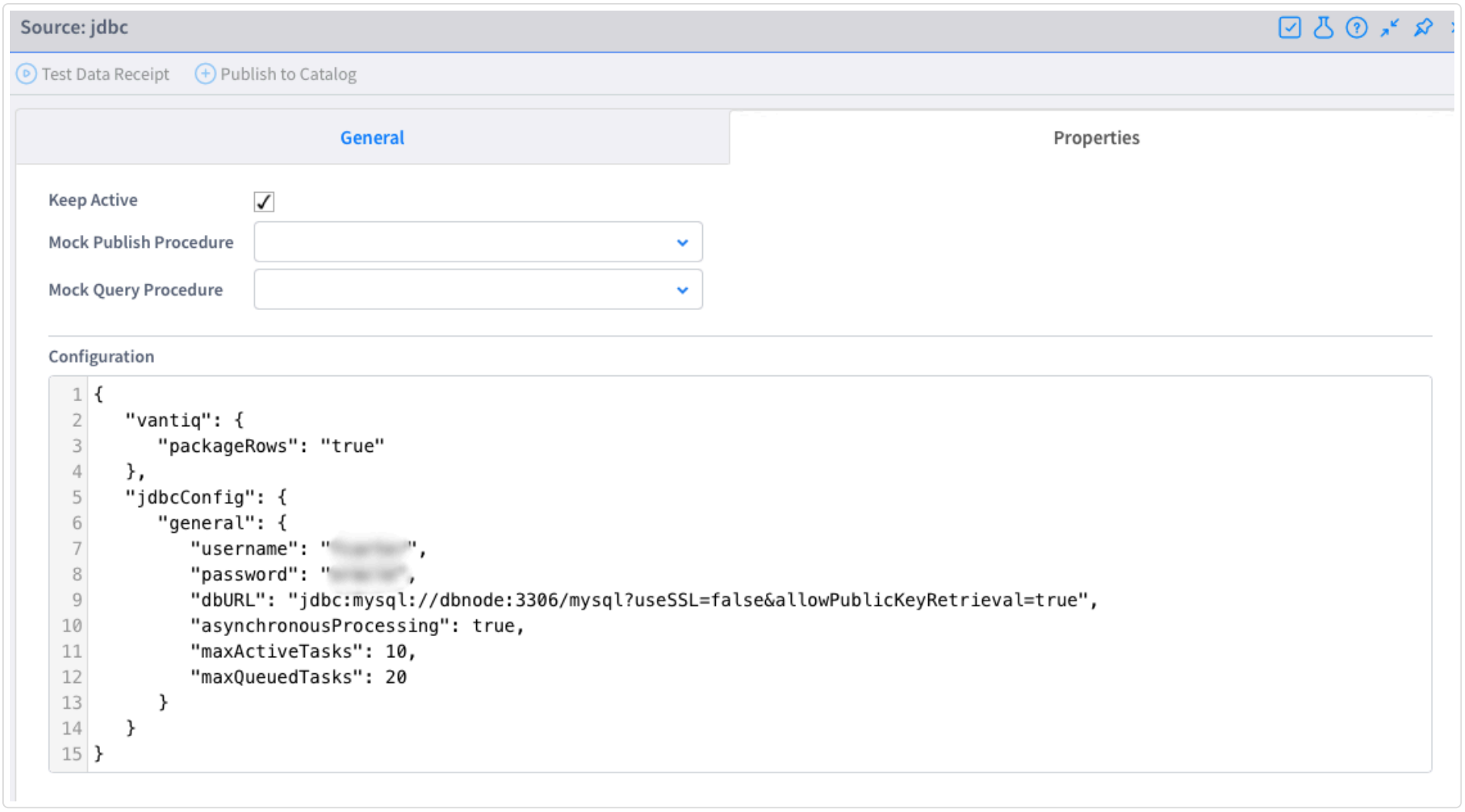
With this, our K8s Installation configuration is complete, and we can press the *Deploy* button. The result will be that the deployment will begin when the K8s Worker picks up the K8s Workitem. (Once this is done, you may see a message about *work items in the queue* on the cluster panel. Such message(s) will remain until the work item is acted upon by the K8s Worker.)

The server configuration information contained in the `jdbc-server-config` Kubernetes secret is as follows.

```
targetServer=http://vantiqhost:8080
sources=jdbc
authToken=tJ_MT50JUj1gX...
```

Note that the `targetServer` property uses `vantiqhost` (the *host alias* we created above) in its URL.

The configuration for the associated source looks like the following.

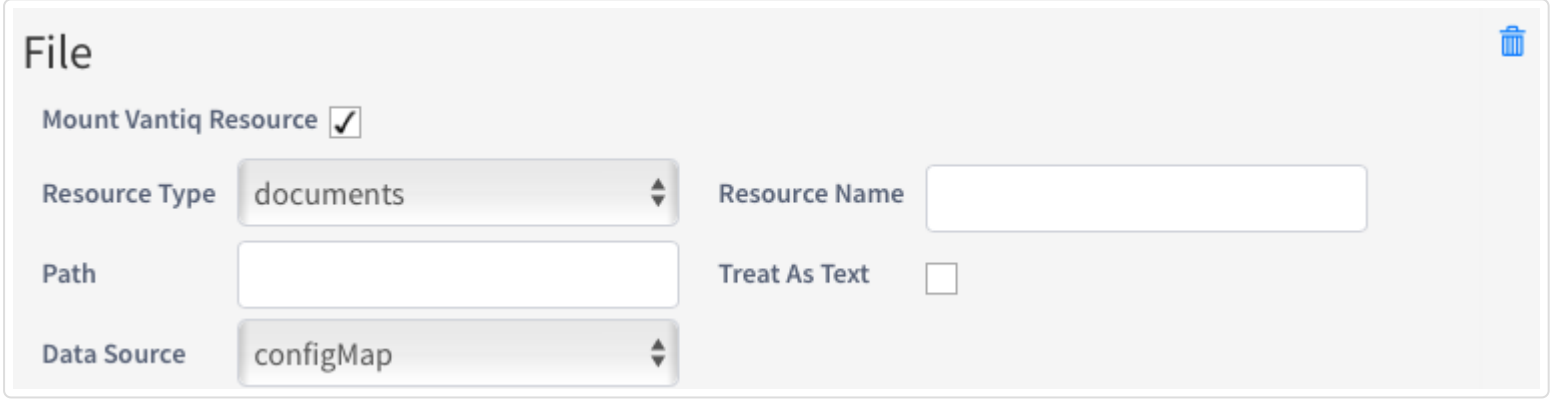


You can see that the `dbURL` property contains a reference to the node `dbnode`. This was provided via the hostAlias configuration option.

Other Configuration Options

This example did not use all of the configuration options. The other options and their properties are described in the K8s Clusters section of the Resource Reference Guide.

Note that the *content* for a file can be obtained from a Vantiq resource (specifically a document, image, or video). This is done by checking the *Mount Vantiq Resource* as shown here.



When the data for the file is to be sourced from a Kubernetes resource, that resource can be either a configMap or a Kubernetes Secret. To use a Kubernetes Secret, change the *Data Source* to **secret** as shown above.

Use of the *self* Cluster

As noted above, with sufficient quota, you may see the *self* cluster listed with your K8sClusters. The `self` cluster is the cluster in which Vantiq runs. Deployments here are quota controlled, and, because of that, any deployments must be performed by the organization admin for the namespace's owning organization.

If you are deploying to the `self` K8s Cluster, you must specify the `resourceRequest` or `resourceLimit` to be used for your installation. If you specify both, the maximum value for these resources is charged against your organization's quota before deployment. See Deploying a K8s Installation for further details.

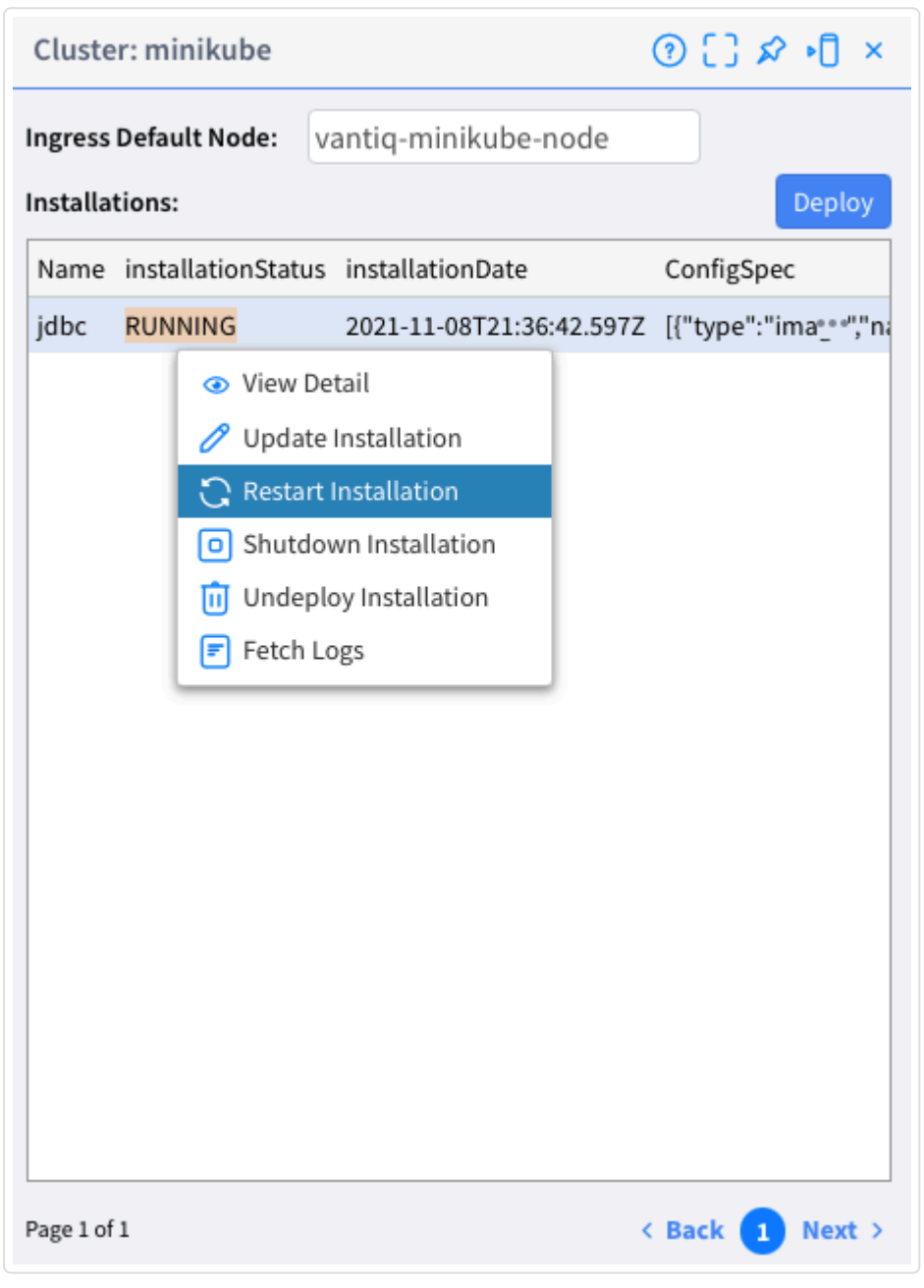
Also, once you deploy to the `self` cluster, you may see changes to the configuration provided. Specifically, you may see items of type `hardAffinity` and/or `label` added. These additions are used by Vantiq in its management of resources within the `self` cluster.

Managing K8s Installations

Once a K8s Installation has been deployed and is running, you can *update* (change the configuration), *shutdown*, *restart*, or *undeploy* it by selecting the appropriate item from the drop-down menu on the installation listed in the cluster panel.

For example, to restart this K8s Installation, find the appropriate cluster

and select the *Restart Installation* menu item from the installation drop-down on the cluster panel.



Other K8s Installation Operations

The Vantiq system performs a couple of checks in the background of which you should be aware.

Verify Installation

Periodically, the Vantiq system will ask the K8s Worker to check the deployed installations. This involves checking to make certain that the components deployed are still there. The Vantiq-Kubernetes linkage is a *federated* one, meaning that a Kubernetes administrator could make alterations to the Kubernetes cluster. If that is done, the installation verification will report that the K8s Installation has a status of `DAMAGED`. This state does not guarantee that the installation will not work, but there is a good chance that it will fail. When this happens, an error will be reported to the Vantiq namespace admin.

If you see an installation with this status, you should contact your Kubernetes administrator to determine what happened.

Delayed Processing

Similarly, the Vantiq system will periodically look for K8s Workitems in the Vantiq system that have not been processed in a timely manner. If K8s Workitems are found that are delayed, an error will be reported to the Vantiq namespace admin. Generally, this indicates that the K8s Worker is not functioning as it should.

Links

- K8s Worker Installation Templates

References

Kubernetes is a registered trademark of The Linux Foundation.