Protegrity Application Protector Go Immutable Policy User Guide for OpenShift Gen2 9.1.0.0

Created on: Nov 19, 2024

Copyright

Table of Contents

Copyright.. 2

Chapter 1 Protegrity Security Operations Cluster using Kubernetes........................ 6
1.1 Business Problem........................6
1.2 Protegrity Solution........................6
1.3 Architecture and Components........................ 7

Chapter 2 Immutable Application Protector (IAP) Deployment Model........................ 11
2.1 Verifying the Prerequisites........................11
2.2 Downloading the Installation Package........................ 12
2.3 Initializing the Linux Instance........................13
2.4 Creating Certificates and Keys for TLS Authentication........................14
2.5 Creating the Cloud Runtime Environment........................ 15
2.5.1 Creating the OpenShift Runtime Environment........................ 15
2.5.1.1 Logging in to the OpenShift Environment........................15
2.6 Deploying the Containers with Security Context Constraints (SCC) and Role-Based Access Control (RBAC)................ 16
2.6.1 Applying the SCC........................17
2.6.2 Applying RBAC........................ 18
2.6.3 Retrieving the Policy from the ESA........................ 19
2.6.4 Copying the Policy Package to a Mount Directory on the NFS Server........................21
2.6.5 Deploying the Sample Application Container on the Kubernetes Cluster........................ 22
2.6.6 Upgrading the Helm Charts........................26
2.7 Deploying the Containers without RBAC........................ 32
2.7.1 Applying the SCC........................32
2.7.2 Retrieving the Policy from the ESA........................ 34
2.7.3 Copying the Policy Package to a Mount Directory on the NFS Server........................35
2.7.4 Deploying the Sample Application Container on the Kubernetes Cluster........................ 37
2.7.5 Upgrading the Helm Charts........................40

Chapter 3 Accessing Logs Using Splunk........................ 47
3.1 Understanding the Logging Architecture........................ 47
3.2 Setting Up Splunk........................48
3.3 Configuring Splunk........................ 49
3.4 Deploying the Splunk Connect for Kubernetes Helm Chart on the Kubernetes Cluster........................54

Chapter 4 Protecting Data Using the Sample Application Container Deployment........................58
4.1 Running Security Operations........................58
4.2 Viewing Logs in Splunk........................ 59
4.3 Autoscaling the Protegrity Reference Deployment........................ 61

Chapter 5 Appendix B: Sample Application Helm Chart Details........................62
5.1 Chart.yaml........................ 62
5.2 Values.yaml........................62
5.2.1 Image Pull Secrets........................ 62
5.2.2 Name Override........................ 63
5.2.3 Container Image Repository........................63
5.2.4 Resources........................63
5.2.5 Pod Service Account........................ 64
5.2.6 Liveliness and Readiness Probe Settings........................64
5.2.7 Pod Security Context........................64
5.2.8 Container Security Context........................ 64
5.2.9 Persistent Volume Claim........................ 65
5.2.10 PBE Secrets........................ 65

5.2.11 Deployment.......... 65
5.2.12 Autoscaling.......... 66
5.2.13 Service Configuration.......... 67
5.2.14 Routes Configuration.......... 67

Chapter 6 Appendix C: Using the Dockerfile to Build Custom Images.......... 69
6.1 Creating Custom Images Using Dockerfile.......... 69

Chapter 7 Appendix D: Applying the NSS Wrapper to the Customer Application Container Image.......... 71

Chapter 1

Protegrity Security Operations Cluster using Kubernetes

1.1 Business Problem
1.2 Protegrity Solution
1.3 Architecture and Components

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 Go form factor:
  - The delivery form factor for cloud deployments is a container. Protegrity has developed an AP Go Container form factor based on the Application Protector form factor that has been delivered for several years.
  - The AP Go Container is a standard Docker Container that is familiar and expected in cloud deployments. Customers can create their own container image by integrating their applications with the AP Go libraries.
  - The AP Go Container form factor makes the container a lightweight deployment of the AP Go.
- 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 Go Container with Policy instantiating a new one.
  - Changes to Policy or the AP Go 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 autoscale 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 autoscale when the node thresholds are reached.
- 3rd Party Integration
    - 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 autoscale, when needed.

1.3 Architecture and Components

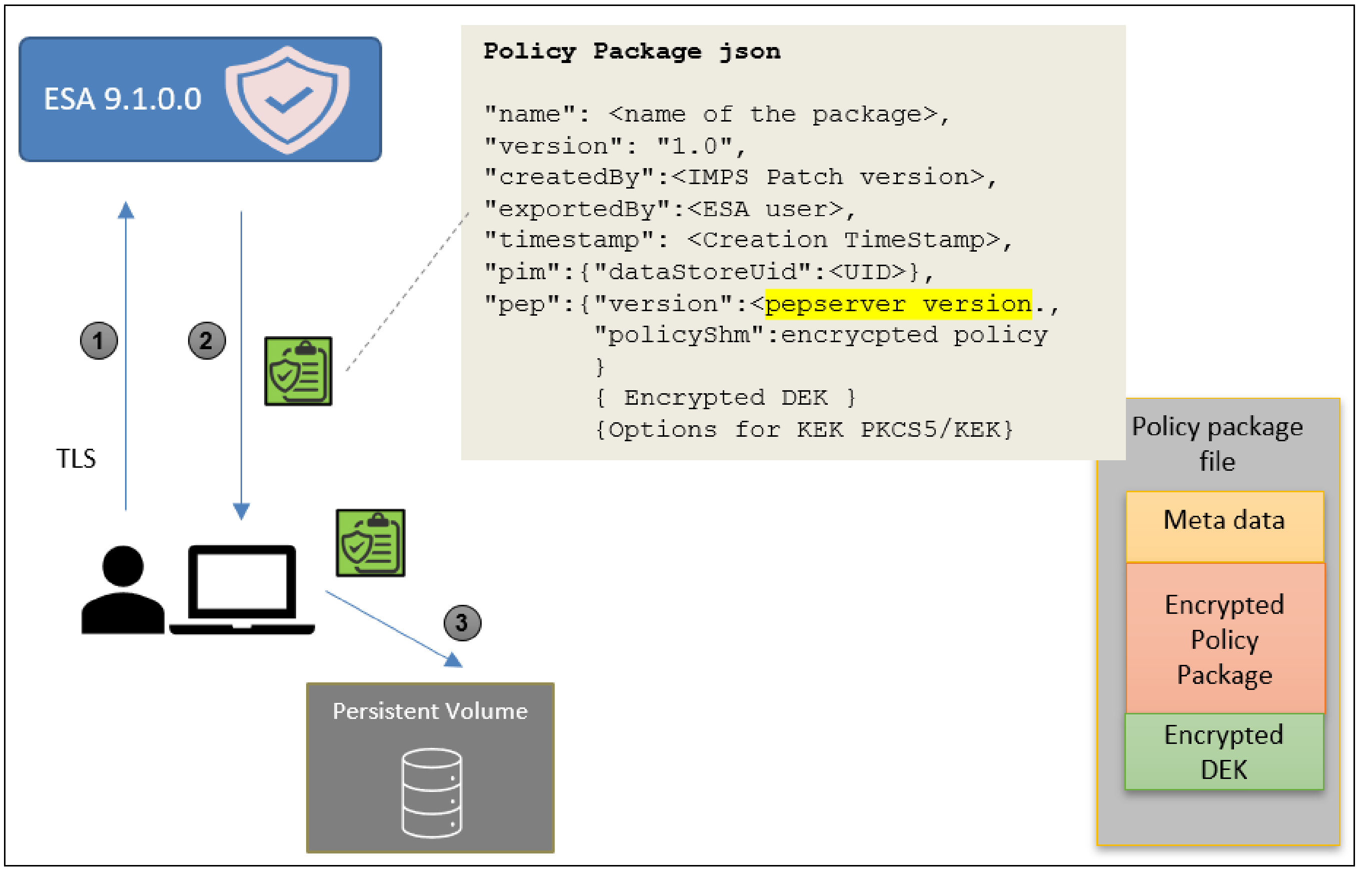
This section will describe the architecture, the individual components, and the workflow of the Protegrity Immutable Application Protector solution.

The IAP Gen2 deployment consists of the following two stages:

- Stage 1: Running a cURL command to fetch the ESA policy and create the policy snapshot.
- Stage 2: Deploying the Sample Application on a Kubernetes cluster, which uses the policy snapshot created in stage 1.

Note: In the production environment, you will deploy the Customer Application instead of the Sample Application.

The following figure describes the architecture diagram and the steps for Stage 1 - Retrieving the policy from the ESA.



```
Policy Package json

"name": <name of the package>,
"version": "1.0",
"createdBy":<IMPS Patch version>,
"exportedBy":<ESA user>,
"timestamp": <Creation TimeStamp>,
"pim":{"dataStoreUid":<UID>},
"pep":{"version":<pepserver version.,
       "policyShm":encrycpted policy
       }
       { Encrypted DEK }
       {Options for KEK PKCS5/KEK}
```

Figure 1-1: Stage 1: Fetching the ESA Policy and Creating the Policy Package

The following steps describe the workflow of retrieving the policy from the ESA.

1. The user sends the IMPS API request to the ESA to create a policy package. You can choose to encrypt the policy package using a Passphrase-Based Encryption or a Key Management System (KMS). You need to specify the encryption parameters as part of the IMPS API request body.
2. The ESA generates a JSON file for the policy package, which contains metadata and the encrypted policy package.
3. The user needs to upload the policy package to an Object Store or a File Store.

The following figure describes the architecture diagram and the steps for Stage 2 - Deploying the Sample Application on a Kubernetes cluster.

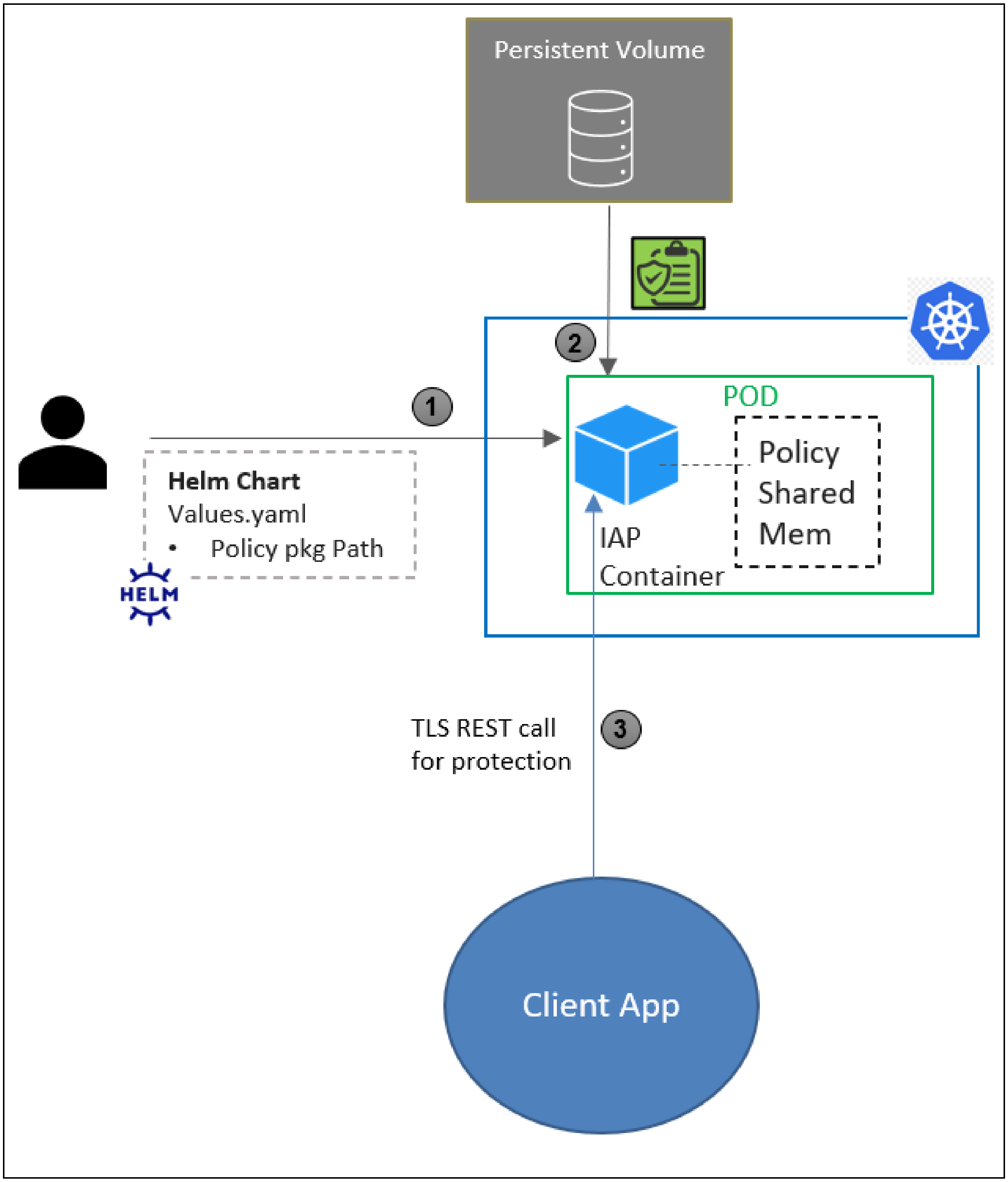


Figure 1-2: Stage 2: Deploying the Sample Application Container

The following steps describe the workflow of deploying the Sample Application container on a Kubernetes cluster.

1. The user runs the Helm chart to deploy the Sample Application to the Kubernetes cluster. The Sample Application is integrated with the Application Protector Go libraries.

   Note: The Sample Application has been used for demonstration purposes. You can choose to create a custom image by integrating your custom application with the Application Protector Go libraries.

2. The Sample Container reads the immutable policy snapshot from a persistent volume, decrypts it using the Kubernetes secret configured for Passphrase-Based Encryption or KMS-related secrets, and stores the policy in the shared memory of the Pod.
3. The Client Application sends a request to the Sample application over REST for protecting, unprotecting, and reprotecting data. The Sample Application internally sends the request to the Application Protector Go, and then returns the protected, unprotected, or reprotected value to the client application.

Chapter 2

Immutable Application Protector (IAP) Deployment Model

2.1 Verifying the Prerequisites
2.2 Downloading the Installation Package
2.3 Initializing the Linux Instance
2.4 Creating Certificates and Keys for TLS Authentication
2.5 Creating the Cloud Runtime Environment
2.6 Deploying the Containers with Security Context Constraints (SCC) and Role-Based Access Control (RBAC)
2.7 Deploying the Containers without RBAC

This section describes the IAP Reference Deployment model that customers can use to deploy the Sample Application containers on a Kubernetes cluster.

2.1 Verifying the Prerequisites

Ensure that the following prerequisites are met:

Reference Solution Deployment

- *IAP_RHUBI-ALL-64_x86-64_ALL.K8S.GO-1_<IAP_version>.tgz* – Installation package for the IAP Gen2 Deployment. This package contains the following packages:
  - *APGoIMSetup_<OS>_x64_<App_Protector_version>.tgz* - Immutable AP Go installation package.
  - *IAP-SAMPLE-APP-HELM_ALL-ALL-ALL_x86-64_K8_<version>.tgz* – Package containing the Helm charts, which are used to deploy the sample application on the Kubernetes cluster.
  - *IAP-SAMPLE-APP_SRC_<version_number>.tgz* - Package containing the Dockerfile that can be used to create a custom image for the Sample Application 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*.
  - *PolicyUtil_Linux_x64_<version>.tgz* - Binary file containing the scripts used to retrieve the policy package from the ESA.
  - *HOW-TO-BUILD* - Text file specifying how to build the Sample Application and 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.

For more information about defining a security policy, refer to the section *Creating and Deploying Policies* in the *Protegrity Policy Management Guide 9.1.0.5*.

- Trusted Application - Create a Trusted Application with the name as *restapi* and user name as *ptyituser*.

  Note: If you are deploying a Customer Application container, then you must specify the application name and user name, which is defined in the Customer Application container image, that is used to run the Application Protector Go.

  Important: If you are deploying a Customer Application container, then you must apply the NSS wrapper, which is a *nss-wrapper* library for the Name Service Switch (NSS) API, to the Customer Application container image.

  For more information about applying the NSS wrapper to the Customer Application container image, refer to the section *Appendix D: Applying the NSS Wrapper to the Customer Application Container Image*.

  For more information about the *nss_wrapper* library, refer to the section *The nss_wrapper library*.

  For more information about setting up a Trusted Application, refer to the section *Working with Trusted Applications* in the *Protegrity Policy Management Guide 9.1.0.5*.

- ESA user - Create an ESA user that will be used to invoke the IMP REST API for retrieving the security policy and the certificates from the ESA 9.1.0.0. Ensure that the user is assigned the *IMP Export* role.

  For more information about assigning roles, refer to the section *Managing Roles* in the *Protegrity Appliances Overview Guide 9.1.0.5*.

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 Sample Application container image.

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_RHUBI-ALL-64_x86-64_ALL.K8S.GO-1_<IAP_version>.tgz* file on the Linux instance.
2. Run the following command to extract the files from the *IAP_RHUBI-ALL-64_x86-64_ALL.K8S.GO-1_<IAP_version>.tgz* file.

   ```
   tar -xvf IAP_RHUBI-ALL-64_x86-64_ALL.K8S.GO-1_<IAP_version>.tgz
   ```

   The following files are extracted:

   - *APGoIMSetup_<OS>_x64_<App_Protector_version>.tgz*
   - *IAP-SAMPLE-APP-HELM_ALL-ALL-ALL_x86-64_K8_<version>.tgz*
   - *IAP-SAMPLE-APP_SRC_<version_number>.tgz*
   - *PolicyUtil_Linux_x64_<version>.tgz*
   - *HOW-TO-BUILD*
   - *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 4.9*
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 Sample Application container.

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

For example:

```
tar -xvf IAP-SAMPLE-APP-HELM_ALL-ALL-ALL_x86-64_OPENSHIFT.K8S_<version>.tgz
```

The extracted package contains the *sample-app* 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

5. 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:

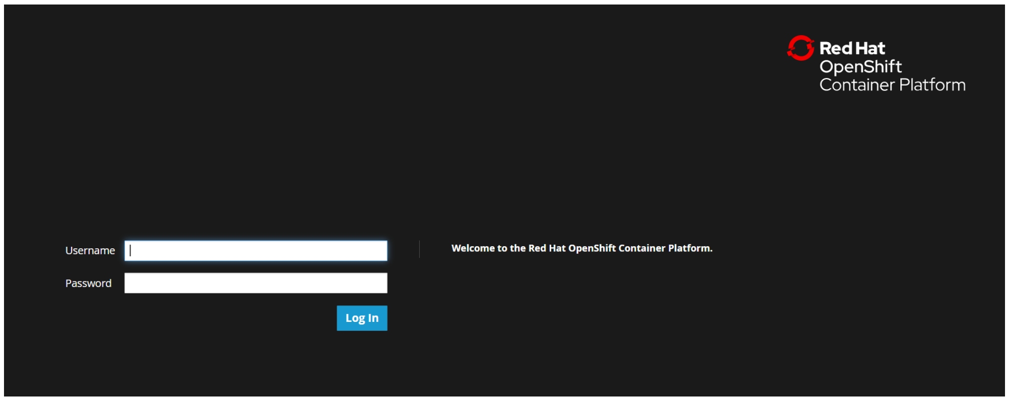
1. Access OpenShift at the following URL:

   https://<OpenShift Console URL>/dashboards

   The OpenShift home screen appears.

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

The Sign in screen appears.

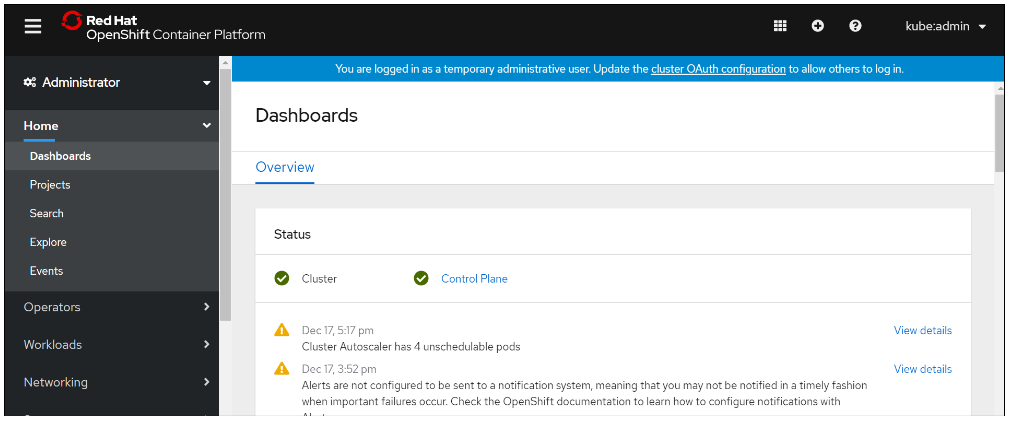


Welcome to the Red Hat OpenShift Container Platform.

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 Sample Application container 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 a Kubernetes service account that will install the Helm charts for deploying the Sample Application container.

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>
   ```

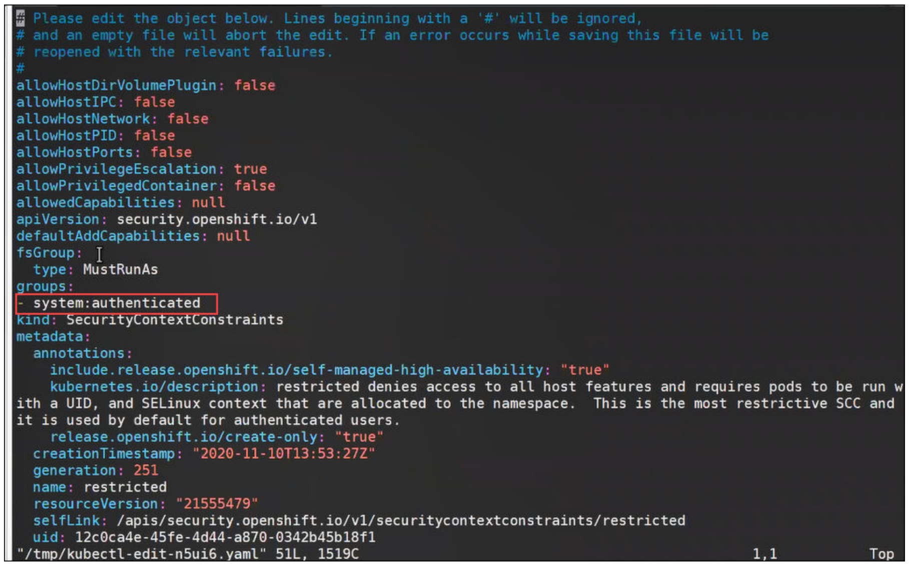
   For example:

   ```
   oc new-project iap
   ```

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 Sample Application 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 the 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 Sample Application, and edit the *pty-scc.yaml* file to replace the *<NAMESPACE>* placeholder in the following snippet with the namespace where the Sample Application will be deployed.

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

   For example:

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

   For more information about the extracted Helm charts, refer to the *step 4* 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 Sample Application container.

   > **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 Sample Application container.

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

      For example:

      ```
      oc create serviceaccount iap-deployer --namespace iap
      ```

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

   b. 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 iap-deployer --namespace iap -o jsonpath='{.secrets}'
```

The output displays the following token names.

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

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

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

```
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
-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
```

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

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

2. Navigate to the directory where you have extracted the Helm charts for deploying the Sample Application container, 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-op.yaml -n <Namespace>
```

For example:

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

For more information about the extracted Helm charts, refer to the *step 4* of the section *Initializing the Linux Instance*.

2.6.3 Retrieving the Policy from the ESA

This section describes how to invoke the Immutable Service APIs to retrieve the policy package using the ESA.

For more information about the Immutable Service APIs, refer to the section *Appendix B: APIs for Immutable Protectors* in the *Protegrity APIs, UDFs, Commands Reference Guide 9.1.0.5*.

➤ To retrieve the policy package from the ESA:

1. Run the following command to verify the PEP server versions that are supported by the Immutable Service on the ESA.

```
curl -v -k -u "<ESA user>:<ESA password>" https://<ESA IP>/pty/v1/imp/version
```

The command returns the following output.

```
{
  "version" : "1.1.0",
  "buildVersion" : "1.1.0+7.g307f.1.1",
  "pepVersions" : [ "1.1.0+85", "1.2.0+16" ]
}
```

Note: Ensure that the PEP server version returned by the IMP version API matches the PEP server version specified in the *manifest.json* file in the installation package.

For more information about the installation package, refer to the section *Downloading the Installation Package*.

2. If you want to download the policy package from the ESA and encrypt the policy package using a passphrase and a salt, then run the following command.

```
curl -v -k -u "<ESA user>:<ESA password>" https://<ESA IP>/pty/v1/imp/export -H
"Content-Type: application/json" -o policy_package_key.tgz --data
'{"name":"policypackage","pepVersion":"1.2.0+16","dataStoreName":"<Value>","emptySt
ring":"<Value>","caseSensitive":"<Value>","kek":{"pkcs5":
{"password":"<Password>","salt":"<Salt>"}}}'
```

For example:

```
curl -v -k -u "IMPUser:<Password>" https://10.49.1.218/pty/v1/imp/export -H
"Content-Type: application/json" -o policy_package_key.json --data
'{"name":"policypackage","pepVersion":"1.2.0+16","dataStoreName":"<Value>","emptySt
ring":"<Value>","caseSensitive":"<Value>","kek":{"pkcs5":
{"password":"<Password>","salt":"<Salt>"}}}'
```

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.

The encrypted policy package is downloaded to your machine.

Important: Ensure that the user is assigned the **Export IMP** permission through roles.

For more information about assigning permissions to user roles, refer to the section *Managing Roles* in the *Appliances Overview 9.1.0.5* guide.

For more information about managing users, refer to the section *Managing Users* in the *Appliances Overview 9.1.0.5* guide.

For more information about the Export API, refer to the section *Sample Immutable Service API - Exporting Policy from a PEP Server Version* in the *Protegrity APIs, UDFs, Commands Reference Guide 9.1.0.5*.

Important: Ensure that the password complexity 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*

3. If you want to download the policy package from the ESA and encrypt the policy package using a KMS, then run the following command.

```
curl -v -k -u "<ESA user>:<ESA password>" https://<ESA IP>/pty/v1/imp/export -H
"Content-Type: application/json" -o <Policy_package_name>.tgz --data
'{"name":"policypackage","pepVersion":"1.2.0+16","dataStoreName":"<value>","emptySt
ring":"<value>","caseSensitive":"<value>","kek":{"publicKey": {"value":"Public Key
of the AWS Customer Key used to encrypt the policy"}}}'
```

For example:

```
curl -v -k -u "IMPUser:<password>" https://10.49.1.218/pty/v1/imp/export -H
"Content-Type: application/json" -o policy_package_key.tgz --data
'{"name":"policypackage","pepVersion":"1.2.0+16","dataStoreName":"<value>","emptySt
ring":"<value>","caseSensitive":"<value>","kek":{"publicKey": {"value":"-----BEGIN
PUBLIC KEY-----
\nMIIBojANBgkqhkiG9w0BAQEFAAOCAY8AMIIBigKCAYEAjwxqqrrNQ1kV84GEfYLR\nVK/
SyOkGTs7kxEImEYMGtugSal2G9m7hFdVlWhHH/OvLDHwdOYe6GLXHVwycfbM/
\nS6pzQPS5ujzyJEaLWYAfcRgAHi9g8GvazDiv34Z+VO5FRbJzP9u9/23J4hExc+e1\nTezKsoBj6Ypx/
zBkE4dGqdamfHJUF3ptB82nhJT/Pth15ejL/SVrD/
q8sORU838O\negcIfQYmd5ziZJgbWLVzFaM9QZXH9hD/
0JxAhqfHP3Fnhmc5MDVdg7D0SQkufIjf\nu7djMy2I3ZRMgnkGxjq8PuBUFaH2s6DvAa4e6K0ppGjFoByVV
e2tumrUVEvJg2EM\ndD/
Pl15kzLelbDKjxrI8T1D8cWuGh43wf1xC+uRZsfoMSgwDWdpBmaOFP+q2k0d4\nrQWKIgZw71mxYADPKBAU
UwMgD/aREuvTDlpvpl0Vk2Y5i/4Xx/fK7GV5DCdy9TFM\nZgzAA/
O9z5tuVzqcgodMu06+iqtXdTUoOafA+BTvSIXLAgMBAAE=\n-----END PUBLIC
KEY-----","algorithm":"RSA_OAEP_SHA256"}}}'
```

Note: Ensure that the new line in the public key is replaced with the *\n* character.

The policy package is downloaded to your machine.

4. Copy the policy package file to an NFS server.

2.6.4 Copying the Policy Package to a Mount Directory on the NFS Server

This section describes how to create a mount point on the NFS server, where you can copy the policy package.

To deploy a release on the Kubernetes cluster:

Before you begin

Ensure that you have set up an NFS server with an NFS share.

1. On the Linux instance, navigate to the location where you have extracted the Helm charts to deploy the Sample Application container.

   For more information about the extracted Helm charts, refer to the *step 4* 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 sample-app/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 amount of storage.

   ```
   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 sample-app/nfs-pvc.yaml -n iap
   ```

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. Copy the policy package, which you have retrieved in the section *Retrieving the Policy from the ESA*, to the mount directory.

2.6.5 Deploying the Sample Application Container on the Kubernetes Cluster

This section describes how to deploy the Sample Application 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 Sample Application container.

   For more information about the extracted Helm charts, refer to the *step 4* 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 *iap-deployer* service account.
   a. Run the following command to create user credentials in the *kube-config* file, using the token created for the service account to deploy the Sample Application container in *step 1d* of the section *Applying RBAC*.
      ```
      oc config set-credentials <username> --token <TOKEN from step 1d 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 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. If you have used Passphrase-Based Encryption to encrypt the policy package, then run the following command to create a passphrase secret, which is used by the Sample Application container to decrypt the policy.
   ```
   oc create secret generic pbe-secret --from-literal='passphrase=<passphrase>' --
   from-literal='salt=<salt>' -n iap
   ```
   Ensure that you specify the same passphrase and salt that you have used in *step 2* of the section *Retrieving the Policy from the ESA* for encrypting the policy package.

   You need to specify the PBE secret name as the value of the *pbeSecrets* parameter in the *values.yaml* file in *step 5*.
5. Modify the default values in the *values.yaml* file as required.

   The *sample-app* > *values.yaml* file contains the default configuration values for deploying the Sample Application 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
...
...

# 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

## start with blue deployment first
deploymentInUse: blue

## Policy metadata for Blue Deployments
blueDeployment:
  enabled: true
  # Choose your storage source. Currently we support <VolumeMount | ObjectStore>
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV or ObjectStore where the policy dump file will be stored. <eg. /
test/xyz/policy >
    # If Object Store make sure first name in path is bucket name i.e. /test will be our
existing bucket
    filePath: ABSOULTE_POLICY_PATH

## Policy metadata for Green Deployments
greenDeployment:
  enabled: false
  # Choose your storage source. Currently we support <VolumeMount | ObjectStore>
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV or ObjectStore where the policy dump file will be stored. <eg. /
test/xyz/policy >
    # If Object Store make sure first name in path is bucket name i.e. /test will be our
existing bucket
    filePath: ABSOULTE_POLICY_PATH

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

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

## specify the ports exposed in your springapp 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.
sampleappService:
    name: "restapi"
    port: 8080
    targetPort: 8080

## 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
```

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: Sample Application 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 **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 Sample Application 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*.
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 in section *Copying the Policy Package to a Mount Directory on the NFS Server*. 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.
sampleappService/name	Specify a name for the tunnel to distinguish between ports.
sampleappService/port	Specify the port number on which you want to expose the Kubernetes service externally.
sampleappService/targetPort	Specify the port on which the Sample Application container is running inside the Docker container. **Important:** Do not change this value.
routes/prodService/hostname	Specify the hostname of the production service.
routes/stagingService/hostname	Specify the hostname of the staging service.

6. Run the following command to deploy the Sample Application container on the Kubernetes cluster:
   ```
   helm install <Release Name> --namespace <Namespace where you want to deploy the
   Sample Application container> <Location of the directory that contains the Helm
   charts>
   ```
   For example:
   ```
   helm install iap-go --namespace iap sample-app
   ```
7. 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
      ```
   b. Run the following command to get the status of the routes.
      ```
      oc get routes -n <namespace>
      ```
      For example:
      ```
      oc get routes -n iap
      ```

2.6.6 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 Sample Application container.

   For more information about the extracted Helm charts, refer to the step *4* of the section *Initializing the Linux Instance*.

   The *values.yaml* file contains the default configuration values for deploying the Sample Application 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
...
...

## persistent volume name e.g nfs-pvc
pvcName: PVC_NAME

## Choose your private key helper. Currently we support [PKCS5, AWS_KMS]
privateKeySource: PKCS5

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

## k8s secret containing the private key id for decrypting policy
privateKeySecret: PKEY_SECRET_NAME

## start with blue deployment first
deploymentInUse: blue

## Policy metadata for Blue Deployments
blueDeployment:
  enabled: true
  policy:
    # absolute path in ObjectStore from where the policy dump file will be fetched.
    # <S3> "s3://bucketName/pathToPolicy"
    # absolute path in PV mount from where the policy dump file will be fetched.
    # <VOLUME> "file://var/data/ptypolicy/pathInVolumeToPolicy"
    # relative path in PV mount from where the policy dump file will be fetched.
    # it will prepend the '/var/data/ptypolicy' to below path
    # <VOLUME> "file:///pathInVolumeToPolicy"
    # <VOLUME> "pathInVolumeToPolicy" # without any leading '/'
    filePath: "POLICY_PATH"

## Policy metadata for Green Deployments
greenDeployment:
  enabled: false
  policy:
    # absolute path in ObjectStore from where the policy dump file will be fetched.
    # <S3> "s3://bucketName/pathToPolicy"
    # absolute path in PV mount from where the policy dump file will be fetched.
    # <VOLUME> "file://var/data/ptypolicy/pathInVolumeToPolicy"
    # relative path in PV mount from where the policy dump file will be fetched.
    # it will prepend the '/var/data/ptypolicy' to below path
    # <VOLUME> "file:///pathInVolumeToPolicy"
    # <VOLUME> "pathInVolumeToPolicy" # without any leading '/'
    filePath: "POLICY_PATH"

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

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

## specify the ports exposed in your Sample app 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.
sampleappService:

  # only applicable if ingress is disabled
  # allows you to configure service type: LoadBalancer or ClusterIP
  type: LoadBalancer

  # only apply, if the ingress is disabled
  # Specify service related annotations here
  annotations:
    ##AWS
    # service.beta.kubernetes.io/aws-load-balancer-internal: "true"
    ##AZURE
    # service.beta.kubernetes.io/azure-load-balancer-internal: "true"
    ##GCP
    # networking.gke.io/load-balancer-type: "Internal"

  name: "restapi"
  port: 8080
  targetPort: 8080

## Ingress enables testing of Green (V2/Staging) Deployments alongside Blue ones
simultaneously.
## This is realized by exposing both deployments on the same External IP but on different
URL Paths.
## If you don't want expose both the deployments at the same time, keep 'ingress.enabled'
as false.
## This means that only Blue deployment gets exposed on the Load Balancer's External IP
via prod service.
ingress:
  enabled: true

  # specify ingressClass cluster resource. It associates which controller will implement
the resource.
  ingressClassName: nginx-iap

  ## if required, specify a different Ingress Controller and related annotations here
  annotations:
    ## Enable client certificate authentication
    #nginx.ingress.kubernetes.io/auth-tls-verify-client: "on"

    ## Name of the Secret along with the Namespace, where its created,
    ## that contains the full CA chain ca.crt,
    ## that is enabled to authenticate against this Ingress
    #nginx.ingress.kubernetes.io/auth-tls-secret: "<NAMESPACE>/ca-secret"

  ## specify the host names and paths for Ingress rules
  ## prod and staging http paths can be used as optional fields.
  hosts:
    - host: "prod.example.com"
      httpPaths:
      - port: 8080
        prod: "/"
    ## Add host section with the hostname used as CN while creating server certificates.
    ## For accessing the staging end-points also, we need hostname as that of CN in
server certs.
    ## While creating the certificates you can use *.example.com as CN for the below
example
    # - host: "prod.example.com"
    #   httpPaths:
    #   - port: 8080
    #     prod: "/"
    # - host: "stage.example.com"
    #   httpPaths:
    #   - port: 8080
    #     staging: "/"
    #
  ## If TLS is terminated on the Ingress Controller Load Balancer,
  ## K8s TLS Secret containing the certificate and key must also be provided
  # tls:
  #  - secretName: example-tls
  #    hosts:
  #      - prod.example.com

###ONLY TO BE USED WITH OPENSHIFT
## 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:
  enabled: false
  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
```

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 section *Appendix B: Sample Application 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 **Note:** Set the value of the *fsGroup* parameter to a non-root value.

Field	Description
	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 Sample Application 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*.
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.
sampleappService/name	Specify a name for the tunnel to distinguish between ports.
sampleappService/port	Specify the port number on which you want to expose the Kubernetes service externally.
sampleappService/targetPort	Specify the port on which the Sample Application container is running inside the Docker container. **Important:** Do not change this value.
routes/prodService/hostname	Specify the hostname of the production service.
routes/stagingService/hostname	Specify the hostname of the staging service.

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

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

   For example:

   ```
   helm upgrade iap-go --namespace iap sample-app
   ```

   Using this configuration ensures that the Sample Application 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 section *Running Security Operations*.

   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 Sample Application 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
   Sample Application container> <Location of the directory that contains the Helm
   charts>
   ```

   For example:

   ```
   helm upgrade iap-go --namespace iap sample-app
   ```

   Using this configuration ensures that the Sample Application 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 Sample Application container> <Location of the directory that contains the Helm charts>
   ```

   For example:

   ```
   helm upgrade iap-go --namespace iap sample-app
   ```

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

   If you want to modify any policy or subsequent releases, then 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 Sample Application 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>
   ```

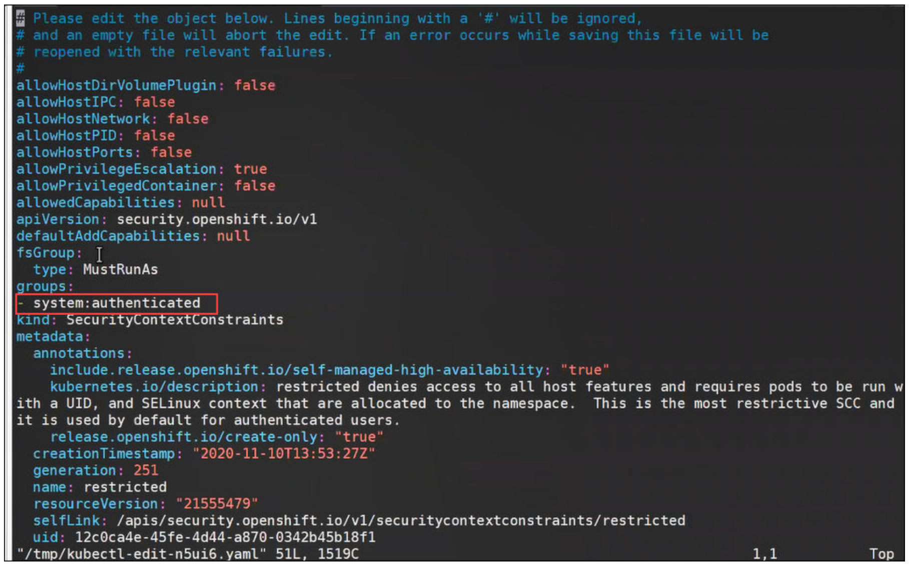
   For example:

   ```
   oc new-project iap
   ```

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 Sample Application 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 the 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 Sample Application, and edit the *pty-scc.yaml* file to replace the *<NAMESPACE>* placeholder in the following snippet with the namespace where the Sample Application will be deployed.

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

   For example:

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

   For more information about the extracted Helm charts, refer to the *step 4* 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 Retrieving the Policy from the ESA

This section describes how to invoke the Immutable Service APIs to retrieve the policy package using the ESA.

For more information about the Immutable Service APIs, refer to the section *Appendix B: APIs for Immutable Protectors* in the *Protegrity APIs, UDFs, Commands Reference Guide 9.1.0.5*.

To retrieve the policy package from the ESA:

1. Run the following command to verify the PEP server versions that are supported by the Immutable Service on the ESA.

   ```
   curl -v -k -u "<ESA user>:<ESA password>" https://<ESA IP>/pty/v1/imp/version
   ```

   The command returns the following output.

   ```
   {
     "version" : "1.1.0",
     "buildVersion" : "1.1.0+7.g307f.1.1",
     "pepVersions" : [ "1.1.0+85", "1.2.0+16" ]
   }
   ```

   Note: Ensure that the PEP server version returned by the IMP version API matches the PEP server version specified in the *manifest.json* file in the installation package.

   For more information about the installation package, refer to the section *Downloading the Installation Package*.

2. If you want to download the policy package from the ESA and encrypt the policy package using a passphrase and a salt, then run the following command.

   ```
   curl -v -k -u "<ESA user>:<ESA password>" https://<ESA IP>/pty/v1/imp/export -H
   "Content-Type: application/json" -o policy_package_key.tgz --data
   '{"name":"policypackage","pepVersion":"1.2.0+16","dataStoreName":"<Value>","emptySt
   ring":"<Value>","caseSensitive":"<Value>","kek":{"pkcs5":
   {"password":"<Password>","salt":"<Salt>"}}}'
   ```

   For example:

   ```
   curl -v -k -u "IMPUser:<Password>" https://10.49.1.218/pty/v1/imp/export -H
   "Content-Type: application/json" -o policy_package_key.json --data
   '{"name":"policypackage","pepVersion":"1.2.0+16","dataStoreName":"<Value>","emptySt
   ring":"<Value>","caseSensitive":"<Value>","kek":{"pkcs5":
   {"password":"<Password>","salt":"<Salt>"}}}'
   ```

   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.

   The encrypted policy package is downloaded to your machine.

   Important: Ensure that the user is assigned the **Export IMP** permission through roles.

For more information about assigning permissions to user roles, refer to the section *Managing Roles* in the *Appliances Overview 9.1.0.5* guide.

For more information about managing users, refer to the section *Managing Users* in the *Appliances Overview 9.1.0.5* guide.

For more information about the Export API, refer to the section *Sample Immutable Service API - Exporting Policy from a PEP Server Version* in the *Protegrity APIs, UDFs, Commands Reference Guide 9.1.0.5*.

Important: Ensure that the password complexity 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*

3. If you want to download the policy package from the ESA and encrypt the policy package using a KMS, then run the following command.

```
curl -v -k -u "<ESA user>:<ESA password>" https://<ESA IP>/pty/v1/imp/export -H
"Content-Type: application/json" -o <Policy_package_name>.tgz --data
'{"name":"policypackage","pepVersion":"1.2.0+16","dataStoreName":"<value>","emptySt
ring":"<value>","caseSensitive":"<value>","kek":{"publicKey": {"value":"Public Key
of the AWS Customer Key used to encrypt the policy"}}}'
```

For example:

```
curl -v -k -u "IMPUser:<password>" https://10.49.1.218/pty/v1/imp/export -H
"Content-Type: application/json" -o policy_package_key.tgz --data
'{"name":"policypackage","pepVersion":"1.2.0+16","dataStoreName":"<value>","emptySt
ring":"<value>","caseSensitive":"<value>","kek":{"publicKey": {"value":"-----BEGIN
PUBLIC KEY-----
\nMIIBojANBgkqhkiG9w0BAQEFAAOCAY8AMIIBigKCAYEAjwxqqrrNQ1kV84GEfYLR\nVK/
SyOkGTs7kxEImEYMGtugSal2G9m7hFdVlWhHH/OvLDHwdOYe6GLXHVwycfbM/
\nS6pzQPS5ujzyJEaLWYAfcRgAHi9g8GvazDiv34Z+VO5FRbJzP9u9/23J4hExc+e1\nTezKsoBj6Ypx/
zBkE4dGqdamfHJUF3ptB82nhJT/Pth15ejL/SVrD/
q8sORU838O\negcIfQYmd5ziZJgbWLVzFaM9QZXH9hD/
0JxAhqfHP3Fnhmc5MDVdg7D0SQkufIjf\nu7djMy2I3ZRMgnkGxjq8PuBUFaH2s6DvAa4e6K0ppGjFoByVV
e2tumrUVEvJg2EM\ndD/
Pl15kzLelbDKjxrI8T1D8cWuGh43wf1xC+uRZsfoMSgwDWdpBmaOFP+q2k0d4\nrQWKIgZw71mxYADPKBAU
UwMgD/aREuvTDlpvpl0Vk2Y5i/4Xx/fK7GV5DCdy9TFM\nZgzAA/
O9z5tuVzqcgodMu06+iqtXdTUoOafA+BTvSIXLAgMBAAE=\n-----END PUBLIC
KEY-----","algorithm":"RSA_OAEP_SHA256"}}}'
```

Note: Ensure that the new line in the public key is replaced with the *\n* character.

The policy package is downloaded to your machine.

4. Copy the policy package file to an NFS server.

2.7.3 Copying the Policy Package to a Mount Directory on the NFS Server

This section describes how to create a mount point on the NFS server, where you can copy the policy package.

➤ To deploy a release on the Kubernetes cluster:

Before you begin
Ensure that you have set up an NFS server with an NFS share.

1. On the Linux instance, navigate to the location where you have extracted the Helm charts to deploy the Sample Application container.

   For more information about the extracted Helm charts, refer to the *step 4* 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 sample-app/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 amount of storage.

```
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 sample-app/nfs-pvc.yaml -n iap
   ```
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. Copy the policy package, which you have retrieved in the section *Retrieving the Policy from the ESA*, to the mount directory.

2.7.4 Deploying the Sample Application Container on the Kubernetes Cluster

This section describes how to deploy the Sample Application container on the Kubernetes cluster by installing the Helm charts.

To deploy a release on the Kubernetes cluster:

1. If you have used Passphrase-Based Encryption to encrypt the policy package, then run the following command to create a passphrase secret, which is used by the Sample Application container to decrypt the policy.

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

   Ensure that you specify the same passphrase and salt that you have used in *step 2* of the section *Retrieving the Policy from the ESA* for encrypting the policy package.

   You need to specify the PBE secret name as the value of the *pbeSecrets* parameter in the *values.yaml* file in *step 3*.

2. On the Linux instance, navigate to the location where you have extracted the Helm charts to deploy the Sample Application container.

   For more information about the extracted Helm charts, refer to the *step 4* of the section *Initializing the Linux Instance*.

   The *sample-app* > *values.yaml* file contains the default configuration values for deploying the Sample Application 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
   ...
   ...

   # 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

   ## start with blue deployment first
   deploymentInUse: blue

   ## Policy metadata for Blue Deployments
   blueDeployment:
     enabled: true
     # Choose your storage source. Currently we support <VolumeMount | ObjectStore>
     securityConfigSource: VolumeMount
     policy:
       # absolute path in PV or ObjectStore where the policy dump file will be stored. <eg. /
   test/xyz/policy >
   ```

```
    # If Object Store make sure first name in path is bucket name i.e. /test will be our
existing bucket
    filePath: ABSOULTE_POLICY_PATH

## Policy metadata for Green Deployments
greenDeployment:
  enabled: false
  # Choose your storage source. Currently we support <VolumeMount | ObjectStore>
  securityConfigSource: VolumeMount
  policy:
    # absolute path in PV or ObjectStore where the policy dump file will be stored. <eg. /
test/xyz/policy >
    # If Object Store make sure first name in path is bucket name i.e. /test will be our
existing bucket
    filePath: ABSOULTE_POLICY_PATH

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

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

## specify the ports exposed in your springapp 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.
sampleappService:
    name: "restapi"
    port: 8080
    targetPort: 8080

## 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
```

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: Sample Application 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

Field	Description
	• *fsGroup* - 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 Sample Application 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*.
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 in section *Copying the Policy Package to a Mount Directory on the NFS Server*. 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.
sampleappService/name	Specify a name for the tunnel to distinguish between ports.
sampleappService/port	Specify the port number on which you want to expose the Kubernetes service externally.
sampleappService/targetPort	Specify the port on which the Sample Application container is running inside the Docker container. **Important:** Do not change this value.
routes/prodService/hostname	Specify the hostname of the production service.
routes/stagingService/hostname	Specify the hostname of the staging service.

4. Run the following command to deploy the Sample Application container on the Kubernetes cluster:

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

   For example:

   ```
   helm install iap-go --namespace iap sample-app
   ```

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
      ```

   b. Run the following command to get the status of the routes.

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

      For example:

      ```
      oc get routes -n iap
      ```

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 Sample Application container.

   For more information about the extracted Helm charts, refer to the step *4* of the section *Initializing the Linux Instance*.

   The *values.yaml* file contains the default configuration values for deploying the Sample Application 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
...
...

## persistent volume name e.g nfs-pvc
pvcName: PVC_NAME

## Choose your private key helper. Currently we support [PKCS5, AWS_KMS]
privateKeySource: PKCS5

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

## k8s secret containing the private key id for decrypting policy
privateKeySecret: PKEY_SECRET_NAME

## start with blue deployment first
deploymentInUse: blue

## Policy metadata for Blue Deployments
blueDeployment:
  enabled: true
  policy:
    # absolute path in ObjectStore from where the policy dump file will be fetched.
    # <S3> "s3://bucketName/pathToPolicy"
    # absolute path in PV mount from where the policy dump file will be fetched.
    # <VOLUME> "file://var/data/ptypolicy/pathInVolumeToPolicy"
    # relative path in PV mount from where the policy dump file will be fetched.
    # it will prepend the '/var/data/ptypolicy' to below path
    # <VOLUME> "file:///pathInVolumeToPolicy"
    # <VOLUME> "pathInVolumeToPolicy" # without any leading '/'
    filePath: "POLICY_PATH"

## Policy metadata for Green Deployments
greenDeployment:
  enabled: false
  policy:
    # absolute path in ObjectStore from where the policy dump file will be fetched.
    # <S3> "s3://bucketName/pathToPolicy"
    # absolute path in PV mount from where the policy dump file will be fetched.
    # <VOLUME> "file://var/data/ptypolicy/pathInVolumeToPolicy"
    # relative path in PV mount from where the policy dump file will be fetched.
    # it will prepend the '/var/data/ptypolicy' to below path
    # <VOLUME> "file:///pathInVolumeToPolicy"
    # <VOLUME> "pathInVolumeToPolicy" # without any leading '/'
    filePath: "POLICY_PATH"

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

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

## specify the ports exposed in your Sample app 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.
sampleappService:

  # only applicable if ingress is disabled
  # allows you to configure service type: LoadBalancer or ClusterIP
  type: LoadBalancer

  # only apply, if the ingress is disabled
  # Specify service related annotations here
  annotations:
    ##AWS
    # service.beta.kubernetes.io/aws-load-balancer-internal: "true"
    ##AZURE
    # service.beta.kubernetes.io/azure-load-balancer-internal: "true"
    ##GCP
    # networking.gke.io/load-balancer-type: "Internal"

  name: "restapi"
  port: 8080
  targetPort: 8080

## Ingress enables testing of Green (V2/Staging) Deployments alongside Blue ones
simultaneously.
## This is realized by exposing both deployments on the same External IP but on different
URL Paths.
## If you don't want expose both the deployments at the same time, keep 'ingress.enabled'
as false.
## This means that only Blue deployment gets exposed on the Load Balancer's External IP
via prod service.
ingress:
  enabled: true

  # specify ingressClass cluster resource. It associates which controller will implement
the resource.
  ingressClassName: nginx-iap

  ## if required, specify a different Ingress Controller and related annotations here
  annotations:
    ## Enable client certificate authentication
    #nginx.ingress.kubernetes.io/auth-tls-verify-client: "on"

    ## Name of the Secret along with the Namespace, where its created,
    ## that contains the full CA chain ca.crt,
    ## that is enabled to authenticate against this Ingress
    #nginx.ingress.kubernetes.io/auth-tls-secret: "<NAMESPACE>/ca-secret"

  ## specify the host names and paths for Ingress rules
  ## prod and staging http paths can be used as optional fields.
  hosts:
    - host: "prod.example.com"
      httpPaths:
      - port: 8080
        prod: "/"
    ## Add host section with the hostname used as CN while creating server certificates.
    ## For accessing the staging end-points also, we need hostname as that of CN in
server certs.
    ## While creating the certificates you can use *.example.com as CN for the below
example
    # - host: "prod.example.com"
    #   httpPaths:
    #   - port: 8080
    #     prod: "/"
    # - host: "stage.example.com"
    #   httpPaths:
    #   - port: 8080
    #     staging: "/"
    #
  ## If TLS is terminated on the Ingress Controller Load Balancer,
  ## K8s TLS Secret containing the certificate and key must also be provided
  # tls:
  #  - secretName: example-tls
  #    hosts:
  #      - prod.example.com

###ONLY TO BE USED WITH OPENSHIFT
## 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:
  enabled: false
  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
```

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 section *Appendix B: Sample Application 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 **Note:** Set the value of the *fsGroup* parameter to a non-root value.

Field	Description
	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 Sample Application 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*.
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.
sampleappService/name	Specify a name for the tunnel to distinguish between ports.
sampleappService/port	Specify the port number on which you want to expose the Kubernetes service externally.
sampleappService/targetPort	Specify the port on which the Sample Application container is running inside the Docker container. **Important:** Do not change this value.
routes/prodService/hostname	Specify the hostname of the production service.
routes/stagingService/hostname	Specify the hostname of the staging service.

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

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

   For example:

   ```
   helm upgrade iap-go --namespace iap sample-app
   ```

   Using this configuration ensures that the Sample Application 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 section *Running Security Operations*.

   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 Sample Application 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
   Sample Application container> <Location of the directory that contains the Helm
   charts>
   ```

   For example:

   ```
   helm upgrade iap-go --namespace iap sample-app
   ```

   Using this configuration ensures that the Sample Application 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
   Sample Application container> <Location of the directory that contains the Helm
   charts>
   ```

   For example:

   ```
   helm upgrade iap-go --namespace iap sample-app
   ```

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

   If you want to modify any policy or subsequent releases, then 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

3.1 Understanding the Logging Architecture
3.2 Setting Up Splunk
3.3 Configuring Splunk
3.4 Deploying the Splunk Connect for Kubernetes Helm Chart on the Kubernetes Cluster

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.

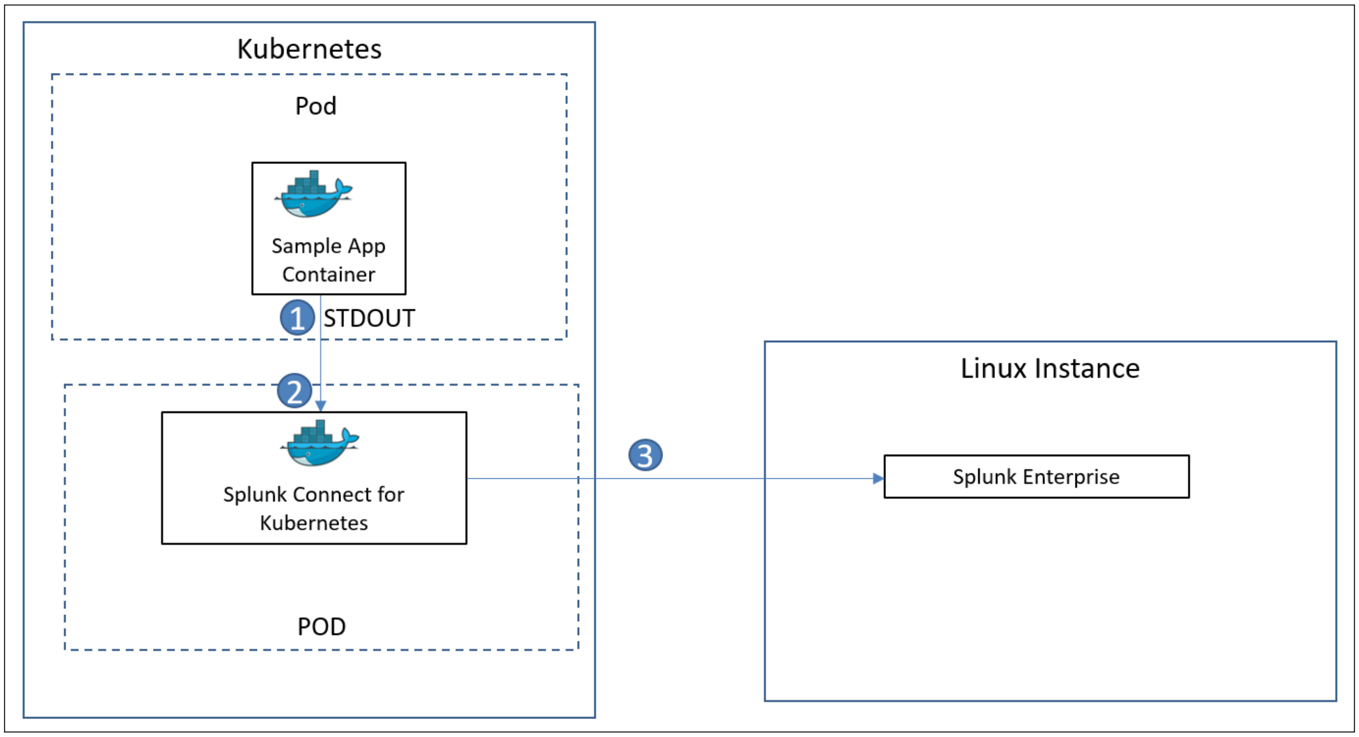


Figure 3-1: Sample Logging Workflow

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

1. The Sample Application container writes 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.

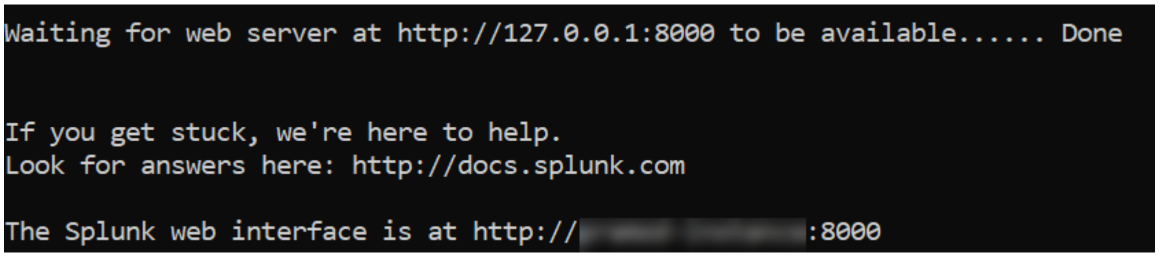
4. Type the username for the administrator account for logging into Splunk Enterprise, and then press **Enter**.

   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*.

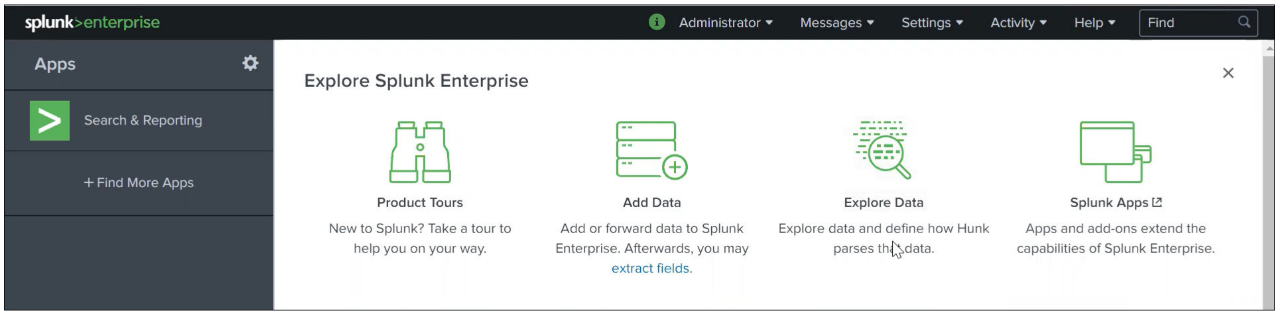
3.3 Configuring Splunk

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.



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.

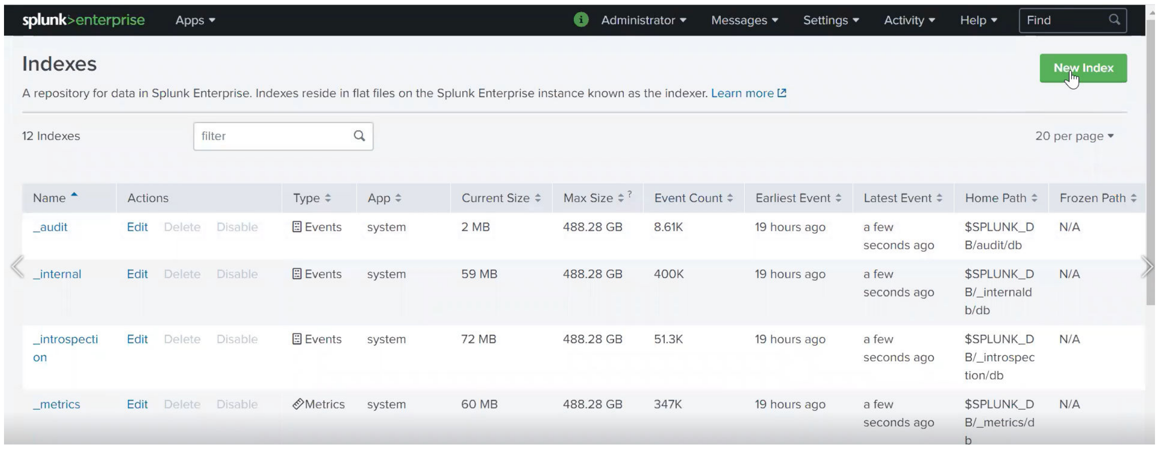
Apps and add-ons extend the capabilities of Splunk Enterprise.

Figure 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

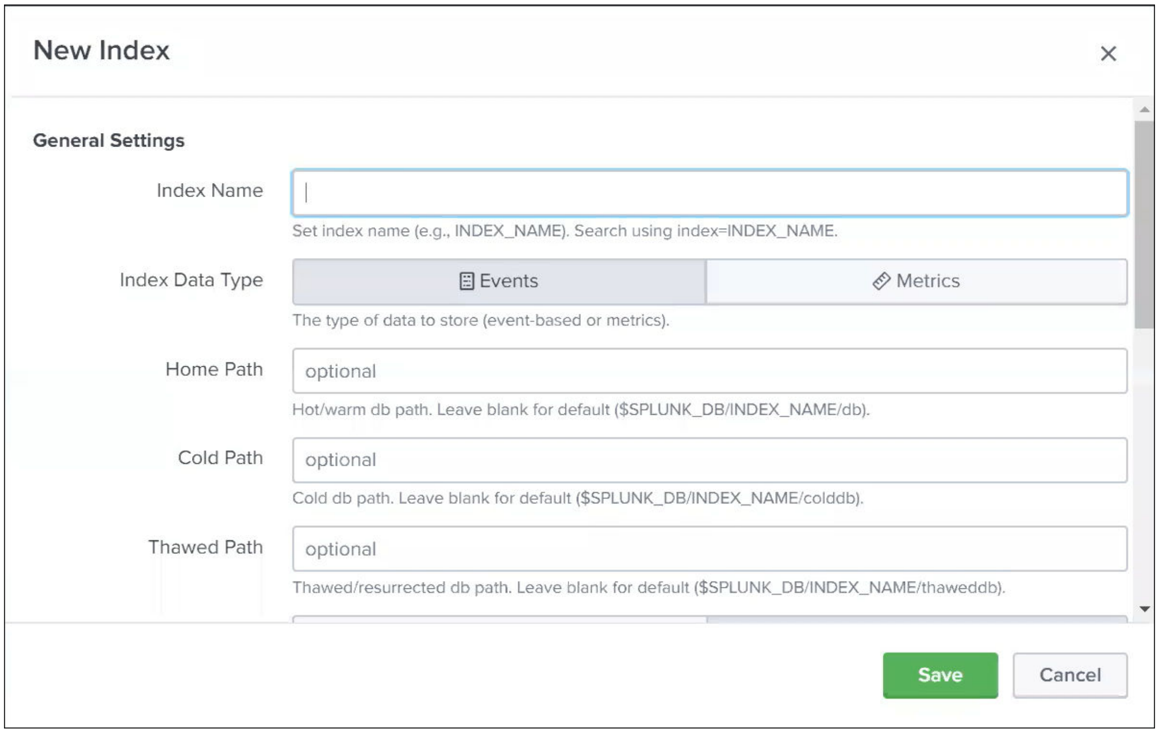
12 Indexes filter 20 per page

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.

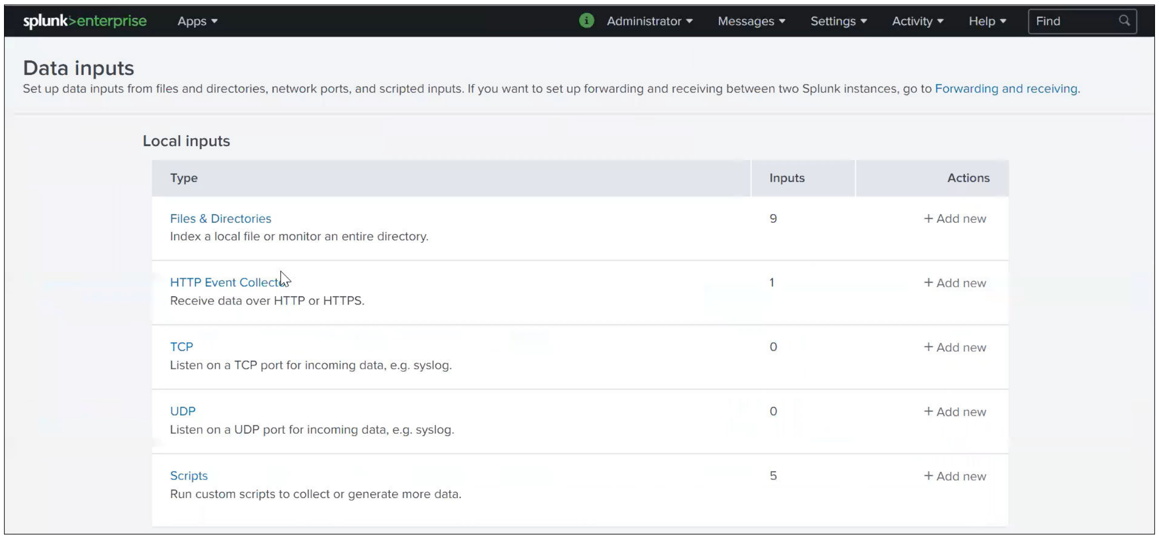
For more information about creating an index in Splunk Enterprise, refer to the section *Create custom indexes* in the Splunk documentation.

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.

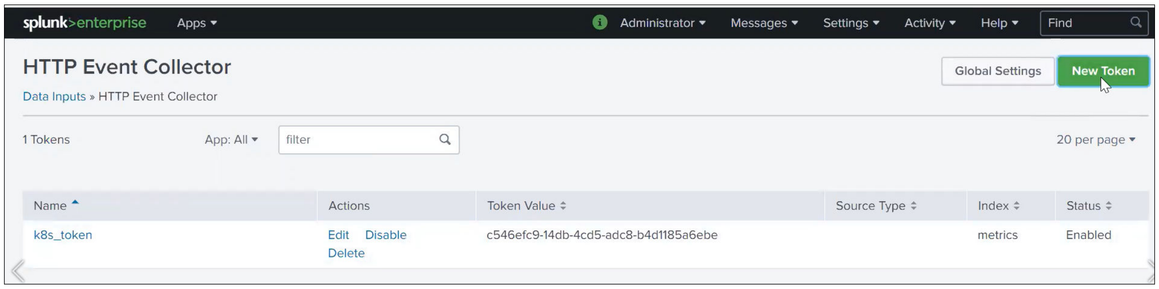
Local inputs

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

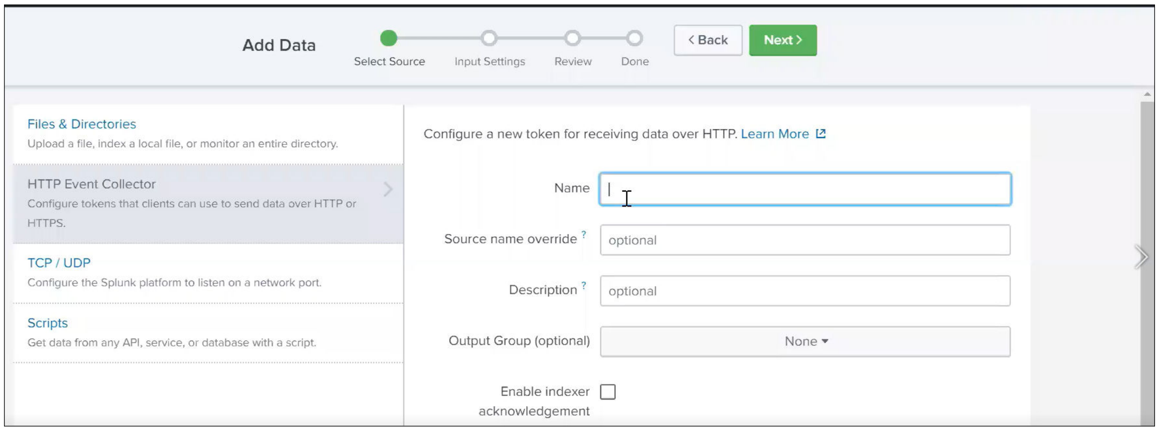
1 Tokens

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.


Add Data
Select Source
Input Settings
Review
Done
< Back
Next >
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

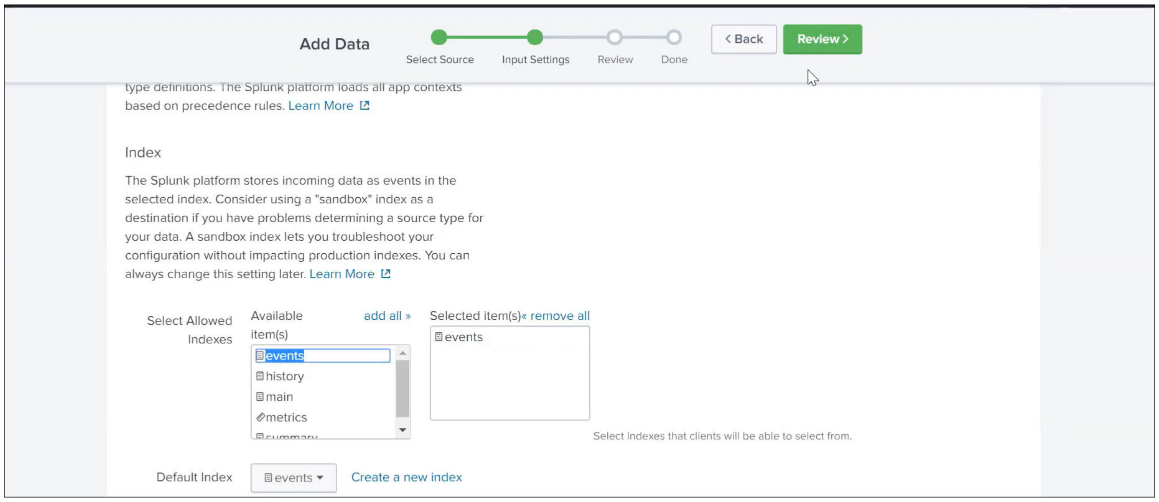
d. In the **Name** field, type a name for the token.

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 — add all » — Selected item(s): events — « remove all

Select indexes that clients will be able to select from.

Default Index: events — Create a new index

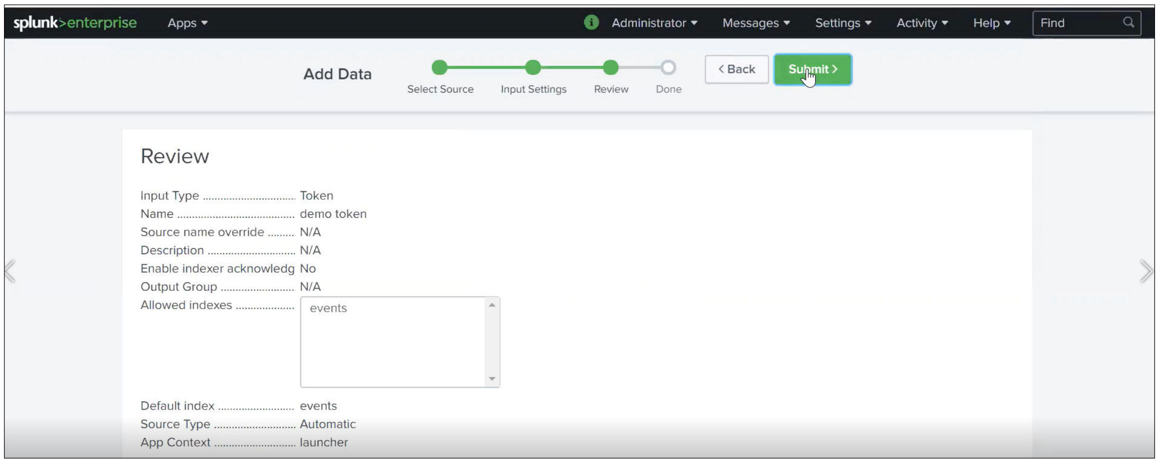
Figure 3-8: Input Settings Screen

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.



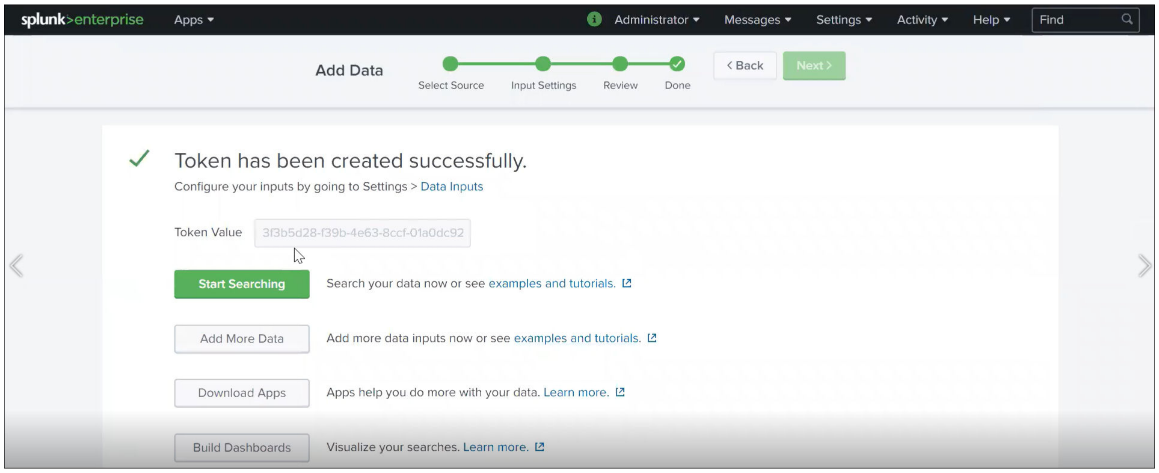
Review

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**.

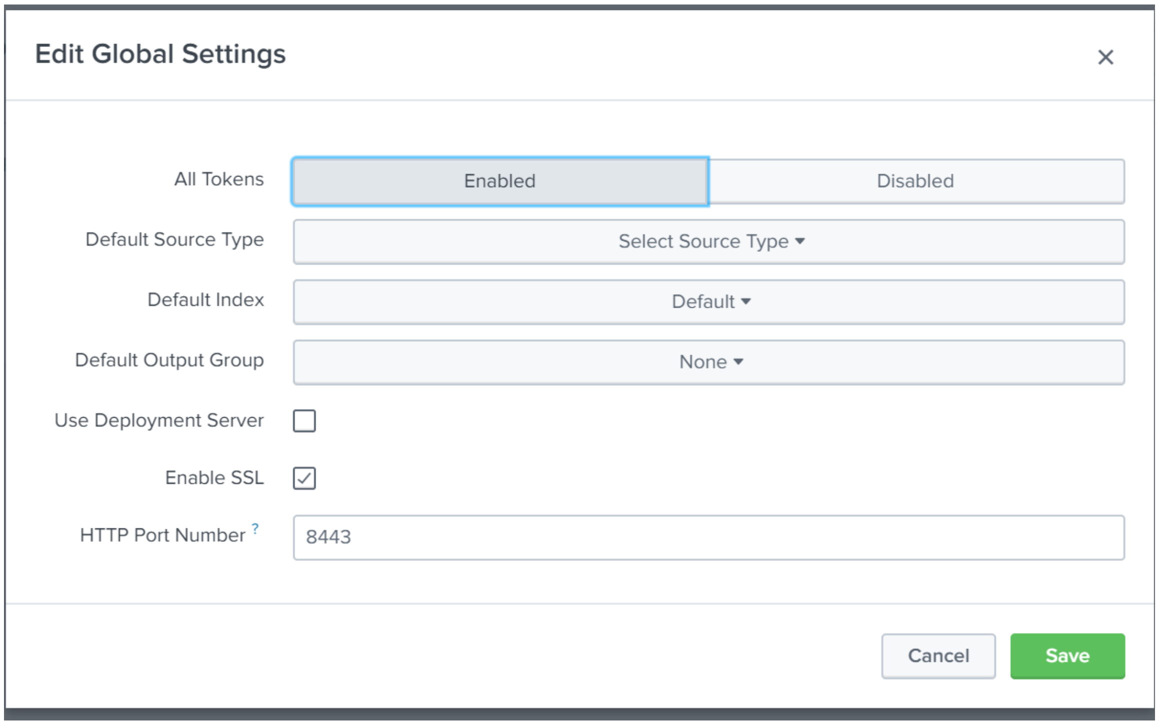
      The **Data inputs** screen appears.

   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 Sample Application 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*.
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*.

Parameter	**Description**
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 Sample Application 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 Sample Application Container has been deployed.

In the following example, the Sample Application Container has been deployed on the *protegrity-iap* 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-iap*.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-iap.']
> ```
>
> 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.

Chapter 4

Protecting Data Using the Sample Application Container Deployment

4.1 Running Security Operations
4.2 Viewing Logs in Splunk
4.3 Autoscaling the Protegrity Reference Deployment

This section describes how to protect data using the Sample Application 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 Sample Application.

4.1 Running Security Operations

This section describes how you can use the Sample Application 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 not using TLS authentication between the Sample Application instance and the REST API client, then send the following cURL request from the Linux instance.

```
curl --location --request POST 'http://<Hostname of the Routes>/protect' --
header 'Host: prod.example.com' --header 'Content-Type: application/json' --data-
raw '{ "dataElement": "Alphanum", "policyUser": "user1", "input": [ "SampleAppGo",
"OnKubernetes" ] }'
```

   In the example, the *<Hostname of the Routes>* value refers to the host name specified in the *routes/prodService/hostName* parameter of the Sample Application *values.yaml* file.

   The Sample Application container instance internally sends this request to the Application Protector Go, and returns the protected value.

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

```
curl --location --request POST 'http://<Hostname of the Routes>/unprotect' --header
'Host: prod.example.com' --header 'Content-Type: application/json' --data-raw
'{ "dataElement": "Alphanum", "policyUser": "user1", "input": [ "iaDDNBdH6EI8U",
"9jB7cRSuk98B" ] }'
```

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

```
curl --location --request POST 'http://<Hostname of the Routes>/reprotect' --
header 'Host: prod.example.com' --header 'Content-Type: application/json' --data-
```

```
raw '{ "dataElement": "Alphanum", "newDataElement": "Alphanum1", "policyUser":
"user1", "input": [ "iaDDNBdH6EI8U", "9jB7cRSuk98B" ] }'
```

2. If you are using TLS authentication between the Sample Application instance and the REST API client, then perform the following steps.

   ```
   curl --location --request POST 'https://<Hostname of the Routes>/protect' --
   header 'Content-Type: application/json' --data-raw '{ "dataElement": "Alphanum",
   "policyUser": "user1", "input": [ "SampleAppGo", "OnKubernetes" ] }' --cacert ./
   iap-ca.crt --cert ./iap-client.crt --key ./iap-client.key
   ```

   In the example, the *<Hostname of the Routes>* value refers to the host name specified in the *routes/prodService/hostName* parameter of the Sample Application *values.yaml* file.

   The Sample Application container instance internally sends this request to the Application Protector Go, and returns the protected value.

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

   ```
   curl --location --request POST 'https://<Hostname of the Routes>/unprotect' --
   header 'Content-Type: application/json' --data-raw '{ "dataElement": "Alphanum",
   "policyUser": "user1", "input": [ "iaDDNBdH6EI8U", "9jB7cRSuk98B" ] }' --cacert ./
   iap-ca.crt --cert ./iap-client.crt --key ./iap-client.key
   ```

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

   ```
   curl --location --request POST 'https://<Hostname of the Routes>/reprotect' --
   header 'Content-Type: application/json' --data-raw '{ "dataElement": "Alphanum",
   "newDataElement": "Alphanum1", "policyUser": "user1", "input": [ "iaDDNBdH6EI8U",
   "9jB7cRSuk98B" ] }' --cacert ./iap-ca.crt --cert ./iap-client.crt --key ./iap-
   client.key
   ```

   For more information about creating the CA certificate, refer to the section *Creating Certificates and Keys for TLS Authentication*.

4.2 Viewing Logs in Splunk

This section describes how you can view the Application Protector Go and Container logs in Splunk Enterprise.

Important: You require a non-privileged role to perform this task.

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.

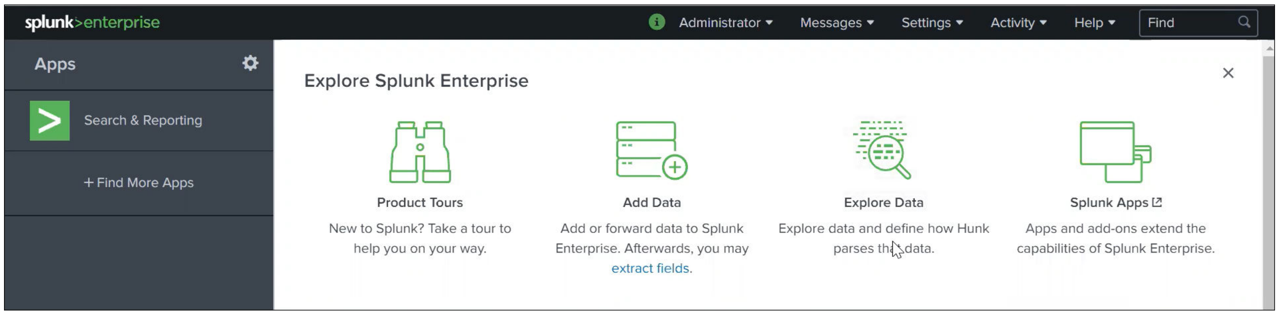

splunk>enterprise
Administrator
Messages
Settings
Activity
Help
Find
Apps
Search & Reporting
+ Find More Apps
Explore Splunk Enterprise
Product Tours
New to Splunk? Take a tour to help you on your way.
Add Data
Add or forward data to Splunk Enterprise. Afterwards, you may extract fields.
Explore Data
Explore data and define how Hunk parses the data.
Splunk Apps
Apps and add-ons extend the capabilities of Splunk Enterprise.

Figure 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.

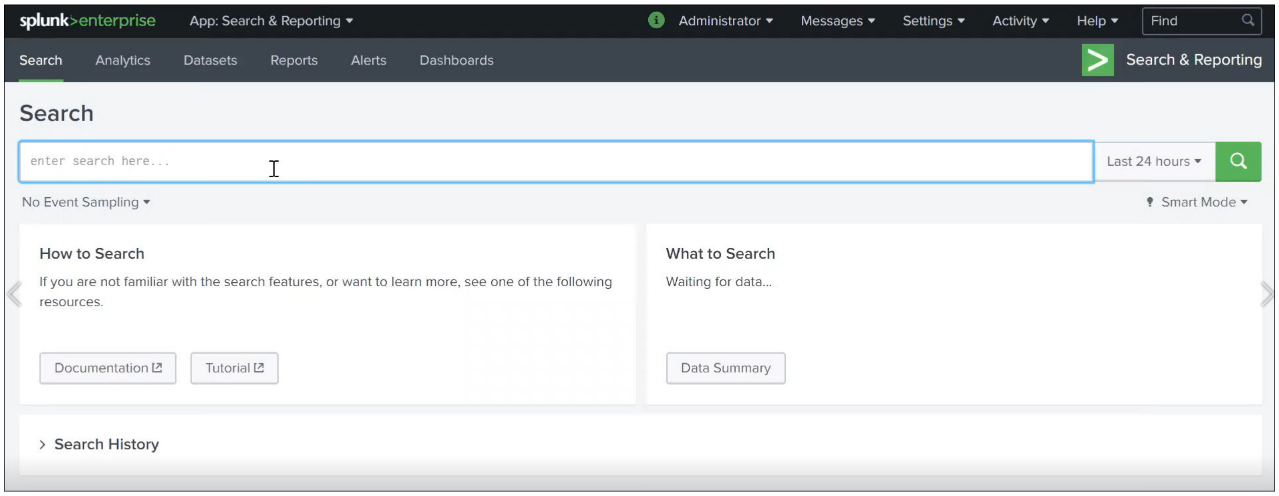


Figure 4-2: Search Screen

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:sampleappService"
```

5. Press **Enter**.

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

6. If you want to view only the logs that are related to the Application Protector Go, 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:sampleappService" "product_id:
Application Protector"
```

7. Press **Enter**.

   The search results display only the logs that are related to the Application Protector Go.

4.3 Autoscaling the Protegrity Reference Deployment

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

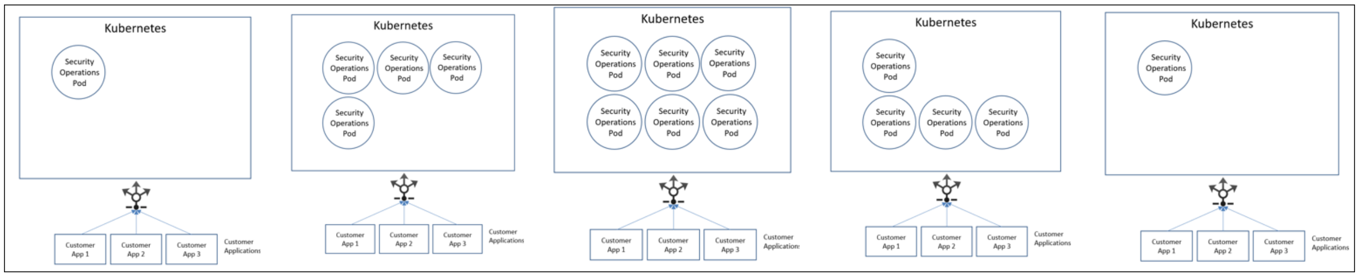


Figure 4-3: Autoscaling Kubernetes Cluster

As illustrated in this figure, the Customer Applications experience an increase and a decrease in workload as required by the businesses. 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 Sample Application appliances to meet the business need and then remove them if not required. Kubernetes performs the task of provisioning the Sample Application instances automatically.

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

Chapter 5

Appendix B: Sample Application Helm Chart Details

5.1 Chart.yaml
5.2 Values.yaml

This section describes the details of the Sample Application 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 Helm versions. These values can be left as default.

```
annotations:
  pepVersion: 1.2.0+16
apiVersion: v1
description: A Sample application protector chart for Kubernetes
name: sample-app
version: 2.0.0
```

5.2 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.

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.

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

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

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: ""
```

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.

```
sampleappImage:
  repository: SAMPLE_APP_IMAGE_REPOSITORY
  tag: SAMPLE_APP_IMAGE_TAG
  pullPolicy: Always
  debug: false # change debug to true to get container debug logs on STDOUT.
```

The following list provides an overview of the *sampleappImage* parameter, which refers to the details for the Sample Application 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 image upload.
- *debug* - Sets the value of this field to *true*, if you want debug logs for the Sample container. By default, the value of this field is set to *false*.

Note: If you are deploying a Customer Application, then you must specify the name of the Customer Application image.

5.2.4 Resources

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

```
sampleContainerResources:
  ## Below allocation covers largest policy that can be imported.
  ## Based on the policy size/application requirement update below limits.
  limits:
    cpu: 500m
    memory: 1750Mi
  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 information about the resource parameters, refer to the section *Managing Resources for Containers* in the Kubernetes documentation.

5.2.5 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>
```

5.2.6 Liveliness and Readiness Probe Settings

This section describes the values that reflect the intervals required to run health checks for the Sample Application container images. Users are recommended not to 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
```

5.2.7 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
  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.

5.2.8 Container Security Context

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

```
## set the Sample App Container's security context object
## leave the field empty if not applicable
sampleContainerSecurityContext:
  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.

5.2.9 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.

5.2.10 PBE Secrets

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

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

The Sample Application container uses the value specified in the *pbeSecrets* parameter to decrypt the policy.

Note: By default, the Passphrase-Based Encryption is used to encrypt the policy package. If you want to use the Azure Key Vault for encrypting the policy package, then you must comment out the *pbeSecrets* entry in the *values.yaml* file or leave the value of the *pbeSecrets* parameter as blank.

Important: Ensure that the password complexity 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*

5.2.11 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 is 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
  ## choose your storage source. Currently we support <VolumeMount | ObjectStore>
  ## default is VolumeMount, change it to ObjectStore if you wish to use object store
  securityConfigSource: VolumeMount
  ## If securityConfigSource is VolumeMount
  ## name of persistent volume claim to be used for this deployment.
  pvcName: PVC_NAME
  ## If securityConfigSource is ObjectStore
  ## specify the respective cloud provider ['AWS', 'AZURE', 'GCP']
  storageProvider:
  ## If securityConfigSource is ObjectStore
  ## specify k8s secret for storing credentials with access to cloud storage.
  storageAccessSecrets:
  policy:
    ## absolute path in PV or ObjectStore where the policy dump file is stored. <eg. /
test/xyz/policy >
    ## If source is ObjectStore, make sure first name in path is bucket name i.e. test will
be our existing bucket
    filePath: ABSOLUTE_POLICY_PATH

## Policy metadata for Green Deployments
greenDeployment:
  enabled: false
  ## choose your storage source. Currently we support <VolumeMount | ObjectStore>
  ## default is VolumeMount, change it to ObjectStore if you wish to use object store
  securityConfigSource: VolumeMount
  ## If securityConfigSource is VolumeMount
  ## name of persistent volume claim to be used for this deployment.
  pvcName: PVC_NAME
  ## If securityConfigSource is ObjectStore
  ## specify the respective cloud provider ['AWS', 'GCP', 'AZURE']
  storageProvider:
  ## If securityConfigSource is ObjectStore
  ## specify k8s secret for storing credentials with access to az storage.
  storageAccessSecrets:
  policy:
    ## absolute path in PV or ObjectStore where the policy dump file is stored. <eg. /
test/xyz/policy >
    ## If source is ObjectStore, make sure first name in path is bucket name i.e. test will
be our existing bucket
    filePath: ABSOLUTE_POLICY_PATH
```

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

For more information about Azure Storage Containers, refer to the *Microsoft Azure documentation*.

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

5.2.12 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.

5.2.13 Service Configuration

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

```
## specify the ports exposed in your Sample app 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.
sampleappService:

  # only applicable if ingress is disabled
  # allows you to configure service type: LoadBalancer or ClusterIP
  type: LoadBalancer

  # only apply, if the ingress is disabled
  # Specify service related annotations here
  annotations:
    ##AWS
    # service.beta.kubernetes.io/aws-load-balancer-internal: "true"
    ##AZURE
    # service.beta.kubernetes.io/azure-load-balancer-internal: "true"
    ##GCP
    # networking.gke.io/load-balancer-type: "Internal"

  name: "restapi"
  port: 8080
  targetPort: 8080
```

- *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 Sample Application is running inside the Docker container.

5.2.14 Routes Configuration

This section explains the Routes configuration for deployment.

```
###ONLY TO BE USED WITH OPENSHIFT
## 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:
  enabled: false
  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 6

Appendix C: Using the Dockerfile to Build Custom Images

6.1 Creating Custom Images Using Dockerfile

This section explains how to use the Dockerfiles, which are provided as part of the installation package, to build custom image for the Sample Application container. 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.

6.1 Creating Custom Images Using Dockerfile

This section describes the steps for creating custom images for the Sample Application 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* 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* file, refer to the section *Verifying the Prerequisites*.

2. Perform the following steps to build a Docker image for the Sample Application container.

   a. Run the following command to extract the files from the *IAP-SAMPLE-APP_OPENSHIFT_SRC_<version_number>.tar.gz* file.

   ```
   tar -xvf IAP-SAMPLE-APP_OPENSHIFT_SRC_<version_number>.tar.gz
   ```

   The following files are extracted:

   - *IAP_RHUBI_SAMPLE-APP_DOCKERFILE_<version_number>*
   - *docker-entrypoint.sh*
   - *passwd.template*

   b. Edit the *IAP_RHUBI_SAMPLE-APP_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. Follow the steps provided in the *HOW-TO-BUILD* text file that is included in the installation package.

d. Run the following command in the directory where you have extracted the contents of the *IAP-SAMPLE-APP_OPENSHIFT_SRC_<version_number>.tar.gz* file.

```
docker build -t <image-name>:<image-tag> -f IAP_RHUBI_SAMPLE-
APP_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*.

e. Run the following command to list the Sample Application container image.

```
docker images
```

f. Push the Sample Application container image to your preferred Container Repository.

For more information about pushing an image to the repository, refer to the *OpenShift documentation*.

Chapter 7

Appendix D: Applying the NSS Wrapper to the Customer Application Container Image

If you run the Customer Application container image with a random user ID, then the */etc/password* file does not contain an entry for the user ID. As a result, the Application Protector Go cannot determine the user name of the application. You can fix this issue by installing the *nss_wrapper* library on the Customer Application container image, and create a file that contains the mapping between the user name and the user ID, as part of the startup command of the image.

This section describes how you can apply the NSS wrapper to the Customer Application container image.

For more information about the *nss_wrapper* library, refer to the section *The nss_wrapper library*.

➤ To apply the NSS wrapper to the Customer Application container image:

1. Create a file named *docker-entrypoint.sh*, which is used as the default argument for the *ENTRYPOINT* instruction in the Dockerfile of the container image.

   Include the following content in the *docker-entrypoint.sh* file.

```
#!/bin/bash
export USER_ID=$(id -u)
export GROUP_ID=$(id -g)
envsubst < /etc/passwd.template > /tmp/passwd
export LD_PRELOAD=/usr/lib64/libnss_wrapper.so:/opt/protegrity/
applicationprotector/go/lib/gopepprovider_imp.plm
export NSS_WRAPPER_PASSWD=/tmp/passwd
export NSS_WRAPPER_GROUP=/etc/group

exec /opt/protegrity/restapi
```

   This command creates file named *passwd.template*.

   The *ENTRYPOINT* instruction specifies the startup command for executing the container image.

   For more information about Dockerfiles, refer to the *Docker* documentation.

2. Modify the *passwd.template* file to include the mapping between the user name of the container application and the user ID that is used to run the container application.

   The following snippet shows a sample *passwd.template* file that has been used for the Sample Application.

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
```

```
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
ptyitusr:x:${USER_ID}:${GROUP_ID}:ptyitusr:/home/ptyituser:/bin/bash
```

You need to replace the *ptyitusr* value with the user name of your Customer Application.

3. Modify the Dockerfile of your Customer Application container image to install the *nss_wrapper* and *gettext* packages, copy the *docker-entrypoint.sh* and *passwd.template* files to the image, and specify the *docker-entrypoint.sh* file as a default argument to the *ENTRYPOINT* instruction.

```
...

# Install the nss_wrapper and gettext packages
RUN yum -y install nss_wrapper gettext

# Copy the passwd.template file to the container image
COPY passwd.template /etc/passwd.template

# Copy the docker-entrypoint.sh file to the container image
COPY docker-entrypoint.sh /opt/docker-entrypoint.sh
...

# Specify the docker-entrypoint.sh file as a default argument for the ENTRYPOINT
instruction
ENTRYPOINT [ "/opt/docker-entrypoint.sh" ]
...
```