Protegrity REST Container Immutable Policy User Guide for OpenShift 9.1.0.0

Created on: Nov 19, 2024

Copyright

Xen, XenServer, and Xen Source are trademarks or registered trademarks of Citrix Systems, Inc. and/or one or more of its subsidiaries, and may be registered in the United States Patent and Trademark Office and in other countries.

VMware, the VMware “boxes” logo and design, Virtual SMP and VMotion are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions.

Amazon Web Services (AWS) and AWS Marks are the registered trademarks of Amazon.com, Inc. in the United States and other countries.

HP is a registered trademark of the Hewlett-Packard Company.

HPE Ezmeral Data Fabric is the trademark of Hewlett Packard Enterprise in the United States and other countries.

Dell is a registered trademark of Dell Inc.

Novell is a registered trademark of Novell, Inc. in the United States and other countries.

POSIX is a registered trademark of the Institute of Electrical and Electronics Engineers, Inc.

Mozilla and Firefox are registered trademarks of Mozilla foundation.

Chrome and Google Cloud Platform (GCP) are registered trademarks of Google Inc.

Swagger Specification and all public tools under the swagger-api GitHub account are trademarks of Apache Software Foundation and licensed under the Apache 2.0 License.

Table of Contents

Chapter 1

Protegrity Security Operations Cluster Using Kubernetes

This section describes the Protegrity security operations cluster based on Kubernetes.

1.1 Business Problem

This section identifies the problems that a company faces while protecting data:

- Protegrity customers are moving to the cloud. This includes data and workloads in support of transactional application and analytical systems.
- It is impossible to keep up with the continual change in workloads by provisioning Protegrity products manually.
- Native Cloud capabilities can be used to solve this problem and deliver the agility and scalability required to keep up with the customers' business.
- Kubernetes can be configured with Protegrity data security components that can leverage the autoscaling capabilities of Kubernetes to scale.

1.2 Protegrity Solution

The Protegrity solution leverages cloud native capabilities to deliver a Protegrity data security operations cluster with the following characteristics:

- Cloud standard Application Protector REST form factor:
  - The delivery form factor for cloud deployments is a container. Protegrity has developed an AP REST Container form factor based on the Application Protector form factor that we have been delivering for several years.
  - The AP REST Container is a standard Docker Container that is familiar and expected in cloud deployments.
  - The AP REST Container form factor makes the container a lightweight deployment of AP REST.
- Immutable Deployment:
  - Cloud deployments or containers do not change dynamically – they are immutable. In other words, when there is a change in Policy, the change is carried out by terminating the existing AP REST Container with Policy and instantiating a new one.
  - Changes to Policy or the AP REST Container itself happen through special deployment strategies available through Kubernetes. For Protegrity deployments, the recommended deployment strategy is a *Blue - Green* strategy.
- Auto Scalability:
  - Kubernetes can be set up to auto scale based on setting a configuration on CPU thresholds.

  - Kubernetes will start with an initial set of Protegrity Pods. These will increase when the CPU threshold is passed, and they will be shrinked when the CPU threshold is under the acceptable limits. This is automatic and hence gives the agility and scalability that is so desired with cloud solutions. Similarly, the nodes on which the pods are run also auto scale when the node thresholds are reached.
- 3rd Party Integration
- ESA Connectivity Not Required: The important implication is that the ESA is no longer required to be up and running when the Kubernetes runtime has all the required components and can auto scale when needed.

1.3 Architecture and Components

This section will describe the architecture, the individual components, and the workflow of the Protegrity Immutable Application Protector REST solution.

The IAP-REST deployment consists of the following two stages:

- Stage 1: Deploying the Get ESA Policy Container on a Kubernetes cluster to fetch the ESA policy and create the policy snapshot.
- Stage 2: Deploying the AP-REST Container on a Kubernetes cluster, which uses the policy snapshot created in stage 1.

The following describes the architecture diagram and the steps for Stage 1 - Deploying the Get ESA Policy container on a Kubernetes cluster.

The following figure represents the architecture for deploying the Get ESA Policy container on a Kubernetes cluster.

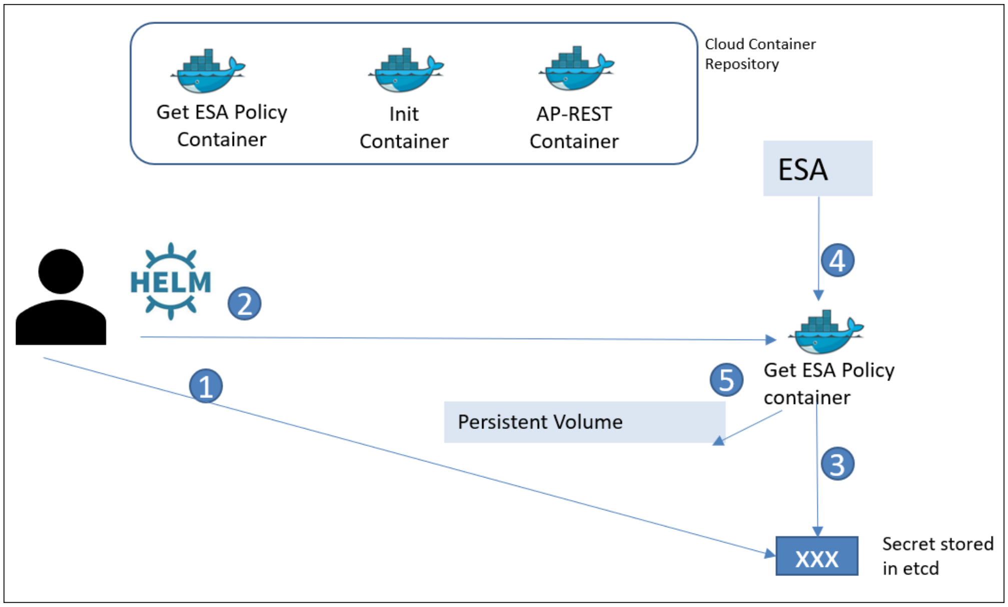


Figure 1-1: Stage 1: Deploying the Get ESA Policy Container

The following steps describe the workflow of deploying the Get ESA Policy container on a Kubernetes cluster.

1. The user stores the ESA credentials in Kubernetes secret. The user also stores the passphrase and the salt for encrypting the policy in a Kubernetes secret.
2. The user runs the Helm chart to deploy the Get ESA Policy container as a Kubernetes job.
3. The Get ESA Policy container retrieves the policy from the ESA and creates a snapshot of the immutable policy and encrypts it.
4. The Get ESA Policy container then uploads the policy snapshot to a persistent volume.
5. The Kubernetes job is completed.

The following describes the architecture diagram and the steps for Stage 2 - Deploying the AP-REST Container on a Kubernetes cluster.

Stage 2: Deploying the AP-REST Container

The following figure represents Pattern 1, which specifies the communication between the AP-REST Container and the customer container pod, which are in different pods but on the same Kubernetes cluster.

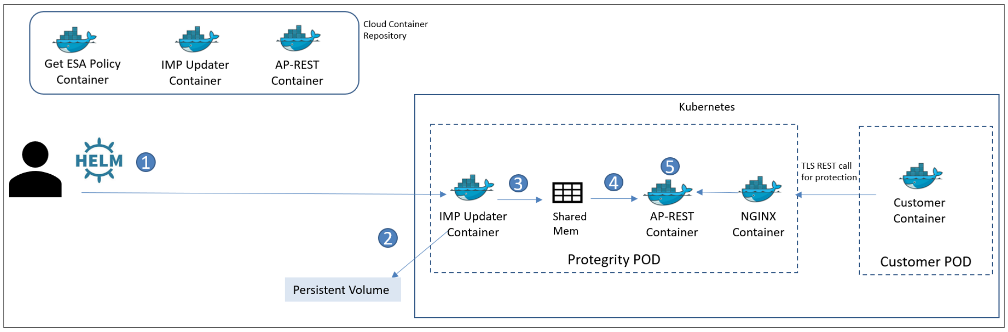


Figure 1-2: Pattern 1 - Communication between Pods in the same Kubernetes Cluster

The following figure represents Pattern 2, which specifies the communication between the AP-REST container and the customer container, which are in different pods and separate Kubernetes clusters.

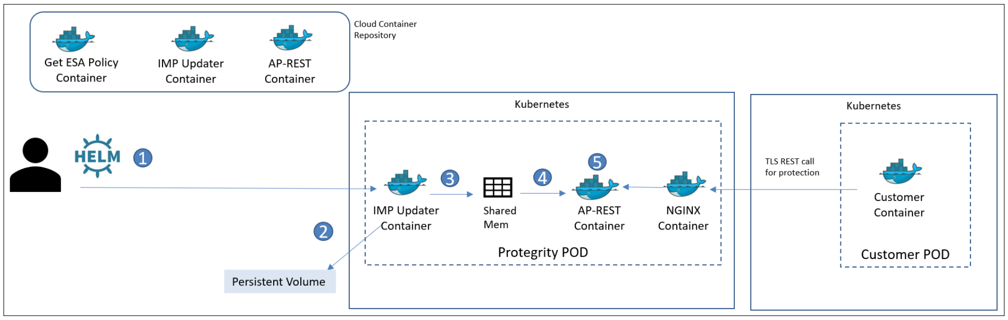


Figure 1-3: Pattern 2 - Communication between Pods in separate Kubernetes Clusters

The following figure represents Pattern 3, which specifies the communication between the AP-REST container and an external client application that is not deployed in a Kubernetes environment.

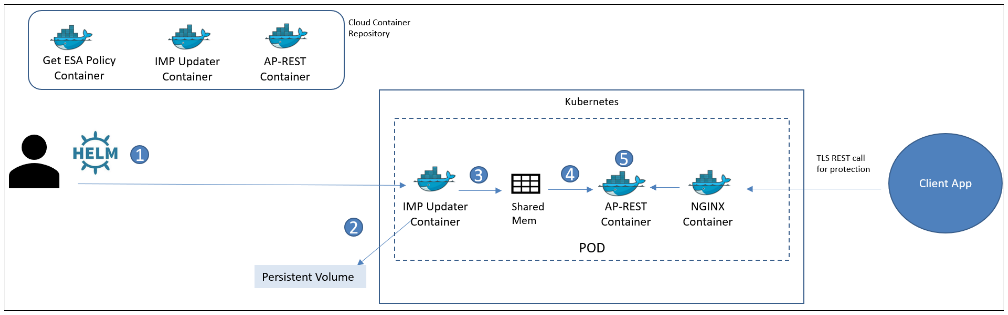


Figure 1-4: Pattern 3 - Communication with an External Application

The following steps describe the workflow of deploying the AP-REST container on a Kubernetes cluster.

1. The user runs the Helm chart to deploy the AP-REST and Init containers to the Kubernetes cluster.
2. The Init Container is initialized first. It reads the immutable policy snapshot from the persistent volume, decrypts it, and uploads the policy in the shared memory of the Pod and then exits.

3. The AP-REST container is then initialized. It uses the policy on the shared memory.
4. The Client Application sends a request to the AP-REST container over REST for protecting, unprotecting, and reprotecting data. The NGINX container is used to terminate the TLS connection from the client application. The client application and the NGINX container communicate with each other over a secure TLS channel, while the NGINX container and the AP-REST container communicate with each other over a non-TLS channel. You can have the following three patterns of communication between the customer application and the AP-REST container:
    - **Pattern 1**: The customer container and the AP-REST container are in different pods but on the same Kubernetes cluster. The AP-REST and the Customer pods are deployed in separate namespaces on the same Kubernetes cluster.
    - **Pattern 2**: The customer container and the AP-REST container are in different pods in separate Kubernetes clusters.
    - **Pattern 3**: The AP-REST container receives the request for protecting, unprotecting, and reprotecting data from an external customer application, which is not deployed in a Kubernetes environment.
5. The AP-REST container returns the response value for the security operation to the client application through the NGINX container.

Chapter 2

AP-REST Deployment Model

This section describes the AP-REST Reference Deployment model that customers can use to deploy Get ESA Policy and the AP-REST containers on a Kubernetes cluster.

2.1 Verifying the Prerequisites

Ensure that the following prerequisites are met:

Reference Solution Deployment

- *IAP-REST_RHUBI-ALL-64_x86-64_OPENSHIFT.K8S_<REST_Container_version>.tgz* – Installation package for the REST Container Deployment. This package contains the following packages:
  - *REST-GET-ESA-POLICY-HELM_ALL-ALL-ALL_x86-64_OPENSHIFT.K8S_<component_version>.tgz* - Package containing the Helm charts, which are used to deploy the Get ESA Policy application on the Kubernetes cluster. Helm is a package manager for Kubernetes.
  - *REST-GET-ESA-POLICY_RHUBI-ALL-64_x86-64_OPENSHIFT.K8S_<component_version>.tar.gz* - Get ESA Policy container image. This image is used to create the Get ESA Policy container, which is used to generate an immutable snapshot of the policy and upload this snapshot to the cloud storage in the Cloud Runtime Environment.
  - *REST-INIT-CONTAINER_RHUBI-ALL-64_x86-64_OPENSHIFT.K8S_<component_version>.tar.gz* - Init container image. This image is used to create the Init container, which is used to retrieve the policy snapshot stored in the persistent volume and make the snapshot available to the Application Protector REST container.
  - *REST-HELM_ALL-ALL-ALL_x86-64_OPENSHIFT.K8S_<component_version>.tgz* – Package containing the Helm charts, which are used to deploy the Application Protector REST on the Kubernetes cluster.
  - *REST_RHEL-ALL-64_x86-64_K8S_<component_version>.tar.gz* - Application Protector REST container image. This image is used to create the Application Protector REST container, which is used to perform security operations.
  - *REST-Samples_Linux-ALL-ALL_x86-64_<component_version>.tgz* - Package containing the sample application for testing the AP-REST containers with sample data.
  - *REST-GET-ESA-POLICY_OPENSHIFT_SRC_<component_version>.tgz* - Package containing the Dockerfile that can be used to create a custom image for the Get ESA Policy container and the associated binary files.

  For more information about building a custom image using the Dockerfile, refer to the section *Creating Custom Images Using Dockerfile*.

  - *REST-INIT-CONTAINER_OPENSHIFT_SRC_<component_version>.tgz* - Package containing the Dockerfile that can be used to create a custom image for the Init container and the associated binary files.

    For more information about building a custom image using the Dockerfile, refer to the section *Creating Custom Images Using Dockerfile*.

  - *REST-SRC_OPENSHIFT_<component_version>.tgz* - Package containing the Dockerfile that can be used to create a custom image for the AP-REST container and the associated binary files.

    For more information about building a custom image using the Dockerfile, refer to the section *Creating Custom Images Using Dockerfile*.

  - *HOW-TO-BUILD-DOCKER-IMAGES* - Text file specifying how to build the Docker images.
  - *manifest.json* - File specifying the name of the product and its components, the version number of the protector, and the build number of product.

Reference Application

The following prerequisites for using the reference application need to be met:

- Policy – Ensure that you have defined the security policy in the ESA. In addition, ensure that you fulfill either of the following requirements:
  - You must attach the policy to the default data store in the ESA.
  - You must add the Kubernetes node or pod IP address ranges, where you want to deploy the containers, as allowed servers to the data store to which the policy is attached.

  For more information about defining a security policy, refer to the section *Creating and Deploying Policies* in the *Policy Management Guide 9.1.0.0*.

  For more information about adding allowed servers to a data store, refer to the section *Adding Allowed Servers to the Data Store* in the *Policy Management Guide 9.1.0.0*.

- User application – For example, Pharmacy application, which contains the patient data that you want to protect using the Application Protector REST.

Additional Prerequisites

The following additional prerequisites need to be met:

- *Linux instance* - This instance can be used to communicate with the Kubernetes cluster. Ensure that Helm is installed on this Linux instance. You can also choose to install Docker on this Linux instance to communicate with the Container Registry, where you want to upload the Docker images.
- Access to an OpenShift account.
- Network File System (NFS) share.
- OpenShift cluster.

  > **Important:** Ensure that you enable the auto scaling of nodes in the cluster.
  >
  > For more information about auto scaling of nodes in an OpenShift cluster, refer to the *OpenShift documentation*.

- Permission to create a persistent volume and persistent volume claim in the cluster.

  For more information about persistent volumes and persistent volume claims, refer to the section *Persistent Volumes* in the Kubernetes documentation.

- Access to a Container registry to upload the Get ESA Policy, Init, and AP-REST container images.

2.2 Downloading the Installation Package

This section describes the steps to download the installation package for the IAP Reference Deployment model.

To download the installation package:

1. Download the *IAP-REST_RHUBI-ALL-64_x86-64_OPENSHIFT.K8S_<REST_Container_version>.tgz* file on the Linux instance.
2. Run the following command to extract the files from the *IAP-REST_RHUBI-ALL-64_x86-64_OPENSHIFT.K8S_<REST_Container_version>.tgz* file.

   ```
   tar -xvf IAP-REST_RHUBI-ALL-64_x86-64_OPENSHIFT.K8S_<REST_Container_version>.tgz
   ```

   The following files are extracted:

   - *REST-GET-ESA-POLICY-HELM_ALL-ALL-ALL_x86-64_OPENSHIFT.K8S_<component_version>.tgz*
   - *REST-GET-ESA-POLICY_RHUBI-ALL-64_x86-64_OPENSHIFT.K8S_<component_version>.tar.gz*
   - *REST-INIT-CONTAINER_RHUBI-ALL-64_x86-64_OPENSHIFT.K8S_<component_version>.tar.gz*
   - *REST-HELM_ALL-ALL-ALL_x86-64_OPENSHIFT.K8S_<component_version>.tgz*
   - *REST_RHEL-ALL-64_x86-64_K8S_<component_version>.tar.gz*
   - *REST-Samples_Linux-ALL-ALL_x86-64_<component_version>.tgz*
   - *REST-GET-ESA-POLICY_OPENSHIFT_SRC_<component_version>.tgz*
   - *REST-INIT-CONTAINER_OPENSHIFT_SRC_<component_version>.tgz*
   - *REST-SRC_OPENSHIFT_<component_version>.tgz*
   - *HOW-TO-BUILD-DOCKER-IMAGES*
   - *manifest.json*

2.3 Initializing the Linux Instance

This section describes how you can initialize the Linux instance.

To initialize the Linux instance:

1. Install OpenShift CLI, which provides a set of command line tools for the OpenShift Cloud Platform.

   For more information about installing OpenShift CLI on the Linux instance, refer to the following website based on the specific version of the OpenShift Container Platform that you want to use:

   - *Getting started with OpenShift CLI for the OpenShift Container Platform version 4.12*
2. Configure the OpenShift CLI on your machine by running the following command.

   ```
   oc login
   ```
3. Run the following commands sequentially to install the Helm client on the Linux instance.

   ```
   curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/
   scripts/get-helm-3
   ```

```
chmod 700 get_helm.sh
```

```
./get_helm.sh
```

For more information about installing a Helm client, refer to *https://helm.sh/docs/using_helm/#installing-helm*.

> **Important:** Verify the version of the Helm client that must be used from the Readme document provided with this build.

You can verify the version of the Helm client installed by running the following command:

```
helm version --client
```

4. Run the following command to extract the *.tgz* package containing the Helm charts for deploying the Get ESA Policy container.

   ```
   tar -xvf <Helm chart package>
   ```

   For example:

   ```
   tar -xvf REST-GET-ESA-POLICY-HELM_ALL-ALL-
   ALL_x86-64_OPENSHIFT.K8S_<component_version>.tgz
   ```

   The extracted package contains the *get-esa-policy* directory, which has the following contents:

   - *Chart.yaml* – A YAML file that provides information regarding the chart
   - *templates* – A directory that contains the templates required to generate the Kubernetes manifest files. Kubernetes uses the manifest files to deploy the application on the Kubernetes cluster.
   - *values.yaml* – The default values for configuring the Helm chart
   - *pty-scc.yaml* – The Protegrity Security Context Constraints (SCC) file that is used to manage the security policies for a pod.
   - *pty-rbac.yaml* – The values for configuring the Protegrity Role Based Access Control (RBAC) for the pods.
   - *nfs-pv.yaml* – The default values for configuring the persistent volume.
   - *nfs-pvc.yaml* – The default values for configuring the persistent volume claim.

5. Run the following command to extract the *.tgz* package containing the Helm charts for deploying the AP-REST container.

   ```
   tar -xvf <Helm chart package>
   ```

   For example:

   ```
   tar -xvf REST-HELM_ALL-ALL-ALL_x86-64_OPENSHIFT.K8S_<component_version>.tgz
   ```

   The extracted package contains the *iap-rest* directory, which has the following contents:

   - *Chart.yaml* – A YAML file that provides information regarding the chart
   - *templates* – A directory that contains the templates required to generate the Kubernetes manifest files. Kubernetes uses the manifest files to deploy the application on the Kubernetes cluster.
   - *values.yaml* – The default values for configuring the Helm chart

6. Run the following command to connect to an OpenShift cluster.

   ```
   oc login <CLUSTER_HOSTS>:<PORT>
   ```

   > **Note:**
   >
   > Before running the command, ensure that you have created an OpenShift cluster.

2.4 Creating Certificates and Keys for TLS Authentication

This section describes the typical steps required to create certificates and keys for establishing secure communication between the client and the server.

> **Note:**
>
> If you already have a server that has been signed by a trusted third-party Certificate Authority (CA), then you do not need create a self-signed server certificate.

Before you begin
Ensure that OpenSSL is installed on the Linux instance to create the required certificates.

To create certificates and keys for TLS authentication:

1. On the Linux instance, run the following command to create a CA certificate and a private key for the certificate.

   ```
   openssl req -x509 -sha256 -newkey rsa:2048 -keyout iap-ca.key -out iap-ca.crt -days 356 -nodes -subj '/CN=IAP Certificate Authority'
   ```

   > **Note:**
   >
   > If you already have a CA certificate and a private key, then you can skip this step.

2. On the Linux instance, create a server certificate and a private key that have been signed using the private key of the CA created in step 1.

   ```
   openssl req -new -newkey rsa:2048 -keyout iap-wildcard.key -out iap-wildcard.csr -nodes -subj '/CN=*.example.com'

   openssl x509 -req -sha256 -days 365 -in iap-wildcard.csr -CA iap-ca.crt -CAkey iap-ca.key -set_serial 04 -out iap-wildcard.crt
   ```

   Ensure that you specify a wildcard character as the subdomain name in the Common Name (CN) of the server certificate. This ensures that the same server certificate is valid for all the subdomains of the given domain name.

   For example, consider that you have separate hostnames for the production and staging environments, *prod.example.com* and *staging.example.com*. By specifying a wildcard character in the Common Name of the server certificate, you can use the same server certificate to authenticate *prod.example.com* and *staging.example.com*.

3. On the Linux instance, create a client certificate and key that have been signed using the private key of the CA created in step 1.

   ```
   openssl req -new -newkey rsa:2048 -keyout iap-client.key -out iap-client.csr -nodes -subj '/CN=My IAP Client'

   openssl x509 -req -sha256 -days 365 -in iap-client.csr -CA iap-ca.crt -CAkey iap-ca.key -set_serial 02 -out iap-client.crt
   ```

4. Copy all the certificates to a common directory.

   For example, create a directory named *iap-certs* and copy all the certificates that have been created to this directory.

2.5 Creating the Cloud Runtime Environment

This section describes how to create the Cloud runtime environment.

2.5.1 Creating the OpenShift Runtime Environment

This section describes how to create the OpenShift runtime environment.

Prerequisites

Before creating the runtime environment on OpenShift, ensure that you have a valid OpenShift account and the following information:

- Login URL for the OpenShift account
- Authentication credentials for the OpenShift account

Audience

It is recommended that you have working knowledge of OpenShift.

2.5.1.1 Logging in to the OpenShift Environment

This section describes how you can login to the OpenShift environment.

➤ To login to the OpenShift environment:

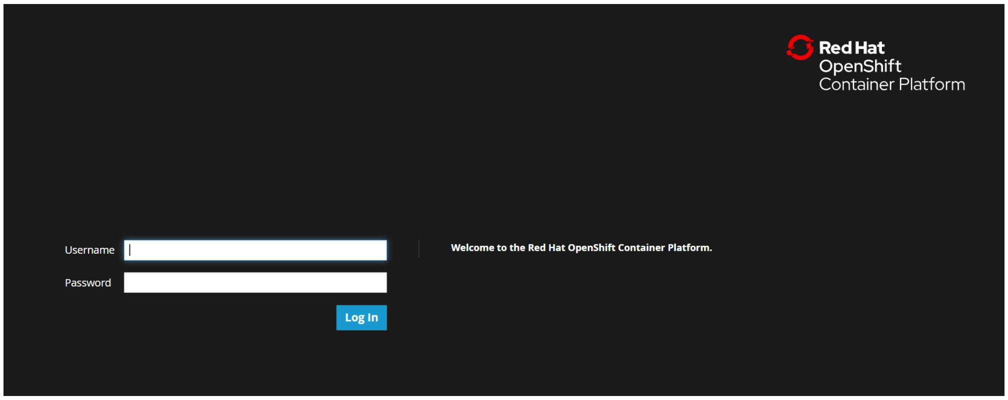
1. Access OpenShift at the following URL:

   https://<OpenShift Console URL>/dashboards

   The OpenShift home screen appears.

2. Click **Sign In to the Console**.

   The Sign in screen appears.



Welcome to the Red Hat OpenShift Container Platform.

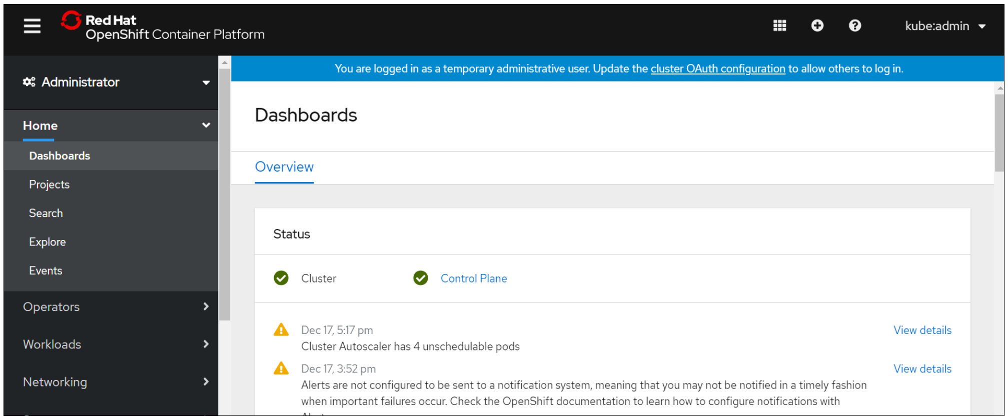
Figure 2-1: OpenShift Container Platform Sign in screen

3. Enter the following details:
   - User name

- Password

4. Click **Log In**.

   After successful authentication, the **Dashboards** screen appears.



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

Dec 17, 5:17 pm
Cluster Autoscaler has 4 unschedulable pods

Dec 17, 3:52 pm
Alerts are not configured to be sent to a notification system, meaning that you may not be notified in a timely fashion when important failures occur. Check the OpenShift documentation to learn how to configure notifications with

Figure 2-2: Dashboards Screen

2.6 Deploying the Containers with Security Context Constraints (SCC) and Role-Based Access Control (RBAC)

This section describes the steps that you need to perform for deploying the Get ESA Policy, Init, and AP-REST containers with Security Context Constraints (SCC) and Role Based Access Control (RBAC). The SCC determines the security policies for running a pod.

The user with the *cluster-admin* role creates the RBAC and applies the SCC. This user also creates two Kubernetes service accounts that will install the Helm charts for deploying the Get ESA Policy and the AP-REST containers.

2.6.1 Applying the SCC

This section describes how to apply the SCC.

> **Note:** Protegrity recommends you to use this SCC. However, you can choose to add additional security configurations in the SCC based on your requirements.

> **Important:** You require a *cluster-admin* role to perform this task.

➤ To apply the SCC:

1. On the Linux instance, run the following command to create project for deploying the containers.

   ```
   oc new-project <Namespace>
   ```

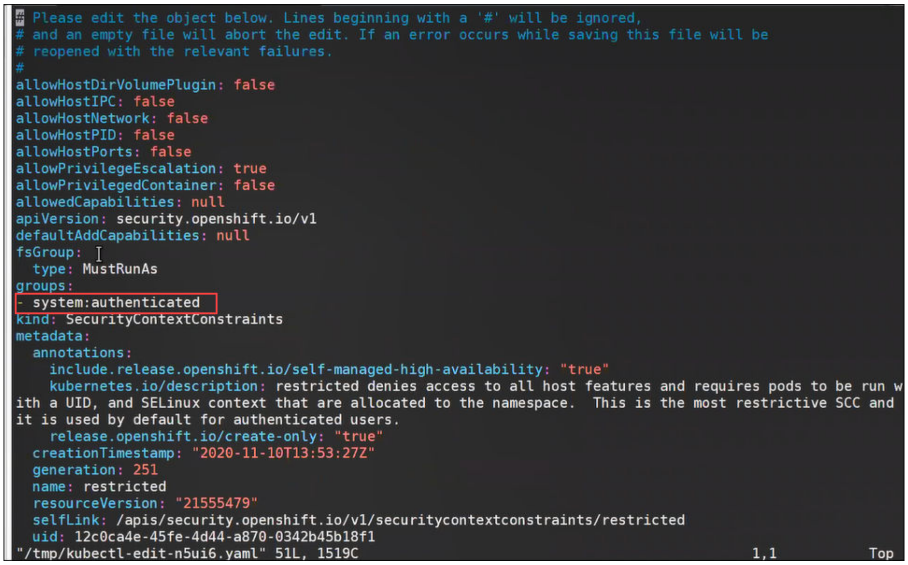
   For example:

   ```
   oc new-project iap-rest
   ```

2. Run the following command to edit the Restricted SCC, which is the default SCC that is applied to all the pods in the Kubernetes cluster.

   ```
   oc edit scc restricted
   ```

The Restricted SCC opens in an editor.



```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
allowHostDirVolumePlugin: false
allowHostIPC: false
allowHostNetwork: false
allowHostPID: false
allowHostPorts: false
allowPrivilegeEscalation: true
allowPrivilegedContainer: false
allowedCapabilities: null
apiVersion: security.openshift.io/v1
defaultAddCapabilities: null
fsGroup:
  type: MustRunAs
groups:
- system:authenticated
kind: SecurityContextConstraints
metadata:
  annotations:
    include.release.openshift.io/self-managed-high-availability: "true"
    kubernetes.io/description: restricted denies access to all host features and requires pods to be run w
ith a UID, and SELinux context that are allocated to the namespace.  This is the most restrictive SCC and
it is used by default for authenticated users.
    release.openshift.io/create-only: "true"
  creationTimestamp: "2020-11-10T13:53:27Z"
  generation: 251
  name: restricted
  resourceVersion: "21555479"
  selfLink: /apis/security.openshift.io/v1/securitycontextconstraints/restricted
  uid: 12c0ca4e-45fe-4d44-a870-0342b45b18f1
"/tmp/kubectl-edit-n5ui6.yaml" 51L, 1519C                                            1,1           Top
```

3. Delete the following line from the Restricted SCC and then save the file to ensure that the Restricted SCC is not applied to the namespace where the AP-REST containers will be deployed.

```
- system:authenticated
```

4. Run the following command to ensure that the Restricted SCC is reapplied to the system-level pods in the Kubernetes cluster.

```
oc get ns | awk '/openshift/{system("oc adm policy add-scc-to-group restricted system:serviceaccounts:" $1)}'
```

This command ensures that the Restricted SCC is applied to OpenShift infrastructure pods.

> **Important:** If you want to apply the Restricted SCC to any other namespace, then you need to run the following command:
>
> ```
> oc adm policy add-scc-to-group restricted system:serviceaccounts:<Namespace>
> ```

5. Navigate to the directory where you have extracted the Helm charts for deploying the Get ESA Policy application, and edit the *pty-scc.yaml* file to replace the *<NAMESPACE>* placeholder in the following snippet with the namespace where the IAP-REST application will be deployed.

```
groups:
- system:serviceaccounts:<NAMESPACE>
```

For example:

```
groups:
- system:serviceaccounts:iap-rest
```

For more information about the extracted Helm charts, refer to the *step 5* of the section *Initializing the Linux Instance*.

6. Run the following command to apply the Protegrity SCC to the Kubernetes cluster.

```
oc apply -f pty-scc.yaml
```

For example:

```
oc apply -f pty-scc.yaml
```

2.6.2 Applying RBAC

This section describes how to apply RBAC.

> **Important:** You require a *cluster-admin* role to perform this task.

To apply an RBAC:

1. Perform the following steps to create service accounts that can be used to deploy the Get ESA Policy, Init, and AP-REST containers.

   > **Important:**
   >
   > You require a *cluster-admin* role to perform these steps.

   a. Run the following command to create a service account that can be used to deploy the Get ESA Policy container.

   ```
   oc create serviceaccount <Service account name> -n <Namespace>
   ```

   For example:

   ```
   oc create serviceaccount get-esa-deployer --namespace iap-rest
   ```

   > **Important:** Ensure that the name of the service account is *get-esa-deployer*, which has been defined in the *pty-rbac.yaml* file.

   If you are using the OpenShift version 4.12, then perform *step 1b* to create a token for the service account.

   If you are using an OpenShift version prior to 4.12, then perform *step 1c* to create a token for the service account.

   b. If you are using the OpenShift version 4.12, then run the following command to create a token for the service account.

   ```
   oc create token <Service account name>
   ```

   c. If you are using an OpenShift version prior to 4.12, then run the following commands to create a token for the service account:

   i. Run the following command to retrieve the token name for the service account.

   ```
   oc get serviceaccount <Service Account Name> --namespace <Namespace> -o
   jsonpath='{.secrets}'
   ```

   For example:

   ```
   oc get serviceaccount get-esa-deployer --namespace iap-rest -o
   jsonpath='{.secrets}'
   ```

   The output displays the following token names.

   ```
   [{"name":"get-esa-deployer-dockercfg-ntmbt"},{"name":"get-esa-deployer-token-s9l6p"}]
   ```

   The first token name is for the Docker configuration, while the second token name is for the service account.

   ii. Run the following command to obtain the service account token, which is stored as a Kubernetes secret, using the token name retrieved in *step 1c > i*.

   ```
   oc get secret <service account token name> --context <Valid admin context> --
   namespace <Namespace> -o jsonpath='{.data.token}'
   ```

For example:

```
oc get secret <token name> --context kubernetes-admin@kubernetes --namespace
iap-rest -o jsonpath='{.data.token}'
```

A Kubernetes context specifies the configuration for a specific Kubernetes cluster that is associated with a namespace and a corresponding service account.

You can obtain the context by running the following command:

```
oc config current-context
```

iii. Run the following command to decode the service account token obtained in *step 1c > ii*.

```
TOKEN=`echo -n <Token from step3> | base64 -d`
```

d. If you are using the OpenShift version 4.12, then repeat *step 1a* and *step 1b* to create a service account that can be used to deploy the AP-REST container.

If you are using an OpenShift version prior to 4.12, then repeat *step 1a* and *step 1c*.

> **Important:** Ensure that the name of the service account is *aprest-deployer*, which has been defined in the *pty-rbac.yaml* file.

2. Navigate to the directory where you have extracted the Helm charts for deploying the Get ESA Policy application, and run the following command to apply the Protegrity RBAC to the Kubernetes cluster.

> **Important:**
>
> You require a *cluster-admin* role to perform these steps.

```
oc apply -f pty-rbac.yaml -n <Namespace>
```

For example:

```
oc apply -f pty-rbac.yaml -n iap-rest
```

For more information about the extracted Helm charts, refer to the *step 5* of the section *Initializing the Linux Instance*.

2.6.3 Applying a Network Policy

A network policy enables you to allow or prevent the requests sent to the AP-REST container for performing security operations. This section describes how to apply a network policy.

In this approach, you first need to add a label to the namespace where the pods that will be used to send the request are deployed. You then apply a network policy that allows requests from only those namespaces that have the matching label as defined in the policy.

> **Important:** The network policy is only applicable to the requests sent by a pod in the same Kubernetes cluster where the AP-REST container has been deployed, which is indicated by *Pattern 1* of the deployment architecture.

> **Warning:** Do not apply the network policy to the *Pattern 2* or *Pattern 3* deployment architecture.

If you apply the network policy to the Pattern 2 or Pattern 3 deployment architecture, then the OpenShift *routes*, which are used to expose the Kubernetes service externally, stop working.

Note: This procedure is optional and you can use custom network policies instead of using the policy specified in this procedure.

Important: You require a *cluster-admin* role to perform this task.

To apply a network policy:

1. Run the following command to apply a label to the namespace where you have deployed the pod that will be used to send a request to the AP-REST container.

   ```
   kubectl label namespace <Namespace> purpose=<Label>
   ```

   For example:

   ```
   kubectl label namespace default purpose=production
   ```

   In this example, a label named *production* has been applied to the default *namespace*, which is the namespace where the pod that will be used to send a request has been deployed.

2. Create a custom *aprest_policy.yaml* file for specifying the network policy for the AP-REST container.

   ```
   kind: NetworkPolicy
   apiVersion: networking.k8s.io/v1
   metadata:
     name: aprest-allow-traffic
   spec:
     podSelector:
       matchLabels:
         app.kubernetes.io/name: iap-rest
     ingress:
     - ports:
       - port: 8443
       from:
       - namespaceSelector:
           matchLabels:
             purpose: production
   ```

   Important: If you want to copy the contents of the *aprest_policy.yaml* file, then ensure that you indent the file as per YAML requirements.

   The network policy specified in this example will allow only those requests to be forwarded to the port *8443* of the service that have originated from a pod, which has been deployed in the namespace that has the label as *production*.

3. Run the following command to apply the network policy to the AP-REST container.

   ```
   kubectl apply aprest_policy.yaml -n <iap-rest namespace>
   ```

   This network policy is applicable to the protect, unprotect, and reprotect requests sent by a pod within the same Kubernetes cluster where the AP-REST container has been deployed.

   For more information about the security operations impacted by this network policy, refer to *step 2* of the section *Running Security Operations*.

2.6.4 Setting up the Environment for Deploying the Get ESA Policy Container

This section describes how to set up the environment for deploying the Get ESA Policy container on the Kubernetes cluster.

> **Important:** You require a *cluster-admin* role to perform this task.

➤ To set up the environment for deploying the Get ESA Policy container:

1. On the Linux instance, navigate to the location where you have extracted the Helm charts for deploying the Get ESA Policy application.

   For more information about the extracted Helm charts, refer to the *step 5* of the section *Initializing the Linux Instance*.

2. Modify the *nfs-pv.yaml* file to update the persistent volume claim details, by specifying the value of the path and server parameters.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
  labels:
    purpose: policy-store
spec:
  capacity:
    # The amount of storage allocated to this volume.
    # e.g. 10Gi
    storage: AMOUNT_OF_STORAGE
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    # The path that is exported by the NFS server
    path: MOUNTH_PATH
    # The host name or IP address of the NFS server.
    server: SERVER_NAME_OR_IP
    readOnly: false
```

3. Run the following command to create a persistent volume.

```
oc apply -f get-esa-policy/nfs-pv.yaml
```

4. Modify the *nfs-pvc.yaml* file to update the persistent volume claim details, by specifying the name of the persistent volume claim and the storage amount.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pvc
spec:
  storageClassName: ""
  selector:
    matchLabels:
      purpose: "policy-store"
  accessModes:
  - ReadWriteMany
  resources:
     requests:
        # The amount of storage allocated to this volume.
        # e.g. 10Gi
       storage: AMOUNT_OF_STORAGE
```

5. Run the following command to create a persistent volume claim.

```
oc apply -f get-esa-policy/nfs-pvc.yaml -n iap-rest
```

6. On the Linux instance, create a mount point for the NFS by running the following command.

   ```
   mkdir /nfs
   ```

   This command creates a mount point *nfs* on the file system.

7. Run the following mount command to mount the NFS on the directory created in *step 6*.

   ```
   sudo mount <nfs server IP>:<Path to the shared folder> /nfs
   ```

2.6.5 Deploying the Get ESA Policy Container on the Kubernetes Cluster

This section describes how to deploy the Get ESA Policy container on the Kubernetes cluster by installing the Helm charts.

➤ To deploy the Get ESA Policy container on the Kubernetes cluster:

1. On the Linux instance, navigate to the location where you have extracted the Helm charts for deploying the Get ESA Policy application.
2. Run the following command to update the *kube-config* file, which is the configuration file that is generated when you create a Kubernetes cluster.

   ```
   oc config set-cluster <Name of the Kubernetes Cluster> --server=https://<DNS or IP
   address of the server where the Kubernetes Cluster has been deployed>:<Port number>
   --certificate-authority=path/to/certificate/authority
   ```

   The *path/to/certificate/authority* value indicates the path where the Certificate Authority (CA) certificate file is stored.

   If you do not want to verify the connection between the OpenShift CLI and the Kubernetes cluster using TLS, then use the following command to update the *kube-config* file:

   ```
   oc config set-cluster <Name of the Kubernetes Cluster> --server=https://DNS or IP
   address of the server where the Kubernetes Cluster has been deployed:<Port number>
   --insecure-skip-tls-verify=true
   ```

3. Perform the following steps to set up a Kubernetes context for the *get-esa-deployer* service account.

   a. If you are using the OpenShift version 4.12, then run the following command to create user credentials in the *kube-config* file, using the token created for the service account to deploy the GET ESA Policy container in *step 1b* of the section *Applying RBAC*.

      ```
      oc config set-credentials <username> --token <TOKEN from step 1b of the section
      Applying Role-Based Access Control (RBAC)>
      ```

      If you are using an OpenShift version prior to 4.12, then then run the following command to create user credentials in the *kube-config* file, using the token created for the service account to deploy the AP-REST container in *step 1c* of the section *Applying RBAC*.

      ```
      oc config set-credentials <username> --token <TOKEN from step 1c of the section
      Applying Role-Based Access Control (RBAC)>
      ```

   b. Run the following command to create a context in the *kubeconfig* file for communicating with the Kubernetes cluster.

      ```
      oc config set-context <Context Name> --cluster=<Name of the Kubernetes Cluster in
      the kube-config file> --user=<Service account name>
      ```

      > **Note:** Ensure that the name of the Kubernetes cluster is the same as that you have specified in *step 2*.
      >
      > You can obtain the name of the Kubernetes cluster by running the `oc config get-contexts` command.

c. Run the following command to set the Kubernetes context to the newly created context in *step 3b*.

```
oc config use-context <Context name from step 2b>
```

4. Run the following command to create the *esa-creds* secret, which is used by the Get ESA Policy container to access the ESA using the non-administrator credentials.

```
oc create secret generic esa-creds --from-literal=esa_user=<ESA user> --from-
literal= esa_pass=<ESA user password> --namespace <NAMESPACE>

oc create secret generic esa-creds --from-literal=esa_user=non-admin --from-
literal= esa_pass=<ESA user password> --namespace iap-rest
```

> **Important:** Ensure that the user, who will be used to access the ESA, has been assigned the *Export Certificates* and *Appliance CLI Viewer* permissions through a custom role. Roles are templates that include permissions and users can be assigned one or more roles.
>
> For more information about managing roles, refer to the section *Managing Roles* in the *Appliances Overview 9.1.0.0* guide.
>
> For more information about managing users, refer to the section *Managing Users* in the *Appliances Overview 9.1.0.0* guide.

5. Run the following command to create a Kubernetes secret for accessing the image registry using existing Docker credentials.

```
oc create secret generic regcred --from-file=.dockerconfigjson=<PATH_TO_DOCKER_CONF
IG>/config.json --type=kubernetes.io/dockerconfigjson --namespace <NAMESPACE>

oc create secret generic regcred --from-file=.dockerconfigjson=<PATH_TO_DOCKER_CONF
IG>/config.json --type=kubernetes.io/dockerconfigjson --namespace iap-rest
```

For more information about the credentials required to access the image repository, refer to the section *Image Pull Secrets*.

6. Run the following command to create a passphrase secret, which is used by the Get ESA Policy container to encrypt the policy.

```
oc create secret generic pbe-secret --from-literal='passphrase=<passphrase>' --
from-literal='salt=<salt>' -n iap-rest
```

This command is used to implement the Password-Based Encryption (PBE) for encrypting the policy.

You need to provide a passphrase and a salt as a Kubernetes secret to the Get ESA Policy Container. The policy is encrypted using the key generated by using the passphrase and salt. The salt is used as an initialization vector for generating the key.

> **Important:** Ensure that the passphrase has a minimum length of 12 characters, with atleast 1 uppercase letter, 2 digits, 2 special characters, and 2 non-letters (such as, digits or special characters). The passphrase must also have a minimum value of entropybits as *30* and a minimum strength of *0.66*.
>
> Entropybits defines the randomness of the password and the strength defines the complexity of the password.
>
> For more information about the password strength, refer to the section *Password Strength*.

7. Modify the default values in the *values.yaml* file as required.

The *values.yaml* file contains the default configuration values for deploying the Get ESA Policy application on the Kubernetes cluster.

```
...
..
.

## pod service account to be used
## leave the field empty if not applicable
```

```
serviceAccount:

## set the Pod's security context object
## leave the field empty if not applicable
podSecurityContext:
  runAsUser: 1000
  runAsGroup: 1000
  fsGroup: 1000
  supplementalGroups: [1000]

....
....
....

## k8s secret for storing ESA credentials
esaCredSecrets: ESA_CREDENTIAL_SECRET_NAME

## k8s secret containing passphrase for encrypting policy
pbeSecrets: PBE_SECRET_NAME

securityConfigDestination: VolumeMount

## If securityConfigDestination is Volume Mount,
## specify the name of persistent volume claim to be used for this deployment.
pvcName: PVC_NAME

policy:
  esaIP: ESA_IP

  ## pepserver configurations that can be changed.
    ## emptystring: Set the output behavior for empty strings
      ## null = (default) null value is returned for empty input string
      ## empty = empty value is returned for empty input string
      ## encrypt = encrypted values is returned for empty input string
    ## logLevel: Specifies the logging level for pepserver
      ## OFF (no logging)
      ## SEVERE
      ## WARNING
      ## INFO
      ## CONFIG
      ## ALL (default)
    ## caseSensitive: Specifies how policy users are checked against policy
      ## yes = (The default) policy users are treated in case sensitive manner
      ## no = policy users are treated in case insensitive manner.
  pepserverConfig:
    emptyString: "null"
    logLevel: "ALL"
    caseSensitive: "yes"
   ## absolute path in PV where the policy dump file will be stored. <eg. /test/xyz/
policy >
  filePath: ABSOLUTE_POLICY_PATH

  ## time (in mins) to wait till the policies get downloaded
  ## default is 15 (mins)
  timeout: 15
```

For more information about Helm charts and their default values, refer to the *Appendix A: Get ESA Policy Helm Chart Details*.

Field	**Description**
podSecurityContext	Specify the privilege and access control settings for the pod. The default values are set as follows: • *runAsUser* - 1000 • *runAsGroup* - 1000 • *fsGroup* - 1000

Field	Description
	- *supplementalGroups* - 1000 **Note:** Set the value of the *fsGroup* parameter to a non-root value. For example, you can set the value of the *fsGroup* parameter to any value between *1* and *65534*. **Note:** Ensure that at any given point of time, the value of at least one of the four parameters must be *1000*. This ensures that the pod has the required permissions to access the *pty* directories in the containers. For more information about the Pod Security Context, refer to the section *Configure a Security Context for a Pod or Container* in the Kubernetes documentation. For more information about the parameters, refer to the section *PodSecurityContext v1 Core* in the Kubernetes API documentation.
esaCredSecrets	Specify the value of the secret that is used to store the ESA credentials. This is used by the Get ESA Policy container to retrieve the policy from the ESA.
pbeSecrets	Specify the value of the secret that is used to store the passphrase and salt. The Get ESA Policy container encrypts the policy using the key generated by using the passphrase and salt. **Important:** Ensure that the password meets the following requirements: - The password must contain a minimum of 10 characters and a maximum of 128 characters. - The user should be able to specify Unicode characters, such as Emoji, in the password. - The user should avoid commonly used passwords, such as, *123456789*, *11111111*, *password*, and *qwertyuiop*.
securityConfigDestination	Specify the value as *VolumeMount*.
pvcName	Specify the name of the Persistent Volume Claim where you want to store the policy snapshot.
policy/ESA_IP	Specify the IP address of the ESA from where you want to download the policy.
policy/pepserverConfig/emptyString	Defines the behavior when the data to protect is an empty string. The following are the possible values: - *empty* - An empty value is returned for an empty input string. - *encrypt* - An encrypted value is returned for an empty input string. - *null* - A null value is returned for an empty input string. By default, the value of this parameter is set to *null*.

Field	Description
	For more information about empty string handling by protectors, refer to the section *Appendix C: Empty String Handling by Protectors* in the *Protection Methods Reference Guide 9.1.0.0*.
policy/pepserverConfig/logLevel	Specifies the logging level for the PEP server. You can specify one of the following values: • *OFF* - No logging • *SEVERE* • *WARNING* • *INFO* • *CONFIG* • *ALL* By default, the value of this parameter is set to *ALL*.
policy/pepserverConfig/caseSensitive	Specifies how policy users are checked against the policy. This parameter enables you to configure the case-sensitivity of the policy users. If this parameter is set to *no*, then the PEP server considers the policy user names as case insensitive. If this parameter is set to *yes* or if it is commented in the file, then the PEP server considers the policy user names as case-sensitive. The default value is *yes*. By default, the value of this parameter is set to *yes*.
policy/filePath	Specify the path in the persistent volume where you want to store the immutable policy package. For example, if you are using NFS as the persistent volume, then you can specify the file path as */<Mount directory>/xyz/imp*. In this case, a policy with the name as *imp* will be stored in the */<Mount direc tory>/xyz* directory in the required persistent volume. **Note:** The *<Mount directory>* is a directory inside the Mount point. **Important:** Use only uppercase and lowercase letters, numbers, dot, and underscore in the file path. Do not use special characters in the file path. **Note:** This value is case-sensitive.
policy/timeout	Specify the time for which the Get ESA Policy container tries to communicate with the ESA for fetching the policies, after which the

Field	Description
	connection times out. By default, this value is set to 15. The unit of the timeout parameter is minutes.

8. Run the following command to deploy the Get ESA Policy application on the Kubernetes cluster.

   ```
   helm install <Release Name> --namespace <Namespace where you want to deploy the Get
   ESA Policy application> <Location of the directory that contains the Helm charts>
   ```

   For example:

   ```
   helm install get-policy-esa --namespace iap-rest get-esa-policy
   ```

9. Run the following command to check the deployment status.

   ```
   oc get pods -n iap-rest
   ```

   After the Get ESA Policy container is up, it retrieves the policy from the ESA and uploads it to the persistent volume specified in the *values.yaml* file. After uploading the policy snapshot, the Kubernetes job is completed.

2.6.6 Verifying the AP-REST Policy Snapshot on the NFS

This section describes how you can verify the AP-REST policy snapshot on the NFS.

The Get ESA Policy container retrieves the policy from the ESA and uploads a snapshot of this immutable policy to the NFS.

➤ To verify the policy snapshot on the NFS:

Perform the following tasks to verify whether the policy package file has been uploaded to the NFS:

1. On the Linux instance, navigate to the location where you have mounted the NFS.

   For more information about the location where you have mounted the NFS, refer to *step 6* in the section *Deploying the Get ESA Policy Container on the Kubernetes Cluster*.
2. List all the files within the directory to access the policy snapshot.

2.6.7 Setting up the Environment for Deploying the AP-REST Container

This section describes how to set up the environment for deploying the AP-REST container on the Kubernetes cluster.

> **Important:** You require a *cluster-admin* role to perform this task.

➤ To set up the environment for deploying the AP-REST container:

Run the following command to create the *nginxCertsSecrets* secret, which stores the TLS certificates for the NGINX container.

```
oc create -n iap-rest secret generic nginx-certificates --from-file=tls.crt=iap-
wildcard.crt --from-file=tls.key=iap-wildcard.key --from-file=ca.crt=iap-ca.crt
```

The TLS certificates are created using the procedure described in the section *Creating Certificates and Keys for TLS Authentication*.

2.6.8 Deploying the AP-REST Container on the Kubernetes Cluster

This section describes how to deploy the AP-REST container on the Kubernetes cluster by installing the Helm charts.

➤ To deploy a release on the Kubernetes cluster:

1. On the Linux instance, navigate to the location where you have extracted the Helm charts to deploy the AP-REST container.

   For more information about the extracted Helm charts, refer to the *step 6* of the section *Initializing the Linux Instance*.
2. Run the following command to update the *kube-config* file, which is the configuration file that is generated when you create a Kubernetes cluster.

   ```
   oc config set-cluster <Name of the Kubernetes Cluster> --server=https://<DNS or IP address of the server where the Kubernetes Cluster has been deployed>:<Port number> --certificate-authority=path/to/certificate/authority
   ```

   The *path/to/certificate/authority* value indicates the path where the Certificate Authority (CA) certificate file is stored.

   If you do not want to verify the connection between the OpenShift CLI and the Kubernetes cluster using TLS, then use the following command to update the *kube-config* file:

   ```
   oc config set-cluster <Name of the Kubernetes Cluster> --server=https://DNS or IP address of the server where the Kubernetes Cluster has been deployed:<Port number> --insecure-skip-tls-verify=true
   ```
3. Perform the following steps to set up a Kubernetes context for the *aprest-deployer* service account.
   a. If you are using the OpenShift version 4.12, then run the following command to create user credentials in the *kube-config* file, using the token created for the service account to deploy the AP-REST container in *step 1b* of the section *Applying RBAC*.

      ```
      oc config set-credentials <username> --token <TOKEN from step 1b of the section Applying Role-Based Access Control (RBAC)>
      ```

      If you are using an OpenShift version prior to 4.12, then then run the following command to create user credentials in the *kube-config* file, using the token created for the service account to deploy the AP-REST container in *step 1c* of the section *Applying RBAC*.

      ```
      oc config set-credentials <username> --token <TOKEN from step 1c of the section Applying Role-Based Access Control (RBAC)>
      ```
   b. Run the following command to create a context in the *kubeconfig* file for communicating with the Kubernetes cluster.

      ```
      oc config set-context <Context Name> --cluster=<Name of the Kubernetes Cluster in the kube-config file> --user=<Service account name>
      ```

      > **Note:** Ensure that the name of the Kubernetes cluster is the same as that you have specified in *step 2*.
      >
      > You can obtain the name of the Kubernetes cluster by running the `oc config get-contexts` command.
   c. Run the following command to set the Kubernetes context to the newly created context in *step 3b*.

      ```
      oc config use-context <Context name from step 2b>
      ```
4. Modify the default values in the *values.yaml* file as required.

The *iap-rest > values.yaml* file contains the default configuration values for deploying the AP-REST container on the Kubernetes cluster.

```
...
..
.

## pod service account to be used
## leave the field empty if not applicable
serviceAccount:


...
...


## set the Pod's security context object
## leave the field empty if not applicable
podSecurityContext:
  runAsUser: 1000
  runAsGroup: 1000
  fsGroup: 1000
  supplementalGroups: [1000]
...
...

# If Volume Mount, name of persistent volume claim to be used for this deployment.
pvcName: PVC_NAME

## k8s secret containing passphrase for encrypting policy
pbeSecrets: PBE_SECRET_NAME

## k8s secret containing TLS certificates for nginx sidecar
nginxCertsSecrets: NGINX_SECRET_NAME

## start with blue deployment first
deploymentInUse: blue

## Policy metadata for Blue Deployments
blueDeployment:
  enabled: true
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV where the policy dump file will be stored. <eg. /test/xyz/
policy >
    filePath: ABSOULTE_POLICY_PATH

## Policy metadata for Green Deployments
greenDeployment:
  enabled: false
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV where the policy dump file will be stored. <eg. /test/xyz/
policy >
    filePath: ABSOULTE_POLICY_PATH

## specify the initial no. of iaprest Pod replicas
replicaCount: 1

## HPA configuration
autoScaling:
  minReplicas: 1
  maxReplicas: 10
  targetCPU: 80

## specify the ports exposed in your iaprest configurations where,
## name - distinguishes between different ports.
## port - the port on which you wan't to expose the service externally.
## targetPort - the port no. configured while creating Tunnel.
iaprestService:
    name: "iaprest"
    port: 8443
    targetPort: 8443

## specify hostname, path and annotations for prod and staging routes in this section.
## staging routes will be enabled only if both blue & green deployments are enabled.
routes:
  prodService:
    hostName: "prod.example.com"
    annotations:
      haproxy.router.openshift.io/balance: roundrobin
      # specify route annotations if any
      # e.g. haproxy.router.openshift.io/ip_whitelist: 192.168.1.10

  stagingService:
    hostName: "staging.example.com"
    annotations:
      haproxy.router.openshift.io/balance: roundrobin
      # specify route annotations if any
      # e.g. haproxy.router.openshift.io/ip_whitelist: 192.168.1.10
```

In a *Blue-Green* deployment strategy, the Release 1 is considered as the *Blue* deployment.

For more information about Helm charts and their default values, refer to the section *Appendix B: AP-REST Helm Chart Details*.

Field	Description
podSecurityContext	Specify the privilege and access control settings for the pod. The default values are set as follows: • *runAsUser* - 1000 • *runAsGroup* - 1000 • *fsGroup* - 1000 • *supplementalGroups* - 1000 **Note:** Set the value of the *fsGroup* parameter to a non-root value. For example, you can set the value of the *fsGroup* parameter to any value between *1* and *65534*. **Note:** Ensure that at any given point of time, the value of at least one of the four parameters must be *1000*. This ensures that the pod has the required permissions to access the *pty* directories in the containers. For more information about the Pod Security Context, refer to the section *Configure a Security Context for a Pod or Container* in the Kubernetes documentation. For more information about the parameters, refer to the section *PodSecurityContext v1 Core* in the Kubernetes API documentation.
pvcName	Specify the name of the Persistent Volume Claim where you want to store the policy snapshot.
pbeSecrets	Specify the Kubernetes secret that contains the passphrase and the salt that were used to generate key, which encrypted the policy.

Field	Description
	The AP-REST container uses the value specified in the *pbeSecrets* parameter to decrypt the policy. **Important:** Ensure that the password meets the following requirements: • The password must contain a minimum of 10 characters and a maximum of 128 characters. • The user should be able to specify Unicode characters, such as Emoji, in the password. • The user should avoid commonly used passwords, such as, *123456789*, *11111111*, *password*, and *qwertyuiop*.
nginxCertsSecrets	Specify the Kubernetes secret that contains the TLS certificates for the NGINX container. The TLS certificates are created using the procedure described in the section *Creating Certificates and Keys for TLS Authentication*.
deploymentInUse	Specify the value as *blue*. This indicates that the *Blue* deployment strategy is in use.
blueDeployment/securityConfigSource	Specify the value as *VolumeMount*.
blueDeployment/policy/filePath	Specify the path in the persistent volume where you have stored the policy. For example, if you are using NFS as the persistent volume, then you can specify the file path as */<Mount directory>/xyz/imp*. In this case, a policy with the name as *imp* will be stored in the */<Mount directory>/xyz* directory in the required persistent volume. **Important:** Use only uppercase and lowercase letters, numbers, dot, and underscore in the file path. Do not use special characters in the file path. **Note:** This value is case-sensitive.
greenDeployment/enabled	Specify the value as *false*. While deploying Release 1, the *Green* deployment strategy is always disabled.
iaprestService/name	Specify a name for the tunnel to distinguish between ports.
iaprestService/port	Specify the port number on which you want to expose the Kubernetes service externally.
iaprestService/targetPort	Specify the port on which the AP-REST container is running inside the Docker container. **Important:** Do not change this value.
routes/prodService/hostname	Specify the hostname of the production service.
routes/prodService/annotations/haproxy.router.openshift.io/balance	Specify the algorithm for the load balancer. By default, the value is set to *roundrobin*.
routes/stagingService/hostname	Specify the hostname of the staging service.
routes/stagingService/annotations/haproxy.router.openshift.io/balance	Specify the algorithm for the load balancer. By default, the value is set to *roundrobin*.

5. Run the following command to deploy the AP-REST container on the Kubernetes cluster:

   ```
   helm install <Release Name> --namespace <Namespace where you want to deploy the AP-REST container> <Location of the directory that contains the Helm charts>
   ```

   For example:

   ```
   helm install ap-rest --namespace iap-rest iap-rest
   ```

6. Perform the following steps to check the status of the Kubernetes resources.

   a. Run the following command to check the status of the pods.

      ```
      oc get pods -n <namespace>
      ```

      For example:

      ```
      oc get pods -n iap-rest
      ```

2.6.9 Upgrading the Helm Charts

This section describes the steps for upgrading the Helm charts.

Protegrity recommends using the *Blue-Green* method of upgrading the Helm charts. In this method, you use two modes of deploying the application, production mode and staging mode. In the staging mode, you can test the changes to the application in a staging environment, without impacting the production environment. After you have tested the application in the staging environment, you can switch the environments by making the staging environment as the new production environment and the production environment as the new staging environment.

In this way, you can seamlessly divert the customer traffic to the modified application without any production downtime. After your modified application is running on the new production environment, you can shutdown the application that is running on the staging environment. In the *Blue-Green* method of deploying applications, only one Release is active at a given point of time.

To upgrade the Helm charts:

1. On the Linux instance, navigate to the location where you have extracted the Helm charts to deploy the AP-REST container.

   For more information about the extracted Helm charts, refer to the step *6* of the section *Initializing the Linux Instance*.

   The *values.yaml* file contains the default configuration values for deploying the AP-REST container on the Kubernetes cluster.

   ```
   ...
   ..
   .

   ## pod service account to be used
   ## leave the field empty if not applicable
   serviceAccount:


   ...
   ...


   ## set the Pod's security context object
   ## leave the field empty if not applicable
   podSecurityContext:
     runAsUser: 1000
     runAsGroup: 1000
     fsGroup: 1000
     supplementalGroups: [1000]
   ```

```
...
...

# If Volume Mount, name of persistent volume claim to be used for this deployment.
pvcName: PVC_NAME

## k8s secret containing passphrase for encrypting policy
pbeSecrets: PBE_SECRET_NAME

## k8s secret containing TLS certificates for nginx sidecar
nginxCertsSecrets: NGINX_SECRET_NAME

## start with blue deployment first
deploymentInUse: blue

## Policy metadata for Blue Deployments
blueDeployment:
  enabled: true
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV where the policy dump file will be stored. <eg. /test/xyz/
policy >
    filePath: ABSOULTE_POLICY_PATH

## Policy metadata for Green Deployments
greenDeployment:
  enabled: false
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV where the policy dump file will be stored. <eg. /test/xyz/
policy >
    filePath: ABSOULTE_POLICY_PATH

## specify the initial no. of iaprest Pod replicas
replicaCount: 1

## HPA configuration
autoScaling:
  minReplicas: 1
  maxReplicas: 10
  targetCPU: 80

## specify the ports exposed in your iaprest configurations where,
## name - distinguishes between different ports.
## port - the port on which you wan't to expose the service externally.
## targetPort - the port no. configured while creating Tunnel.
iaprestService:
    name: "iaprest"
    port: 8443
    targetPort: 8443

## specify hostname, path and annotations for prod and staging routes in this section.
## staging routes will be enabled only if both blue & green deployments are enabled.
routes:
  prodService:
    hostName: "prod.example.com"
    annotations:
      haproxy.router.openshift.io/balance: roundrobin
      # specify route annotations if any
      # e.g. haproxy.router.openshift.io/ip_whitelist: 192.168.1.10

  stagingService:
    hostName: "staging.example.com"
    annotations:
      haproxy.router.openshift.io/balance: roundrobin
      # specify route annotations if any
      # e.g. haproxy.router.openshift.io/ip_whitelist: 192.168.1.10
```

In a *Blue-Green* deployment strategy, the Release 2 is considered as the *Green* deployment.

2. Modify the default values in the *values.yaml* file as required.

For more information about Helm charts and their default values, refer to the *Appendix: Helm Chart Details*.

Field	Description
podSecurityContext	Specify the privilege and access control settings for the pod. The default values are set as follows: • *runAsUser* - 1000 • *runAsGroup* - 1000 • *fsGroup* - 1000 • *supplementalGroups* - 1000 **Note:** Set the value of the *fsGroup* parameter to a non-root value. For example, you can set the value of the *fsGroup* parameter to any value between *1* and *65534*. **Note:** Ensure that at any given point of time, the value of at least one of the four parameters must be *1000*. This ensures that the pod has the required permissions to access the *pty* directories in the containers. For more information about the Pod Security Context, refer to the section *Configure a Security Context for a Pod or Container* in the Kubernetes documentation. For more information about the parameters, refer to the section *PodSecurityContext v1 Core* in the Kubernetes API documentation.
pvcName	Specify the name of the Persistent Volume Claim where you want to store the policy snapshot.
pbeSecrets	Specify the Kubernetes secret that contains the passphrase and the salt that were used to generate key, which encrypted the policy. The AP-REST container uses the value specified in the *pbeSecrets* parameter to decrypt the policy. **Important:** Ensure that the password meets the following requirements: • The password must contain a minimum of 10 characters and a maximum of 128 characters. • The user should be able to specify Unicode characters, such as Emoji, in the password. • The user should avoid commonly used passwords, such as, *123456789*, *11111111*, *password*, and *qwertyuiop*.
nginxCertsSecrets	Specify the Kubernetes secret that contains the TLS certificates for the NGINX container. The TLS certificates are created using the procedure described in the section *Creating Certificates and Keys for TLS Authentication*.

Field	Description
deploymentInUse	Specify the value as *blue*. This indicates that the *Blue* deployment strategy is in use.
greenDeployment\enabled	Specify the value as *true*. While upgrading the release to Release 2, the *Green* deployment strategy is enabled.
greenDeployment/securityConfigSource	Specify the value as *VolumeMount*. By default, the value is set to *VolumeMount*.
greenDeployment/policy/filePath	Specify the path in the persistent volume where you have stored the policy. For example, if you are using NFS as the persistent volume, then you can specify the file path as */<Mount directory>/xyz/imp*. In this case, a policy with the name as *imp* will be stored in the */<Mount directory>/xyz* directory in the required persistent volume. **Important:** Use only uppercase and lowercase letters, numbers, dot, and underscore in the file path. Do not use special characters in the file path. **Note:** This value is case-sensitive.
iaprestService/name	Specify a name for the tunnel to distinguish between ports.
iaprestService/port	Specify the port number on which you want to expose the Kubernetes service externally.
iaprestService/targetPort	Specify the port on which the AP-REST container is running inside the Docker container. **Important:** Do not change this value.
routes/prodService/hostname	Specify the hostname of the production service.
routes/prodService/annotations/haproxy.router.openshift.io/balance	Specify the algorithm for the load balancer. By default, the value is set to *roundrobin*.
routes/stagingService/hostname	Specify the hostname of the staging service.
routes/stagingService/annotations/haproxy.router.openshift.io/balance	Specify the algorithm for the load balancer. By default, the value is set to *roundrobin*.

3. Run the following command to upgrade the Helm charts.

   ```
   helm upgrade <Release Name> --namespace <Namespace where you want to deploy the
   AP-REST container > <Location of the directory that contains the Helm charts>
   ```

   For example:

   ```
   helm upgrade ap-rest --namespace iap-rest iap-rest
   ```

   Using this configuration ensures that the AP-REST container is deployed with the updated policy snapshot on the staging environment.

4. Verify whether the updated configuration is working accurately by running security operations on the staging environment.

   For more information about running security operations, refer to the *Running Security Operations* section.

After you have verified that the updated configuration is working accurately, you can switch the production and staging environments so that all the production traffic is directed to the AP-REST container with the updated configuration.

5. Modify the *values.yaml* file to change the value of the *deploymentInUse* field to *green*.

   This ensures that the *Green* deployment is now considered as the production environment, and the updated configuration handles all the production traffic from the customer application.
6. Run the following command to upgrade the Helm charts.

   ```
   helm upgrade <Release Name> --namespace <Namespace where you want to deploy the AP-REST container> <Location of the directory that contains the Helm charts>
   ```

   For example:

   ```
   helm upgrade ap-rest --namespace iap-rest iap-rest
   ```

   Using this configuration ensures that the AP-REST container is deployed with the updated policy snapshot on the staging environment.

   After the update configuration is working accurately on the new production environment, you can shutdown the pods that are running the Release 1 containers.
7. Modify the *values.yaml* file to change the value of the *blueDeployment/enabled* field to *false*.

   This will shut down all the pods that were deployed as part of the *Blue* deployment.

   > **Note:**
   >
   > If you do not change the value of this parameter to *false* and upgrade the Helm charts, then the pods in the staging environment will keep on consuming resources.
8. Run the following command to upgrade the Helm charts.

   ```
   helm upgrade <Release Name> ---namespace <Namespace where you want to deploy the AP-REST container> <Location of the directory that contains the Helm charts>
   ```

   For example:

   ```
   helm upgrade ap-rest --namespace iap-rest iap-rest
   ```

   Now, only the production environment is running with the updated configurations.

   If you want to modify any policy or subsequent releases, you need to repeat steps 1 to 8. The only difference is that you need to toggle the value of the *deploymentInUse* field from *green* to *blue* and vice versa depending on which deployment contains your modified configurations.

2.7 Deploying the Containers without RBAC

This section describes the steps that you need to perform for deploying the Get ESA Policy, Init, and AP-REST containers without RBAC.

> **Note:** The procedures performed in this section require the user to have the *cluster-admin* role.

2.7.1 Applying the SCC

This section describes how to apply the SCC.

> **Note:** Protegrity recommends you to use this SCC. However, you can choose to add additional security configurations in the SCC based on your requirements.

> **Important:** You require a *cluster-admin* role to perform this task.

➤ To apply the SCC:

1. On the Linux instance, run the following command to create project for deploying the containers.

   ```
   oc new-project <Namespace>
   ```

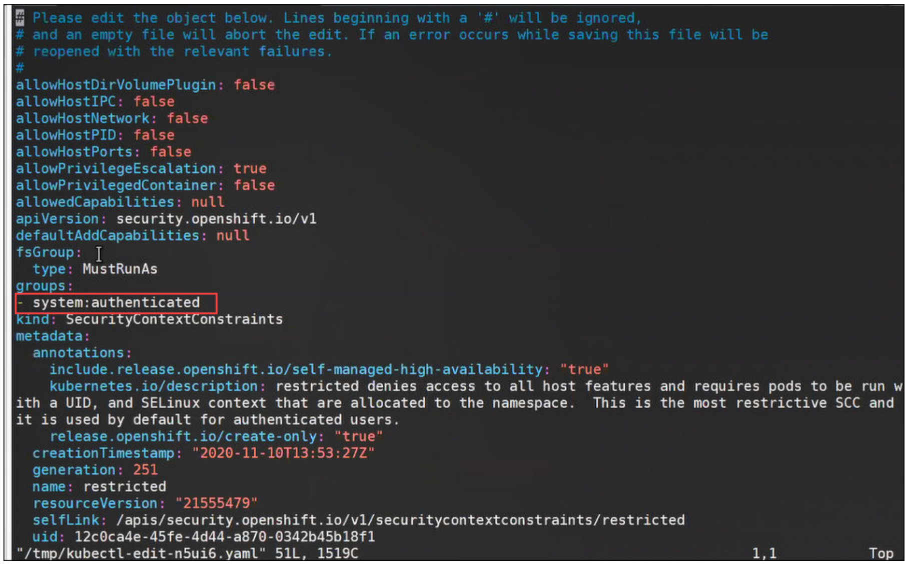
   For example:

   ```
   oc new-project iap-rest
   ```

2. Run the following command to edit the Restricted SCC, which is the default SCC that is applied to all the pods in the Kubernetes cluster.

   ```
   oc edit scc restricted
   ```

   The Restricted SCC opens in an editor.

   ```
   # Please edit the object below. Lines beginning with a '#' will be ignored,
   # and an empty file will abort the edit. If an error occurs while saving this file will be
   # reopened with the relevant failures.
   #
   allowHostDirVolumePlugin: false
   allowHostIPC: false
   allowHostNetwork: false
   allowHostPID: false
   allowHostPorts: false
   allowPrivilegeEscalation: true
   allowPrivilegedContainer: false
   allowedCapabilities: null
   apiVersion: security.openshift.io/v1
   defaultAddCapabilities: null
   fsGroup:
     type: MustRunAs
   groups:
   - system:authenticated
   kind: SecurityContextConstraints
   metadata:
     annotations:
       include.release.openshift.io/self-managed-high-availability: "true"
       kubernetes.io/description: restricted denies access to all host features and requires pods to be run w
   ith a UID, and SELinux context that are allocated to the namespace.  This is the most restrictive SCC and
   it is used by default for authenticated users.
       release.openshift.io/create-only: "true"
     creationTimestamp: "2020-11-10T13:53:27Z"
     generation: 251
     name: restricted
     resourceVersion: "21555479"
     selfLink: /apis/security.openshift.io/v1/securitycontextconstraints/restricted
     uid: 12c0ca4e-45fe-4d44-a870-0342b45b18f1
   "/tmp/kubectl-edit-n5ui6.yaml" 51L, 1519C                                                 1,1           Top
   ```

3. Delete the following line from the Restricted SCC and then save the file to ensure that the Restricted SCC is not applied to the namespace where the AP-REST containers will be deployed.

   ```
   - system:authenticated
   ```

4. Run the following command to ensure that the Restricted SCC is reapplied to the system-level pods in the Kubernetes cluster.

   ```
   oc get ns | awk '/openshift/{system("oc adm policy add-scc-to-group restricted
   system:serviceaccounts:" $1)}'
   ```

This command ensures that the Restricted SCC is applied to OpenShift infrastructure pods.

> **Important:** If you want to apply the Restricted SCC to any other namespace, then you need to run the following command:
>
> ```
> oc adm policy add-scc-to-group restricted system:serviceaccounts:<Namespace>
> ```

5. Navigate to the directory where you have extracted the Helm charts for deploying the Get ESA Policy application, and edit the *pty-scc.yaml* file to replace the *<NAMESPACE>* placeholder in the following snippet with the namespace where the IAP-REST application will be deployed.

   ```
   groups:
   - system:serviceaccounts:<NAMESPACE>
   ```

   For example:

   ```
   groups:
   - system:serviceaccounts:iap-rest
   ```

   For more information about the extracted Helm charts, refer to the *step 5* of the section *Initializing the Linux Instance*.

6. Run the following command to apply the Protegrity SCC to the Kubernetes cluster.

   ```
   oc apply -f pty-scc.yaml
   ```

   For example:

   ```
   oc apply -f pty-scc.yaml
   ```

2.7.2 Deploying the Get ESA Policy Container on the Kubernetes Cluster

This section describes how to deploy the Get ESA Policy container on the Kubernetes cluster by installing the Helm charts.

➤ To deploy the Get ESA Policy container on the Kubernetes cluster:

1. On the Linux instance, navigate to the location where you have extracted the Helm charts for deploying the Get ESA Policy application.
2. Modify the *nfs-pv.yaml* file to update the persistent volume claim details, by specifying the value of the path and server parameters.

   ```
   apiVersion: v1
   kind: PersistentVolume
   metadata:
     name: nfs-pv
     labels:
       purpose: policy-store
   spec:
     capacity:
       # The amount of storage allocated to this volume.
       # e.g. 10Gi
       storage: AMOUNT_OF_STORAGE
     accessModes:
       - ReadWriteMany
     persistentVolumeReclaimPolicy: Retain
     nfs:
       # The path that is exported by the NFS server
       path: MOUNTH_PATH
       # The host name or IP address of the NFS server.
   ```

```
    server: SERVER_NAME_OR_IP
    readOnly: false
```

3. Run the following command to create a persistent volume.

   ```
   oc apply -f get-esa-policy/nfs-pv.yaml
   ```

4. Modify the *nfs-pvc.yaml* file to update the persistent volume claim details, by specifying the name of the persistent volume claim and the storage amount.

   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
     name: nfs-pvc
   spec:
     storageClassName: ""
     selector:
       matchLabels:
         purpose: "policy-store"
     accessModes:
     - ReadWriteMany
     resources:
        requests:
           # The amount of storage allocated to this volume.
           # e.g. 10Gi
          storage: AMOUNT_OF_STORAGE
   ```

5. Run the following command to create a persistent volume claim.

   ```
   oc apply -f get-esa-policy/nfs-pvc.yaml -n iap-rest
   ```

6. On the Linux instance, create a mount point for the NFS by running the following command.

   ```
   mkdir /nfs
   ```

   This command creates a mount point *nfs* on the file system.

7. Run the following mount command to mount the NFS on the directory created in *step 6*.

   ```
   sudo mount <nfs server IP>:<Path to the shared folder> /nfs
   ```

8. Run the following command to create the *esa-creds* secret, which is used by the Get ESA Policy container to access the ESA using the non-administrator credentials.

   ```
   oc create secret generic esa-creds --from-literal=esa_user=<ESA user> --from-literal= esa_pass=<ESA user password> --namespace <NAMESPACE>

   oc create secret generic esa-creds --from-literal=esa_user=non-admin --from-literal= esa_pass=<ESA user password> --namespace iap-rest
   ```

   > **Important:** Ensure that the user, who will be used to access the ESA, has been assigned the *Export Certificates* and *Appliance CLI Viewer* permissions through a custom role. Roles are templates that include permissions and users can be assigned one or more roles.
   >
   > For more information about managing roles, refer to the section *Managing Roles* in the *Appliances Overview 9.1.0.0* guide.
   >
   > For more information about managing users, refer to the section *Managing Users* in the *Appliances Overview 9.1.0.0* guide.

9. Run the following command to create a Kubernetes secret for accessing the image registry using existing Docker credentials.

   ```
   oc create secret generic regcred --from-file=.dockerconfigjson=<PATH_TO_DOCKER_CONFIG>/config.json --type=kubernetes.io/dockerconfigjson --namespace <NAMESPACE>

   oc create secret generic regcred --from-file=.dockerconfigjson=<PATH_TO_DOCKER_CONFIG>/config.json --type=kubernetes.io/dockerconfigjson --namespace iap-rest
   ```

   For more information about the credentials required to access the image repository, refer to the section *Image Pull Secrets*.

10. Run the following command to create a passphrase secret, which is used by the Get ESA Policy container to encrypt the policy.

    ```
    oc create secret generic pbe-secret --from-literal='passphrase=<passphrase>' --from-literal='salt=<salt>' -n iap-rest
    ```

    This command is used to implement the Password-Based Encryption (PBE) for encrypting the policy.

    You need to provide a passphrase and a salt as a Kubernetes secret to the Get ESA Policy Container. The policy is encrypted using the key generated by using the passphrase and salt. The salt is used as an initialization vector for generating the key.

    > **Important:** Ensure that the password meets the following requirements:
    >
    > - The password must contain a minimum of 10 characters and a maximum of 128 characters.
    > - The user should be able to specify Unicode characters, such as Emoji, in the password.
    > - The user should avoid commonly used passwords, such as, *123456789*, *11111111*, *password*, and *qwertyuiop*.

11. Modify the default values in the *values.yaml* file as required.

    The *values.yaml* file contains the default configuration values for deploying the Get ESA Policy application on the Kubernetes cluster.

```
...
..
.

## pod service account to be used
## leave the field empty if not applicable
serviceAccount:

## set the Pod's security context object
## leave the field empty if not applicable
podSecurityContext:
  runAsUser: 1000
  runAsGroup: 1000
  fsGroup: 1000
  supplementalGroups: [1000]

....
....
....

## k8s secret for storing ESA credentials
esaCredSecrets: ESA_CREDENTIAL_SECRET_NAME

## k8s secret containing passphrase for encrypting policy
pbeSecrets: PBE_SECRET_NAME

securityConfigDestination: VolumeMount

## If securityConfigDestination is Volume Mount,
## specify the name of persistent volume claim to be used for this deployment.
pvcName: PVC_NAME

policy:
  esaIP: ESA_IP

  ## pepserver configurations that can be changed.
    ## emptystring: Set the output behavior for empty strings
      ## null = (default) null value is returned for empty input string
      ## empty = empty value is returned for empty input string
      ## encrypt = encrypted values is returned for empty input string
    ## logLevel: Specifies the logging level for pepserver
      ## OFF (no logging)
      ## SEVERE
      ## WARNING
      ## INFO
      ## CONFIG
```

```
      ## ALL (default)
    ## caseSensitive: Specifies how policy users are checked against policy
      ## yes = (The default) policy users are treated in case sensitive manner
      ## no = policy users are treated in case insensitive manner.
  pepserverConfig:
    emptyString: "null"
    logLevel: "ALL"
    caseSensitive: "yes"
  ## absolute path in PV where the policy dump file will be stored. <eg. /test/xyz/policy
>
  filePath: ABSOLUTE_POLICY_PATH

  ## time (in mins) to wait till the policies get downloaded
  ## default is 15 (mins)
  timeout: 15
```

For more information about Helm charts and their default values, refer to the *Appendix A: Get ESA Policy Helm Chart Details*.

Field	Description
podSecurityContext	Specify the privilege and access control settings for the pod. The default values are set as follows: - *runAsUser* - 1000 - *runAsGroup* - 1000 - *fsGroup* - 1000 - *supplementalGroups* - 1000 **Note:** Set the value of the *fsGroup* parameter to a non-root value. For example, you can set the value of the *fsGroup* parameter to any value between *1* and *65534*. **Note:** Ensure that at any given point of time, the value of at least one of the four parameters must be *1000*. This ensures that the pod has the required permissions to access the *pty* directories in the containers. For more information about the Pod Security Context, refer to the section *Configure a Security Context for a Pod or Container* in the Kubernetes documentation. For more information about the parameters, refer to the section *PodSecurityContext v1 Core* in the Kubernetes API documentation.
esaCredSecrets	Specify the value of the secret that is used to store the ESA credentials. This is used by the Get ESA Policy container to retrieve the policy from the ESA.
pbeSecrets	Specify the value of the secret that is used to store the passphrase and salt. The Get ESA Policy container encrypts the policy using the key generated by using the passphrase and salt. **Important:** Ensure that the password meets the following requirements:

Field	Description
	• The password must contain a minimum of 10 characters and a maximum of 128 characters. • The user should be able to specify Unicode characters, such as Emoji, in the password. • The user should avoid commonly used passwords, such as, *123456789*, *11111111*, *password*, and *qwertyuiop*.
securityConfigDestination	Specify the value as *VolumeMount*.
pvcName	Specify the name of the Persistent Volume Claim where you want to store the policy snapshot.
policy/ESA_IP	Specify the IP address of the ESA from where you want to download the policy.
policy/pepserverConfig/emptyString	Defines the behavior when the data to protect is an empty string. The following are the possible values: • *empty* - An empty value is returned for an empty input string. • *encrypt* - An encrypted value is returned for an empty input string. • *null* - A null value is returned for an empty input string. By default, the value of this parameter is set to *null*. For more information about empty string handling by protectors, refer to the section *Appendix C: Empty String Handling by Protectors* in the *Protection Methods Reference Guide 9.1.0.0*.
policy/pepserverConfig/logLevel	Specifies the logging level for the PEP server. You can specify one of the following values: • *OFF* - No logging • *SEVERE* • *WARNING* • *INFO* • *CONFIG* • *ALL* By default, the value of this parameter is set to *ALL*.
policy/pepserverConfig/caseSensitive	Specifies how policy users are checked against the policy. This parameter enables you to configure the case-sensitivity of the policy users. If this parameter is set to *no*, then the PEP server considers the policy user names that are case insensitive. If this parameter is set to *yes* or if it is commented in the file, then the PEP server considers the policy user names that are case-sensitive. The default value is *yes*. By default, the value of this parameter is set to *yes*.
policy/filePath	Specify the path in the persistent volume where you want to store the immutable policy package.

Field	Description
	For example, if you are using NFS as the persistent volume, then you can specify the file path as */<Mount directory>/xyz/imp*. In this case, a policy with the name as *imp* will be stored in the */<Mount direc tory>/xyz* directory in the required persistent volume. **Note:** The *<Mount directory>* is a directory inside the Mount point. **Important:** Use only uppercase and lowercase letters, numbers, dot, and underscore in the file path. Do not use special characters in the file path. **Note:** This value is case-sensitive.
policy/timeout	Specify the time for which the Get ESA Policy container tries to communicate with the ESA for fetching the policies, after which the connection times out. By default, this value is set to 15. The unit of the timeout parameter is minutes.

12. Run the following command to deploy the Get ESA Policy application on the Kubernetes cluster.

    ```
    helm install <Release Name> --namespace <Namespace where you want to deploy the Get
    ESA Policy application> <Location of the directory that contains the Helm charts>
    ```

    For example:

    ```
    helm install get-policy-esa --namespace iap-rest get-esa-policy
    ```

13. Run the following command to check the deployment status.

    ```
    oc get pods -n iap-rest
    ```

    After the Get ESA Policy container is up, it retrieves the policy from the ESA and uploads it to the persistent volume specified in the *values.yaml* file. After uploading the policy snapshot, the Kubernetes job is completed.

2.7.3 Verifying the AP-REST Policy Snapshot on the NFS

This section describes how you can verify the AP-REST policy snapshot on the NFS.

The Get ESA Policy container retrieves the policy from the ESA and uploads a snapshot of this immutable policy to the NFS.

➤ To verify the policy snapshot on the NFS:

Perform the following tasks to verify whether the policy package file has been uploaded to the NFS:

1. On the Linux instance, navigate to the location where you have mounted the NFS.

For more information about the location where you have mounted the NFS, refer to *step 6* in the section *Deploying the Get ESA Policy Container on the Kubernetes Cluster*.

2. List all the files within the directory to access the policy snapshot.

2.7.4 Deploying the AP-REST Container on the Kubernetes Cluster

This section describes how to deploy the AP-REST container on the Kubernetes cluster by installing the Helm charts.

To deploy a release on the Kubernetes cluster:

1. Run the following command to create the *nginxCertsSecrets* secret, which stores the TLS certificates for the NGINX container.

   ```
   oc create -n iap-rest secret generic nginx-certificates --from-file=tls.crt=iap-wildcard.crt --from-file=tls.key=iap-wildcard.key --from-file=ca.crt=iap-ca.crt
   ```

   The TLS certificates are created using the procedure described in the section *Creating Certificates and Keys for TLS Authentication*.

2. On the Linux instance, navigate to the location where you have extracted the Helm charts to deploy the AP-REST container.

   For more information about the extracted Helm charts, refer to the *step 6* of the section *Initializing the Linux Instance*.

   The *iap-rest > values.yaml* file contains the default configuration values for deploying the AP_REST container on the Kubernetes cluster.

```
...
..
.

## pod service account to be used
## leave the field empty if not applicable
serviceAccount:


...
...


## set the Pod's security context object
## leave the field empty if not applicable
podSecurityContext:
  runAsUser: 1000
  runAsGroup: 1000
  fsGroup: 1000
  supplementalGroups: [1000]
...
...

# If Volume Mount, name of persistent volume claim to be used for this deployment.
pvcName: PVC_NAME

## k8s secret containing passphrase for encrypting policy
pbeSecrets: PBE_SECRET_NAME

## k8s secret containing TLS certificates for nginx sidecar
nginxCertsSecrets: NGINX_SECRET_NAME

## start with blue deployment first
deploymentInUse: blue

## Policy metadata for Blue Deployments
blueDeployment:
```

```
  enabled: true
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV where the policy dump file will be stored. <eg. /test/xyz/
policy >
    filePath: ABSOULTE_POLICY_PATH

## Policy metadata for Green Deployments
greenDeployment:
  enabled: false
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV where the policy dump file will be stored. <eg. /test/xyz/
policy >
    filePath: ABSOULTE_POLICY_PATH

## specify the initial no. of iaprest Pod replicas
replicaCount: 1

## HPA configuration
autoScaling:
  minReplicas: 1
  maxReplicas: 10
  targetCPU: 80

## specify the ports exposed in your iaprest configurations where,
## name - distinguishes between different ports.
## port - the port on which you wan't to expose the service externally.
## targetPort - the port no. configured while creating Tunnel.
iaprestService:
    name: "iaprest"
    port: 8443
    targetPort: 8443

## specify hostname, path and annotations for prod and staging routes in this section.
## staging routes will be enabled only if both blue & green deployments are enabled.
routes:
  prodService:
    hostName: "prod.example.com"
    annotations:
      haproxy.router.openshift.io/balance: roundrobin
      # specify route annotations if any
      # e.g. haproxy.router.openshift.io/ip_whitelist: 192.168.1.10

  stagingService:
    hostName: "staging.example.com"
    annotations:
      haproxy.router.openshift.io/balance: roundrobin
      # specify route annotations if any
      # e.g. haproxy.router.openshift.io/ip_whitelist: 192.168.1.10
```

In a *Blue-Green* deployment strategy, the Release 1 is considered as the *Blue* deployment.

3. Modify the default values in the *values.yaml* file as required.

   For more information about Helm charts and their default values, refer to the section *Appendix B: AP-REST Helm Chart Details*.

Field	Description
podSecurityContext	Specify the privilege and access control settings for the pod. The default values are set as follows: • *runAsUser* - 1000 • *runAsGroup* - 1000 • *fsGroup* - 1000

Field	Description
	- *supplementalGroups* - 1000 **Note:** Set the value of the *fsGroup* parameter to a non-root value. For example, you can set the value of the *fsGroup* parameter to any value between *1* and *65534*. **Note:** Ensure that at any given point of time, the value of at least one of the four parameters must be *1000*. This ensures that the pod has the required permissions to access the *pty* directories in the containers. For more information about the Pod Security Context, refer to the section *Configure a Security Context for a Pod or Container* in the Kubernetes documentation. For more information about the parameters, refer to the section *PodSecurityContext v1 Core* in the Kubernetes API documentation.
pvcName	Specify the name of the Persistent Volume Claim where you want to store the policy snapshot.
pbeSecrets	Specify the Kubernetes secret that contains the passphrase and the salt that were used to generate key, which encrypted the policy. The AP-REST container uses the value specified in the *pbeSecrets* parameter to decrypt the policy. **Important:** Ensure that the password meets the following requirements: - The password must contain a minimum of 10 characters and a maximum of 128 characters. - The user should be able to specify Unicode characters, such as Emoji, in the password. - The user should avoid commonly used passwords, such as, *123456789*, *11111111*, *password*, and *qwertyuiop*.
nginxCertsSecrets	Specify the Kubernetes secret that contains the TLS certificates for the NGINX container. The TLS certificates are created using the procedure described in the section *Creating Certificates and Keys for TLS Authentication*.
deploymentInUse	Specify the value as *blue*. This indicates that the *Blue* deployment strategy is in use.
blueDeployment/securityConfigSource	Specify the value as *VolumeMount*.
blueDeployment/policy/filePath	Specify the path in the persistent volume where you have stored the policy. For example, if you are using NFS as the persistent volume, then you can specify the file path as */<Mount directory>/xyz/imp*. In this

Field	Description
	case, a policy with the name as *imp* will be stored in the */<Mount directory>/xyz* directory in the required persistent volume. **Important:** Use only uppercase and lowercase letters, numbers, dot, and underscore in the file path. Do not use special characters in the file path. **Note:** This value is case-sensitive.
greenDeployment/enabled	Specify the value as *false*. While deploying Release 1, the *Green* deployment strategy is always disabled.
iaprestService/name	Specify a name for the tunnel to distinguish between ports.
iaprestService/port	Specify the port number on which you want to expose the Kubernetes service externally.
iaprestService/targetPort	Specify the port on which the AP-REST container is running inside the Docker container. **Important:** Do not change this value.
routes/prodService/hostname	Specify the hostname of the production service.
routes/prodService/annotations/haproxy.router.openshift.io/balance	Specify the algorithm for the load balancer. By default, the value is set to *roundrobin*.
routes/stagingService/hostname	Specify the hostname of the staging service.
routes/stagingService/annotations/haproxy.router.openshift.io/balance	Specify the algorithm for the load balancer. By default, the value is set to *roundrobin*.

4. Run the following command to deploy the AP-REST container on the Kubernetes cluster:

   ```
   helm install <Release Name> --namespace <Namespace where you want to deploy the AP-REST container> <Location of the directory that contains the Helm charts>
   ```

   For example:

   ```
   helm install ap-rest --namespace iap-rest iap-rest
   ```

5. Perform the following steps to check the status of the Kubernetes resources.

   a. Run the following command to check the status of the pods.

      ```
      oc get pods -n <namespace>
      ```

      For example:

      ```
      oc get pods -n iap-rest
      ```

2.7.5 Upgrading the Helm Charts

This section describes the steps for upgrading the Helm charts.

Protegrity recommends using the *Blue-Green* method of upgrading the Helm charts. In this method, you use two modes of deploying the application, production mode and staging mode. In the staging mode, you can test the changes to the application in a staging environment, without impacting the production environment. After you have tested the application in the staging environment, you can switch the environments by making the staging environment as the new production environment and the production environment as the new staging environment.

In this way, you can seamlessly divert the customer traffic to the modified application without any production downtime. After your modified application is running on the new production environment, you can shutdown the application that is running on the staging environment. In the *Blue-Green* method of deploying applications, only one Release is active at a given point of time.

To upgrade the Helm charts:

1. On the Linux instance, navigate to the location where you have extracted the Helm charts to deploy the AP-REST container.

   For more information about the extracted Helm charts, refer to the step *6* of the section *Initializing the Linux Instance*.

   The *values.yaml* file contains the default configuration values for deploying the AP-REST container on the Kubernetes cluster.

```
...
..
.

## pod service account to be used
## leave the field empty if not applicable
serviceAccount:


...
...


## set the Pod's security context object
## leave the field empty if not applicable
podSecurityContext:
  runAsUser: 1000
  runAsGroup: 1000
  fsGroup: 1000
  supplementalGroups: [1000]

...
...

# If Volume Mount, name of persistent volume claim to be used for this deployment.
pvcName: PVC_NAME

## k8s secret containing passphrase for encrypting policy
pbeSecrets: PBE_SECRET_NAME

## k8s secret containing TLS certificates for nginx sidecar
nginxCertsSecrets: NGINX_SECRET_NAME

## start with blue deployment first
deploymentInUse: blue

## Policy metadata for Blue Deployments
blueDeployment:
  enabled: true
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV where the policy dump file will be stored. <eg. /test/xyz/
policy >
    filePath: ABSOULTE_POLICY_PATH

## Policy metadata for Green Deployments
greenDeployment:
  enabled: false
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV where the policy dump file will be stored. <eg. /test/xyz/
policy >
    filePath: ABSOULTE_POLICY_PATH
```

```
## specify the initial no. of iaprest Pod replicas
replicaCount: 1

## HPA configuration
autoScaling:
  minReplicas: 1
  maxReplicas: 10
  targetCPU: 80

## specify the ports exposed in your iaprest configurations where,
## name - distinguishes between different ports.
## port - the port on which you wan't to expose the service externally.
## targetPort - the port no. configured while creating Tunnel.
iaprestService:
    name: "iaprest"
    port: 8443
    targetPort: 8443

## specify hostname, path and annotations for prod and staging routes in this section.
## staging routes will be enabled only if both blue & green deployments are enabled.
routes:
  prodService:
    hostName: "prod.example.com"
    annotations:
      haproxy.router.openshift.io/balance: roundrobin
      # specify route annotations if any
      # e.g. haproxy.router.openshift.io/ip_whitelist: 192.168.1.10

  stagingService:
    hostName: "staging.example.com"
    annotations:
      haproxy.router.openshift.io/balance: roundrobin
      # specify route annotations if any
      # e.g. haproxy.router.openshift.io/ip_whitelist: 192.168.1.10
```

In a *Blue-Green* deployment strategy, the Release 2 is considered as the *Green* deployment.

2. Modify the default values in the *values.yaml* file as required.

   For more information about Helm charts and their default values, refer to the *Appendix: Helm Chart Details*.

Field	Description
podSecurityContext	Specify the privilege and access control settings for the pod. The default values are set as follows: • *runAsUser* - 1000 • *runAsGroup* - 1000 • *fsGroup* - 1000 • *supplementalGroups* - 1000 **Note:** Set the value of the *fsGroup* parameter to a non-root value. For example, you can set the value of the *fsGroup* parameter to any value between *1* and *65534*. **Note:** Ensure that at any given point of time, the value of at least one of the four parameters must be *1000*.

Field	Description
	This ensures that the pod has the required permissions to access the *pty* directories in the containers. For more information about the Pod Security Context, refer to the section *Configure a Security Context for a Pod or Container* in the Kubernetes documentation. For more information about the parameters, refer to the section *PodSecurityContext v1 Core* in the Kubernetes API documentation.
pvcName	Specify the name of the Persistent Volume Claim where you want to store the policy snapshot.
pbeSecrets	Specify the Kubernetes secret that contains the passphrase and the salt that were used to generate key, which encrypted the policy. The AP-REST container uses the value specified in the *pbeSecrets* parameter to decrypt the policy. **Important:** Ensure that the password meets the following requirements: • The password must contain a minimum of 10 characters and a maximum of 128 characters. • The user should be able to specify Unicode characters, such as Emoji, in the password. • The user should avoid commonly used passwords, such as, *123456789*, *11111111*, *password*, and *qwertyuiop*.
nginxCertsSecrets	Specify the Kubernetes secret that contains the TLS certificates for the NGINX container. The TLS certificates are created using the procedure described in the section *Creating Certificates and Keys for TLS Authentication*.
deploymentInUse	Specify the value as *blue*. This indicates that the *Blue* deployment strategy is in use.
greenDeployment\enabled	Specify the value as *true*. While upgrading the release to Release 2, the *Green* deployment strategy is enabled.
greenDeployment/securityConfigSource	Specify the value as *VolumeMount*. By default, the value is set to *VolumeMount*.
greenDeployment/policy/filePath	Specify the path in the persistent volume where you have stored the policy. For example, if you are using NFS as the persistent volume, then you can specify the file path as */<Mount directory>/xyz/imp*. In this

Field	Description
	case, a policy with the name as *imp* will be stored in the */<Mount directory>/xyz* directory in the required persistent volume. **Important:** Use only uppercase and lowercase letters, numbers, dot, and underscore in the file path. Do not use special characters in the file path. **Note:** This value is case-sensitive.
iaprestService/name	Specify a name for the tunnel to distinguish between ports.
iaprestService/port	Specify the port number on which you want to expose the Kubernetes service externally.
iaprestService/targetPort	Specify the port on which the AP-REST container is running inside the Docker container. **Important:** Do not change this value.
routes/prodService/hostname	Specify the hostname of the production service.
routes/prodService/annotations/haproxy.router.openshift.io/balance	Specify the algorithm for the load balancer. By default, the value is set to *roundrobin*.
routes/stagingService/hostname	Specify the hostname of the staging service.
routes/stagingService/annotations/haproxy.router.openshift.io/balance	Specify the algorithm for the load balancer. By default, the value is set to *roundrobin*.

3. Run the following command to upgrade the Helm charts.

   ```
   helm upgrade <Release Name> --namespace <Namespace where you want to deploy the
   AP-REST container > <Location of the directory that contains the Helm charts>
   ```

   For example:

   ```
   helm upgrade ap-rest --namespace iap-rest iap-rest
   ```

   Using this configuration ensures that the AP-REST container is deployed with the updated policy snapshot on the staging environment.

4. Verify whether the updated configuration is working accurately by running security operations on the staging environment.

   For more information about running security operations, refer to the *Running Security Operations* section.

   After you have verified that the updated configuration is working accurately, you can switch the production and staging environments so that all the production traffic is directed to the AP-REST container with the updated configuration.

5. Modify the *values.yaml* file to change the value of the *deploymentInUse* field to *green*.

   This ensures that the *Green* deployment is now considered as the production environment, and the updated configuration handles all the production traffic from the customer application.

6. Run the following command to upgrade the Helm charts.

   ```
   helm upgrade <Release Name> --namespace <Namespace where you want to deploy the
   AP-REST container> <Location of the directory that contains the Helm charts>
   ```

   For example:

   ```
   helm upgrade ap-rest --namespace iap-rest iap-rest
   ```

Using this configuration ensures that the AP-REST container is deployed with the updated policy snapshot on the staging environment.

After the update configuration is working accurately on the new production environment, you can shutdown the pods that are running the Release 1 containers.

7. Modify the *values.yaml* file to change the value of the *blueDeployment/enabled* field to *false*.

   This will shut down all the pods that were deployed as part of the *Blue* deployment.

   > **Note:**
   >
   > If you do not change the value of this parameter to *false* and upgrade the Helm charts, then the pods in the staging environment will keep on consuming resources.

8. Run the following command to upgrade the Helm charts.

   ```
   helm upgrade <Release Name> ---namespace <Namespace where you want to deploy the AP-REST container> <Location of the directory that contains the Helm charts>
   ```

   For example:

   ```
   helm upgrade ap-rest --namespace iap-rest iap-rest
   ```

   Now, only the production environment is running with the updated configurations.

   If you want to modify any policy or subsequent releases, you need to repeat steps 1 to 8. The only difference is that you need to toggle the value of the *deploymentInUse* field from *green* to *blue* and vice versa depending on which deployment contains your modified configurations.

Chapter 3

Accessing Logs Using Splunk

This section describes how to access the Application Protector and Container logs using a third-party tool, such as Splunk, which is used as an example to process the logs from the IAP deployment.

Important: Ensure that you have a valid license for using Splunk.

3.1 Understanding the Logging Architecture

This section describes the sample architecture that is used to access the logs that are generated on the Kubernetes cluster.

The following figure represents a sample workflow that is used for accessing the Application Protector and Container logs. In this workflow, a third-party tool, such as Splunk, is used to view the generated logs.

For more information about Splunk, refer to the *Splunk documentation* website.

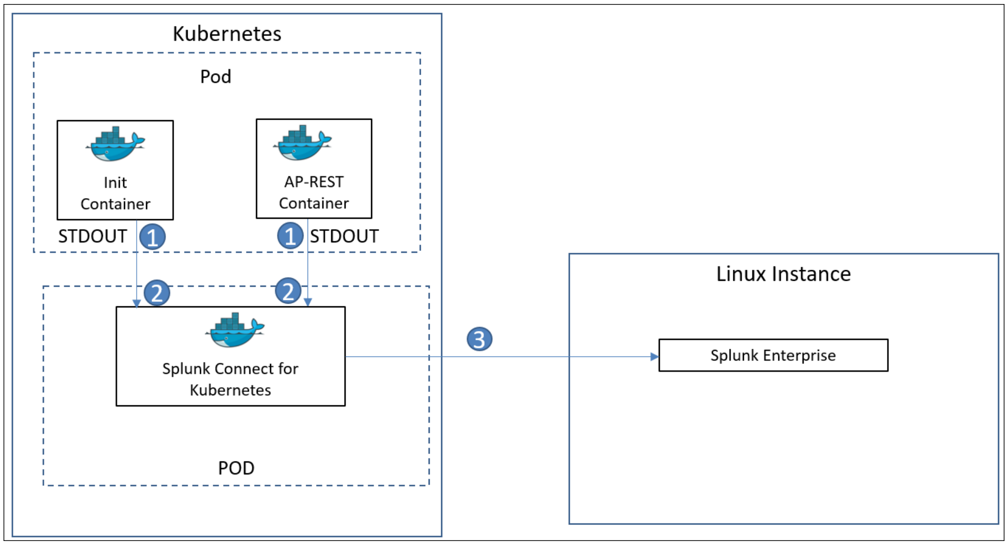


Figure 3-1: Sample Logging Workflow

The following steps describe the workflow of accessing the Application Protector and Container logs.

1. The Init and AP-REST container write the logs to the Standard Output (STDOUT) stream.
2. The Splunk Connect for Kubernetes is used to collect the logs from the STDOUT stream.

   For more information about Splunk Connect for Kubernetes, refer to the *Splunk Connect for Kubernetes* page on Github.

3. The Splunk Connect for Kubernetes then forwards the logs to the Splunk Enterprise, which is used to view the logs.

3.2 Setting Up Splunk

This section describes how you can set up Splunk for accessing the Application Protector and Container logs.

To set up Splunk:

1. Create a Linux instance, either in an on-premise or on a Cloud environment.
2. Install the latest version of Splunk Enterprise on the Linux instance.

   By default, Splunk is installed in the */opt/splunk* directory.

   For more information about installing Splunk Enterprise on a Linux instance, refer to the section *Install Splunk Enterprise* in the Splunk documentation.

3. Navigate to the */opt/splunk/bin* directory and start Splunk using the following command.

   ```
   ./splunk start --accept license
   ```

   You are prompted to create a username that will be used to login to Splunk Enterprise.

   For more information about starting Splunk Enterprise on Linux, refer to the section *Start Splunk Enterprise on Linux* in the Splunk documentation.

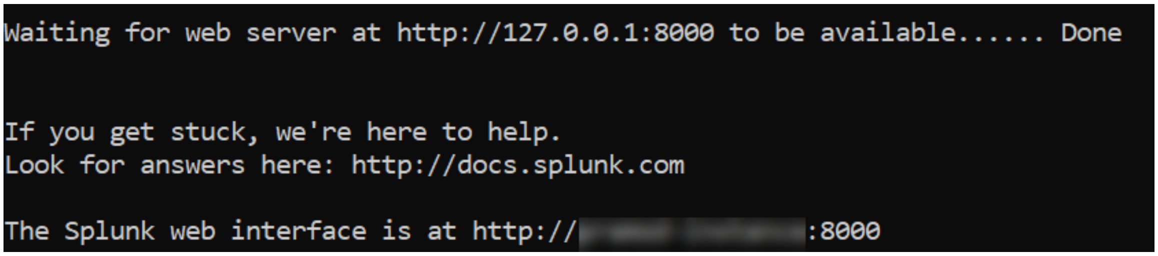
4. Type the username for the administrator account for logging into Splunk Enterprise, and then press **Enter**.

   You are prompted to create a password for the created user.

   > **Note:** If you press **Enter** without specifying any username, then *admin* is used as the default username.

5. Type a password for the user that you have created in *step 4*, and then press **Enter**.

   The following message appears on the console.



```
Waiting for web server at http://127.0.0.1:8000 to be available...... Done


If you get stuck, we're here to help.
Look for answers here: http://docs.splunk.com

The Splunk web interface is at http://[redacted]:8000
```

   By default, you can access the web interface of Splunk Enterprise from the port number *8000* of your Linux instance.

   For example, if the IP address of your Linux instance is *10.xx.xx.xx*, then you can access Splunk Web at *http://10.xx.xx.xx:8000*.

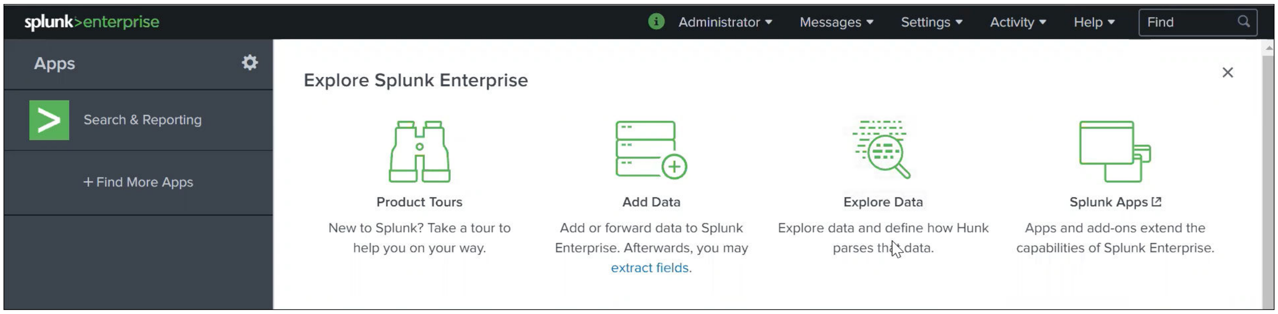
3.3 Configuring Splunk

This section describes how you can configure Splunk Enterprise for accessing the Sample Application and Container logs.

To configure Splunk Enterprise:

1. Login to Splunk Web UI at *http://<IP of Linux instance>:8000*.
2. On the Splunk Web UI, type the username and password that you have created while installing Splunk Enterprise on the Linux instance.

   The Splunk Enterprise screen appears.



Explore Splunk Enterprise

Product Tours
New to Splunk? Take a tour to help you on your way.

Add Data
Add or forward data to Splunk Enterprise. Afterwards, you may extract fields.

Explore Data
Explore data and define how Hunk parses the data.

Splunk Apps
Apps and add-ons extend the capabilities of Splunk Enterprise.

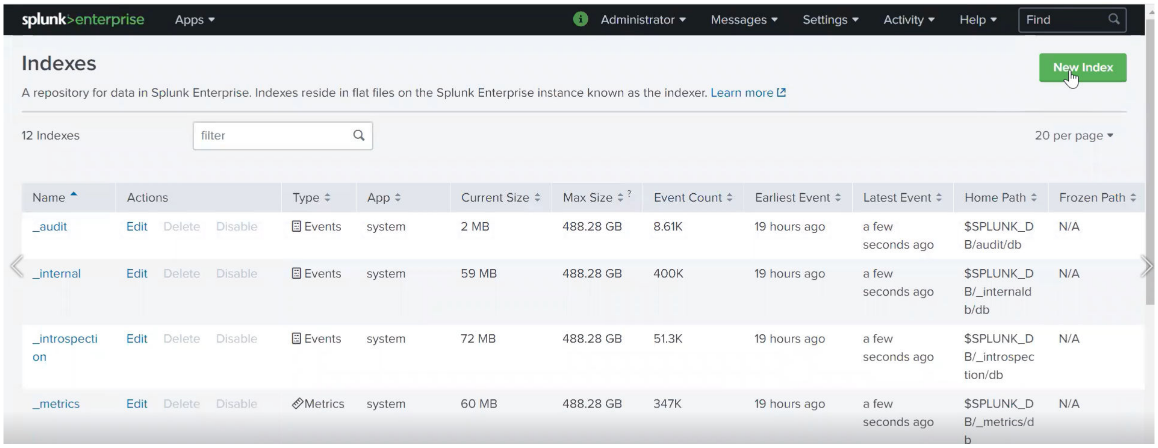
Figure 3-2: Splunk Enterprise Web UI

For more information about the user credentials, refer to the section *Setting Up Splunk*.

3. Perform the following steps to create an index, which is a repository for storing the Splunk Enterprise data.

   a. Navigate to **Settings** > **Index**.

   The **Indexes** screen appears.



Indexes

A repository for data in Splunk Enterprise. Indexes reside in flat files on the Splunk Enterprise instance known as the indexer. Learn more

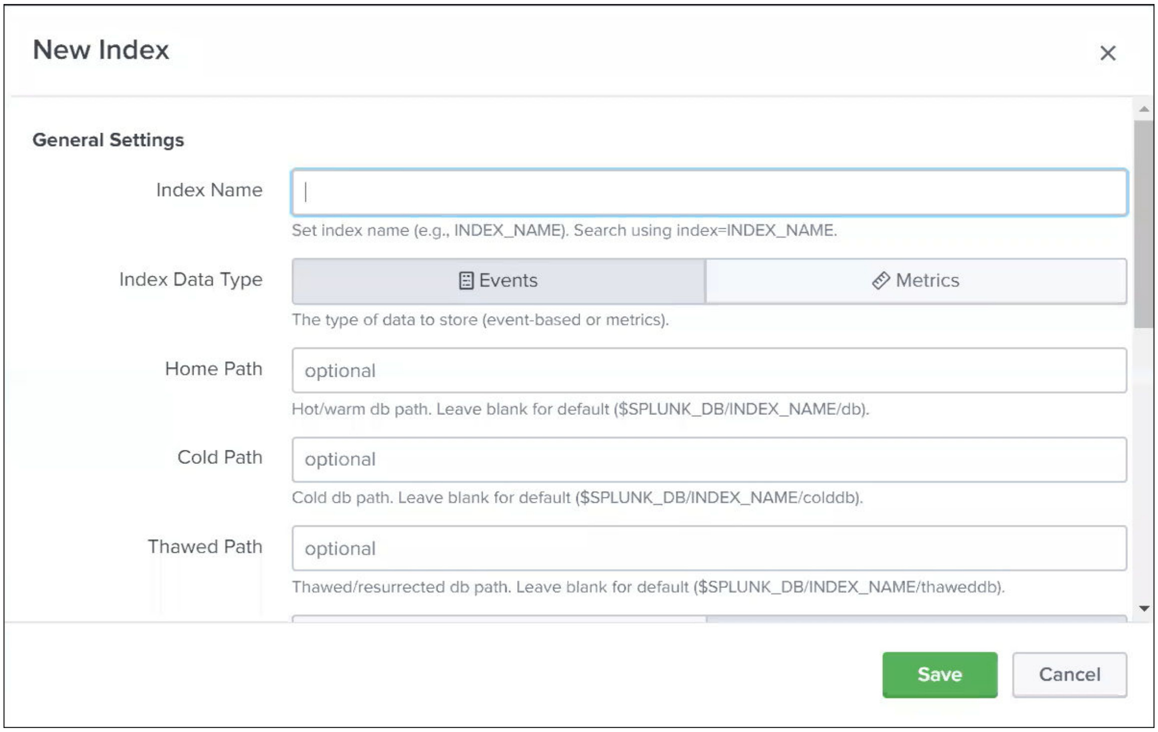
12 Indexes

Name	Actions	Type	App	Current Size	Max Size	Event Count	Earliest Event	Latest Event	Home Path	Frozen Path
_audit	Edit Delete Disable	Events	system	2 MB	488.28 GB	8.61K	19 hours ago	a few seconds ago	$SPLUNK_DB/audit/db	N/A
_internal	Edit Delete Disable	Events	system	59 MB	488.28 GB	400K	19 hours ago	a few seconds ago	$SPLUNK_DB/_internaldb/db	N/A
_introspection	Edit Delete Disable	Events	system	72 MB	488.28 GB	51.3K	19 hours ago	a few seconds ago	$SPLUNK_DB/_introspection/db	N/A
_metrics	Edit Delete Disable	Metrics	system	60 MB	488.28 GB	347K	19 hours ago	a few seconds ago	$SPLUNK_DB/_metrics/db	N/A

Figure 3-3: Indexes Screen

   b. Click **New Index**.

   The **New Index** dialog box appears.



New Index

General Settings

Index Name: |

Set index name (e.g., INDEX_NAME). Search using index=INDEX_NAME.

Index Data Type: Events | Metrics

The type of data to store (event-based or metrics).

Home Path: optional

Hot/warm db path. Leave blank for default ($SPLUNK_DB/INDEX_NAME/db).

Cold Path: optional

Cold db path. Leave blank for default ($SPLUNK_DB/INDEX_NAME/colddb).

Thawed Path: optional

Thawed/resurrected db path. Leave blank for default ($SPLUNK_DB/INDEX_NAME/thaweddb).

Figure 3-4: New Index Dialog Box

c. In the **General Settings** > **Index Name** field, type a name for the index that you want to create.

d. Click **Save**.

By default, an index with an index data type of *events* is created. This events index is used to store the Sample Application and Container logs.

For more information about indexes used in Splunk Enterprise, refer to the section *About managing indexes* in the Splunk documentation.

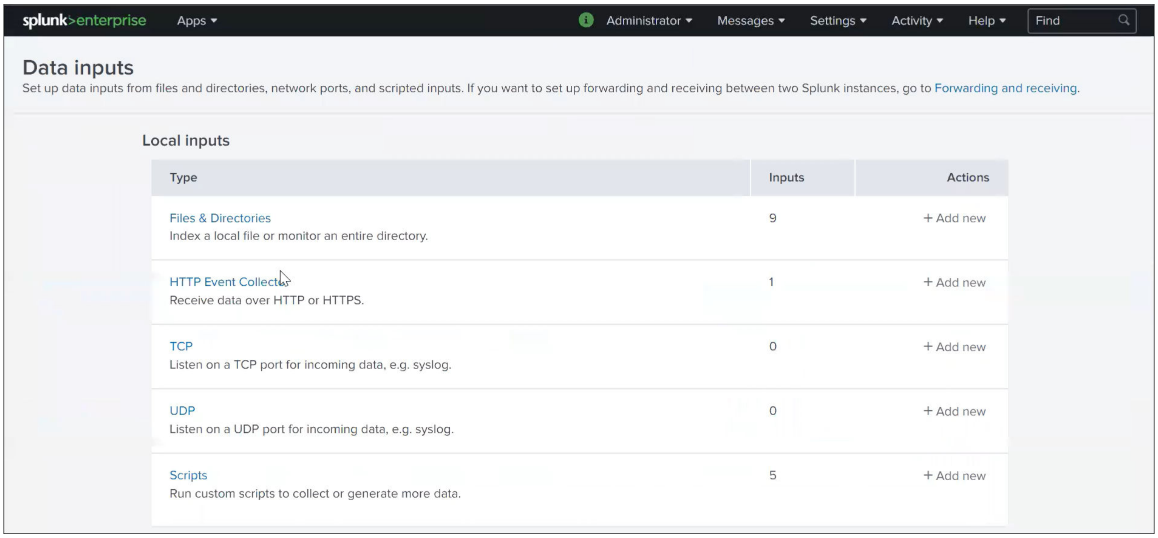
For more information about creating an index in Splunk Enterprise, refer to the section *Create custom indexes* in the Splunk documentation.

4. Perform the following steps to set up HTTP Event Collector (HEC) in Splunk Web. The HEC enables you to receive the Sample Application and Container logs sent from Kubernetes.

   For more information about HEC, refer to the section *Set up and use HTTP Event Collector in Splunk Web* in the Splunk documentation.

   a. Navigate to **Settings** > **Data inputs**.

      The **Data inputs** screen appears.



Data inputs

Set up data inputs from files and directories, network ports, and scripted inputs. If you want to set up forwarding and receiving between two Splunk instances, go to Forwarding and receiving.

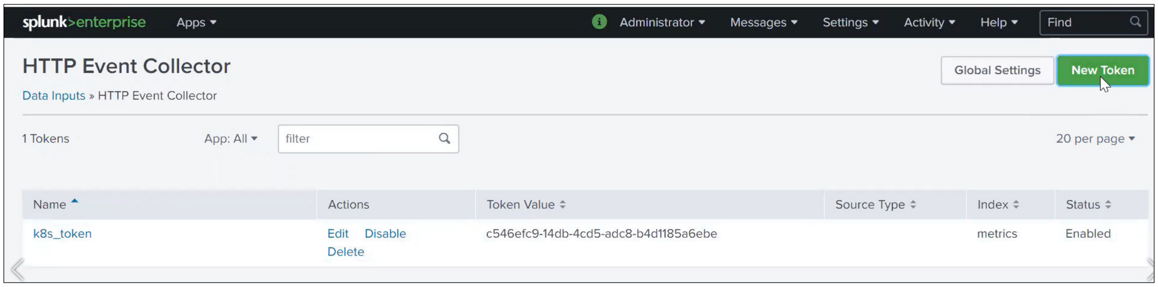
Local inputs

Type	Inputs	Actions
Files & Directories Index a local file or monitor an entire directory.	9	+ Add new
HTTP Event Collector Receive data over HTTP or HTTPS.	1	+ Add new
TCP Listen on a TCP port for incoming data, e.g. syslog.	0	+ Add new
UDP Listen on a UDP port for incoming data, e.g. syslog.	0	+ Add new
Scripts Run custom scripts to collect or generate more data.	5	+ Add new

Figure 3-5: Data inputs Screen

b. Click **HTTP Event Collector**.

The **HTTP Event Collector** screen appears.



HTTP Event Collector

Data Inputs » HTTP Event Collector

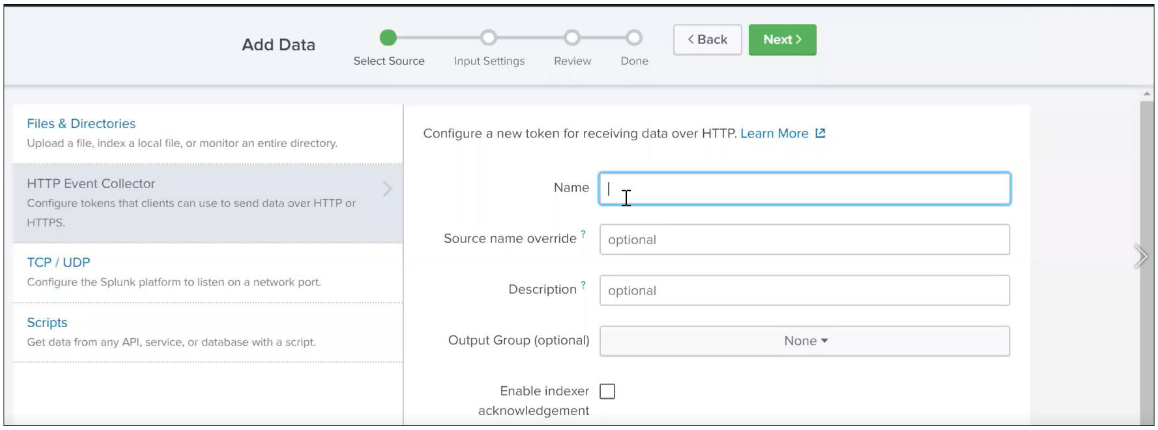
1 Tokens

Name	Actions	Token Value	Source Type	Index	Status
k8s_token	Edit Disable Delete	c546efc9-14db-4cd5-adc8-b4d1185a6ebe		metrics	Enabled

Figure 3-6: HTTP Event Collector Screen

c. Click **New Token**.

The **Add Data** > **Select Source** screen appears.



Files & Directories
Upload a file, index a local file, or monitor an entire directory.

HTTP Event Collector
Configure tokens that clients can use to send data over HTTP or HTTPS.

TCP / UDP
Configure the Splunk platform to listen on a network port.

Scripts
Get data from any API, service, or database with a script.

Configure a new token for receiving data over HTTP. Learn More

Name:

Source name override ?: optional

Description ?: optional

Output Group (optional): None ▾

Enable indexer acknowledgement □

Figure 3-7: Select Source Screen

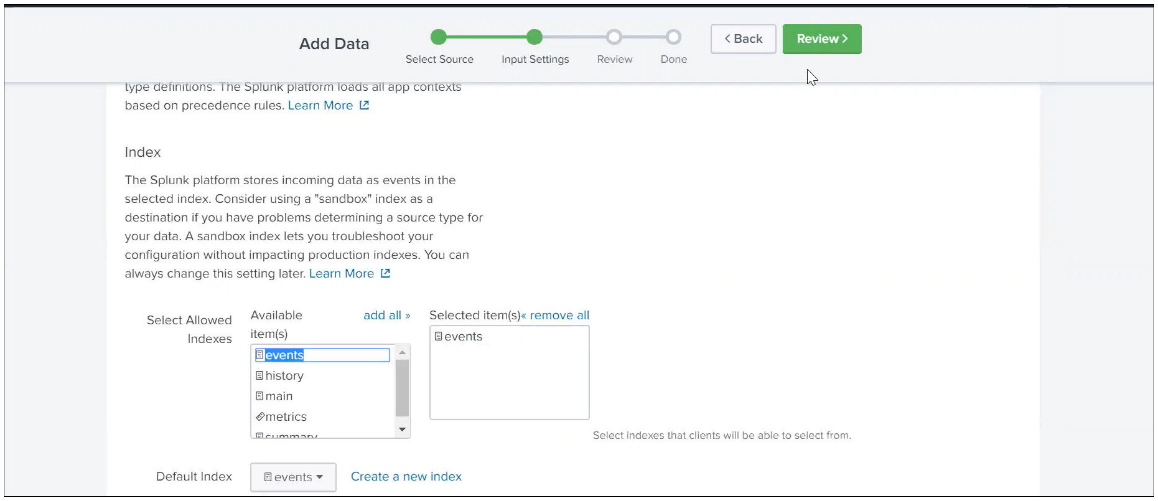
d. In the **Name** field, type a name for the token.

You need to create a token for receiving data over HTTPS.

For more information about HEC tokens, refer to the section *About Event Collector* tokens in the *Splunk documentation*.

e. Click **Next**.

The **Input Settings** screen appears.



type definitions. The Splunk platform loads all app contexts based on precedence rules. Learn More

Index

The Splunk platform stores incoming data as events in the selected index. Consider using a "sandbox" index as a destination if you have problems determining a source type for your data. A sandbox index lets you troubleshoot your configuration without impacting production indexes. You can always change this setting later. Learn More

Select Allowed Indexes — Available item(s): events, history, main, metrics, summary — Selected item(s): events

Select indexes that clients will be able to select from.

Default Index: events

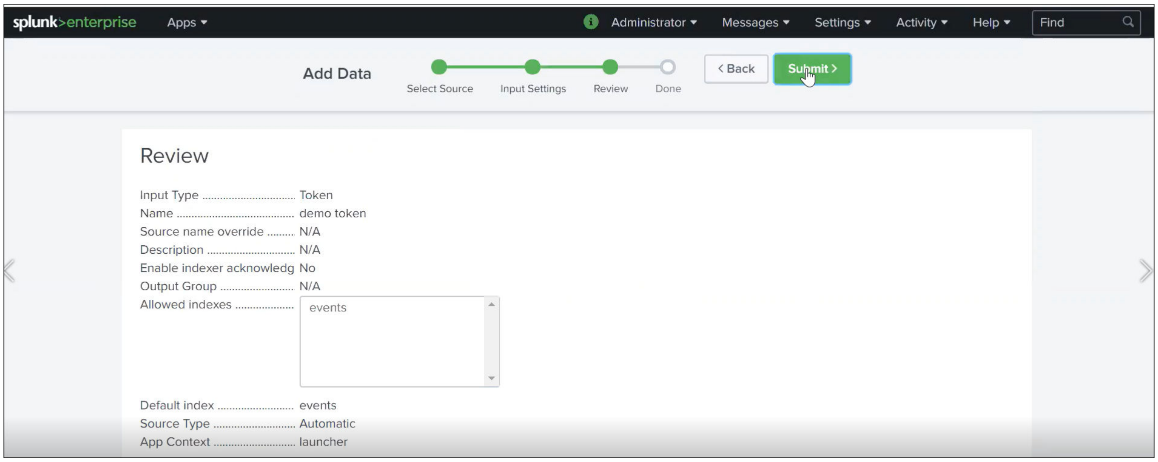
Figure 3-8: Input Settings Screen

f. In the **Available item(s)** list in the **Index** > **Select Allowed Indexes** section, select the index that you have created in *step 3*.

g. Double-click the index so that it appears in the **Selected item(s)** list.

h. Click **Review**.

The **Review** screen appears.



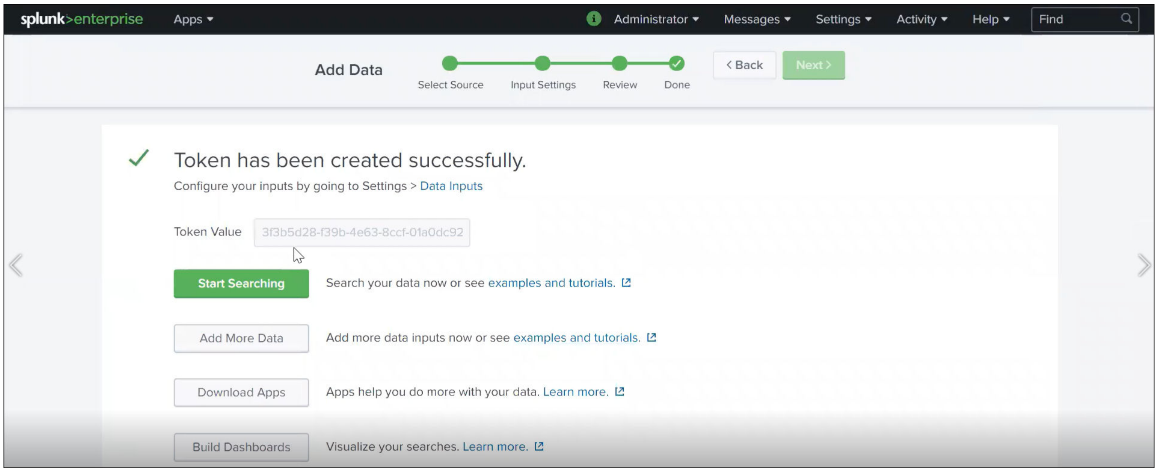
Review

Input Type .... Token
Name .... demo token
Source name override .... N/A
Description .... N/A
Enable indexer acknowledg No
Output Group .... N/A
Allowed indexes .... events
Default index .... events
Source Type .... Automatic
App Context .... launcher

Figure 3-9: Review Screen

i. Click **Submit.**

The **Done** screen appears. The **Token Value** field displays the HEC token that you have created. You must specify this token value in the *values.yaml* file that you will use to deploy Splunk Connect for Kubernetes.



Token has been created successfully.

Configure your inputs by going to Settings > Data Inputs

Token Value 3f3b5d28-f39b-4e63-8ccf-01a0dc92

Search your data now or see examples and tutorials.

Add more data inputs now or see examples and tutorials.

Apps help you do more with your data. Learn more.

Visualize your searches. Learn more.

Figure 3-10: Done Screen

5. Perform the following steps to configure the global settings for the HEC.

   a. Navigate to **Settings** > **Data inputs**.

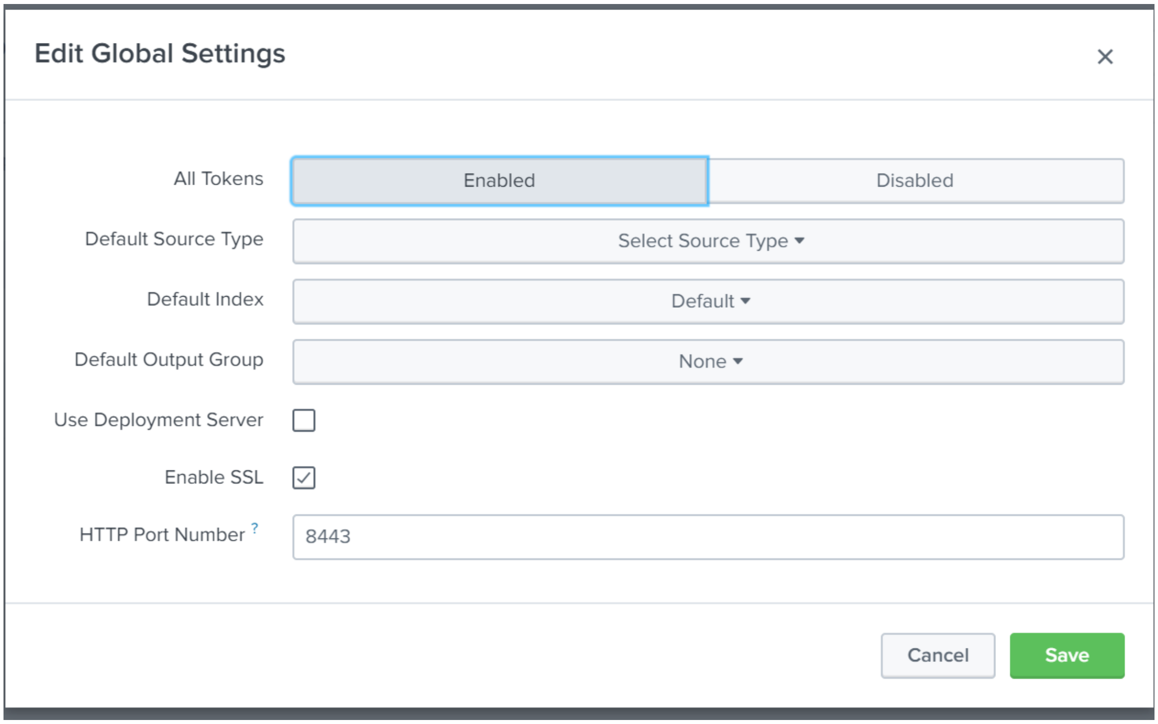
      The **Data inputs** screen appears.

   b. Click **HTTP Event Collector**.

      The **HTTP Event Collector** screen appears.

   c. Click **Global Settings**.

      The **Edit Global Settings** dialog box appears.



Edit Global Settings

All Tokens: Enabled | Disabled

Default Source Type: Select Source Type

Default Index: Default

Default Output Group: None

Use Deployment Server ☐

Enable SSL ☑

HTTP Port Number ? 8443

Figure 3-11: Edit Global Settings Dialog Box

   d. Ensure the **Enable SSL** check box is selected for SSL communication between the Splunk Enterprise and the Splunk Connect for Kubernetes.

   e. In the **HTTP Port Number** field, type a port number that will be used for the communication between the Splunk Enterprise and the Splunk Connect for Kubernetes.

> **Important:** Ensure that the pods in the Kubernetes cluster can communicate with the HEC on the configured port.

3.4 Deploying the Splunk Connect for Kubernetes Helm Chart on the Kubernetes Cluster

This section describes how you can deploy the Splunk Connect for Kubernetes Helm chart on the same Kubernetes cluster where you have deployed the AP-REST container.

To deploy the Splunk Connect for Kubernetes Helm chart:

1. Run the following command to create a service account that can be used by the pod where the Splunk Connect for Kubernetes will be deployed.

   ```
   oc create serviceaccount <Service_account_name>
   ```

   For example:

   ```
   oc create serviceaccount splunk-logging
   ```

2. Run the following command to assign privileged permission to the service account that you have created in *step 1*.

   ```
   oc adm policy add-scc-to-user privileged -z <Service_account_name>
   ```

   For example:

   ```
   oc adm policy add-scc-to-user privileged -z splunk-logging
   ```

3. Create a custom *custom-values.yaml* file for deploying the Splunk Connect for Kubernetes.

   The following snippet shows a sample *custom-values.yaml* file.

   ```
   global:
     splunk:
       hec:
         protocol: https
         insecureSSL: true
         token: <Token value>
         host: <Splunk server IP>
         port: 8443
         indexName: <Index name>
   serviceAccount:
     create: false
     name: splunk-logging
   containers:
     logFormatType: cri
     logFormat: "%Y-%m-%dT%H:%M:%S.%N%:z"
   ```

   > **Important:**
   >
   > If you want to copy the contents of the *custom-values.yaml* file, then ensure that you indent the file as per YAML requirements.

4. Modify the default values in the *custom-values.yaml* file as required.

Parameter	Description
global/splunk/hec/protocol	Specify the protocol that is used to communicate between the Splunk Connect for Kubernetes and the Splunk Enterprise installed on the Linux instance. By default, the value of this parameter is set to *https*.

Parameter	Description
global/splunk/hec/insecureSSL	Specify whether an insecure SSL connection over HTTPS should be allowed between the Splunk Connect for Kubernetes and the Splunk Enterprise installed on the Linux instance. Specify the value of this parameter to *true*.
global/splunk/hec/token	Specify the value of the token that you have created in *step 4i* of the section *Configuring Splunk*.
global/splunk/hec/host	Specify the IP address of the Linux instance where you have installed Splunk Enterprise.
global/splunk/hec/port	Specify the port number that you have used to configure the HTTP Event Collector of the Splunk Enterprise in *step 5e* of the section *Configuring Splunk*.
global/splunk/hec/indexName	Specify the name of the index that you have created in *step 3* of the section *Configuring Splunk*.
serviceAccount/create	Specify whether you want to create a new service account or use an existing service account. Specify the value of this parameter as *false*, so that you can use the service account that you have created in *step 1*.
serviceAccount/name	Specify the name of the service account that you have created in *step 1*.
container/logFormatType	Specify the log format type as *cri*, as this is the default log format for the OpenShift Container logs. **Important:** If you specify another value, such as, *json*, instead of *cri*, then the logs will not be parsed.
container/logFormat	Specify the log format as *%Y-%m-%dT%H:%M:%S.%N%:z*. This is the format of the timestamp that is included in each Container log.

For more information about the complete list of parameters that you can specify in the *custom-values.yaml* file, refer to the default *values.yaml* file in the Helm chart for Splunk Connect for Kubernetes.

5. If you want to forward only the Protegrity logs to the Splunk Enterprise, and do not want to forward the other logs, such as, the Kubernetes logs or the Container logs, then modify the *custom-values.yaml* file as follows.

```
global:
  splunk:
    hec:
      protocol: https
      insecureSSL: true
      token: <Token value>
      host: <Splunk server IP>
      port: 8443
      indexName: <Index name>
serviceAccount:
  create: false
  name: splunk-logging
containers:
  logFormatType: cri
  logFormat: "%Y-%m-%dT%H:%M:%S.%N%:z"
fluentd:
  path: /var/log/containers/*<Namespace where the AP-REST Container has been
deployed>*.log
```

All the container logs are stored in the */var/log/containers* path. You can filter out the Protegrity namespace logs based on the namespace where the AP-REST Container has been deployed.

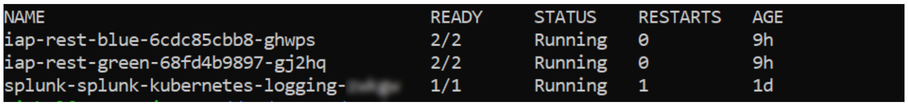
In the following example, the AP-REST Container has been deployed on the *protegrity-rest* namespace. You can then filter out the Protegrity logs by specifying the value of the *path* parameter as shown in the following code snippet. This ensures that only the Protegrity logs are forwarded to the Splunk Enterprise.

```
global:
  splunk:
    hec:
      protocol: https
      insecureSSL: true
      token: <Token value>
      host: <Splunk server IP>
      port: 8443
      indexName: <Index name>
serviceAccount:
  create: false
  name: splunk-logging
containers:
  logFormatType: cri
  logFormat: "%Y-%m-%dT%H:%M:%S.%N%:z"
fluentd:
  path: /var/log/containers/*protegrity-rest*.log
```

> **Note:** If you want to use another log collector, such as, Filebeat, alongside Fluentd, and you want to forward all the logs, except the Protegrity logs, to an Elasticsearch server instead of Splunk Enterprise, then you need to create another *custom-values2.yaml* file for configuring Filebeat, as shown in the following snippet.
>
> ```
> filebeatConfig:
>   filebeat.yml: |
>     filebeat.inputs:
>     - type: container
>       paths:
>         - /var/log/containers/*.log
>       exclude_files: ['.protegrity-rest.']
> ```
>
> The highlighted line in the snippet is used to exclude the Protegrity logs and forward the rest of the logs to the Elasticsearch server.
>
> You must run the following command to deploy Filebeat Helm chart on the Kubernetes cluster.
>
> ***helm repo add elastic https://helm.elastic.cohelm install filebeat -f custom-values2.yaml elastic/filebeat***

6. Run the following command to deploy the Splunk Connect for Kubernetes Helm chart on the Kubernetes cluster.
   helm install kube-splunk -f custom-values.yaml https://github.com/splunk/splunk-connect-for-kubernetes/releases/download/1.4.2/splunk-kubernetes-logging-1.4.2.tgz
7. Run the following command to list all the pods that are deployed.
   oc get pods

   The splunk-splunk-kubernetes-logging-XXXXX pod is listed on the console.



```
NAME                                     READY   STATUS    RESTARTS   AGE
iap-rest-blue-6cdc85cbb8-ghwps           2/2     Running   0          9h
iap-rest-green-68fd4b9897-gj2hq          2/2     Running   0          9h
splunk-splunk-kubernetes-logging-[redacted]   1/1     Running   1          1d
```

Chapter 4

Protecting Data Using the AP-REST Container Deployment

This section describes how to protect data using the AP-REST Container that has been deployed on the Kubernetes cluster using Postman client as an example. The Postman client is used to make REST calls to the AP-REST.

4.1 Running Security Operations

This section describes how you can use the AP-REST instances running on the Kubernetes cluster to protect the data that is sent by a REST API client.

➤ To run security operations:

1. If you are communicating with the AP-REST container from outside the Kubernetes cluster, then send the following cURL request from the Linux instance:

```
curl -v https://prod.example.com/rest-v1/protect -H 'Content-Type: application/json' --data '{ "protect": { "policyusername": "user1", "dataelementname": "Alphanum", "bulk": { "id": 1, "data": [ { "id": 1, "content": "<Data encoded in base64>" }, { "id": 2, "content": "Data encoded inbase64" } ] } } }' --cacert iap-rest/certs/iap-ca.crt --cert iap-rest/certs/iap-client.crt --key iap-rest/certs/iap-client.key
```

> **Important:**
>
> The input data must be Base64 encoded.

The AP-REST container instance returns the following protected output.

```
{"protect":{"bulk":{"id":1,"returntype":"success","data":[{"id":1,"returncode":"/rest-v1/
returncodes/id/
6","returntype":"success","content":"AHMAVABLAGMARwBSAHEAbQBxAEMAMQAyADMANAA1"},
{"id":2,"returncode":"/rest-v1/returncodes/id/
6","returntype":"success","content":"AHIASABGAE4AdgBVAFIARgBEAFMAWgBmAGYAZABPAEsAcg=="}]}}
}
```

If you want to unprotect the data, then you can run the following command:

```
curl -v POST https://prod.example.com/rest-v1/unprotect -H 'Content-
Type:application/json' --data'{ "unprotect": { "policyusername": "user1",
"dataelementname": "Alphanum", "bulk": { "id": 1, "data": [ { "id": 1, "content":
"<Data encoded in base64>" }, { "id": 2, "content": "<Data encoded in
base64>" } ] } } }' --cacert iap-rest/certs/iap-ca.crt --cert iap-rest/certs/iap-
client.crt --key iap-rest/certs/iap-client.key
```

If you want to reprotect the data, then you can run the following command:

```
curl -v POST https://prod.example.com/rest-v1/reprotect -H 'Content-
Type:application/json' --data '{ "reprotect": { "policyusername": "user1",
"olddataelementname": "Alphanum", "newdataelementname": "Alphanum1", "bulk":
{ "id": 1, "data": [ { "id": 1, "content": "<Dataencoded in base64>" }, { "id": 2,
"content": "<Data encoded in base64>" } ] } } }' --cacert iap-rest/certs/iap-ca.crt
--cert iap-rest/certs/iap-client.crt --key iap-rest/certs/iap-client.key
```

You can use these commands with the following patterns:

- External customer application communicating with the AP-REST container
- Customer container and the AP-REST container are in different pods in different Kubernetes clusters

2. If the AP-REST container and the Customer container are in separate pods, but on the same Kubernetes cluster, then send the following cURL request from the Linux instance:

```
curl -v https://<Kubernetes service name of AP-REST container>.<Namespace
of AP-REST container>.svc.cluster.local:8443/rest-v1/protect -H 'Content-
Type: application/json' --data '{ "protect": { "policyusername": "user1",
"dataelementname": "Alphanum", "bulk": { "id": 1, "data": [ { "id": 1,
"content": "<Data encoded in base64>" }, { "id": 2, "content": "Data encoded
inbase64" } ] } } }' --cacert iap-rest/certs/iap-ca.crt --cert iap-rest/certs/iap-
client.crt --key iap-rest/certs/iap-client.key
```

> **Important:**
>
> The input data must be Base64 encoded.

The AP-REST container instance returns the following protected output.

```
{"protect":{"bulk":{"id":1,"returntype":"success","data":[{"id":1,"returncode":"/rest-v1/
returncodes/id/
6","returntype":"success","content":"AHMAVABLAGMARwBSAHEAbQBxAEMAMQAyADMANAA1"},
{"id":2,"returncode":"/rest-v1/returncodes/id/
6","returntype":"success","content":"AHIASABGAE4AdgBVAFIARgBEAFMAWgBmAGYAZABPAEsAcg=="}]}}
}
```

If you want to unprotect the data, then you can run the following command:

```
curl -v POST https://<Kubernetes service name of AP-REST container>.<Namespace
of AP-REST container>.svc.cluster.local:8443/rest-v1/unprotect -H 'Content-
Type: application/json' --data'{ "unprotect": { "policyusername": "user1",
"dataelementname": "Alphanum", "bulk": { "id": 1, "data": [ { "id": 1, "content":
"<Data encoded in base64>" }, { "id": 2, "content": "<Data encoded in
base64>" } ] } } }' --cacert iap-rest/certs/iap-ca.crt --cert iap-rest/certs/iap-
client.crt --key iap-rest/certs/iap-client.key
```

If you want to reprotect the data, then you can run the following command:

```
curl -v POST https://<Kubernetes service name of AP-REST container>.<Namespace
of AP-REST container>.svc.cluster.local:8443/rest-v1/reprotect -H 'Content-
Type:application/json' --data '{ "reprotect": { "policyusername": "user1",
"olddataelementname": "Alphanum", "newdataelementname": "Alphanum1", "bulk":
```

```
{ "id": 1, "data": [ { "id": 1, "content": "<Dataencoded in base64>" }, { "id": 2,
"content": "<Data encoded in base64>" } ] } } }' --cacert iap-rest/certs/iap-ca.crt
--cert iap-rest/certs/iap-client.crt --key iap-rest/certs/iap-client.key
```

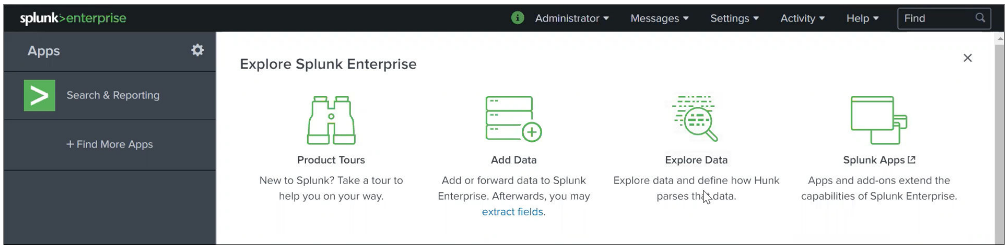
4.2 Viewing Logs in Splunk

This section describes how you can view the AP-REST and Container logs in Splunk Enterprise.

➤ To view logs:

1. Login to Splunk Web at *http://<IP of Linux instance>:8000.*
2. On the Splunk Web UI, type the username and password that you have created while installing Splunk Enterprise on the Linux instance.

   The Splunk Enterprise screen appears.



New to Splunk? Take a tour to help you on your way.

Add or forward data to Splunk Enterprise. Afterwards, you may extract fields.

Explore data and define how Hunk parses the data.

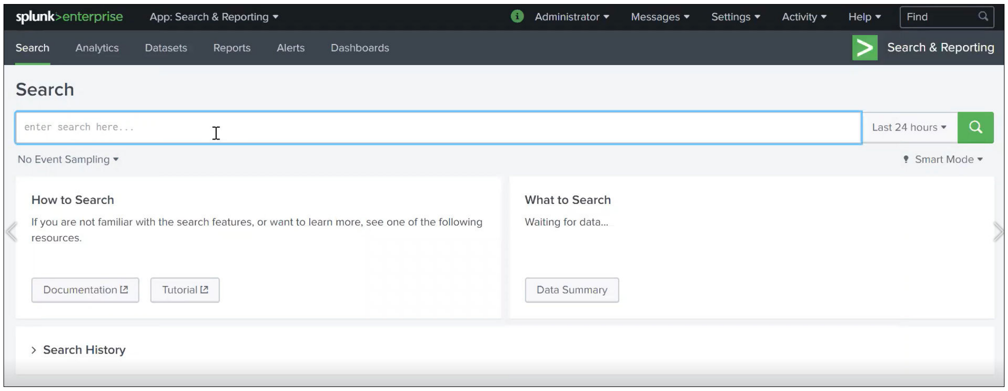
Apps and add-ons extend the capabilities of Splunk Enterprise.

Figure 4-1: Splunk Enterprise Web UI

   For more information about the user credentials, refer to the section *Setting Up Splunk*.

3. On the left pane, click **Search & Reporting**.

   The **Search** screen appears.



If you are not familiar with the search features, or want to learn more, see one of the following resources.

Waiting for data...

Figure 4-2: Search Screen

4. In the **Search** field, type the following text to filter the logs.

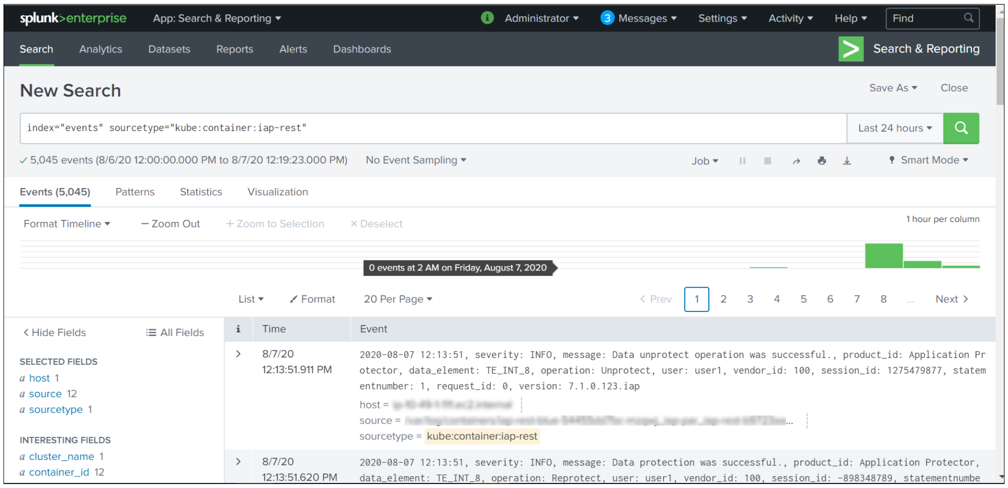
```
index="<Index_name>" sourcetype="kube:container:<Container_name>"
```

   For example:

```
index="events" sourcetype="kube:container:iap-rest"
```

5. Press **Enter**.

   The search results appear in the **Events** tab. The search results display all the STDOUT logs from the specified container.



```
index="events" sourcetype="kube:container:iap-rest"
```

✓ 5,045 events (8/6/20 12:00:00.000 PM to 8/7/20 12:19:23.000 PM)

8/7/20 12:13:51.911 PM

```
2020-08-07 12:13:51, severity: INFO, message: Data unprotect operation was successful., product_id: Application Pr
otector, data_element: TE_INT_8, operation: Unprotect, user: user1, vendor_id: 100, session_id: 1275479877, statem
entnumber: 1, request_id: 0, version: 7.1.0.123.iap
```

host = [redacted]
source = [redacted]
sourcetype = kube:container:iap-rest

8/7/20 12:13:51.620 PM

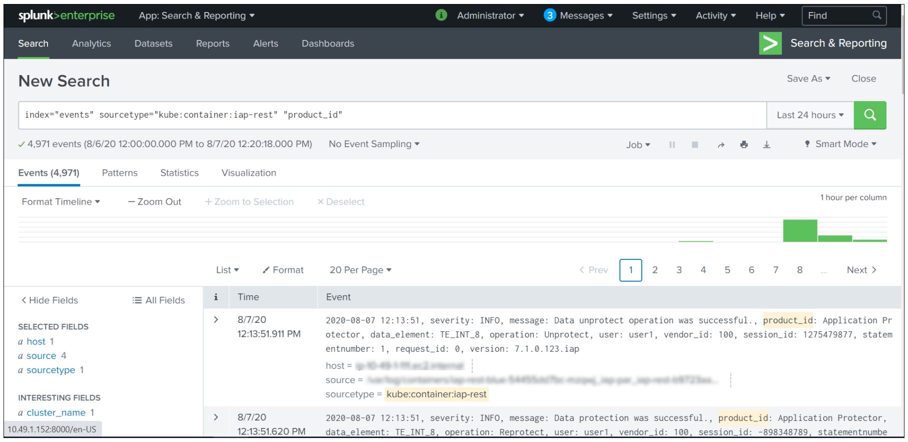
```
2020-08-07 12:13:51, severity: INFO, message: Data protection was successful., product_id: Application Protector,
data_element: TE_INT_8, operation: Reprotect, user: user1, vendor_id: 100, session_id: -898348789, statementnumbe
```

6. If you want to view only the logs that are related to the AP-REST, then you can modify the text in the **Search** field to filter the logs, as shown in the following snippet.

```
index="<Index_name>" sourcetype="kube:container:iap-rest"
```

7. Press **Enter**.

   The search results display only the logs that are related to the AP-REST.



```
index="events" sourcetype="kube:container:iap-rest" "product_id"
```

✓ 4,971 events (8/6/20 12:00:00.000 PM to 8/7/20 12:20:18.000 PM)

8/7/20 12:13:51.911 PM

```
2020-08-07 12:13:51, severity: INFO, message: Data unprotect operation was successful., product_id: Application Pr
otector, data_element: TE_INT_8, operation: Unprotect, user: user1, vendor_id: 100, session_id: 1275479877, statem
entnumber: 1, request_id: 0, version: 7.1.0.123.iap
```

host = [redacted]
source = [redacted]
sourcetype = kube:container:iap-rest

8/7/20 12:13:51.620 PM

```
2020-08-07 12:13:51, severity: INFO, message: Data protection was successful., product_id: Application Protector,
data_element: TE_INT_8, operation: Reprotect, user: user1, vendor_id: 100, session_id: -898348789, statementnumbe
```

4.3 Autoscaling the Protegrity Reference Deployment

You can use the autoscaling property of Kubernetes to autoscale the AP-REST instances based on a specific load. After the load crosses a pre-defined threshold, Kubernetes automatically scales up the AP-REST instances. Similarly, as the load dips below the pre-defined threshold, Kubernetes automatically scales down the AP-REST instances.

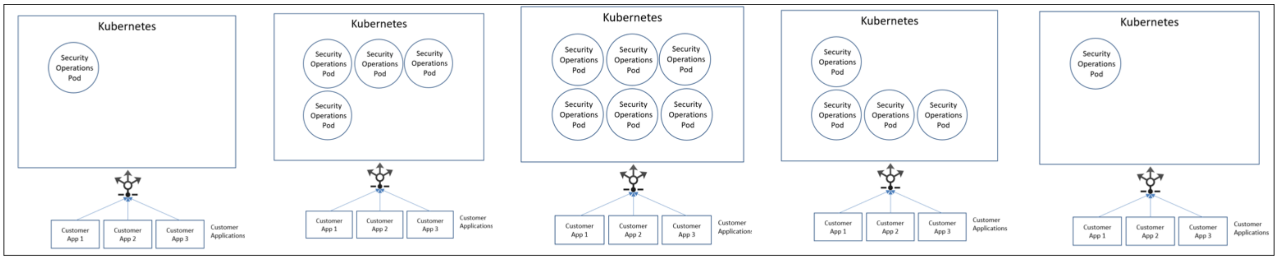


Figure 4-3: Autoscaling Kubernetes Cluster

As illustrated in the figure, the Customer Applications experience an increase and a decrease in workload required by the business. Kubernetes will automatically grow the Protegrity Security Operation pods to meet business needs. If the workloads reduce, then Kubernetes will shrink the cluster.

You do not have to provision additional AP-REST appliances to meet the business needs and then remove them if not required. Kubernetes performs the task of provisioning the AP-REST instances automatically.

The auto scaling capability ensures that the business needs are met dynamically while optimizing the costs.

Chapter 5

Appendix A: Get ESA Policy Helm Chart Details

This section describes the details of the Get ESA Policy Helm chart structure and the entities that the user needs to modify for adapting the chart for their environments.

5.1 Chart.yaml

The *Chart.yaml* file contains information about the Helm chart. These values can be left as default.

```
apiVersion: v1
description: A chart for getting policy from ESA
name: get-esa-policy
version: 1.0.0
```

5.2 Values.yaml

The *Values.yaml* file contains important fields that need to be edited by the user as per the requirements of each environment. The following sections will explain the details of each field that needs to be replaced.

5.2.1 Image Pull Secrets

This section specifies the credentials that are required to access the image repository.

```
## create image pull secrets and specify the name here.
## remove the [] after 'imagePullSecrets:' once you specify the secrets
imagePullSecrets: []
#  - name: regcred
```

The user is required to create a Kubernetes secret for accessing the image registry. Kubernetes secrets can be created using existing Docker credentials.

For example:

```
oc create secret generic regcred --from-file=.dockerconfigjson=<PATH_TO_DOCKER_CONFIG>/config.json --type=kubernetes.io/dockerconfigjson --namespace <NAMESPACE>
```

5.2.2 Name Override

These values indicate how naming for releases are managed for deployment.

> **Note:** Protegrity recommends that these values should not be changed by the user.

If the ***fullnameoverride*** parameter value is specified, then the deployment names will use that value.

If the ***nameOverride*** parameter value is specified, then the deployment names will be a combination of the ***nameOverride*** parameter value and the Helm chart release name.

If both the values are kept empty, then the Helm chart release name will be used.

```
nameOverride: ""
fullnameOverride: ""
```

5.2.3 Container Image Repository

This section provides the path for the images in the fields that the users are required to enter after loading the images into their registry.

```
getEsaPolicyImage:
  repository: GET_ESA_POLICY_IMAGE_REPOSITORY
  tag: GET_ESA_POLICY_IMAGE_TAG
  pullPolicy: Always
  debug: false
```

The following list provides an overview of the *getEsaPolicyImage* parameter, which refers to details for the Get ESA Policy container image:

- *repository* – Refers to the name of the registry.

  > **Note:** Ensure that you only add the Container registry path as the repository value. Do not include the tag name in this value. You need to add the tag name as the value in the *tag* field.

- *tag* – Refers to the tag mentioned at the time of the image upload.
- *debug* - Sets the value of this field to *true*, if you want debug logs for the Get ESA Policy container. By default, the value of this field is set to *false*.

5.2.4 Resources

This section specifies the resource limits that can be set for the containers in a pod.

```
resources:
  ## We usually recommend not to specify default resources and to leave this as a conscious
  ## choice for the user. This also increases chances charts run on environments with little
  ## resources, such as Minikube. If you do want to specify resources, uncomment the following
  ## lines, adjust them as necessary, and remove the curly braces after 'resources:'.
  # limits:
  #   cpu: 500m
  #   memory: 3500Mi
  requests:
    cpu: 200m
    memory: 200Mi
```

You can specify the following parameters:

- *limits* - Specify the resource limits for the containers in a pod. These limits ensure that the running container does not use additional resources than the limit you set.

- *requests* - Specify the resource request for the containers in a pod. This information determines the nodes for deploying the pods.

For more about the resource parameters, refer to the section *Managing Resources for Containers* in the Kubernetes documentation.

5.2.5 Pod Security Context

This section specifies the privilege and access control settings for the pod.

```
## set the Pod's security context object
## leave the field empty if not applicable
podSecurityContext:
  runAsUser: 1000
  runAsGroup: 1000
  fsGroup: 1000
  supplementalGroups: [1000]
```

For more information about the Pod Security Context, refer to the section *Configure a Security Context for a Pod or Container* in the Kubernetes documentation.

For more information about the parameters, refer to the section *PodSecurityContext v1 Core* in the Kubernetes API documentation.

5.2.6 Persistent Volume Claim

This section specifies the value of the persistent volume claim that will be used to store the immutable policy package.

```
## If securityConfigDestination is Volume Mount,
## specify the name of persistent volume claim to be used for this deployment.
pvcName: PVC_NAME
```

For more information about persistent volumes and persistent volume claims, refer to the section *Persistent Volumes* in the Kubernetes documentation.

5.2.7 ESA Credential Secrets

This section provides information for the credentials that are required to access the ESA for retrieving the policy.

```
## k8s secret for storing ESA credentials
esaCredSecrets: ESA_CREDENTIAL_SECRET_NAME
```

The user is required to create a Kubernetes secret for accessing the ESA. Kubernetes secrets can be created using existing Docker credentials.

For example:

```
oc create secret generic esa-creds --from-literal=esa_user=<ESA user> --from-literal=
esa_pass=<ESA user password> --namespace <NAMESPACE>
```

```
oc create secret generic esa-creds --from-literal=esa_user=non-admin --from-literal=
esa_pass=<ESA user password> --namespace iap-rest
```

> **Important:** Ensure that the user, who will be used to access the ESA, has been assigned the *Export Certificates* and *Appliance CLI Viewer* permissions through a custom role. Roles are templates that include permissions and users can be assigned one or more roles.
>
> For more information about managing roles, refer to the section *Managing Roles* in the *Appliances Overview 9.1.0.0* guide.

For more information about managing users, refer to the section *Managing Users* in the *Appliances Overview 9.1.0.0* guide.

5.2.8 PBE Secrets

This section provides information for the passphrase and the salt that are used to generate a key.

```
## k8s secret containing passphrase for encrypting policy
pbeSecrets: PBE_SECRET_NAME
```

The Get ESA Policy container uses this key to encrypt the policy. This is used to implement Passphrase Based Encryption (PBE).

For example:

```
oc create secret generic <PBE_SECRET_NAME> --from-literal='passphrase=<passphrase>' --from-literal='salt=<salt>' -n <NAMESPACE>
```

```
oc create secret generic pbe-secret --from-literal='passphrase=<passphrase>' --from-literal='salt=<salt>' -n iap-rest
```

5.2.9 Security Config Destination

This section specifies the destination where you want to store the policy package.

```
## choose your storage destination. Currently we support <VolumeMount | ObjectStore>
## default is VolumeMount, change it to ObjectStore if you wish to use object store
securityConfigDestination: VolumeMount
```

The value of the *securityConfigDestination* parameter is set to *VolumeMount*.

5.2.10 Pod Service Account

This section provides information on how you can specify the name of the service account that you have created for the pod in the Kubernetes Cluster. Do not edit the field if not applicable.

```
serviceAccount: <Name of the service account created for the pod>
```

Do not edit the field if not applicable.

5.2.11 Policy

This section specifies the configurations related to the ESA policy for the current release.

```
policy:
  esaIP: ESA_IP

  ## pepserver configurations that can be changed.
    ## emptystring: Set the output behavior for empty strings
      ## null = (default) null value is returned for empty input string
      ## empty = empty value is returned for empty input string
      ## encrypt = encrypted values is returned for empty input string
    ## level: Specifies the logging level for pepserver
      ## OFF (no logging)
      ## SEVERE
      ## WARNING
      ## INFO
      ## CONFIG
      ## ALL (default)
    ## caseSensitive: Specifies how policy users are checked against policy
      ## yes = (The default) policy users are treated in case sensitive manner
      ## no = policy users are treated in case insensitive manner.
  pepserverConfig:
    emptyString: "null"
    logLevel: "ALL"
    caseSensitive: "yes"
  ## absolute path in PV or ObjectStore where the policy dump file will be stored. <eg. /
test/xyz/policy >
  ## If destination is ObjectStore, make sure first name in path is bucket name i.e. test
will be our existing bucket
  filePath: ABSOLUTE_POLICY_PATH

  ## time (in mins) to wait till the policies get downloaded
  ## default is 15 (mins)
  timeout: 15
```

The *esaIP* field denotes the IP address of the ESA from where you want to retrieve the policy.

The *pepserverConfig* field enables you to specify the PEP server configurations.

Users are required to update the file path where you want to upload the policy package.

You can also specify the maximum time until which the Get ESA Policy container tries to establish communication with the ESA for fetching the policies, after which the connection times out. The default value is 15 minutes.

5.3 pty-scc.yaml

The *pty-scc.yaml* file contains the Protegrity Security Context Constraints (SCC).

> **Important:** Do not modify the values.

```
kind: SecurityContextConstraints
apiVersion: security.openshift.io/v1
metadata:
  annotations:
    kubernetes.io/description: This is a restrictive SCC.
  name: pty-scc
allowHostDirVolumePlugin: false
allowHostIPC: false
allowHostNetwork: false
allowHostPorts: false
allowHostPID: false
allowPrivilegeEscalation: false
allowPrivilegedContainer: false
defaultAddCapabilities: []
allowedCapabilities: []
readOnlyRootFilesystem: true
groups:
- system:serviceaccounts:<NAMESPACE>
requiredDropCapabilities:
- ALL
runAsUser:
  type: MustRunAsNonRoot
seLinuxContext:
  type: RunAsAny
supplementalGroups:
  type: MustRunAs
  ranges:
    - min: 1
      max: 65535
fsGroup:
  type: MustRunAs
  ranges:
    - min: 1
      max: 65535
volumes:
- persistentVolumeClaim
- secret
- configMap
- emptyDir
```

5.4 pty-rbac.yaml

The *pty-rbac.yaml* file defines the roles that are assigned the Protegrity Security Context Constraints (SCC).

> **Important:** Do not modify the values.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: iap-install-upgrade
rules:
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["create", "get" ,"list", "patch" ,"watch","delete"]
- apiGroups: ["autoscaling"] # "" indicates the core API group
  resources: ["horizontalpodautoscalers"]
  verbs: ["create", "get" ,"list", "patch" ,"watch","delete"]
- apiGroups: ["extensions"] # "" indicates the core API group
  resources: ["ingresses"]
  verbs: ["create", "get" ,"list", "patch" ,"watch","delete"]
- apiGroups: [""] # "" indicates the core API group
  resources: ["services" ]
  verbs: ["create", "get" ,"list", "patch" , "watch","delete"]
- apiGroups: [""] # "" indicates the core API group
  resources: ["persistentvolumeclaims"]
  verbs: ["create", "get" ,"list", "watch"]
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods", "pods/log" ]
  verbs: [ "get" , "list" , "watch"]
- apiGroups: [""] # "" indicates the core API group
  resources: ["events" ]
  verbs: [ "get" , "list" , "watch"]
- apiGroups: ["metrics.k8s.io"] # Top pods permission
  resources: ["pods" ]
  verbs: [ "get" , "list" , "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: iap-install-upgrade
subjects:
- kind: ServiceAccount
  name: iap-deployer
roleRef:
  kind: Role
  name: iap-install-upgrade
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: get-esa-policy-install-delete
rules:
- apiGroups: ["batch"]
  resources: ["jobs"]
  verbs: ["create", "get","delete"]
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["create"]
- apiGroups: [""] # "" indicates the core API group
  resources: ["persistentvolumeclaims"]
  verbs: ["create", "get" ,"list", "watch"]
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods", "pods/log" ]
  verbs: [ "get" , "list" , "watch"]
- apiGroups: [""] # "" indicates the core API group
```

```
  resources: ["events" ]
  verbs: [ "get" , "list" , "watch"]
- apiGroups: ["metrics.k8s.io"] # Top pods permission
  resources: ["pods" ]
  verbs: [ "get" , "list" , "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: get-esa-policy-install-delete
subjects:
- kind: ServiceAccount
  name: get-esa-deployer
roleRef:
  kind: Role
  name: get-esa-policy-install-delete
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: iap-secrets-manager
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["create", "get" ,"list", "delete" , "update" ]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: iap-secrets-manager
subjects:
- kind: ServiceAccount
  name: get-esa-deployer
- kind: ServiceAccount
  name: iap-deployer
roleRef:
  kind: Role
  name: iap-secrets-manager
  apiGroup: rbac.authorization.k8s.io
```

Chapter 6

Appendix B: AP-REST Helm Chart Details

This section describes the details of the AP-REST Helm chart structure and the entities that the user needs to modify for adapting the chart for their environments.

6.1 Chart.yaml

The *Chart.yaml* file contains information regarding Helm versions.

These values can be left as default.

```
apiVersion: v1
appVersion: "1.0"
description: Immutable Application Protector REST Chart for Kubernetes
name: iap-rest
version: 1.0.0
```

6.2 Values.yaml

The *Values.yaml* file contains important fields that need to be edited by the user as per the specifics of each environment.

The following sections will explain the details of each field that needs to be replaced.

6.2.1 Image Pull Secrets

This section specifies the credentials that are required to access the image repository.

```
## create image pull secrets and specify the name here.
## remove the [] after 'imagePullSecrets:' once you specify the secrets
imagePullSecrets: []
#  - name: regcred
```

The user is required to create a Kubernetes secret for accessing the image registry. Kubernetes secrets can be created using existing Docker credentials.

> **Important:** Both privileged and non-privileged roles can perform this task.

For example:

```
kubectl create secret generic regcred --from-
file=.dockerconfigjson=<PATH_TO_DOCKER_CONFIG>/config.json --type=kubernetes.io/
dockerconfigjson --namespace <NAMESPACE>
```

6.2.2 Name Override

This section specifies the values that indicate how naming for releases are managed for deployment.

> **Note:** Protegrity recommends that these values should not be changed by the user.

If the `fullnameoverride` parameter value is specified, then the deployment names will use that value.

If the `nameOverride` parameter value is specified, then the deployment names will be a combination of the `nameOverride` parameter value and the Helm chart release name.

If both the values are kept empty, then the Helm chart release name will be used.

```
nameOverride: ""
fullnameOverride: ""
```

6.2.3 Container Image Repository

This section provides the path for the images in the fields that the users are required to enter after loading the images into their registry.

```
initImage:
  repository: INIT_CONTAINER_IMAGE_REPOSITORY
  tag: INIT_CONTAINER_IMAGE_TAG
  pullPolicy: Always
  debug: false # enable this flag to get imp container debug logs on STDOUT.

iaprestImage:
  repository: IAPREST_IMAGE_REPOSITORY
  tag: IAPREST_IMAGE_TAG
  pullPolicy: Always

# Docker Hub Image (Root User): docker.io/nginx:stable
# To use nginx image that runs with non-root permissions
# Ref. https://hub.docker.com/r/nginxinc/nginx-unprivileged
nginxImage:
  repository: NGINX_IMAGE_REPOSITORY
  tag: NGINX_IMAGE_TAG
  pullPolicy: Always
```

The following list provides an overview for the parameters in the Get ESA Policy container image:

- *initImage* – Refers to details for the Init container image:
  - *repository* – Refers to the name of the registry

    > **Note:**
    >
    > Ensure that you only add the Container registry path as the repository value. Do not include the tag name in this value. You need to add tag name as the value in the *tag* field.

  - *tag* – Refers to the tag mentioned at the time of image upload
  - *debug* - Set the value of this field to *true*, if you want debug logs for the Init container. By default, the value of this field is set to *false*.
- *iaprestImage* – Refers to details for the AP-REST container image

- *nginxImage* – Refers to details for the NGINX container image

  > **Note:**
  >
  > If you want to use the NGINX image that runs with root user, then download the NGINX image from the Docker hub *docker.io/nginx:stable*.
  >
  > If you want to use NGINX image that runs with non-root permissions, then refer to the website *Docker Hub for NGINX Docker Images*.

6.2.4 Resources

This section specifies the resource limits that can be set for the containers in a pod.

```
initContainerResources:
# Important: Set the resource based on the policy metadata dump size.
# Default is set for a policy metadata dump size upto ~350Mb.
  limits:
    cpu: 300m
    memory: 3096Mi
  requests:
    cpu: 200m
    memory: 512Mi

iaprestResources:
  limits:
    cpu: 500m
    memory: 512Mi
  requests:
    cpu: 200m
    memory: 200Mi

nginxResources:
  limits:
    cpu: 500m
    memory: 512Mi
  requests:
    cpu: 200m
    memory: 200Mi
```

You can specify the following parameters:

- *limits* - Specify the resource limits for the containers in a pod. These limits ensure that the running container does not use more resources than the limit you set.
- *requests* - Specify the resource request for the containers in a pod. This information determines the nodes for deploying the pods.

For more about the resource parameters, refer to the section *Managing Resources for Containers* in the Kubernetes documentation.

6.2.5 Pod Service Account

This section specifies the name of the service account that you have created for the pod in the Kubernetes Cluster. Do not edit the field if not applicable.

```
## pod service account to be used
## leave the field empty if not applicable
serviceAccount: <Name of the service account created for the pod>
```

6.2.6 Liveliness and Readiness Probe Settings

This section provides information for the intervals required to run health checks for the AP-REST container images. Users need not change these settings.

```
## Configure the delays for Liveness Probe here
livenessProbe:
  initialDelaySeconds: 15
  periodSeconds: 20

## Configure the delays for Readiness Probe here
readinessProbe:
  initialDelaySeconds: 15
  periodSeconds: 20
```

6.2.7 Pod Security Context

This section specifies the privilege and access control settings for the pod.

```
## set the Pod's security context object
## leave the field empty if not applicable
podSecurityContext:
  runAsUser: 1000
  runAsGroup: 1000
  fsGroup: 1000
```

For more information about the Pod Security Context, refer to the section *Configure a Security Context for a Pod or Container* in the Kubernetes documentation.

For more information about the parameters, refer to the section *PodSecurityContext v1 Core* in the Kubernetes API documentation.

6.2.8 Container Security Context

This section specifies the privilege and access control settings for the container.

```
## set the Init Container's security context object
## leave the field empty if not applicable
initContainerSecurityContext:
  capabilities:
    drop:
    - ALL
  allowPrivilegeEscalation: false
  privileged : false
  runAsNonRoot : true
  readOnlyRootFilesystem: true

## set the iapRest Container's security context object
## leave the field empty if not applicable
iaprestContainerSecurityContext:
  capabilities:
    drop:
    - ALL
  allowPrivilegeEscalation: false
  privileged : false
  runAsNonRoot : true
  readOnlyRootFilesystem: true

## set the nginx Container's security context object
## leave the field empty if not applicable
nginxContainerSecurityContext:
  capabilities:
    drop:
    - ALL
  allowPrivilegeEscalation: false
  privileged : false
  runAsNonRoot : true
  readOnlyRootFilesystem: true
```

The default values in the Container Security Context ensure that the container is not run in privileged mode.

> **Note:** It is recommended not to edit the default values.

For more information about the Container Security Context, refer to the section *Configure a Security Context for a Pod or Container* in the Kubernetes documentation.

For more information about the parameter, refer to the section *SecurityContext v1 core* in the Kubernetes API documentation.

6.2.9 Persistent Volume Claim

This section specifies the value of the persistent volume claim that will be used to store the immutable policy package.

```
#Name of persistent volume claim to be used for this deployment.
pvcName: PVC_NAME
```

For more information about persistent volumes and persistent volume claims, refer to the section *Persistent Volumes* in the Kubernetes documentation.

6.2.10 PBE Secrets

This section uses the value specified in the *pbeSecrets* parameter to decrypt the policy in the AP-REST container.

```
pbeSecrets: PBE_SECRET_NAME
```

6.2.11 NGINX Secrets

This section describes the *nginxCertsSecrets* Kubernetes secret. The *nginxCertsSecrets* Kubernetes secret contains the TLS certificates for the NGINX container.

```
## k8s secret containing TLS certificates for nginx sidecar
nginxCertsSecrets: NGINX_SECRET_NAME
```

6.2.12 Deployment

This section provides an overview on the configurations that refer to the policy package information for the current release. The first release is considered to be *Blue* deployment.

When the ***`blueDeployment/enabled: true`*** parameter is enabled, the configurations mentioned under the policy section are deployed.

```
blueDeployment:
  enabled: true
```

```
### Policy metadata for Blue Deployments
blueDeployment:
  enabled: true
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV where the policy dump file will be stored. <eg. /test/xyz/policy >
    filePath: ABSOULTE_POLICY_PATH

## Policy metadata for Green Deployments
greenDeployment:
  enabled: false
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV where the policy dump file will be stored. <eg. /test/xyz/policy >
    filePath: ABSOULTE_POLICY_PATH
```

The users are required to update the file path where the policy package has been uploaded. Note that all entries are case-sensitive.

6.2.13 Autoscaling

This section provides an overview for the parameters used for autoscaling of pods on a cluster.

```
## specify the initial no. of springapp Pod replicas
replicaCount: 1

## HPA configuration
autoScaling:
  minReplicas: 1
  maxReplicas: 10
  targetCPU: 50
```

The following list provides a description for the parameters:

- *replicaCount* - Indicates the number of pods that get instantiated at the start of the deployment.
- *minReplicas* - Indicates the minimum number of pods that will keep running in the deployment for a cluster.
- *maxReplicas* - Indicates the maximum number of pods that will keep running in the deployment for a cluster.
- *targetCPU* - Horizontal Pod Autoscaler (HPA) will increase and decrease the number of replicas (through the deployment) to maintain an average CPU utilization across all Pods based on the % value specified in the *targetCPU* setting.

6.2.14 Service Configuration

This section specifies the configurations for the REST service that needs to be used on the cluster running the AP-REST containers.

```
### specify the ports exposed in your iaprest configurations where,
## name - distinguishes between different ports.
## port - the port on which you wan't to expose the service externally.
## targetPort - the port no. configured while creating Tunnel.
iaprestService:
    name: "iaprest"
    port: 8443
    targetPort: 8443
```

- *name* - Specifies the name of the Kubernetes service. The name must contain only lowercase alphanumeric characters, which is based on standard Kubernetes naming convention.
- *port* - Specifies the port on which you want to expose the service externally.
- *targetPort* - Specifies the port on which the AP-REST is running inside the Docker container.

6.2.15 Routes Configuration

This section explains the Routes configuration for deployment.

```
## specify hostname, path and annotations for prod and staging routes in this section.
## staging routes will be enabled only if both blue & green deployments are enabled.
routes:
  prodService:
    hostName: "prod.example.com"
    httpPath: "/"
    annotations:
      # specify route annotations if any
      # e.g. haproxy.router.openshift.io/ip_whitelist: 192.168.1.10

  stagingService:
    hostName: "staging.example.com"
    httpPath: "/"
    annotations:
      # specify route annotations if any
      # e.g. haproxy.router.openshift.io/ip_whitelist: 192.168.1.10
```

The *prodService* parameters specify the configurations for the production environment, while the *stagingService* parameters specify the configurations for the staging environment.

Chapter 7

Appendix C: Using the Samples

This section explains details and usage of the components included in the *REST-Samples_Linux-ALL-ALL_x86-64_<component_version>.tgz* archive.

7.1 Overview

Protegrity delivers a sample application as part of the AP-REST Container installation package. The sample application constitutes a canned policy (to get imported to the ESA), sample Postman collection (to test AP-REST Container Pods serving deployed IMP) and an auto-scaling script (to push more load to the Kubernetes cluster to force auto-scaling of the AP-REST Container Pods). On consuming the deliverables in the AP-REST Container installation package, we expect customers to run this sample application end-to-end, as a sanity test, to confirm the installation was completed accurately. In this section, we are providing details on the exact steps that the customer needs to follow to run the sample application end-to-end. This section explains details and usage of the components included in *REST-Samples_Linux-ALL-ALL_x86-64_<component_version>.tgz* archive.

The following components are included in the *REST-Samples_Linux-ALL-ALL_x86-64_<component_version>.tgz* archive.

> **Note:**
> The sample included refers to the deployment pattern where the AP-REST container receives the request for protecting, unprotecting, and reprotecting data from an external customer application, which is not deployed in a Kubernetes environment.
>
> For more information about the deployment patterns, refer to the section *Architecture and Components*.

7.1.1 Policy Sample

Package – *Sample_App_Policy.tgz*

This component consists of the sample policy that can be imported on the ESA 9.1.0.5 for getting started with the AP-REST Containers use case. The following are the details for the policy.

Policy Name - Sample_policy

Token Type	**Data Element Name**
Alphanumeric	Alphanum
Alphanumeric	Alphanum1

7.1.2 Autoscaling Script

Package – *Sample_App_autoscale.sh*

Script for making 10,000 REST calls to AP-REST. This script can be triggered to test the autoscaling of the pods.

7.1.3 PostMan Collection

PKG – *Sample_App_PostMan_Collection.json*

This collection can be used to make REST calls to AP-REST for protecting the data. The JSON file contains the following collections:

- Release 1 protect request
- Release 1 unprotect request
- Release 1 reprotect request
- Release 2 protect request

Release 1 protect request

Post Request Path: - ***`https://prod.example.com/rest-v1/protect`***

Release 1 unprotect request

Post Request Path: - ***`https://prod.example.com/rest-v1/unprotect`***

Release 1 protect request

Post Request Path: - ***`https://prod.example.com/rest-v1/reprotect`***

Release 2 protect request

Post Request Path: - ***`https://prod.example.com/rest-v1/protect`***

> **Note:**
> The POST requests will change depending on the deployment pattern used.
>
> For more information about the deployment patterns, refer to the section *Architecture and Components*.
>
> For more information about the POST requests for each deployment patterns, refer to the section *Deploying the AP-REST Container on the Kubernetes Cluster*.

7.2 Using the Samples

1. Ensure that the prerequisites mentioned in the subsection *Additional Prerequisites* within the section *Verifying the Prerequisites* are followed.
2. An OpenShift environment is created.

   For more information about creating the cloud runtime environment, refer to the section *Creating the OpenShift Runtime Environment*.

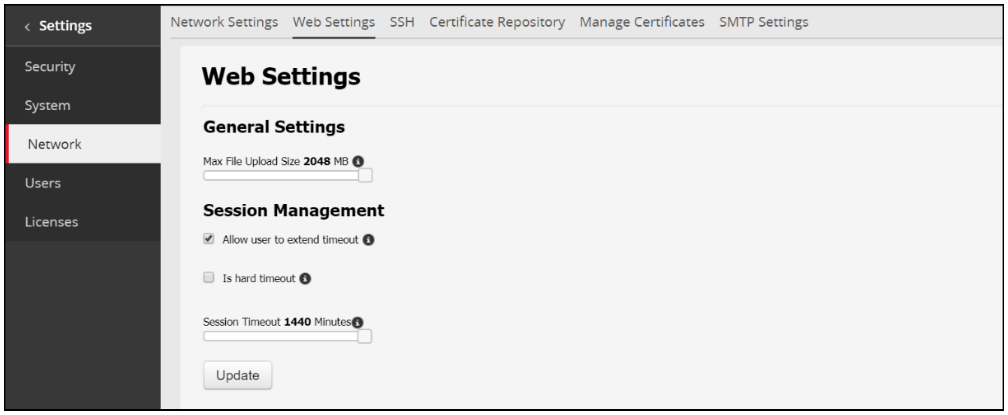
The user must perform the following tasks.

7.2.1 Importing Policy Sample on the ESA

The user needs to perform the following steps to import the *Sample_App_Policy_ V2.tgz* file on the ESA.

➤ To import policy sample on the ESA:

1. Login to the ESA as *admin*.
2. Navigate to **Settings** > **Network** > **Web Settings**.



Max File Upload Size **2048** MB

☑ Allow user to extend timeout

☐ Is hard timeout

Session Timeout **1440** Minutes

Figure 7-1: ESA Web Settings

3. In the **General Settings** section, change the **Max File Upload Size** value to the maximum value.
4. In the **Session Management** section, change the **Session Timeout** value to the maximum value.
5. Click **Update**.
6. Navigate to **Settings** > **System** > **File Upload**.
7. Click **Choose File** to select the *Sample_App_Policy_ V2.tgz* file that you want to upload.
8. Enter the admin password.
9. Navigate to **System** > **Backup and Restore** > **Import**, and select the *Sample_App_Policy_ V2.tgz* file from the drop down list and click **Import**.
10. Provide the password as `admin1234` and click **Import**..

    After successful import, the *Sample_policy* should be available in the *Policies* section.

7.2.2 Creating IMP Packages

For Release 1

Set the following values for creating IMP package for release 1 and create IMP packages:

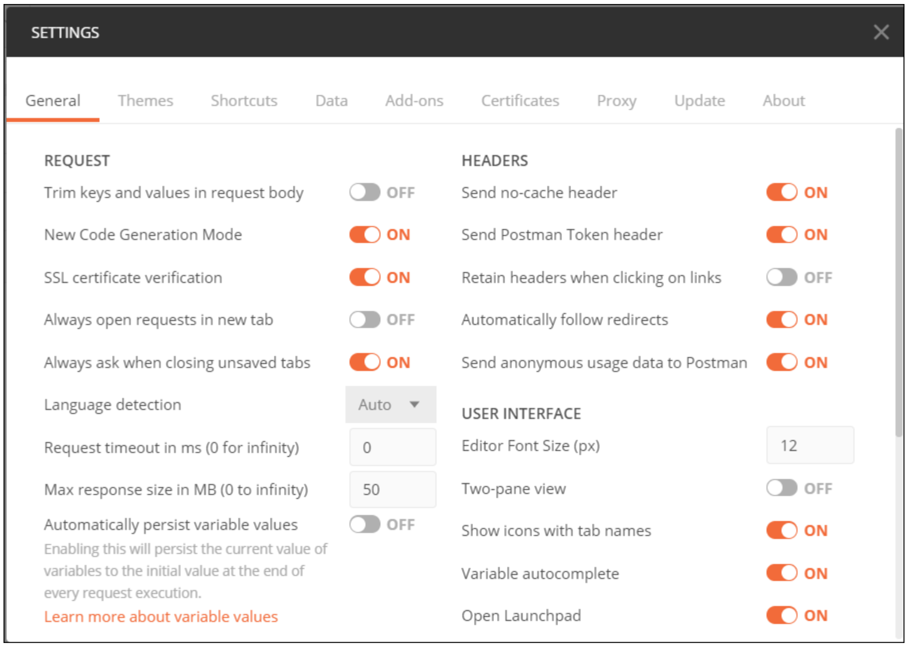
- securityConfigDestination: *VolumeMount*
- policy/filePath: *test/release1/policy*

7.2.3 Importing Certificates to the Postman Client

This section describes the steps to import the CA certificate, the client certificates, and keys to the Postman client, if you want to ensure secure communication between the Postman client and the NGINX container using TLS.

To import certificates to the Postman client:

1. Open the Postman client.
2. In the header of the Postman client, click , and then select **Settings**.
   The **SETTINGS** dialog box appears.



SETTINGS

REQUEST

Trim keys and values in request body	OFF
New Code Generation Mode	ON
SSL certificate verification	ON
Always open requests in new tab	OFF
Always ask when closing unsaved tabs	ON
Language detection	Auto
Request timeout in ms (0 for infinity)	0
Max response size in MB (0 to infinity)	50
Automatically persist variable values	OFF

Enabling this will persist the current value of variables to the initial value at the end of every request execution.
Learn more about variable values

HEADERS

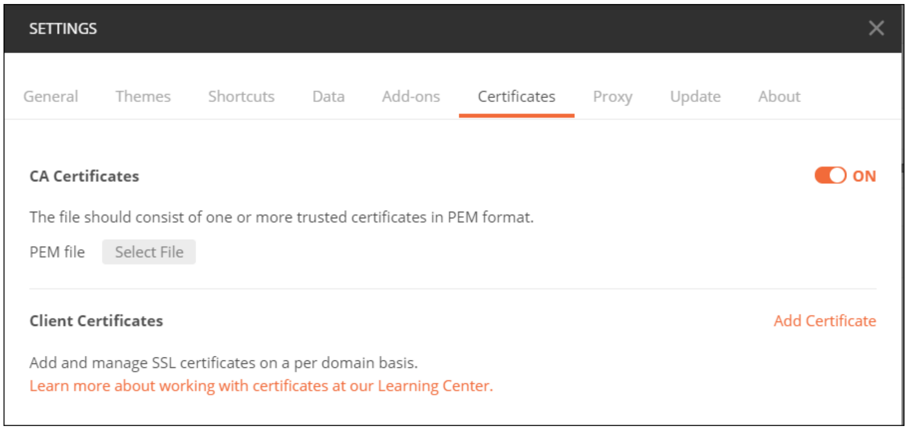
Send no-cache header	ON
Send Postman Token header	ON
Retain headers when clicking on links	OFF
Automatically follow redirects	ON
Send anonymous usage data to Postman	ON

USER INTERFACE

Editor Font Size (px)	12
Two-pane view	OFF
Show icons with tab names	ON
Variable autocomplete	ON
Open Launchpad	ON

Figure 7-2: SETTING Dialog Box

3. Navigate to the **Certificates** tab.



SETTINGS

CA Certificates ON

The file should consist of one or more trusted certificates in PEM format.

PEM file Select File

Client Certificates Add Certificate

Add and manage SSL certificates on a per domain basis.
Learn more about working with certificates at our Learning Center.

Figure 7-3: Certificates Tab

4. In the **CA Certificates** section, click **Select File** and browse for the *iap-ca.crt* file in the *iap-certs* directory that you have created in the section *Creating Certificates and Keys for TLS Authentication*.
5. In the **Client Certificates** section, click **Add Certificate**.
   The **Add Certificate** screen appears.

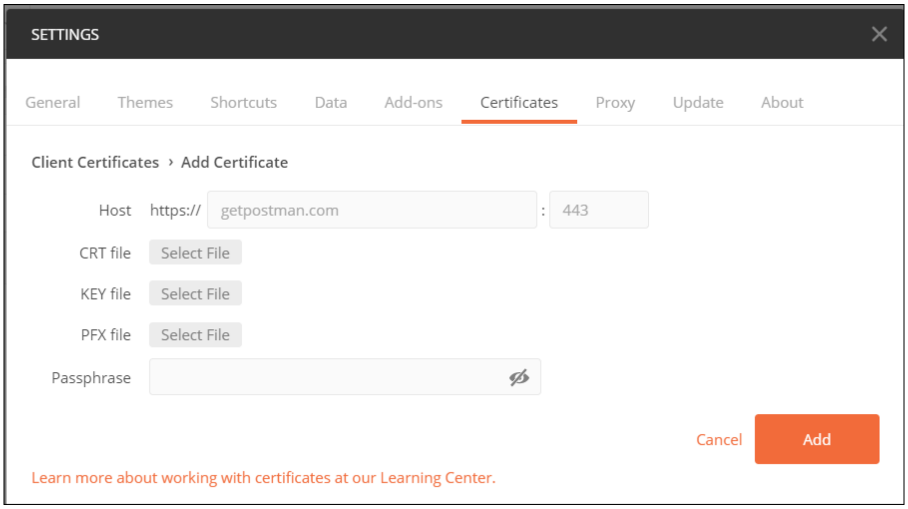


Figure 7-4: Add Certificate Screen

6. In the **Host** field, specify the value as *prod.example.com*.

   You need to specify the ingress port number, which is *8443* by default.
7. In the **CRT file** field, click **Select File** to browse for the *iap-client.crt* file in the *iap-certs* directory.
8. In the **KEY file** field, click **Select File** to browse for the *iap-client.key* file in the *iap-certs* directory.
9. Click **Add** to add the client certificate and key to the Postman client.
10. Repeat steps 5 to 9 for adding client certificates for the host *staging.example.com*.

7.2.4 Deploying Release 1

1. Ensure that the *Values.yaml* file has the following configuration set on the Linux node.

```
blueDeployment:
  enabled: true
  securityConfigSource: VolumeMount
  policy:
    filepath: test/release1/policy
```

2. Deploy Release 1 on the cluster.

   For more information about deploying the AP-REST Container on the Kubernetes Cluster, refer to the section *Deploying the AP-REST Container on the Kubernetes Cluster*.

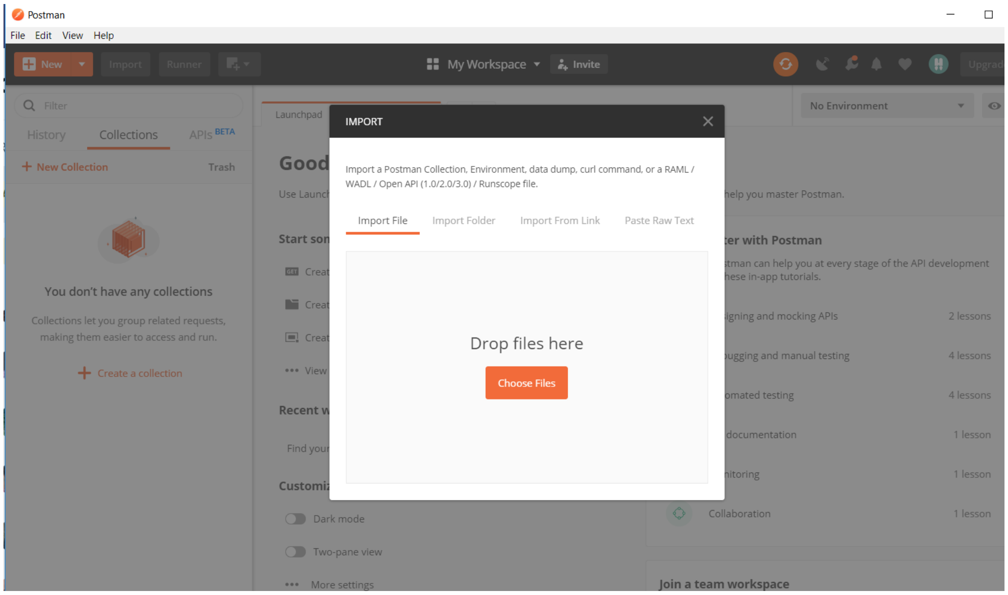
7.2.5 Running the Postman Collection

This section describes the steps for protecting data for Release 1 and Release 2.

The component consists of the Postman JSON file to generate the REST request for protecting data.

For protecting data with Release 1

1. Import the Postman collection.



Import a Postman Collection, Environment, data dump, curl command, or a RAML / WADL / Open API (1.0/2.0/3.0) / Runscope file.

2. After import, the following four collections should be available:
   - Release 1 protect request
   - Release 1 unprotect request
   - Release 1 reprotect request
   - Release 2 protect request
3. Add an entry in your hosts file of the node from where you are making the REST API call.

   For Windows:

   Navigate to `C:\Windows\System32\drivers\etc` directory. Edit the *hosts* file in any text editor, such as Notepad, as an administrator, and add the following entry:

   ```
   <Host name in your Ruleset> <OPENSHIFT_ENDPOINT_IP>
   ```

   For example:

   ```
   prod.example.com <OPENSHIFT_ENDPOINT_IP>
   ```

   For Linux:

   Run the following command `vi /etc/hosts`, and add the following entry:

   ```
   <Host name in your Ruleset> <OPENSHIFT_ENDPOINT_IP>
   ```

   For example:

   ```
   prod.example.com <OPENSHIFT_ENDPOINT_IP>
   ```

4. Select **Release 1 protect request** in AP_REST SAMPLE and click **Send**.

   The user should get response as successful *200 OK* and receive protected data.

5. Select **Release 1 unprotect request** in AP_REST SAMPLE and click **Send**.

   The user should get response as successful *200 OK* and receive unprotected data.

6. Select **Release 1 reprotect request** in AP_REST SAMPLE and click **Send**.

The user should get response as successful *200 OK* and receive reprotected data.

For protecting data with Release 2

1. Add the data element for *Alpha* on the ESA.
2. Create the IMP package R2.

For Release 2

Set the following values for creating the IMP package for release 1:

- securityConfigDestination: *VolumeMount*
- policy/filePath: *test/release2/policy*

1. Run Helm upgrade for switching to Release 2.

   For more information about upgrading the Helm Charts, refer to the section *Upgrading the Helm Charts*.

2. Ensure that the *Values.yaml* file has the following configurations set on the Linux node.

```
greenDeployment:
enabled: true
securityConfigSource: VolumeMount
policy:
filePath: test/release2/policy
```

3. Add an entry in your hosts file of the node from where you are making the REST API call.

   For Windows:

   Navigate to `C:\Windows\System32\drivers\etc` directory. Edit the *hosts* file in any text editor, such as Notepad, as an administrator, and add the following entry:

   ```
   <Host name in your Ruleset> <OPENSHIFT_ENDPOINT_IP>
   ```

   For example:

   ```
   staging.example.com <OPENSHIFT_ENDPOINT_IP>
   ```

   For Linux:

   Run the following command `vi /etc/hosts`, and add the following entry:

   ```
   <Host name in your Ruleset> <OPENSHIFT_ENDPOINT_IP>
   ```

   For example:

   ```
   staging.example.com <OPENSHIFT_ENDPOINT_IP>
   ```

4. Select **Release 2 protect request** in AP_REST SAMPLE and click **Send**.

   The user should get response as successful *200 OK* and receive protected data.

7.2.6 Running the Autoscaling Script

This section provides an overview on the Autoscaling script.

The Autoscaling script is used to issue continuous requests on the AP-REST containers. When the number of REST requests hitting the container are increased, with Horizontal Pod Autoscaling, Kubernetes automatically scales the number of pods based on the CPU utilization observed.

The user can run the autoscaling script from any Linux node, which can connect the external IP for the deployment.

Usage

```
./Sample_App_autoscale.sh <OPENSHIFT_ENDPOINT> <CA certificate path> <Client
certificate path> <Client key path>
```

After running this script, the user can observe that new pods are created to handle to the incoming traffic.

Chapter 8

Appendix D: Application Protector API on REST

This section describes the AP REST protector APIs that are available for protection and unprotection of data. In addition, it lists the error handling capabilities provided by the AP API on REST.

8.1 AP REST APIs

This section describes the AP REST APIs available for protection and unprotection of data.

8.1.1 HTTP GET version

This API displays the version of the AP REST protector API being used.

Request

Method	URI
GET	https://hostname/rest-v1/version

Parameters

Hostname	Resource
Hostname as defined in the AP-REST deployment	/rest-v1/version

Result

This function returns the current version of the AP REST protector API.

Response

Status	Response
200	`{"version":"9.1.0.0.13","components":{"jpepVersion":"9.1.0.0.15","coreVersion":"1.1.0+76.ge82e5.1.1"}}`

Example

```
$ curl 'https://<HostName>/rest-v1/version' --cacert iap-rest-ca.crt --cert iap-rest-client.crt  --key iap-rest-client.key
```

8.1.2 HTTP POST protect

This API returns protected data.

URI

https://hostname/rest-v1/protect

Method

POST

Parameters

Hostname: Host name of the endpoint, as defined in the AP-REST deployment

Resource: The resource to be used, which is */rest-v1/protect*

Result

This API returns protected data.

Example 1 - without external IV and external tweak

```
$ curl --location --request POST 'https://<hostname>/rest-v1/protect' \
--connect-to  "<hostname>:443:<Openshift LoadBalancer>:443"  \
--header 'Content-Type: application/json' \
 --cacert iap-rest-ca.crt --cert iap-rest-client.crt  --key iap-rest-client.key  --data
'{
  "protect": {
    "policyusername": "Uername",
    "dataelementname": "DataElement1",
    "bulk":{
      "id": 1,
      "data": [
        {
          "id": 1,
          "content": "AFAAcgBvAHQAZQBnAHIAaQB0AHkAMQAyADMANA=="
        },
                {
          "id": 2,
          "content": "AFAAcgBvAHQAZQBnAHIAaQB0AHkAMQAyADMANA=="
        }
      ]
    }
  }
}'
```

Response 1 - without external IV and external tweak

The following response appears for the status code 200, if the API is invoked successfully.

```
{
   "protect":{
      "bulk":{
         "id":1,
         "returntype":"success",
         "data":[
            {
               "id":1,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"AGoAZABzAHIAdQBlAGMAagBaAEMAMQAyADMANA=="
            },
            {
               "id":2,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"AGoAZABzAHIAdQBlAGMAagBaAEMAMQAyADMANA=="
            }
         ]
      }
   }
}
```

Example 2 - with external IV

```
curl --location --request POST 'https://<hostname>/rest-v1/protect' \
--connect-to  "<hostname>:443:<Openshift LoadBalancer>:443"  \
--header 'Content-Type: application/json' \
 --cacert iap-rest-ca.crt --cert iap-rest-client.crt --key iap-rest-client.key  --data
'{
  "protect": {
    "policyusername": "UserName",
    "dataelementname": "DataElement1",
    "externaliv":"ZXh0ZXJuYWpdg==",
    "bulk":{
      "id": 1,
      "data": [
        {
          "id": 1,
          "content": "RW5eEN2RGZZaw=="
        },
                {
          "id": 2,
          "content": "cmZBcnJTRg=="
        }
      ]
    }
  }
}'
```

Response 2 - with external IV

The following response appears for the status code 200, if the API is invoked successfully.

```
{
   "protect":{
      "bulk":{
         "id":1,
         "returntype":"success",
         "data":[
            {
               "id":1,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"OG8xZW0QlQ3MQ=="
            },
            {
               "id":2,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"blg2Qm5Ddg=="
            }
         ]
      }
   }
}
```

Example 3 - with external tweak

```
curl --location --request POST 'https://<hostname>/rest-v1/protect' \
--header 'Host: <hostname>' \
--connect-to  "<hostname>:443:<Openshift LoadBalancer>:443"  \
--header 'Content-Type: application/json' \
 --cacert iap-rest-ca.crt --cert iap-rest-client.crt --key iap-rest-client.key  --data
'{
  "protect": {
    "policyusername": "UserName",
    "dataelementname": "DataElement2_FPE",
    "externaltweak":"ZXh0ZXJuYWpdg==",
    "bulk":{
      "id": 1,
      "data": [
        {
          "id": 1,
          "content": "RW5eEN2RGZZaw=="
        },
                {
          "id": 2,
          "content": "cmZBcnJTRg=="
        }
      ]
    }
  }
}'
```

Response 3 - with external tweak

The following response appears for the status code 200, if the API is invoked successfully.

```
{
   "protect":{
      "bulk":{
         "id":1,
         "returntype":"success",
         "data":[
            {
               "id":1,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"MHM4OVpsRndIbA=="
            },
            {
               "id":2,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"VzFsNmd1Ng=="
            }
         ]
      }
   }
}
```

8.1.3 HTTP POST unprotect

This API unprotects the protected data.

URI

https://hostname/rest-v1/unprotect

Method

POST

Parameters

Hostname: Host name of the endpoint, as defined in the AP-REST deployment

Resource: The resource to be used, which is */rest-v1/unprotect*

Result

This API returns unprotected data.

Example 1 - without external IV and external tweak

```
$ curl --location --request POST 'https://<hostname>/rest-v1/unprotect' \
--connect-to  "<hostname>:443:<OpenShift LoadBalancer>:443"  \
--header 'Content-Type: application/json' \
 --cacert iap-rest-ca.crt --cert iap-rest-client.crt  --key iap-rest-client.key  --data
'{
  "unprotect": {
    "policyusername": "UserName",
    "dataelementname": "DataElement1",
    "bulk":{
      "id": 1,
      "data": [
        {
          "id": 1,
          "content": "AFAAcgBvAHQAZQBnAHIAaQB0AHkAMQAyADMANA=="
        },
                {
          "id": 2,
          "content": "AFAAcgBvAHQAZQBnAHIAaQB0AHkAMQAyADMANA=="
        }
      ]
    }
  }
}'
```

Response 1 - without external IV and external tweak

The following response appears for the status code 200, if the API is invoked successfully.

```
{
   "unprotect":{
      "bulk":{
         "id":1,
         "returntype":"success",
         "data":[
            {
               "id":1,
               "returncode":"/rest-v1/returncodes/id/8",
               "returntype":"success",
               "content":"AGwATgBWAEwATAByAFIAUAB2AGcAMQAyADMANA=="
            },
            {
               "id":2,
               "returncode":"/rest-v1/returncodes/id/8",
               "returntype":"success",
               "content":"AGwATgBWAEwATAByAFIAUAB2AGcAMQAyADMANA=="
            }
         ]
      }
   }
}
```

Example 2 - with external IV

```
$ curl --location --request POST 'https://<hostname>/rest-v1/unprotect' \
--connect-to  "<hostname>:443:<OpenShift LoadBalancer>:443"  \
--header 'Content-Type: application/json' \
 --cacert iap-rest-ca.crt --cert iap-rest-client.crt  --key iap-rest-client.key  --data
'{
  "unprotect": {
    "policyusername": "UserName",
    "dataelementname": "DataElement1",
    "externaliv": "ZXh0ZXJuYWpdg==",
    "bulk":{
      "id": 1,
      "data": [
        {
          "id": 1,
          "content": "OG8xZW0QlQ3MQ=="
        },
                {
          "id": 2,
          "content": "blg2Qm5Ddg=="
        }
      ]
    }
  }
}'
```

Response 2 - with external IV

The following response appears for the status code 200, if the API is invoked successfully.

```
{
   "unprotect":{
      "bulk":{
         "id":1,
         "returntype":"success",
         "data":[
            {
               "id":1,
               "returncode":"/rest-v1/returncodes/id/8",
               "returntype":"success",
               "content":"RW5eEN2RGZZaw=="
            },
            {
               "id":2,
               "returncode":"/rest-v1/returncodes/id/8",
               "returntype":"success",
               "content":"cmZBcnJTRg=="
            }
         ]
      }
   }
}
```

Example 3 - with external tweak

```
$ curl --location --request POST 'https://<hostname>/rest-v1/unprotect' \
--connect-to  "<hostname>:443:<OpenShift LoadBalancer>:443"  \
--header 'Content-Type: application/json' \
 --cacert iap-rest-ca.crt --cert iap-rest-client.crt  --key iap-rest-client.key  --data
'{
  "unprotect": {
    "policyusername": "UserName",
    "dataelementname": "DataElement2_FPE",
    "externaltweak": "ZXh0ZXJuYWpdg==",
    "bulk":{
      "id": 1,
      "data": [
        {
          "id": 1,
          "content": "MHM4OVpsRndIbA=="
        },
                {
          "id": 2,
          "content": "VzFsNmd1Ng=="
        }
      ]
    }
  }
}'
```

Response - with external tweak

The following response appears for the status code 200, if the API is invoked successfully.

```
{
   "unprotect":{
      "bulk":{
         "id":1,
         "returntype":"success",
         "data":[
            {
               "id":1,
               "returncode":"/rest-v1/returncodes/id/8",
               "returntype":"success",
               "content":"RW5eEN2RGZZaw=="
            },
            {
               "id":2,
               "returncode":"/rest-v1/returncodes/id/8",
               "returntype":"success",
               "content":"cmZBcnJTRg=="
            }
         ]
      }
   }
}
```

8.1.4 HTTP POST reprotect

This API reprotects the data.

URI

`https://hostname/rest-v1/reprotect`

Method

POST

Parameters

Hostname: Host name of the endpoint, as defined in the AP-REST deployment

Resource: The resource to be used, which is */rest-v1/reprotect*

Result

This API reprotects the data.

Example 1 - without external IV and external tweak

```
$ curl --location --request POST 'https://<hostname>/rest-v1/reprotect' \
--connect-to  "<hostname>:443:<OpenShift LoadBalancer>:443"  \
--header 'Content-Type: application/json' \
 --cacert iap-rest-ca.crt --cert iap-rest-client.crt  --key iap-rest-client.key  --data
'{
  "reprotect": {
    "policyusername": "UserName",
    "olddataelementname": "DataElement1", "newdataelementname": "DataElement2",
    "bulk":{
      "id": 1,
      "data": [
        {
          "id": 1,
          "content": "AFAAcgBvAHQAZQBnAHIAaQB0AHkAMQAyADMANA=="
        },
                {
          "id": 2,
          "content": "AFAAcgBvAHQAZQBnAHIAaQB0AHkAMQAyADMANA=="
        }
      ]
    }
  }
}'
```

Response 1 - without external IV and external tweak

The following response appears for the status code 200, if the API is invoked successfully.

```
{
   "reprotect":{
      "bulk":{
         "id":1,
         "returntype":"success",
         "data":[
            {
               "id":1,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"AFAAcgBvAHQAZQBnAHIAaQB0AHkAMQAyADMANA=="
            },
            {
               "id":2,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"AFAAcgBvAHQAZQBnAHIAaQB0AHkAMQAyADMANA=="
            }
         ]
      }
   }
}
```

Example 2 - with external IV

```
curl --location --request POST 'https://<hostname>/rest-v1/reprotect' \
--connect-to  "<hostname>:443:<Openshift LoadBalancer>:443"  \
--header 'Content-Type: application/json' \
 --cacert iap-rest-ca.crt --cert iap-rest-client.crt --key iap-rest-client.key  --data
'{
  "reprotect": {
    "policyusername": "UserName",
    "olddataelementname": "DataElement1",
    "newdataelementname": "DataElement2",
    "oldexternaliv":"MTIzNDVhYmNzIyQlXiM2Nzg5MFMrTlNBQkNTRA=",
    "newexternaliv":"MTIzNDVhYmNzIyQlXiM2Nzg5MFMrTlNBQkNTRA=",
    "bulk":{
      "id": 1,
      "data": [
        {
          "id": 1,
          "content": "MTA1MTYwNTk1MjE5OTY3OTU="
        },
                {
          "id": 2,
          "content": "MTA1MTYwNTk1MjE5OTY3OTU="
        }
      ]
    }
  }
}'
```

Response 2 - with external IV

The following response appears for the status code 200, if the API is invoked successfully.

```
{
   "reprotect":{
      "bulk":{
         "id":1,
         "returntype":"success",
         "data":[
            {
               "id":1,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"Q09udGFpbmVyVGVhbTEyMzQ1Njc="
            },
            {
               "id":2,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"AFAAcgBvAHQAZQBnAHIAaQB0AHkAMQAyADMANAA1"
            }
         ]
      }
   }
}
```

Example 3 - with external tweak

```
curl --location --request POST 'https://<hostname>/rest-v1/reprotect' \
--header 'Host: <hostname>' \
--connect-to  "<hostname>:443:<Openshift LoadBalancer>:443"  \
--header 'Content-Type: application/json' \
 --cacert iap-rest-ca.crt --cert iap-rest-client.crt --key iap-rest-client.key  --data
'{
  "reprotect": {
    "policyusername": "UserName",
    "olddataelementname": "DataElement1",
    "newdataelementname": "DataElement2",
    "oldexternaltweak":"MTIzNDVhYmNzIyQlXiM2Nzg5MFMrTlNBQkNTRA=",
    "newexternaltweak":"MTIzNDVhYmNzIyQlXiM2Nzg5MFMrTlNBQkNTRA=",
    "bulk":{
      "id": 1,
      "data": [
        {
          "id": 1,
          "content": "MTA1MTYwNTk1MjE5OTY3OTU="
        },
                {
          "id": 2,
          "content": "MTA1MTYwNTk1MjE5OTY3OTU="
        }
      ]
    }
  }
}'
```

Response 3 - with external tweak

The following response appears for the status code 200, if the API is invoked successfully.

```
{
   "reprotect":{
      "bulk":{
         "id":1,
         "returntype":"success",
         "data":[
            {
               "id":1,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"AFAAYQByAGgAbQBoAFAAawBMAGcAZQBaAFgAaABtAGEAcg"
            },
            {
               "id":2,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"ADEAMgAzADQANQA2ADcAOAA5ADA"
            }
         ]
      }
   }
}
```

8.1.5 HTTP Headers

The client should send the required HTTP headers to the server to specify the type of data being sent in the payload. The content type also specifies the type of result being sent by the server to the client.

To send a JSON request and get a JSON response, specify the following HTTP header:

`Content-Type: application/json`

8.2 Error Handling

For record error handling, the *bulk id* and *data id* fields are used, which enable tracking of the errors from the client side.

The following table lists the record error handling status codes, which are sent from the server to the client.

Table 8-1: Record Error Handling Status Codes

<table>
<tr><th>Status Code</th><th>Responses</th></tr>
<tr><td>Success</td><td>

```
{
      "bulk":{
         "id":1,
         "returntype":"success",
         "data":[
            {
               "id":1,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"AGoAZABzAHIAdQBlAGMAagBaAEMAMQAyADMANA=="
            },
            {
               "id":2,
               "returncode":"/rest-v1/returncodes/id/6",
               "returntype":"success",
               "content":"AGoAZABzAHIAdQBlAGMAagBaAEMAMQAyADMANA=="
            }
         ]
      }
   }
```

</td></tr>
<tr><td>Success, with warning</td><td>

```
{
   "bulk":{
      "id":1,
      "returntype":"warning",
      "data":[
         {
            "id":1,
            "returntype":"warning",
            "content":null
         },
         {
            "id":2,
            "returntype":"warning",
            "content":null
         }
      ]
   }
}
```

</td></tr>
<tr><td>Error type of log return code</td><td>

```
{
   "bulk":{
      "id":1,
      "returntype":"error",
      "data":[
         {
            "id":1,
            "message":"Data is too short to be protected/unprotected.",
            "returncode":"/rest-v1/returncodes/id/22",
            "returntype":"error"
         },
         {
            "id":2,
            "message":"Data is too short to be protected/unprotected.",
            "returncode":"/rest-v1/returncodes/id/22",
            "returntype":"error"
         }
      ]
   }
}
```

</td></tr>
</table>

<table>
<tr><th>Status Code</th><th>Responses</th></tr>
<tr><td>Error type of log return code (different)</td><td>

```
{
   "bulk":{
      "id":1,
      "returntype":"error",
      "data":[
         {
            "id":1,
            "message":"Data is too short to be protected/unprotected.",
            "returncode":"/rest-v1/returncodes/id/22",
            "returntype":"error"
         },
         {
            "id":2,
            "returncode":"/rest-v1/returncodes/id/6",
            "returntype":"success",
            "content":"AGoAZABzAHIAdQBlAGMAagBaAEMAMQAyADMANA=="
         }
      ]
   }
}
```

</td></tr>
</table>

8.3 AP REST API Return Codes

When you develop an application using the Application Protector (AP) REST APIs, you may encounter the errors described in this section. You can avoid most of the errors if you accurately use the API.

For more information, refer to the section *AP REST APIs*.

8.3.1 AP REST API Log Return Error Codes

This section lists the log return error codes returned by the AP REST API as a result of policy exceptions. Audits are generated in the ESA for these errors.

Table 8-2: AP REST API Log Return Codes

Code	Error Message	Error Description
1	1, No such user.	The user name could not be found in the policy residing in the shared memory.
2	2, Data element not found.	The data element could not be found in the policy residing in the shared memory.
3	3, Permission denied.	The user does not have the required permissions to perform the requested operation.
5	5, Integrity check failed.	The data integrity check failed when decrypting using a Data Element with CRC enabled.
6	6, Protect success.	The operation to protect the data was successful.
7	7, Protect failed.	The operation to protect the data failed.
8	8, Unprotect success.	The operation to unprotect the data was successful.
9	9, Unprotect failed.	The operation to unprotect the data failed.
10	10, Policy check OK.	The user has the required permissions to perform the requested operation. This return code ensures a verification and no data is protected or unprotected.

Code	Error Message	Error Description
11	11, Inactive key id used.	The operation to unprotect the data was successful using an inactive Key ID.
12	12, Invalid parameter.	Input parameters are either NULL or not within allowed limits.
13	13, Internal error.	Internal error occurring in a function call after the PEP provider has been opened. For example: - `failed to get mutex / semaphore, - unexpected null param`.
14	14, Failed to load key.	A key for a data element could not be loaded from shared memory into the Crypto engine.
15	15, Tweak input is too long.	Tweak input is too long.
17	17, Init failed.	The PEP server failed to initialize, which is a fatal error.
19	19, Unsupported tweak action for the specified fpe dataelement	The external tweak is not supported for the specified FPE data element.
20	20, Out of memory.	Failed to allocate memory.
21	21, Buffer too small.	The input or output buffer is very small.
22	22, Input too short.	The data is too short to be protected or unprotected.
23	23, Input too long.	The data is too long to be protected or unprotected.
25	25, User name too long.	The user name is longer than the maximum supported length of the user name that can be used for protect or unprotect operations.
26	26, Unsupported.	The algorithm or action for the specific data element is unsupported.
31	31, Empty policy.	The policy residing in the shared memory is empty.
39	39, Policy locked.	The policy residing in the shared memory is locked. This error can be caused by a *Disk Full* alert.
40	40, License expired.	The license is invalid or the current date is beyond the license expiry date.
41	41, Method restricted.	The use of the Protection method is restricted by the license.
42	42, Invalid license.	The license is invalid or the time is prior to the start of the license tenure.
44	44, Invalid format.	The content of the input data is invalid.

8.3.2 AP REST API PEP Result Codes

This section lists the PEP result codes returned by AP REST APIs as a result of system exceptions. Audits are not generated in the ESA for these errors.

Table 8-3: AP REST API PEP Result Codes

Code	Error Message	Error Description
-1	-1, Invalid parameter	The parameter is invalid.
-7	-7, Error when parsing contents, e.g.	The error occurred when the contents were parsed.
-8	-8, Not found!	The search operation was not successful.
-16	-16, Buffer is too small	The buffer size is very small.
-43	-43, Invalid format	The format is invalid.
-46	-46, Requesting service/function on an object that is not initialized	The service requested or function is performed on an object that is not initialized.
-47	-47, Policy locked for some reason	The Policy is locked.

8.4 AP REST HTTP Response Codes

This section lists the response codes generated for the HTTP REST requests sent to the AP REST APIs. It also specifies the corresponding audit code generated in the logs.

Table 8-4: AP REST HTTP Response Codes

Scenario	Operation	*Audit Code in Logs*	HTTP Response Code
Failed to decode Base64	• Protect • Unprotect • Reprotect	No audit code generated	400
User name not present in policy	Protect	1	401
User name not present in policy	Unprotect	1	200
User name not present in policy	Reprotect	1	401
Data element not found in policy	• Protect • Unprotect	2	400
Data element not found in policy	Reprotect	2	400
No access to data element	Protect	3	403
No access to data element	Unprotect (The API returns an Exception)	3	403
No access to data element	Unprotect (The API returns the protected value)	3	200
No access to data element	Unprotect (The API returns a Null value)	3	200
No access to data element	Reprotect	3	403
Integrity check failed	Unprotect	5	400
Invalid parameter	• Protect • Unprotect • Reprotect	12	400
Failed to load the data encryption key	Unprotect	14	400
Input too short	• Protect • Unprotect • Reprotect	22	200
Input too long	• Protect • Unprotect • Reprotect	23	200
Unsupported algorithm or unsupported action for the specific data element	Unprotect	26	400
Invalid format	• Protect • Unprotect • Reprotect	44	200
Any other *policy exception errors* apart from the ones already listed in this table **Important:** This scenario is only applicable for the 9.1.0.0 release and	Any	Any	400

Scenario	Operation	*Audit Code in Logs*	HTTP Response Code
not for the 7.1 MR4 and 9.0.0.0 releases.			

Important: The HTTP response codes generated for the 9.1.0.0 release differ from the ones that are generated for the 7.1 MR4 and 9.0.0.0 releases.

Chapter 9

Appendix E: Using the Dockerfile to Build Custom Images

This section explains how to use the Dockerfiles, which are provided as part of the installation package, to build custom images for the Get ESA Policy, Init, and AP-REST containers. You can use the custom image instead of the default images that are provided by Protegrity as part of the installation package. This enables you to use a base image of your choice, instead of using the default RHEL Universal Base Image, which is used as the default base image for the Protegrity images.

9.1 Creating Custom Images Using Dockerfile

This section describes the steps for creating custom images for the Get ESA Policy, Init, and AP-REST containers, using the Dockerfile provided with the installation package.

➤ To create custom images:

1. Download the installation package.

   For more information about downloading the installation package, refer to the section *Downloading the Installation Package*.

   > **Important:** The dependency packages required for building the Docker images are specified in the *HOW-TO-BUILD-DOCKER-IMAGES* file, which is a part of the installation package. You must ensure that these dependency packages can be downloaded either from the Internet or from your internal repository.
   >
   > For more information about the *HOW-TO-BUILD-DOCKER-IMAGES* file, refer to the section *Verifying the Prerequisites*.

2. Perform the following steps to build a Docker image for the Get ESA Policy container.

   a. Run the following command to extract the files from the *REST-GET-ESA-POLICY_OPENSHIFT_SRC_<component_version>.tgz* file.

   ```
   tar -xvf REST-GET-ESA-POLICY_OPENSHIFT_SRC_<component_version>.tgz
   ```

   The following files are extracted:

   - *GET-ESA-POLICY_DOCKERFILE_<version_number>*
   - *PepServerSetup_Linux_x64_<version_number>.iap.tgz*
   - *HealthCheckSetup_Linux_x64_<version_number>.tgz*
   - *PtyShmCatSetup_Linux_x64_<version_number>.tgz*

   b. Edit the *GET-ESA-POLICY_DOCKERFILE_<version_number>* file and in the following code snippet, replace the placeholder *RHEL-MINIMAL-UBI* with the name of the repository from where you want to download the custom RHEL UBI.

      ```
      FROM RHEL-MINIMAL-UBI
      ```

   c. Run the following command in the directory where you have extracted the contents of the *REST-GET-ESA-POLICY_OPENSHIFT_SRC_<component_version>.tgz* file.

      ```
      docker build -t <image-name>:<image-tag> -f GET-ESA-POLICY_DOCKERFILE_<version_number> .
      ```

      For more information the Docker build command, refer to the *Docker* documentation.

      For more information about tagging an image, refer to the *OpenShift documentation*.

   d. Run the following command to list the Get ESA Policy container image.

      ```
      docker images
      ```

   e. Push the Get ESA Policy container image to your preferred Container Repository.

      For more information about pushing an image to the repository, refer to the *OpenShift documentation*.

3. Perform the following steps to build a Docker image for the Init container.

   a. Run the following command to extract the files from the *REST-INIT-CONTAINER_OPENSHIFT_SRC_<component_version>.tgz* file.

      ```
      tar -xvf REST-INIT-CONTAINER_OPENSHIFT_SRC_<component_version>.tgz
      ```

      The following files are extracted:

      - *INIT-CONTAINER_DOCKERFILE_<version_number>*
      - *PtyShmPutSetup_Linux_x64_<version_number>.tgz*

   b. Edit the *INIT-CONTAINER_DOCKERFILE_<version_number>* file and in the following code snippet, replace the placeholder *RHEL-MINIMAL-UBI* with the name of the repository from where you want to download the custom RHEL UBI.

      ```
      FROM RHEL-MINIMAL-UBI
      ```

   c. Run the following command in the directory where you have extracted the contents of the *REST-INIT-CONTAINER_OPENSHIFT_SRC_<component_version>.tgz* file.

      ```
      docker build -t <image-name>:<image-tag> -f INIT-CONTAINER_DOCKERFILE_<version_number> .
      ```

      For more information the Docker build command, refer to the *Docker* documentation.

      For more information about tagging an image, refer to the *OpenShift documentation*.

   d. Run the following command to list the Init container image.

      ```
      docker images
      ```

   e. Push the Init container image to your preferred Container Repository.

      For more information about pushing an image to the repository, refer to the *OpenShift documentation*.

4. Perform the following steps to build a Docker image for the AP-REST container.

   a. Run the following command to extract the files from the *REST-SRC_OPENSHIFT_<component_version>.tgz* file.

      ```
      tar -xvf REST-SRC_OPENSHIFT_<component_version>.tgz
      ```

      The following files are extracted:

      - *REST_RHUBI_DOCKERFILE_<version_number>*
      - *ImmutableApplicationProtectorRESTLinux_x64_<version_number>.iap.tgz*
      - *HealthCheckSetup_Linux_x64_<version_number>.tgz*

- *docker-entrypoint.sh*

b. Edit the *REST_RHUBI_DOCKERFILE_<version_number>* file and in the following code snippet, replace the placeholder *RHEL-MINIMAL-UBI* with the name of the repository from where you want to download the custom RHEL UBI.

```
FROM RHEL-MINIMAL-UBI
```

c. Run the following command in the directory where you have extracted the contents of the *REST-SRC_OPENSHIFT_<component_version>.tgz* file.

```
docker build -t <image-name>:<image-tag> -f
REST_RHUBI_DOCKERFILE_<version_number> .
```

For more information the Docker build command, refer to the *Docker* documentation.

For more information about tagging an image, refer to the *OpenShift documentation*.

d. Run the following command to list the AP-REST container image.

```
docker images
```

e. Push the AP-REST container image to your preferred Container Repository.

For more information about pushing an image to the repository, refer to the *OpenShift documentation*.