



Protegrity Application Protector Java Immutable Policy User Guide for OpenShift 9.1.0.0

Created on: Nov 19, 2024

Copyright

Copyright © 2004-2024 Protegrity Corporation. All rights reserved.

Protegrity products are protected by and subject to patent protections;

Patent: <https://support.protegrity.com/patents/>.

The Protegrity logo is the trademark of Protegrity Corporation.

NOTICE TO ALL PERSONS RECEIVING THIS DOCUMENT

Some of the product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective owners.

Windows, Azure, MS-SQL Server, Internet Explorer and Internet Explorer logo, Active Directory, and Hyper-V are registered trademarks of Microsoft Corporation in the United States and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SCO and SCO UnixWare are registered trademarks of The SCO Group.

Sun, Oracle, Java, and Solaris are the registered trademarks of Oracle Corporation and/or its affiliates in the United States and other countries.

Teradata and the Teradata logo are the trademarks or registered trademarks of Teradata Corporation or its affiliates in the United States and other countries.

Hadoop or Apache Hadoop, Hadoop elephant logo, Hive, and Pig are trademarks of Apache Software Foundation.

Cloudera and the Cloudera logo are trademarks of Cloudera and its suppliers or licensors.

Hortonworks and the Hortonworks logo are the trademarks of Hortonworks, Inc. in the United States and other countries.

Greenplum Database is the registered trademark of VMware Corporation in the U.S. and other countries.

Pivotal HD is the registered trademark of Pivotal, Inc. in the U.S. and other countries.

PostgreSQL or Postgres is the copyright of The PostgreSQL Global Development Group and The Regents of the University of California.

AIX, DB2, IBM and the IBM logo, and z/OS are registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Utimaco Safeware AG is a member of the Sophos Group.

Jaspersoft, the Jaspersoft logo, and JasperServer products are trademarks and/or registered trademarks of Jaspersoft Corporation in the United States and in jurisdictions throughout the world.

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

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

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

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

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

Dell is a registered trademark of Dell Inc.

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

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

Mozilla and Firefox are registered trademarks of Mozilla foundation.

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

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

Table of Contents

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.....	9
2.1 Verifying the Prerequisites.....	9
2.2 Downloading the Installation Package.....	11
2.3 Initializing the Linux Instance.....	12
2.4 Creating Certificates and Keys for TLS Authentication.....	13
2.5 Creating the Cloud Runtime Environment.....	14
2.5.1 Creating the OpenShift Runtime Environment.....	14
2.5.1.1 Logging in to the OpenShift Environment.....	15
2.5.2 Creating the Azure Runtime Environment.....	16
2.5.2.1 Logging in to the Azure Environment.....	16
2.5.2.2 Registering an Application.....	17
2.5.2.3 Creating an Azure Storage Container.....	18
2.6 Deploying the Containers with Security Context Constraints (SCC) and Role-Based Access Control (RBAC).....	19
2.6.1 Applying the SCC.....	20
2.6.2 Applying RBAC.....	21
2.6.3 Setting up the Environment for Deploying the Get ESA Policy Container.....	23
2.6.4 Deploying the Get ESA Policy Container on the Kubernetes Cluster.....	24
2.6.5 Verifying the IAP Policy Snapshot on the NFS.....	30
2.6.6 Verifying the IAP Policy Package on the Azure Storage Container.....	31
2.6.7 Setting up the Environment for Deploying the Sample Application Container.....	32
2.6.8 Deploying the Sample Application Container on the Kubernetes Cluster.....	32
2.6.9 Upgrading the Helm Charts.....	37
2.7 Deploying the Containers without RBAC.....	43
2.7.1 Applying the SCC.....	43
2.7.2 Deploying the Get ESA Policy Container on the Kubernetes Cluster.....	44
2.7.3 Verifying the IAP Policy Snapshot on the NFS.....	51
2.7.4 Verifying the IAP Policy Package on the Azure Storage Container.....	52
2.7.5 Deploying the Sample Application Container on the Kubernetes Cluster.....	52
2.7.6 Upgrading the Helm Charts.....	57
Chapter 3 Accessing Logs Using Splunk.....	63
3.1 Understanding the Logging Architecture.....	63
3.2 Setting Up Splunk.....	64
3.3 Configuring Splunk.....	65
3.4 Deploying the Splunk Connect for Kubernetes Helm Chart on the Kubernetes Cluster.....	70
Chapter 4 Protecting Data Using the Sample Application Container Deployment.....	74
4.1 Running Security Operations.....	74
4.2 Viewing Logs in Splunk.....	75
4.3 Autoscaling the Protegrity Reference Deployment.....	77
Chapter 5 Appendix A: Get ESA Policy Helm Chart Details.....	79
5.1 Chart.yaml.....	79
5.2 pty-scc.yaml.....	79
5.3 pty-rbac.yaml.....	80
5.4 nfs-pv.yaml.....	81



5.5 nfs-pvc.yaml.....	82
5.6 Credentials.json.....	82
5.7 Values.yaml.....	82
5.7.1 Image Pull Secrets.....	83
5.7.2 Name Override.....	83
5.7.3 Proxy Config.....	83
5.7.4 Container Image Repository.....	83
5.7.5 Resources.....	84
5.7.6 Pod Security Context.....	84
5.7.7 Container Security Context.....	85
5.7.8 ESACred Secrets.....	85
5.7.9 PBE Secrets.....	85
5.7.10 Security Config Destination.....	86
5.7.11 Persistent Volume Claim.....	86
5.7.12 Storage Provider.....	86
5.7.13 Storage Access Secrets.....	87
5.7.14 Pod Service Account.....	87
5.7.15 Policy.....	87
Chapter 6 Appendix B: Sample Application Helm Chart Details.....	88
6.1 Chart.yaml.....	88
6.2 Credentials.json.....	88
6.3 Values.yaml.....	88
6.3.1 Image Pull Secrets.....	89
6.3.2 Name Override.....	89
6.3.3 Container Image Repository.....	89
6.3.4 Resources.....	90
6.3.5 Pod Service Account.....	90
6.3.6 Liveness and Readiness Probe Settings.....	90
6.3.7 Pod Security Context.....	91
6.3.8 Container Security Context.....	91
6.3.9 PBE Secrets.....	92
6.3.10 Deployment.....	92
6.3.11 Autoscaling.....	93
6.3.12 Service Configuration.....	93
6.3.13 Routes Configuration.....	94
Chapter 7 Appendix C: Using the Dockerfile to Build Custom Images.....	95
7.1 Creating Custom Images Using Dockerfile.....	95
Chapter 8 Appendix D: Applying the NSS Wrapper to the Customer Application Container Image.....	98

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 Java form factor:
 - The delivery form factor for cloud deployments is a container. Protegrity has developed an AP Java Container form factor based on the Application Protector form factor that we have been delivering for several years.
 - The AP Java 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 Java libraries.
 - The AP Java Container form factor makes the container a lightweight deployment of the AP Java.
- 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 Java Container with Policy instantiating a new one.
 - Changes to Policy or the AP Java 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 shrunk 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 deployment consists of the following two stages:

- Stage 1: Deploying the Get ESA Policy Container on a Kubernetes cluster to fetch the ESA policy and create the policy snapshot.
- Stage 2: Deploying the 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 figures represents the architecture for deploying the Get ESA Policy container on a Kubernetes cluster.

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

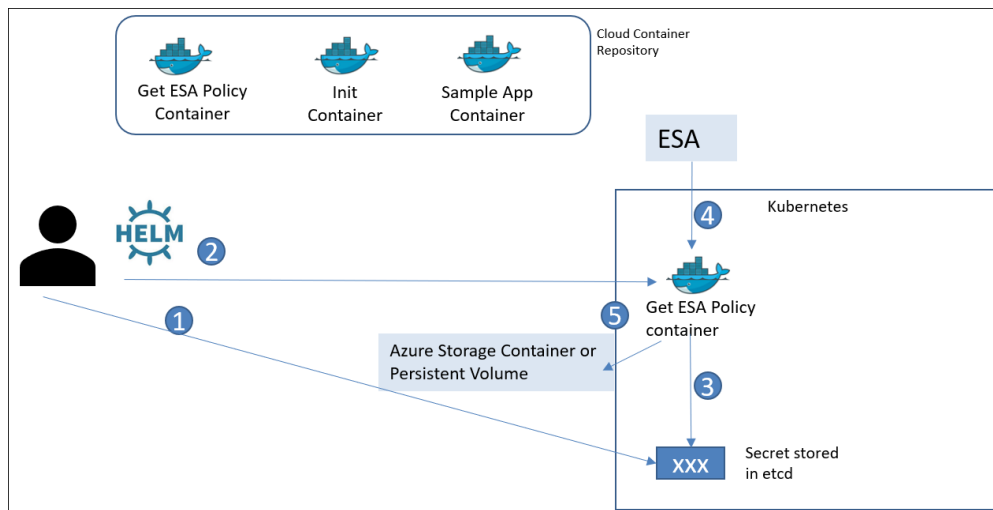


Figure 1-1: Stage 1: Deploying the Get ESA Policy Container

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

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

5. The Kubernetes job is completed.

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

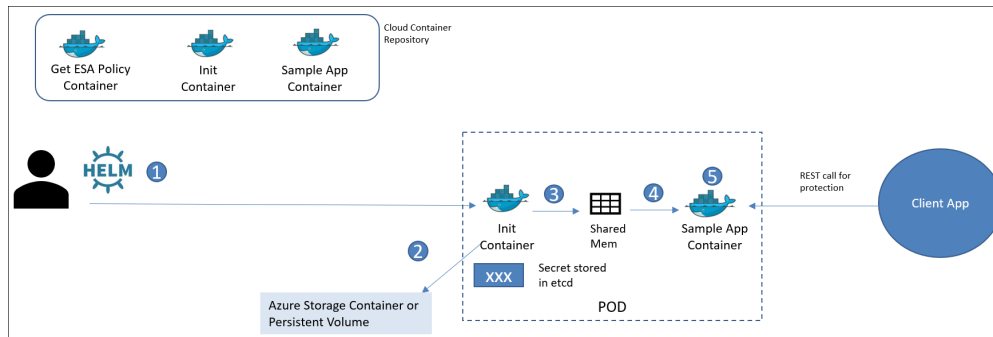


Figure 1-2: Stage 2: Deploying the Sample Application Container

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 and Init containers to the Kubernetes cluster. The Sample Application is integrated with the Application Protector Java 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 Java libraries.

2. The Init Container is initialized first. It reads the immutable policy package from the Object Storage, such as Azure storage container, or a persistent volume, decrypts it using the Kubernetes secret, and stores the policy in the shared memory of the Pod and then exits.
3. The Sample Application container is then initialized. It uses the policy on the shared memory.
4. The Client Application sends a request to the Sample application over REST for protecting, unprotecting, and reprotecting data.
5. The Sample Application internally sends the request to the Application Protector Java, and then returns the accurate 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 Get ESA Policy and Sample Application containers on a Kubernetes cluster.

2.1 Verifying the Prerequisites

Ensure that the following prerequisites are met:

Reference Solution Deployment

- *IAP_RHUBI-8-64_x86-64_OPENSHIFT.K8S.JRE-1.8_<IAP_version>.tgz* – Installation package for the IAP Deployment. This package contains the following packages:
 - *IAP-GET-ESA-POLICY-HELM_ALL-ALL-ALL_x86-64_OPENSHIFT.K8S_<version_number>.tgz* - Package containing the Helm charts, which are used to deploy the Get ESA Policy application on the Kubernetes cluster. Helm is a package manager for Kubernetes.
 - *IAP-GET-ESA-POLICY_RHUBI-8-64_x86-64_OPENSHIFT.K8S_<version_number>.tar.gz* - Get ESA Policy container image. This image is used to create the Get ESA Policy container, which is used to generate an immutable snapshot of the policy and upload this snapshot to the cloud storage in the Cloud Runtime Environment.
 - *IAP-INIT-CONTAINER_RHUBI-8-64_x86-64_OPENSHIFT.K8S_<version_number>.tar.gz* - Init container image. This image is used to create the Init container, which is used to retrieve the policy snapshot stored in the persistent volume and make the snapshot available to the Sample Application container.
 - *APJavaIMSetup_<OS>_x64_<App_Protector_version>.tgz* - Immutable AP Java installation package.
 - *IAP-SAMPLE-APP-HELM_ALL-ALL-ALL_x86-64_K8_<version_number>.tgz* – Package containing the Helm charts, which are used to deploy the sample application on the Kubernetes cluster.
 - *IAP-GET-ESA-POLICY_OPENSHIFT_SRC_<version_number>.tgz* - Package containing the Dockerfile that can be used to create a custom image for the Get ESA Policy container and the associated binary files.

For more information about building a custom image using the Dockerfile, refer to the section [Creating Custom Images Using Dockerfile](#).

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

For more information about building a custom image using the Dockerfile, refer to the section [Creating Custom Images Using Dockerfile](#).

- *IAP-SAMPLE-APP_OPENSHIFT_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](#).

- *HOW-TO-BUILD-DOCKER-IMAGES* - 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. In addition, ensure that you fulfill either of the following requirements:
 - You must attach the policy to the default data store in the ESA.
 - You must add the Kubernetes node or pod IP address ranges, where you want to deploy the containers, as allowed servers to the data store to which the policy is attached.

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

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

- Trusted Application - Create a Trusted Application with the name as *com.protegrity.sample.apjavarest.APJavaSpringApp* 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 Java.

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 [Policy Management Guide 9.1.0.0](#).

- Non-admin ESA user - Create a non-admin ESA user that will be used by the Get ESA Policy container to retrieve the security policy and the certificates from the ESA. Ensure that the user is assigned the *Export Certificates* and the *Appliance CLI Viewer* roles.

For more information about assigning roles, refer to the section *Managing Roles* in the [Appliances Overview Guide 9.1.0.0](#).

Additional Prerequisites

The following additional prerequisites need to be met:

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

Important: Ensure that you enable the auto scaling of nodes in the cluster.

For more information about auto scaling of nodes in an OpenShift cluster, refer to the [OpenShift documentation](#).

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

For more information about persistent volumes and persistent volume claims, refer to the section [Persistent Volumes](#) in the Kubernetes documentation.

- Access to a Container registry to upload the Get ESA Policy, Init, and Sample Application container images.
- Access to an Azure subscription and a valid Azure Active Directory user for creating the following Azure services:

Important: This prerequisite is optional, and is required only if you want to use the Azure Storage Container storing the policy package, instead of using the persistent volume.

- Resource Group - Used to organize resources. All the remaining Azure services are created within a Resource Group.
- Azure Storage Account for creating the Azure Storage Container, where the immutable policy package will be uploaded.
- Azure Storage Container - This prerequisite is optional, and is required only if you want to use the Azure Storage Container for storing the policy package.
- One service principal with *Read* and *Write* permissions on the Azure Storage Account.

Note: A service principal is created when you register an application on the Azure portal.

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-8-64_x86-64_OPENSHIFT.K8S.JRE-1.8_<IAP_version>.tgz* file on the Linux instance.
2. Run the following command to extract the files from the *IAP_RHUBI-8-64_x86-64_OPENSHIFT.K8S.JRE-1.8_<IAP_version>.tgz* file.

```
tar -xvf IAP_RHEL-ALL-64_x86-64_OPENSHIFT.K8S.JRE-1.7_<IAP_version>.tgz
```

The following files are extracted:

- *APJavaIMSetup_<OS>_x64_<App_Protector_version>.tgz*
- *IAP-GET-ESA-POLICY-HELM_ALL-ALL-ALL_x86-64_OPENSHIFT.K8S_<version_number>.tgz*
- *IAP-GET-ESA-POLICY_RHUBI-8-64_x86-64_OPENSHIFT.K8S_<version_number>.tar.gz*
- *IAP-INIT-CONTAINER_RHUBI-8-64_x86-64_OPENSHIFT.K8S_<version_number>.tar.gz*
- *IAP-SAMPLE-APP-HELM_ALL-ALL-ALL_x86-64_OPENSHIFT.K8S_<version_number>.tgz*
- *IAP-GET-ESA-POLICY_OPENSHIFT_SRC_<version_number>.tgz*
- *IAP-INIT-CONTAINER_OPENSHIFT_SRC_<version_number>.tgz*

- *IAP-SAMPLE-APP_OPENSIFT_SRC_<version_number>.tgz*
- *HOW-TO-BUILD-DOCKER-IMAGES*
- *manifest.json*

2.3 Initializing the Linux Instance

This section describes how you can initialize the Linux instance.

► To initialize the Linux instance:

1. Install OpenShift CLI, which provides a set of command line tools for the OpenShift Cloud Platform.
For more information about installing OpenShift CLI on the Linux instance, refer to one of the following websites 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 Get ESA Policy container.

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

For example:

```
tar -xvf IAP-GET-ESA-POLICY-HELM_ALL-ALL-ALL_x86-64_OPENSIFT.K8S_<version_number>.tgz
```

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

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

- *pty-scc.yaml* – The Protegrity Security Context Constraints (SCC) file that is used to manage the security policies for a pod.
- *pty-rbac.yaml* – The values for configuring the Protegrity Role Based Access Control (RBAC) for the pods.
- *nfs-pv.yaml* – The default values for configuring the persistent volume.
- *nfs-pvc.yaml* – The default values for configuring the persistent volume claim.
- *credentials.json* – Credentials file for the service principal, which is required for uploading the policy packages that are uploaded on Azure. By default, the *credentials.json* file contains placeholders, which need to be replaced with the actual values applicable for the service principal.

Important: This file is required only if you want to use the Azure Storage Container storing the policy package, instead of using the persistent volume.

5. 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_number>.tgz
```

The extracted package contains the *spring-apjava* 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
- *credentials.json* – Credentials file for the service principal, which is required for reading the policy packages that are uploaded on Azure. By default, the *credentials.json* file contains placeholders, which need to be replaced with the actual values applicable for the service principal.

Important: This file is required only if you want to use the Azure Storage Container storing the policy package, instead of using the persistent volume.

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

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

Note:

Before running the command, ensure that you have created an OpenShift cluster.

2.4 Creating Certificates and Keys for TLS Authentication

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

Note:

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

Before you begin

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

► To create certificates and keys for TLS authentication:

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

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

Note:

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

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

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

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

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

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

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

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

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

4. Copy all the certificates to a common directory.

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

2.5 Creating the Cloud Runtime Environment

This section describes how to create the Cloud runtime environment.

2.5.1 Creating the OpenShift Runtime Environment

This section describes how to create the OpenShift runtime environment.

Prerequisites

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

- Login URL for the OpenShift account

- Authentication credentials for the OpenShift account

Audience

It is recommended that you have working knowledge of OpenShift.

2.5.1.1 Logging in to the OpenShift Environment

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

► To login to the OpenShift environment:

1. Access OpenShift at the following URL:
https://<OpenShift Console URL>/dashboards

The OpenShift home screen appears.

2. Click **Sign In to the Console**.
The Sign in screen appears.

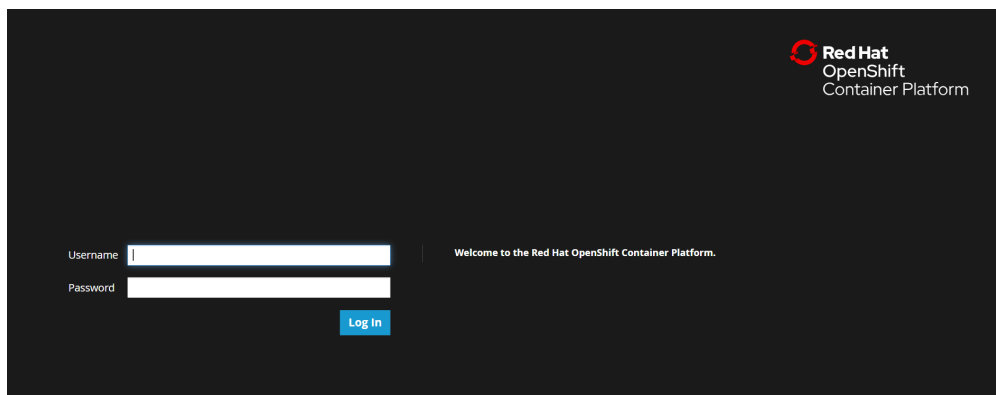


Figure 2-1: OpenShift Container Platform Sign in screen

3. Enter the following details:
 - User name
 - Password
4. Click **Log In**.
After successful authentication, the **Dashboards** screen appears.

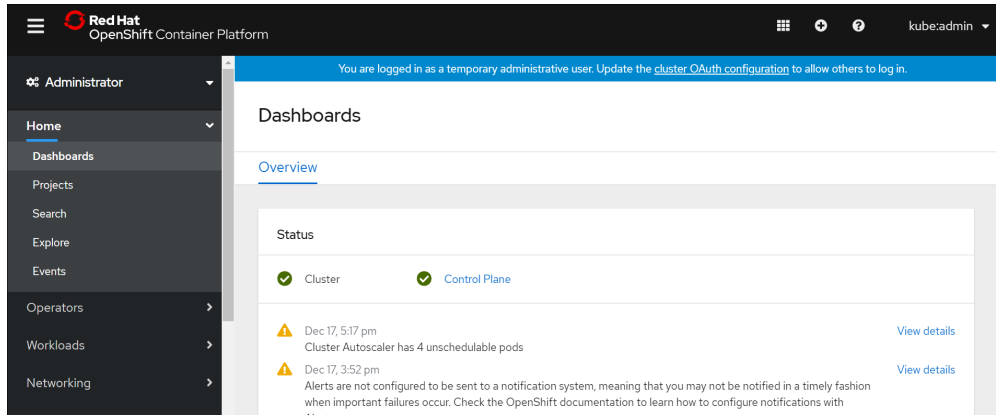


Figure 2-2: Dashboards Screen

2.5.2 Creating the Azure Runtime Environment

This section describes how to create the Azure runtime environment.

Important: This procedure is optional, and is required only if you want to use the Azure Storage Container storing the policy package, instead of using the persistent volume.

Prerequisites

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

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

Audience

It is recommended that you have working knowledge of Azure and knowledge of the following concepts:

- Introduction to Azure Storage
- Introduction to Azure Cybersecurity

2.5.2.1 Logging in to the Azure Environment


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

For more information about how to login to the Azure environment, refer the [Microsoft Azure documentation](#).

► To login to the Azure environment:

1. Access Azure using the following URL:
<https://azure.microsoft.com>
The **Microsoft Azure** home screen appears.
2. Click **Sign in**.
The **Microsoft Sign in** screen appears.
3. On the **Microsoft Sign in** screen, enter the email address for logging in to Azure, and then click **Next**.

The **Enter Password** screen appears.

4. Enter the password for authenticating your email address and click **Next**.
After successful authentication, you are logged in to Microsoft Azure. The Azure home screen appears.
5. Click **Portal**, which is on the top-right side of the Azure home screen.
The Azure Portal home screen appears. By default, the home screen displays a list of frequently used Azure services.
6. Click the **Portal** menu icon ().
The **Portal** menu appears.
7. Click **Resource groups**.
The **Resource groups** screen appears.
8. Select the required resource group from the available list.


Note: If a resource group is not available, then contact your IT administrator for creating a resource group.

2.5.2.2 Registering an Application

This section describes how you can register an application in the Azure Active Directory. After you register an application, a corresponding service principal is automatically created. You then need to generate the secret for the service principal.

For more information about registering an application, refer to the [Microsoft Azure documentation](#).

To register an application:

1. Login to the Azure environment.
For more information about logging in to the Azure environment, refer to the section [Logging in to the Azure Environment](#).
2. Click the **Portal** menu icon ().
The **Portal** menu appears.
3. Navigate to **Azure Active Directory**.
The **Overview** screen of the Azure Active Directory appears.
4. Navigate to **App registrations**.
The **App Registrations** screen appears.
5. Click **New registration**.
The **Register an application** screen appears.
6. In the **Name** field, specify a name for the service principal.
For example, specify **service-principal**.
7. In the **Supported account types** field, retain the default option for single tenant.
8. Click **Register** to register your application.
The application is registered and the **Overview** screen appears, displaying the details about the registered application.

Important: Note the values of the **Application (client) ID** and **Directory (tenant) ID**. You need to specify these values in the credentials file for the corresponding service principal.

For more information about the *credentials.json* file, refer to the section [Credentials.json](#).

9. Perform the following steps to create a client secret for the registered application.

- a. On the left pane, click **Manage > Certificates & secrets**.

The **Certificates & secrets** screen appears.

- b. Click **New client secret**.

The **Add a client secret** screen appears.

- c. In the **Description** field, specify a description for the client secret.

- d. In the **Expires** option, retain the default value, **In 1 year**.

- e. Click **Add**.

The client secret for the registered application appears in the **Client secrets** section.

- f. Click the Copy to clipboard icon  next to the **Value** column to copy the value of the client secret.

Important: Copy the client secret value to a secure location. You need to specify this value in the *credentials.json* file for the corresponding service principal.

For more information about the *credentials.json* file, refer to the section [Credentials.json](#).

2.5.2.3 Creating an Azure Storage Container

This section describes how to create an Azure Storage Container.

Important: This procedure is optional, and is required only if you want to use the Azure Storage Container for storing the policy package, instead of using the Persistent Volume.

For more information about creating an Azure Storage Container, refer to the [Microsoft Azure documentation](#).

► To create an Azure Storage Container:

1. Login to the Azure environment.

For more information about logging in to the Azure environment, refer to the section [Logging in to the Azure Environment](#).

2. Click the **Portal** menu icon ().

The **Portal** menu appears.

3. Click **Storage accounts**.

The **Storage accounts** screen appears.

4. Click **Add**.

The **Create storage account** screen appears. By default, the **Basics** tab appears.

5. In the **Basics** tab, enter the following mandatory details.

Field	Description
Subscription	By default, this value is set to <i>Azure Cloud Platform</i> .

Field	Description
Resource group	Select the required resource group.
Location	Select a region where the Azure storage account will be created. For example, select <i>(US) US East 2</i> .

Note: Retain the default values for all the remaining tabs on the **Create storage account** screen.

6. Click **Review + create** to validate the entered information.
Azure validates the information that you have entered.
7. Click **Create** to create the storage account.
A screen appears indicating that the storage account has been created.
8. Click **Go to resource**.
The **Overview** screen appears, displaying the details of your storage account.
9. Perform the following steps to assign a role to the service principals for accessing the storage account.
 - a. On the left pane, click **Access control (IAM)**.
The **Access control (IAM)** screen appears.
 - b. Click **Add**.
 - c. Select **Add role assignment**.
The **Add role assignment** dialog box appears.
 - d. In the **Role** list, select the required role.
For example, for service principal, select the role as *Storage Blob Data Contributor* and *Storage Blob Data Reader*.
 - e. In the **Assign access to** list, retain the default value *Azure AD user, group, or service principal*.
 - f. In the **Select** list, select service principal.
The service principal has the same name as that of the application that you have registered in the section [Registering an Application](#).
 - g. Click **Save** to assign the role to the service principal.
10. Perform the following steps to create a storage container.
 - a. On the left pane, click **Blob service > Containers**.
The **Containers** screen appears.
 - b. Click **Container**.
The **New Container** screen appears.
 - c. Specify a name for the container in the **Name** field.
 - d. Retain the default value for the **Public access level** list.
The default value is set to *Private (no anonymous access)*.
 - e. Click **OK** to create the container.

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

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

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

2.6.1 Applying the SCC

This section describes how to apply the SCC.

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

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

► To apply the SCC:

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

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

For example:

```
oc new-project iap
```

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/kubectrl-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 OpenShift infrastructure pods.

Important: If you want to apply the Restricted SCC to any other namespace, then you need to run the following command:

```
oc adm policy add-scc-to-group restricted system:serviceaccounts:<Namespace>
```

5. Navigate to the directory where you have extracted the Helm charts for deploying the Get ESA Policy application, and edit the `pty-scc.yaml` file to replace the `<NAMESPACE>` placeholder in the following snippet with the namespace where the 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 Get ESA Policy, Init, and Sample Application containers.

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

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

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

For example:

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

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

If you are using the OpenShift version 4.12, then perform [step 1b](#) to create a token for the service account.

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

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

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

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

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

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

For example:

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

The output displays the following token names.

```
[{"name": "get-esa-deployer-dockercfg-ntmbt"}, {"name": "get-esa-deployer-token-s916p"}]
```

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

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

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

For example:

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

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

You can obtain the context by running the following command:

```
oc config current-context
```

- iii. Run the following command to decode the service account token obtained in [step 1c > ii](#).

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

- d. If you are using the OpenShift version 4.12, then repeat [step 1a](#) and [step 1b](#) to create a service account that can be used to deploy the Sample Application container.

If you are using an OpenShift version prior to 4.12, then repeat [step 1a](#) and [step 1c](#).

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

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

Important:

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

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

For example:

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

For more information about the extracted Helm charts, refer to the [step 4](#) of the section [Initializing the Linux Instance](#).

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

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

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

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

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

For more information about the extracted Helm charts, refer to the [step 4](#) of the section [Initializing the Linux Instance](#).

2. If you want to use an Azure Storage Container for storing the policy package instead of the persistent volume, then modify the *credentials.json* file to update the credentials for the service principal.

The *credentials.json* file has the following format.

```
{
  "storage_account_name" : "AZURE_STORAGE_ACCOUNT_NAME",
  "tenant_id" : "AZURE_TENANT_ID",
  "client_id" : "AZURE_CLIENT_ID",
  "client_secret": "AZURE_CLIENT_SECRET"
}
```

For more information about the values for the *tenant_id*, *client_id*, and *client_secret* keys, refer to the section [Registering an Application](#).

For more information about the structure of the *credentials.json* file, refer to the section [Credentials.json](#).

3. If you want to use an Azure Storage Container for storing the policy package, instead of the persistent volume, then run the following command to create the *create-obj-access* secret, which is used by the Get ESA Policy container to upload the immutable policy package to the Azure storage container.

```
kubectl create secret generic create-obj-access --from-file=get-esa-policy/
credentials.json --namespace <NAMESPACE>
```

```
kubectl create secret generic create-obj-access --from-file=get-esa-policy/
credentials.json --namespace iap
```

For more information about the *credentials.json* file, refer to the section [Credentials.json](#).

Note: Ensure that you specify the name of the secret as the value of the *storageAccessSecrets* field in the *values.yaml* file.

For more information about the *storageAccessSecrets* parameters, refer to the section [Storage Access Secrets](#).

- If you want to use a persistent volume for storing the policy package, instead of the Azure Storage Container, then 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: MOUNT_PATH
    # The host name or IP address of the NFS server.
    server: SERVER_NAME_OR_IP
    readOnly: false
```

- Run the following command to create a persistent volume.
`oc apply -f get-esa-policy/nfs-pv.yaml`
- Modify the *nfs-pvc.yaml* file to update the persistent volume claim details, by specifying the name of the persistent volume claim and the storage amount.

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

- Run the following command to create a persistent volume claim.
`oc apply -f get-esa-policy/nfs-pvc.yaml -n iap`
- 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.

- Run the following mount command to mount the NFS on the directory created in [step 6](#).
`sudo mount <nfs server IP>:<Path to the shared folder> /nfs`

2.6.4 Deploying the Get ESA Policy Container on the Kubernetes Cluster

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

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

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

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

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

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

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

3. Perform the following steps to set up a Kubernetes context for the *get-esa-deployer* service account.
 - a. If you are using the OpenShift version 4.12, then run the following command to create user credentials in the *kube-config* file, using the token created for the service account to deploy the GET ESA Policy container in [step 1b](#) of the section [Applying RBAC](#).

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

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

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

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

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

Note: Ensure that the name of the Kubernetes cluster is the same as that you have specified in [step 2](#).

You can obtain the name of the Kubernetes cluster by running the `oc config get-contexts` command.

- c. Run the following command to set the Kubernetes context to the newly created context in [step 3b](#).

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

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

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

For example:

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

Important: Ensure that the user, who will be used to access the ESA, has been assigned the *Export Certificates* and *Appliance CLI Viewer* permissions through a custom role. Roles are templates that include permissions and users can be assigned one or more roles.

For more information about managing roles, refer to the section *Managing Roles* in the [Appliances Overview Guide 9.1.0.0](#).

For more information about managing users, refer to the section *Managing Users* in the [Appliances Overview Guide 9.1.0.0](#).

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


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

For example:

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

For more information about the credentials required to access the image repository, refer to the section [Image Pull Secrets](#).

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

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

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

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

Important: Ensure that the passphrase has a minimum length of 12 characters, with at least 1 uppercase letter, 2 digits, 2 special characters, and 2 non-letters (such as, digits or special characters). The passphrase must also have a minimum value of entropybits as 30 and a minimum strength of 0.66.

Entropybits defines the randomness of the password and the strength defines the complexity of the password.

For more information about the password strength, refer to the section [Password Strength](#).

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

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

```
...
..
.

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

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

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

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

```

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

## choose your storage destination. Currently we support <VolumeMount | ObjectStore>
## default is VolumeMount, change it to ObjectStore if you wish to use object store
securityConfigDestination: VolumeMount
## If securityConfigDestination is VolumeMount
## name of persistent volume claim to be used for this deployment.
pvcName: PVC_NAME
## If securityConfigDestination is ObjectStore
## specify the respective cloud provider ['AWS', 'AZURE', 'GCP']
storageProvider:
## If securityConfigDestination is ObjectStore
## specify k8s secret for storing credentials with access to cloud storage.
storageAccessSecrets:

policy:
  esaIP: ESA_IP

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

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

```

For more information about Helm charts and their default values, refer to the [Appendix A: Get ESA Policy Helm Chart Details](#).

Field	Description
podSecurityContext	<p>Specify the privilege and access control settings for the pod.</p> <p>The default values are set as follows:</p> <ul style="list-style-type: none"> • <i>runAsUser</i> - 1000 • <i>runAsGroup</i> - 1000 • <i>fsGroup</i> - 1000 <p>Note: Set the value of the <i>fsGroup</i> parameter to a non-root value.</p>

Field	Description
	<p>For example, you can set the value of the <i>fsGroup</i> parameter to any value between <i>1</i> and <i>65534</i>.</p> <p>Note: Ensure that at any given point of time, the value of at least one of the four parameters must be <i>1000</i>. This ensures that the pod has the required permissions to access the <i>pty</i> directories in the containers.</p> <p>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.</p> <p>For more information about the parameters, refer to the section PodSecurityContext v1 Core in the Kubernetes API documentation.</p>
esaCredSecrets	Specify the value of the secret that is used to store the ESA credentials. This is used by the Get ESA Policy container to retrieve the policy from the ESA.
pbeSecrets	<p>Specify the value of the secret that is used to store the passphrase and salt. The Get ESA Policy container encrypts the policy using the key generated by using the passphrase and salt.</p> <p>Important: Ensure that the passphrase has a minimum length of 12 characters, with atleast 1 uppercase letter, 2 digits, 2 special characters, and 2 non-letters (such as, digits or special characters). The passphrase must also have a minimum value of entropybits as <i>30</i> and a minimum strength of <i>0.66</i>. Entropybits defines the randomness of the password and the strength defines the complexity of the password.</p> <p>For more information about the password strength, refer to the section Password Strength.</p>
securityConfigDestination	<p>Specify one of the following options:</p> <ul style="list-style-type: none"> <i>VolumeMount</i> - Specify this value if you want to store the policy package on a volume that is mounted on the pod. Use this option if you are using the persistent volume for storing the policy package. <i>ObjectStore</i> - Specify this value if you want to store the policy package in the Azure Storage Container. <p>By default, the value is set to <i>VolumeMount</i>.</p>
pvcName	<p>Specify the name of the Persistent Volume Claim where you want to store the policy snapshot.</p> <p>If you want to use the Azure Storage Container for storing the policy package, then leave the value of this field blank.</p>
storageProvider	Specify the value as <i>AZURE</i> if you want to upload the policy package to the Azure storage container.

Field	Description
	If you want to use the persistent volume for storing the policy package, then leave the value of this field blank.
storageAccessSecrets	<p>Specify the value of the secret, which you have created in Step 3, to store the credentials with write access to the Azure storage container. This is used by the Get ESA Policy container to upload the policy package to the Azure storage container.</p> <p>If you want to use the persistent volume for storing the policy package, then leave the value of this field blank.</p> <p>Important: This field is required if you want to store the immutable policy package in the Azure Storage Container.</p>
policy/ESA_IP	Specify the IP address of the ESA from where you want to download the policy.
policy/pepserverConfig/emptyString	<p>Defines the behavior when the data to protect is an empty string. The following are the possible values:</p> <ul style="list-style-type: none"> <i>empty</i> - An empty value is returned for an empty input string. <i>encrypt</i> - An encrypted value is returned for an empty input string. <i>null</i> - A null value is returned for an empty input string. <p>By default, the value of this parameter is set to <i>null</i>.</p> <p>For more information about empty string handling by protectors, refer to the section <i>Appendix C: Empty String Handling by Protectors</i> in the <i>Protection Methods Reference Guide 9.1.0.0</i>.</p>
policy/pepserverConfig/logLevel	<p>Specifies the logging level for the PEP server. You can specify one of the following values:</p> <ul style="list-style-type: none"> <i>OFF</i> - No logging <i>SEVERE</i> <i>WARNING</i> <i>INFO</i> <i>CONFIG</i> <i>ALL</i> <p>By default, the value of this parameter is set to <i>ALL</i>.</p>
policy/pepserverConfig/caseSensitive	<p>Specifies how policy users are checked against the policy. This parameter enables you to configure the case-sensitivity of the policy users.</p> <p>If this parameter is set to <i>no</i>, then the PEP server considers the policy user names as case insensitive.</p> <p>If this parameter is set to <i>yes</i> or if it is commented in the file, then the PEP server considers the policy user names as case-sensitive. The default value is <i>yes</i>.</p> <p>By default, the value of this parameter is set to <i>yes</i>.</p>

Field	Description
policy/filePath	<p>Specify the path in the persistent volume or the object store where you want to store the immutable policy package.</p> <p>For example, if you are using NFS as the persistent volume, then you can specify the file path as <code>/<Mount directory>/xyz/imp</code>. In this case, a policy with the name as <code>imp</code> will be stored in the <code>/<Mount directory>/xyz</code> directory in the required persistent volume.</p> <p>Note: The <code><Mount directory></code> is a directory inside the Mount point.</p> <p>If you want to store the policy package in an Object Store, such as an Azure Storage Container, then you can specify the container name as the first name in the <code>filePath</code> field.</p> <p>For example, you can specify the value of the <code>filePath</code> field, as <code>test/LOB/policy</code>, where <code>test</code> is the container name, <code>LOB</code> is the directory inside the container, and <code>policy</code> is the name of the immutable policy package. In this example, the container name is specified as the first name of the file path.</p> <p>Important: Use only uppercase and lowercase letters, numbers, dot, and underscore in the file path. Do not use special characters in the file path.</p> <p>Note: This value is case-sensitive.</p>
policy/timeout	<p>Specify the time for which the Get ESA Policy container tries to communicate with the ESA for fetching the policies, after which the connection times out. By default, this value is set to 15. The unit of the timeout parameter is minutes.</p>

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

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

For example:

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

- Run the following command to check the deployment status.

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

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

2.6.5 Verifying the IAP Policy Snapshot on the NFS

This section describes how you can verify the IAP policy snapshot on the NFS.

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

Important: This procedure is optional, and is required only if you want to use the persistent volume storing the policy package, instead of using the Azure Storage Container.

► **To verify the policy snapshot on the NFS:**

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

1. On the Linux instance, navigate to the location where you have mounted the NFS.
For more information about the location where you have mounted the NFS, refer to [step 6](#) in the section [Setting up the Environment for Deploying the Get ESA Policy Container](#).
2. List all the files within the directory to access the policy snapshot.


2.6.6 Verifying the IAP Policy Package on the Azure Storage Container

The Get ESA Policy container retrieves the policy from the ESA and uploads a snapshot of this immutable policy to the Azure storage container. This section describes how to verify the policy package in the Azure environment.

Important: This procedure is optional, and is required only if you want to use the Azure Storage Container for storing the policy package, instead of using the persistent volume.

► **To verify the policy package on the Azure Storage Container:**

Perform the following task to verify whether the immutable policy package has been uploaded to the Azure storage container.

1. Login to the Azure environment.
For more information about logging in to the Azure environment, refer to the section [Logging in to the Azure Environment](#).
2. Click the **Portal** menu icon ().
The **Portal** menu appears.
3. Click **Storage accounts**.
The **Storage accounts** screen appears.
4. Navigate to the path that you have specified as the value of the `filePath` parameter in the `values.yaml` file in [step 7](#) of the section [Deploying the Get ESA Policy Container on the Kubernetes Cluster](#).
The immutable policy package appears.
5. Click the immutable policy package to view additional details about the file.

2.6.7 Setting up the Environment for Deploying the Sample Application Container

This section describes how to set up the environment for deploying the Sample Application container on the Kubernetes cluster.

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

► To set up the environment for deploying the Sample Application container:

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

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

The TLS certificates are created using the procedure described in the section [Creating Certificates and Keys for TLS Authentication](#).

2. If you want to use an Azure Storage Container for storing the policy package, instead of the persistent volume, then perform the following steps.
 - a. Modify the *credentials.json* file, which is present inside the directory where the Helm charts for the Sample Application container have been extracted, to update the credentials for the service principal.

The following snippet contains the format of the *credentials.json* file.

```
{
  "storage_account_name" : "AZURE_STORAGE_ACCOUNT_NAME",
  "tenant_id" : "AZURE_TENANT_ID",
  "client_id" : "AZURE_CLIENT_ID",
  "client_secret": "AZURE_CLIENT_SECRET"
}
```

For more information about the values for the *tenant_id*, *client_id*, and *client_secret* keys, refer to the section [Registering an Application](#).

For more information about the structure of the *credentials.json* file, refer to the section [credentials.json](#).

- b. Run the following command to create the *read-obj-access* secret, which is used by the Sample Application container to download the immutable policy package from the Azure storage container to the shared memory.

```
kubect1 create secret generic read-obj-access --from-file=spring-apjava/credentials.json --namespace <NAMESPACE>
```

```
kubect1 create secret generic read-obj-access --from-file=spring-apjava/credentials.json --namespace iap
```

For more information about the *credentials.json* file, refer to the section [credentials.json](#).

Note: Ensure that you specify the name of the secret as the value of the *storageAccessSecrets* field in the *values.yaml* file.

2.6.8 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 5](#) 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. If you are using the OpenShift version 4.12, then run the following command to create user credentials in the *kube-config* file, using the token created for the service account to deploy the Sample Application container in [step 1b](#) of the section [Applying RBAC](#).

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

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

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

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

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

Note: Ensure that the name of the Kubernetes cluster is the same as that you have specified in [step 2](#).

You can obtain the name of the Kubernetes cluster by running the `oc config get-contexts` command.

- c. Run the following command to set the Kubernetes context to the newly created context in [step 3b](#).

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

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

The *spring-apjava > 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
...

## 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>
  ## 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

## 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 want to expose the service externally.
## targetPort - the port no. configured while creating Tunnel.
springappService:
  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	<p>Specify the privilege and access control settings for the pod.</p> <p>The default values are set as follows:</p> <ul style="list-style-type: none"> • <i>runAsUser</i> - 1000 • <i>runAsGroup</i> - 1000 • <i>fsGroup</i> - 1000 <p>Note: Set the value of the <i>fsGroup</i> parameter to a non-root value.</p> <p>For example, you can set the value of the <i>fsGroup</i> parameter to any value between 1 and 65534.</p> <p>Note: Ensure that at any given point of time, the value of at least one of the four parameters must be 1000.</p> <p>This ensures that the pod has the required permissions to access the <i>pty</i> directories in the containers.</p> <p>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.</p>

Field	Description
	For more information about the parameters, refer to the section PodSecurityContext v1 Core in the Kubernetes API documentation.
pbeSecrets	<p>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 <i>pbeSecrets</i> parameter to decrypt the policy.</p> <p>Important: Ensure that the passphrase has a minimum length of 12 characters, with at least 1 uppercase letter, 2 digits, 2 special characters, and 2 non-letters (such as, digits or special characters). The passphrase must also have a minimum value of entropybits as <i>30</i> and a minimum strength of <i>0.66</i>. Entropybits defines the randomness of the password and the strength defines the complexity of the password.</p> <p>For more information about the password strength, refer to the section Password Strength.</p>
deploymentInUse	Specify the value as <i>blue</i> . This indicates that the <i>Blue</i> deployment strategy is in use.
blueDeployment/enabled	Specify the value as <i>true</i> to enable the <i>blue</i> deployment strategy.
blueDeployment/securityConfigSource	<p>Specify one of the following options:</p> <ul style="list-style-type: none"> <i>VolumeMount</i> - Specify this value if you want to store the policy package on a volume that is mounted on the pod. Use this option if you are using the persistent volume for storing the policy package. <i>ObjectStore</i> - Specify this value if you want to store the policy package in the Azure Storage Container. <p>By default, the value is set to <i>VolumeMount</i>.</p>
blueDeployment/pvcName	<p>Specify the name of the Persistent Volume Claim where you want to store the policy snapshot.</p> <p>If you want to use the Azure Storage Container for storing the policy package, then leave the value of this field blank.</p>
blueDeployment/storageProvider	<p>Specify the value as <i>AZURE</i> if you want to upload the policy package to the Azure storage container.</p> <p>If you want to use the persistent volume for storing the policy package, then leave the value of this field blank.</p>
blueDeployment/storageAccessSecrets	<p>Specify the value of the secret, which you have created in Step 2b, to store the credentials with read access to the Azure storage container. This is used by the Sample Application container to read the policy package from the Azure storage container.</p> <p>If you want to use the persistent volume for storing the policy package, then leave the value of this field blank.</p> <p>Important: This field is required if you want to store the immutable policy package in the Azure Storage Container.</p>

Field	Description
blueDeployment/policy/filePath	<p>Specify the path in the persistent volume or the Azure Storage Container where you have stored the policy.</p> <p>Ensure that you specify the same value that you have specified as the value of the <i>filePath</i> parameter in the <i>values.yaml</i> file for the Get ESA Policy container in step 7 of the section Deploying the Get ESA Policy Container on the Kubernetes Cluster.</p> <p>Important: Use only uppercase and lowercase letters, numbers, dot, and underscore in the file path. Do not use special characters in the file path.</p> <p>Note: This value is case-sensitive.</p>
greenDeployment/enabled	Specify the value as <i>false</i> . While deploying Release 1, the <i>Green</i> deployment strategy is always disabled.
springappService/name	Specify a name for the tunnel to distinguish between ports.
springappService/port	Specify the port number on which you want to expose the Kubernetes service externally.
springappService/targetPort	<p>Specify the port on which the Sample Application container is running inside the Docker container.</p> <p>Important: Do not change this value.</p>
routes/prodService/hostname	Specify the hostname of the production service.
routes/stagingService/hostname	Specify the hostname of the staging service.

5. 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-java --namespace iap spring-apjava
```

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

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

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

For example:

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

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

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

For example:

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

2.6.9 Upgrading the Helm Charts

This section describes the steps for upgrading the Helm charts.

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

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

► To upgrade the Helm charts:

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

For more information about the extracted Helm charts, refer to the step 5 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
...
...

## 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>
  ## 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

## 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 want to expose the service externally.
## targetPort - the port no. configured while creating Tunnel.
springappService:
  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 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	<p>Specify the privilege and access control settings for the pod.</p> <p>The default values are set as follows:</p> <ul style="list-style-type: none"> <i>runAsUser</i> - 1000 <i>runAsGroup</i> - 1000 <i>fsGroup</i> - 1000 <p>Note: Set the value of the <i>fsGroup</i> parameter to a non-root value.</p> <p>For example, you can set the value of the <i>fsGroup</i> parameter to any value between 1 and 65534.</p> <p>Note: Ensure that at any given point of time, the value of at least one of the four parameters must be 1000.</p> <p>This ensures that the pod has the required permissions to access the <i>pty</i> directories in the containers.</p> <p>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.</p> <p>For more information about the parameters, refer to the section PodSecurityContext v1 Core in the Kubernetes API documentation.</p>
pbeSecrets	<p>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 <i>pbeSecrets</i> parameter to decrypt the policy.</p> <p>Important: Ensure that the passphrase has a minimum length of 12 characters, with atleast 1 uppercase letter, 2 digits, 2 special characters, and 2 non-letters (such as, digits or special characters). The passphrase must also have a minimum value of entropybits as 30 and a minimum strength of 0.66.</p> <p>Entropybits defines the randomness of the password and the strength defines the complexity of the password.</p> <p>For more information about the password strength, refer to the section Password Strength.</p>
deploymentInUse	Specify the value as <i>blue</i> . This indicates that the <i>Blue</i> deployment strategy is in use.
greenDeployment/enabled	Specify the value as <i>true</i> . While upgrading the release to Release 2, the <i>Green</i> deployment strategy is enabled.
greenDeployment/securityConfigSource	<p>Specify one of the following options:</p> <ul style="list-style-type: none"> <i>VolumeMount</i> - Specify this value if you want to store the policy package on a volume that is mounted on the pod. Use this option if you are using the persistent volume for storing the policy package.

Field	Description
	<ul style="list-style-type: none"> <i>ObjectStore</i> - Specify this value if you want to store the policy package in the Azure Storage Container. <p>By default, the value is set to <i>VolumeMount</i>.</p>
greenDeployment/pvcName	<p>Specify the name of the Persistent Volume Claim where you want to store the policy snapshot.</p> <p>If you want to use the Azure Storage Container for storing the policy package, then leave the value of this field blank.</p>
greenDeployment/storageProvider	<p>Specify the value as <i>AZURE</i> if you want to upload the policy package to the Azure storage container.</p> <p>If you want to use the persistent volume for storing the policy package, then leave the value of this field blank.</p>
greenDeployment/storageAccessSecrets	<p>Specify the value of the secret which you have used to store the credentials with read access to the Azure storage container. This is used by the Sample Application container to read the policy package from the Azure storage container.</p> <p>If you want to use the persistent volume for storing the policy package, then leave the value of this field blank.</p> <p>Important: This field is required if you want to store the immutable policy package in the Azure Storage Container.</p>
greenDeployment/policy/filePath	<p>Specify the path in the persistent volume or the Azure Storage Container where you have stored the policy.</p> <p>Important: Use only uppercase and lowercase letters, numbers, dot, and underscore in the file path. Do not use special characters in the file path.</p> <p>Note: This value is case-sensitive.</p>
springappService/name	Specify a name for the tunnel to distinguish between ports.
springappService/port	Specify the port number on which you want to expose the Kubernetes service externally.
springappService/targetPort	<p>Specify the port on which the Sample Application container is running inside the Docker container.</p> <p>Important: Do not change this value.</p>
routes/prodService/hostname	Specify the hostname of the production service.
routes/stagingService/hostname	Specify the hostname of the staging service.

- 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-java --namespace iap spring-apjava
```

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-java --namespace iap spring-apjava
```

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-java --namespace iap spring-apjava
```

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

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

## 2.7 Deploying the Containers without RBAC

This section describes the steps that you need to perform for deploying the Get ESA Policy, Init, and 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 OpenShift infrastructure pods.

**Important:** If you want to apply the Restricted SCC to any other namespace, then you need to run the following command:

```
oc adm policy add-scc-to-group restricted system:serviceaccounts:<Namespace>
```

5. Navigate to the directory where you have extracted the Helm charts for deploying the Get ESA Policy application, and edit the `pty-scc.yaml` file to replace the `<NAMESPACE>` placeholder in the following snippet with the namespace where the Sample Application container 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 Deploying the Get ESA Policy Container on the Kubernetes Cluster

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

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

1. On the Linux instance, navigate to the location where you have extracted the Helm charts for deploying the Get ESA Policy application.
2. If you want to use an Azure Storage Container for storing the policy package, instead of the persistent volume, then modify the `credentials.json` file to update the credentials for the service principal.

The `credentials.json` file has the following format.

```
{
 "storage_account_name" : "AZURE_STORAGE_ACCOUNT_NAME",
 "tenant_id" : "AZURE_TENANT_ID",
 "client_id" : "AZURE_CLIENT_ID",
```

```
} "client_secret": "AZURE_CLIENT_SECRET"
```

For more information about the values for the *tenant\_id*, *client\_id*, and *client\_secret* keys, refer to the section [Registering an Application](#).

For more information about the structure of the *credentials.json* file, refer to the section [Credentials.json](#).

3. If you want to use an Azure Storage Container for storing the policy package, instead of the persistent volume, then run the following command to create the *create-obj-access* secret, which is used by the Get ESA Policy container to upload the immutable policy package to the Azure storage container.

```
kubectl create secret generic create-obj-access --from-file=get-esa-policy/credentials.json --namespace <NAMESPACE>
```

```
kubectl create secret generic create-obj-access --from-file=get-esa-policy/credentials.json --namespace iap
```

For more information about the *credentials.json* file, refer to the section [Credentials.json](#).

**Note:** Ensure that you specify the name of the secret as the value of the *storageAccessSecrets* field in the *values.yaml* file.

For more information about the *storageAccessSecrets* parameters, refer to the section [Storage Access Secrets](#).

4. If you want to use a persistent volume for storing the policy package, instead of the Azure Storage Container, then 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: MOUNT_PATH
 # The host name or IP address of the NFS server.
 server: SERVER_NAME_OR_IP
 readOnly: false
```

5. Run the following command to create a persistent volume.
- ```
oc apply -f get-esa-policy/nfs-pv.yaml
```
6. Modify the *nfs-pvc.yaml* file to update the persistent volume claim details, by specifying the name of the persistent volume claim and the storage amount.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <PVC Name>
spec:
  selector:
    matchLabels:
      purpose: "policy-store"
  accessModes:
    - ReadWriteMany
  resources:
```

```
requests:
  # The amount of storage allocated to this volume.
  # e.g. 10Gi
  storage: <AMOUNT_OF_STORAGE>
```

7. Run the following command to create a persistent volume claim.
`oc apply -f get-esa-policy/nfs-pvc.yaml -n iap`
8. 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.
9. 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`
10. Run the following command to create the *esa-creds* secret, which is used by the Get ESA Policy container to access the ESA using the non-administrator credentials.

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

For example:

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

Important: Ensure that the user, who will be used to access the ESA, has been assigned the *Export Certificates* and *Appliance CLI Viewer* permissions through a custom role. Roles are templates that include permissions and users can be assigned one or more roles.

For more information about managing roles, refer to the section *Managing Roles* in the [Appliances Overview Guide 9.1.0.0](#).

For more information about managing users, refer to the section *Managing Users* in the [Appliances Overview Guide 9.1.0.0](#).

11. Run the following command to create a Kubernetes secret for accessing the image registry using existing Docker credentials.
`oc create secret generic regcred --from-file=.dockerconfigjson=<PATH_TO_DOCKER_CONFIG>/config.json --type=kubernetes.io/dockerconfigjson --namespace <NAMESPACE>`

For example:

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

For more information about the credentials required to access the image repository, refer to the section [Image Pull Secrets](#).

12. Run the following command to create a passphrase secret, which is used by the Get ESA Policy container to encrypt the policy.
`oc create secret generic pbe-secret --from-literal='passphrase=<passphrase>' --from-literal='salt=<salt>' -n iap`

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

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

Important: Ensure that the passphrase has a minimum length of 12 characters, with atleast 1 uppercase letter, 2 digits, 2 special characters, and 2 non-letters (such as, digits or special characters). The passphrase must also have a minimum value of entropybits as 30 and a minimum strength of 0.66.

Entropybits defines the randomness of the password and the strength defines the complexity of the password.

For more information about the password strength, refer to the section [Password Strength](#).

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

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

```
...
..
.

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

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

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

## k8s secret for storing ESA credentials
esaCredSecrets: ESA_CREDENTIAL_SECRET_NAME

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

## choose your storage destination. Currently we support <VolumeMount | ObjectStore>
## default is VolumeMount, change it to ObjectStore if you wish to use object store
securityConfigDestination: VolumeMount
## If securityConfigDestination is VolumeMount
## name of persistent volume claim to be used for this deployment.
pvcName: PVC_NAME
## If securityConfigDestination is ObjectStore
## specify the respective cloud provider ['AWS', 'AZURE', 'GCP']
storageProvider:
## If securityConfigDestination is ObjectStore
## specify k8s secret for storing credentials with access to cloud storage.
storageAccessSecrets:

policy:
  esaIP: ESA_IP

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

```

## CONFIG
## ALL (default)
## caseSensitive: Specifies how policy users are checked against policy
## yes = (The default) policy users are treated in case sensitive manner
## no = policy users are treated in case insensitive manner.
pepserverConfig:
  emptyString: "null"
  logLevel: "ALL"
  caseSensitive: "yes"
  ## absolute path in PV or ObjectStore where the policy dump file will be stored. <eg. /
  test/xyz/policy >
  ## If destination is ObjectStore, make sure first name in path is bucket name i.e. test
  will be our existing bucket
  filePath: ABSOLUTE_POLICY_PATH

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

```

For more information about Helm charts and their default values, refer to the [Appendix A: Get ESA Policy Helm Chart Details](#).

Field	Description
podSecurityContext	<p>Specify the privilege and access control settings for the pod.</p> <p>The default values are set as follows:</p> <ul style="list-style-type: none"> • <i>runAsUser</i> - 1000 • <i>runAsGroup</i> - 1000 • <i>fsGroup</i> - 1000 <p>Note: Set the value of the <i>fsGroup</i> parameter to a non-root value.</p> <p>For example, you can set the value of the <i>fsGroup</i> parameter to any value between <i>1</i> and <i>65534</i>.</p> <p>Note: Ensure that at any given point of time, the value of at least one of the four parameters must be <i>1000</i>.</p> <p>This ensures that the pod has the required permissions to access the <i>pty</i> directories in the containers.</p> <p>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.</p> <p>For more information about the parameters, refer to the section PodSecurityContext v1 Core in the Kubernetes API documentation.</p>
esaCredSecrets	<p>Specify the value of the secret that is used to store the ESA credentials. This is used by the Get ESA Policy container to retrieve the policy from the ESA.</p>
pbeSecrets	<p>Specify the value of the secret that is used to store the passphrase and salt. The Get ESA Policy container encrypts the policy using the key generated by using the passphrase and salt.</p> <p>Important: Ensure that the passphrase has a minimum length of 12 characters, with atleast 1 uppercase letter, 2 digits, 2</p>

Field	Description
	<p>special characters, and 2 non-letters (such as, digits or special characters). The passphrase must also have a minimum value of entropybits as <i>30</i> and a minimum strength of <i>0.66</i>.</p> <p>Entropybits defines the randomness of the password and the strength defines the complexity of the password.</p> <p>For more information about the password strength, refer to the section Password Strength.</p>
securityConfigDestination	<p>Specify one of the following options:</p> <ul style="list-style-type: none"> <i>VolumeMount</i> - Specify this value if you want to store the policy package on a volume that is mounted on the pod. Use this option if you are using the persistent volume for storing the policy package. <i>ObjectStore</i> - Specify this value if you want to store the policy package in the Azure Storage Container. <p>By default, the value is set to <i>VolumeMount</i>.</p>
pvcName	<p>Specify the name of the Persistent Volume Claim where you want to store the policy snapshot.</p> <p>If you want to use the Azure Storage Container for storing the policy package, then leave the value of this field blank.</p>
storageProvider	<p>Specify the value as <i>AZURE</i> if you want to upload the policy package to the Azure storage container.</p> <p>If you want to use the persistent volume for storing the policy package, then leave the value of this field blank.</p>
storageAccessSecrets	<p>Specify the value of the secret, which you have created in Step 3, to store the credentials with write access to the Azure storage container. This is used by the Get ESA Policy container to upload the policy package to the Azure storage container.</p> <p>If you want to use the persistent volume for storing the policy package, then leave the value of this field blank.</p> <p>Important: This field is required if you want to store the immutable policy package in the Azure Storage Container.</p>
policy/ESA_IP	<p>Specify the IP address of the ESA from where you want to download the policy.</p>
policy/pepserverConfig/emptyString	<p>Defines the behavior when the data to protect is an empty string. The following are the possible values:</p> <ul style="list-style-type: none"> <i>empty</i> - An empty value is returned for an empty input string. <i>encrypt</i> - An encrypted value is returned for an empty input string. <i>null</i> - A null value is returned for an empty input string. <p>By default, the value of this parameter is set to <i>null</i>.</p>

Field	Description
	For more information about empty string handling by protectors, refer to the section <i>Appendix C: Empty String Handling by Protectors</i> in the <i>Protection Methods Reference Guide 9.1.0.0</i> .
policy/pepserverConfig/logLevel	<p>Specifies the logging level for the PEP server.</p> <p>You can specify one of the following values:</p> <ul style="list-style-type: none"> • <i>OFF</i> - No logging • <i>SEVERE</i> • <i>WARNING</i> • <i>INFO</i> • <i>CONFIG</i> • <i>ALL</i> <p>By default, the value of this parameter is set to <i>ALL</i>.</p>
policy/pepserverConfig/caseSensitive	<p>Specifies how policy users are checked against the policy. This parameter enables you to configure the case-sensitivity of the policy users.</p> <p>If this parameter is set to <i>no</i>, then the PEP server considers the policy user names that are case insensitive.</p> <p>If this parameter is set to <i>yes</i> or if it is commented in the file, then the PEP server considers the policy user names that are case-sensitive. The default value is <i>yes</i>.</p> <p>By default, the value of this parameter is set to <i>yes</i>.</p>
policy/filePath	<p>Specify the path in the persistent volume or the object store where you want to store the immutable policy package.</p> <p>For example, if you are using NFS as the persistent volume, then you can specify the file path as <i>/<Mount directory>/xyz/imp</i>. In this case, a policy with the name as <i>imp</i> will be stored in the <i>/<Mount directory>/xyz</i> directory in the required persistent volume.</p> <div style="background-color: #e0f2f1; padding: 10px; margin: 10px 0;"> <p>Note:</p> <p>The <i><Mount directory></i> is a directory inside the Mount point.</p> </div> <p>If you want to store the policy package in an Object Store, such as an Azure Storage Container, then you can specify the container name as the first name in the <i>filePath</i> field.</p> <p>For example, you can specify the value of the <i>filePath</i> field, as <i>test/LOB/policy</i>, where <i>test</i> is the container name, <i>LOB</i> is the directory inside the container, and <i>policy</i> is the name of the immutable policy package. In this example, the container name is specified as the first name of the file path.</p> <div style="background-color: #ffe0e0; padding: 10px; margin: 10px 0;"> <p>Important:</p> </div>

Field	Description
	<p>Use only uppercase and lowercase letters, numbers, dot, and underscore in the file path. Do not use special characters in the file path.</p> <p>Note: This value is case-sensitive.</p>
policy/timeout	Specify the time for which the Get ESA Policy container tries to communicate with the ESA for fetching the policies, after which the connection times out. By default, this value is set to 15. The unit of the timeout parameter is minutes.

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

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

For example:

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

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

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

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

2.7.3 Verifying the IAP Policy Snapshot on the NFS

This section describes how you can verify the IAP policy snapshot on the NFS.

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

Important: This procedure is optional, and is required only if you want to use the persistent volume storing the policy package, instead of using the Azure Storage Container.

► To verify the policy snapshot on the NFS:

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

- On the Linux instance, navigate to the location where you have mounted the NFS.
For more information about the location where you have mounted the NFS, refer to [step 6](#) in the section *Setting up the Environment for Deploying the Get ESA Policy Container*.
- List all the files within the directory to access the policy snapshot.


2.7.4 Verifying the IAP Policy Package on the Azure Storage Container

The Get ESA Policy container retrieves the policy from the ESA and uploads a snapshot of this immutable policy to the Azure storage container. This section describes how to verify the policy package in the Azure environment.

Important: This procedure is optional, and is required only if you want to use the Azure Storage Container storing the policy package, instead of using the persistent volume.

► To verify the policy package on the Azure Storage Container:

Perform the following task to verify whether the immutable policy package has been uploaded to the Azure storage container.

1. Login to the Azure environment.
For more information about logging in to the Azure environment, refer to the section [Logging in to the Azure Environment](#).
2. Click the **Portal** menu icon ().
The **Portal** menu appears.
3. Click **Storage accounts**.
The **Storage accounts** screen appears.
4. Navigate to the path that you have specified as the value of the `filePath` parameter in the `values.yaml` file in [step 13](#) of the section [Deploying the Get ESA Policy Container on the Kubernetes Cluster](#).
The immutable policy package appears.
5. Click the immutable policy package to view additional details about the file.

2.7.5 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. If you want to use an Azure Storage Container for storing the policy package, instead of the persistent volume, then modify the `credentials.json` file to update the credentials for the service principal.

The following snippet contains the format of the `credentials.json` file.

```
{
  "storage_account_name" : "AZURE_STORAGE_ACCOUNT_NAME",
  "tenant_id" : "AZURE_TENANT_ID",
  "client_id" : "AZURE_CLIENT_ID",
  "client_secret": "AZURE_CLIENT_SECRET"
}
```

For more information about the values for the `tenant_id`, `client_id`, and `client_secret` keys, refer to the section [Registering an Application](#).

For more information about the structure of the `credentials.json` file, refer to the section [Credentials.json](#).

- If you want to use an Azure Storage Container for storing the policy package, instead of the persistent volume, then run the following command to create the *read-obj-access* secret, which is used by the Sample Application container to download the immutable policy package from the Azure storage container to the shared memory.

```
kubectl create secret generic read-obj-access --from-file=spring-apjava/credentials.json --namespace <NAMESPACE>
```

```
kubectl create secret generic read-obj-access --from-file=spring-apjava/credentials.json --namespace iap
```

For more information about the *credentials.json* file, refer to the section [Credentials.json](#).

Note: Ensure that you specify the name of the secret as the value of the *storageAccessSecrets* field in the *values.yaml* file.

- 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 5](#) of the section [Initializing the Linux Instance](#).

The *spring-apjava > 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
...
...

## 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>
  ## 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

## 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 want to expose the service externally.
## targetPort - the port no. configured while creating Tunnel.
springappService:
  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.

4. 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	<p>Specify the privilege and access control settings for the pod.</p> <p>The default values are set as follows:</p> <ul style="list-style-type: none"> <i>runAsUser</i> - 1000 <i>runAsGroup</i> - 1000 <i>fsGroup</i> - 1000 <p>Note: Set the value of the <i>fsGroup</i> parameter to a non-root value.</p> <p>For example, you can set the value of the <i>fsGroup</i> parameter to any value between 1 and 65534.</p> <p>Note: Ensure that at any given point of time, the value of at least one of the four parameters must be 1000.</p> <p>This ensures that the pod has the required permissions to access the <i>pty</i> directories in the containers.</p> <p>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.</p> <p>For more information about the parameters, refer to the section PodSecurityContext v1 Core in the Kubernetes API documentation.</p>
pbeSecrets	<p>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 <i>pbeSecrets</i> parameter to decrypt the policy.</p> <p>Important: Ensure that the passphrase has a minimum length of 12 characters, with atleast 1 uppercase letter, 2 digits, 2 special characters, and 2 non-letters (such as, digits or special characters). The passphrase must also have a minimum value of entropybits as 30 and a minimum strength of 0.66.</p> <p>Entropybits defines the randomness of the password and the strength defines the complexity of the password.</p> <p>For more information about the password strength, refer to the section Password Strength.</p>
deploymentInUse	Specify the value as <i>blue</i> . This indicates that the <i>Blue</i> deployment strategy is in use.
blueDeployment/enabled	Specify the value as <i>true</i> to enable the <i>blue</i> deployment strategy.
blueDeployment/securityConfigSource	<p>Specify one of the following options:</p> <ul style="list-style-type: none"> <i>VolumeMount</i> - Specify this value if you want to store the policy package on a volume that is mounted on the pod. Use this option if you are using the persistent volume for storing the policy package.

Field	Description
	<ul style="list-style-type: none"> <i>ObjectStore</i> - Specify this value if you want to store the policy package in the Azure Storage Container. <p>By default, the value is set to <i>VolumeMount</i>.</p>
blueDeployment/pvcName	<p>Specify the name of the Persistent Volume Claim where you want to store the policy snapshot.</p> <p>If you want to use the Azure Storage Container for storing the policy package, then leave the value of this field blank.</p>
blueDeployment/storageProvider	<p>Specify the value as <i>AZURE</i> if you want to upload the policy package to the Azure storage container.</p> <p>If you want to use the persistent volume for storing the policy package, then leave the value of this field blank.</p>
blueDeployment/storageAccessSecrets	<p>Specify the value of the secret, which you have created in Step 2b, to store the credentials with read access to the Azure storage container. This is used by the Sample Application container to read the policy package from the Azure storage container.</p> <p>If you want to use the persistent volume for storing the policy package, then leave the value of this field blank.</p> <p>Important: This field is required if you want to store the immutable policy package in the Azure Storage Container.</p>
blueDeployment/policy/filePath	<p>Specify the path in the persistent volume or the Azure Storage Container where you have stored the policy.</p> <p>Ensure that you specify the same value that you have specified as the value of the <i>filePath</i> parameter in the <i>values.yaml</i> file for the Get ESA Policy container in step 13 of the section Deploying the Get ESA Policy Container on the Kubernetes Cluster.</p> <p>Important: Use only uppercase and lowercase letters, numbers, dot, and underscore in the file path. Do not use special characters in the file path.</p> <p>Note: This value is case-sensitive.</p>
greenDeployment/enabled	<p>Specify the value as <i>false</i>. While deploying Release 1, the <i>Green</i> deployment strategy is always disabled.</p>
springappService/name	<p>Specify a name for the tunnel to distinguish between ports.</p>
springappService/port	<p>Specify the port number on which you want to expose the Kubernetes service externally.</p>
springappService/targetPort	<p>Specify the port on which the Sample Application container is running inside the Docker container.</p> <p>Important: Do not change this value.</p>
routes/prodService/hostname	<p>Specify the hostname of the production service.</p>
routes/stagingService/hostname	<p>Specify the hostname of the staging service.</p>

5. 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-java --namespace iap spring-apjava
```

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

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

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

For example:

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

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

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

For example:

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

2.7.6 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 5 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
...
...

## 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>
  ## 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

## 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 want to expose the service externally.
## targetPort - the port no. configured while creating Tunnel.
springappService:
  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 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	<p>Specify the privilege and access control settings for the pod.</p> <p>The default values are set as follows:</p> <ul style="list-style-type: none"> • <i>runAsUser</i> - 1000 • <i>runAsGroup</i> - 1000 • <i>fsGroup</i> - 1000 <p>Note: Set the value of the <i>fsGroup</i> parameter to a non-root value.</p> <p>For example, you can set the value of the <i>fsGroup</i> parameter to any value between <i>1</i> and <i>65534</i>.</p> <p>Note: Ensure that at any given point of time, the value of at least one of the four parameters must be <i>1000</i>.</p> <p>This ensures that the pod has the required permissions to access the <i>pty</i> directories in the containers.</p> <p>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.</p> <p>For more information about the parameters, refer to the section PodSecurityContext v1 Core in the Kubernetes API documentation.</p>
pbeSecrets	<p>Specify the Kubernetes secret that contains the passphrase and the salt that were used to generate key, which encrypted the policy.</p>

Field	Description
	<p>The Sample Application container uses the value specified in the <i>pbeSecrets</i> parameter to decrypt the policy.</p> <p>Important: Ensure that the passphrase has a minimum length of 12 characters, with atleast 1 uppercase letter, 2 digits, 2 special characters, and 2 non-letters (such as, digits or special characters). The passphrase must also have a minimum value of entropybits as <i>.30</i> and a minimum strength of <i>0.66</i>.</p> <p>Entropybits defines the randomness of the password and the strength defines the complexity of the password.</p> <p>For more information about the password strength, refer to the section Password Strength.</p>
deploymentInUse	Specify the value as <i>blue</i> . This indicates that the <i>Blue</i> deployment strategy is in use.
greenDeployment/enabled	Specify the value as <i>true</i> . While upgrading the release to Release 2, the <i>Green</i> deployment strategy is enabled.
greenDeployment/securityConfigSource	<p>Specify one of the following options:</p> <ul style="list-style-type: none"> <i>VolumeMount</i> - Specify this value if you want to store the policy package on a volume that is mounted on the pod. Use this option if you are using the persistent volume for storing the policy package. <i>ObjectStore</i> - Specify this value if you want to store the policy package in the Azure Storage Container. <p>By default, the value is set to <i>VolumeMount</i>.</p>
greenDeployment/pvcName	<p>Specify the name of the Persistent Volume Claim where you want to store the policy snapshot.</p> <p>If you want to use the Azure Storage Container for storing the policy package, then leave the value of this field blank.</p>
greenDeployment/storageProvider	<p>Specify the value as <i>AZURE</i> if you want to upload the policy package to the Azure storage container.</p> <p>If you want to use the persistent volume for storing the policy package, then leave the value of this field blank.</p>
greenDeployment/storageAccessSecrets	<p>Specify the value of the secret which you have used to store the credentials with read access to the Azure storage container. This is used by the Sample Application container to read the policy package from the Azure storage container.</p> <p>If you want to use the persistent volume for storing the policy package, then leave the value of this field blank.</p> <p>Important: This field is required if you want to store the immutable policy package in the Azure Storage Container.</p>

Field	Description
greenDeployment/policy/filePath	Specify the path in the persistent volume or the Azure Storage Container where you have stored the policy. 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.
springappService/name	Specify a name for the tunnel to distinguish between ports.
springappService/port	Specify the port number on which you want to expose the Kubernetes service externally.
springappService/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.

- 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-java --namespace iap spring-apjava
```

Using this configuration ensures that the Sample Application container is deployed with the updated policy snapshot on the staging environment.

- 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.

- 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.
- 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-java --namespace iap spring-apjava
```

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-java --namespace iap spring-apjava
```

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

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

Chapter 3

Accessing Logs Using Splunk

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.

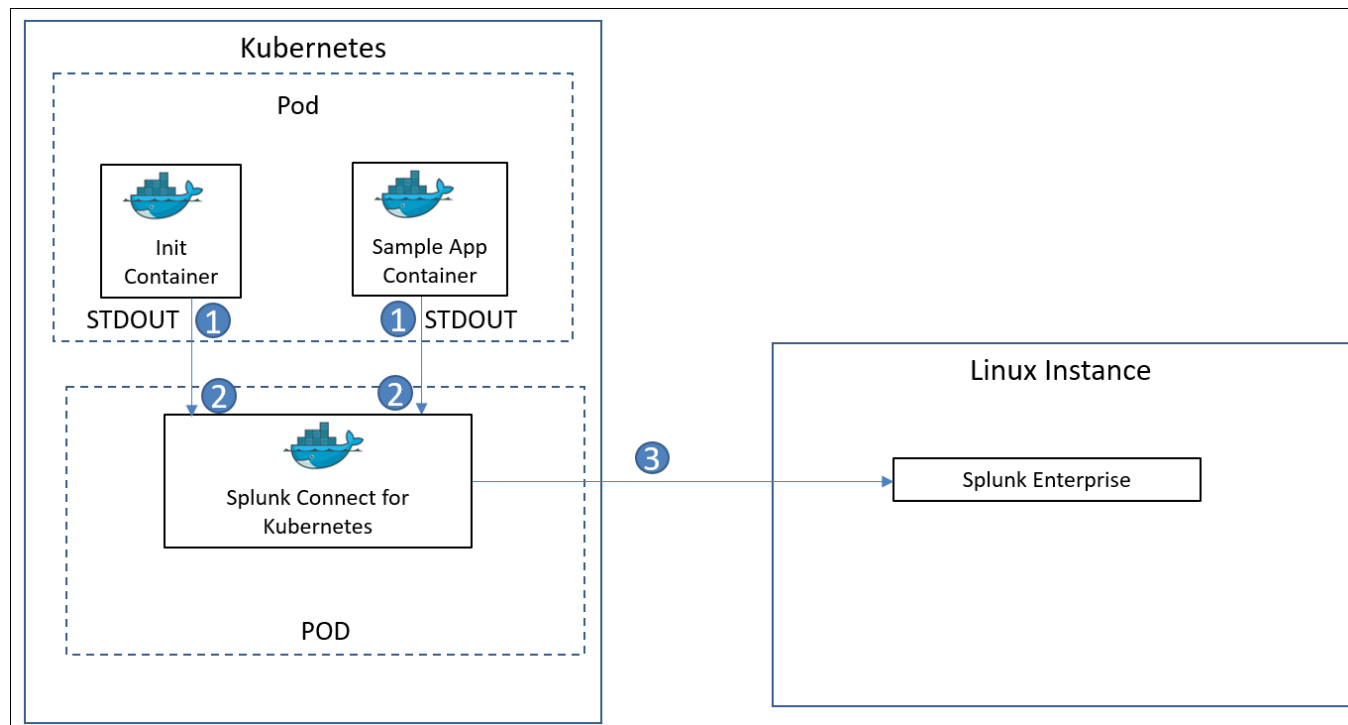


Figure 3-1: Sample Logging Workflow

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

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

For more information about Splunk Connect for Kubernetes, refer to the [Splunk Connect for Kubernetes](#) page on Github.

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

3.2 Setting Up Splunk

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

► To set up Splunk:

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

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

For more information about installing Splunk Enterprise on a Linux instance, refer to the section [Install Splunk Enterprise](#) in the Splunk documentation.

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

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

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

For more information about starting Splunk Enterprise on Linux, refer to the section [Start Splunk Enterprise on Linux](#) in the Splunk documentation.

4. Type the username for the administrator account for logging into Splunk Enterprise, and then press **Enter**.
You are prompted to create a password for the created user.

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

5. Type a password for the user that you have created in [step 4](#), and then press **Enter**.
The following message appears on the console.

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

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

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

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

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

3.3 Configuring Splunk

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

► To configure Splunk Enterprise:

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

The Splunk Enterprise screen appears.

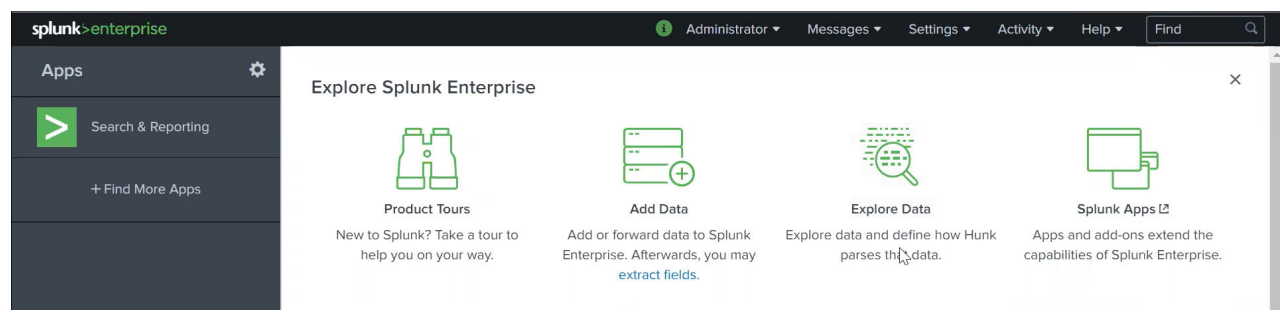


Figure 3-2: Splunk Enterprise Web UI

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.

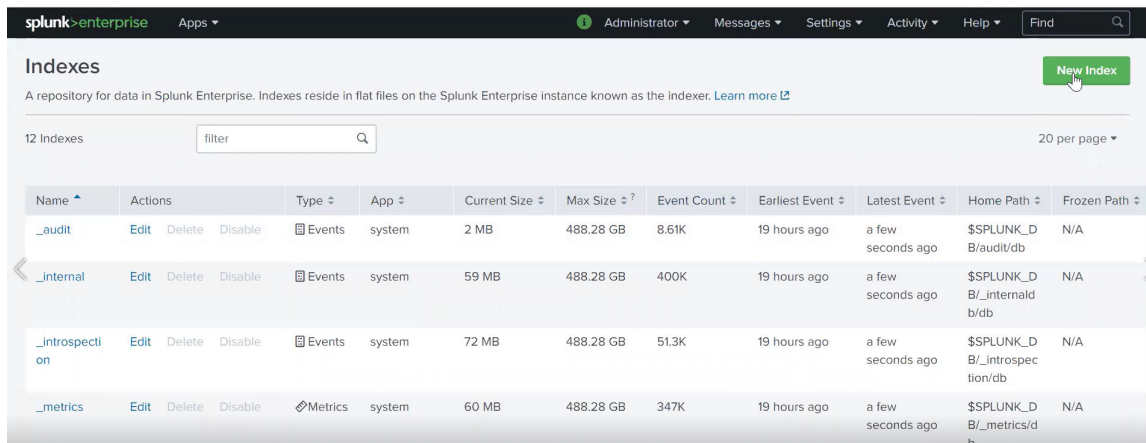


Figure 3-3: Indexes Screen

- b. Click **New Index**.

The **New Index** dialog box appears.

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.

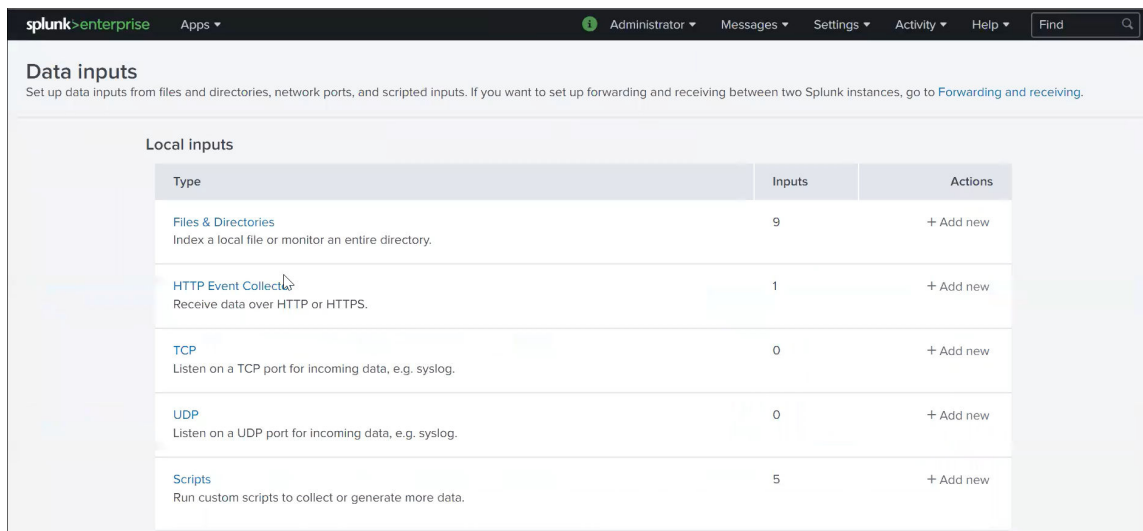


Figure 3-5: Data inputs Screen

- b. Click **HTTP Event Collector**.

The **HTTP Event Collector** screen appears.

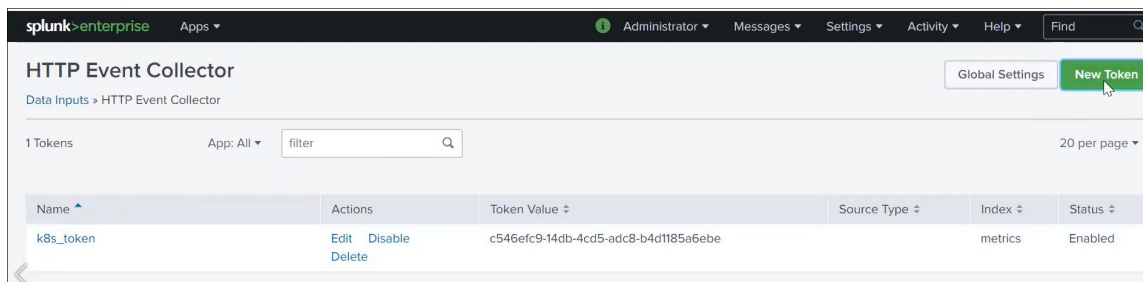


Figure 3-6: HTTP Event Collector Screen

- c. Click **New Token**.

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

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.

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.

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.

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.

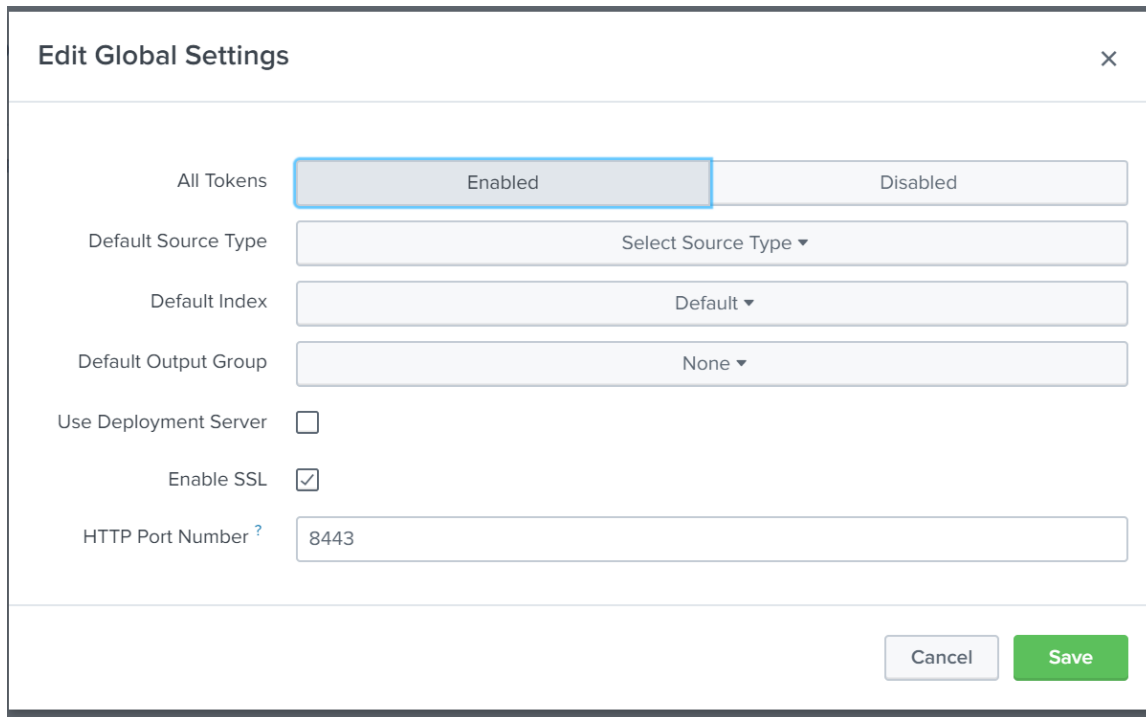
The image shows a dialog box titled "Edit Global Settings" with a close button (X) in the top right corner. The dialog contains several settings: "All Tokens" is a toggle switch currently set to "Enabled"; "Default Source Type" is a dropdown menu showing "Select Source Type"; "Default Index" is a dropdown menu showing "Default"; "Default Output Group" is a dropdown menu showing "None"; "Use Deployment Server" is an unchecked checkbox; "Enable SSL" is a checked checkbox; and "HTTP Port Number" is a text input field containing "8443". At the bottom right, there are "Cancel" and "Save" buttons.

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 <code>https</code> .
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 <code>true</code> .
global/splunk/hec/token	Specify the value of the token that you have created in step 4i of the section <i>Configuring Splunk</i> .
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 <i>Configuring Splunk</i> .
global/splunk/hec/indexName	Specify the name of the index that you have created in step 3 of the section <i>Configuring Splunk</i> .
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 <code>false</code> , 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 <i>cri</i> , as this is the default log format for the OpenShift Container logs. Important: If you specify another value, such as, <i>json</i> , instead of <i>cri</i> , then the logs will not be parsed.
container/logFormat	Specify the log format as <i>%Y-%m-%dT%H:%M:%S.%N%.z</i> . 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.

- 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.co  
helm 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-XXXXXX 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 'https://<Hostname of the Routes>/protect' --header 'Host: prod.example.com' --header 'Content-Type: application/json' --data-raw '{ "dataElement": "Alphanum", "policyUser": "user1", "input": [ "SampleAppJava", "OnKubernetes" ] }'
```

In the above 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 Java, 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 '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 'https://<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": [ "SampleAppJava", "OnKubernetes" ] }' --cacert ./
iap-ca.crt --cert ./iap-client.crt --key ./iap-client.key
```

In the above 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 Java, 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 Java 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.

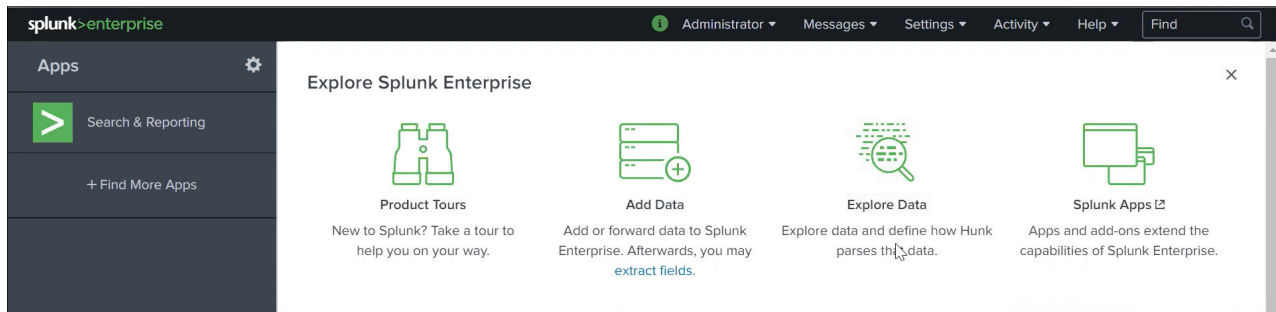


Figure 4-1: Splunk Enterprise Web UI

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.

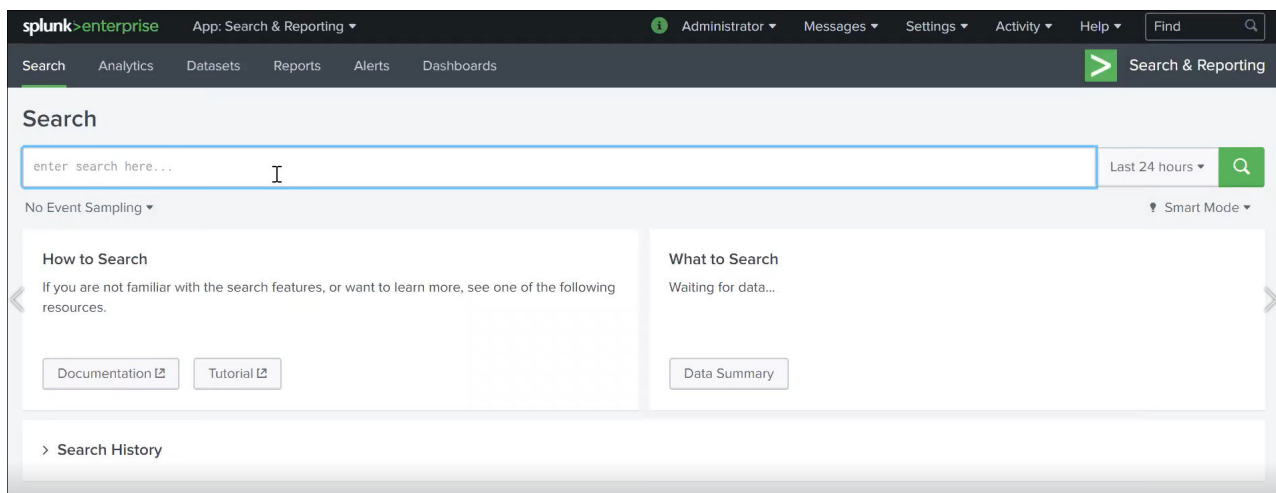


Figure 4-2: Search Screen

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:spring-app-java"
```

5. Press **Enter**.

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

The screenshot shows the Splunk Search & Reporting interface. The search bar contains the query: `index="events" sourcetype="kube:container:spring-apjava"`. The results show 539 events. The table displays the following data:

Time	Event
11/11/20 1:59:35.458 PM	2020-11-11 13:59:35.457 INFO 1 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Completed initial ization in 6 ms host = [REDACTED] source = [REDACTED] sourcetype = kube:container:spring-apjava
11/11/20 1:59:35.451 PM	2020-11-11 13:59:35.451 INFO 1 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Initializing Serv let 'dispatcherServlet'
11/11/20 1:59:35.451 PM	2020-11-11 13:59:35.450 INFO 1 --- [nio-8080-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spr ing DispatcherServlet 'dispatcherServlet'

- If you want to view only the logs that are related to the Application Protector Java, 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:spring-app-java" "product_id: Application Protector"
```

- Press **Enter**.

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

The screenshot shows the Splunk Search & Reporting interface with the search query: `index="events" sourcetype="kube:container:spring-apjava" "product_id: Application Protector"`. The results show 20 events. The table displays the following data:

Time	Event
11/11/20 1:58:53.789 PM	2020-11-11 13:58:53, severity: INFO, message: Data protection was successful., product_id: Application Protector, data_element: TE_A_N_S13_L0R0_N, operation: Protect, user: pyituser, vendor_id: 100, session_id: -1313650089, statementnumber: 1, request_id: 1637752040, version: 7.1.0.151.iap host = [REDACTED] source = [REDACTED] sourcetype = kube:container:spring-apjava
11/11/20 1:57:15.575 PM	2020-11-11 13:57:15, severity: WARNING, message: Permission denied, product_id: Application Protector, data_element: TE_A_N_S13_L0R0_N, operation: Unprotect, user: user2, vendor_id: 100, session_id: 220023257, statementnumber: 1, request_id: 1805652985, version: 7.1.0.151.iap host = [REDACTED] source = [REDACTED] sourcetype = kube:container:spring-apjava

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.

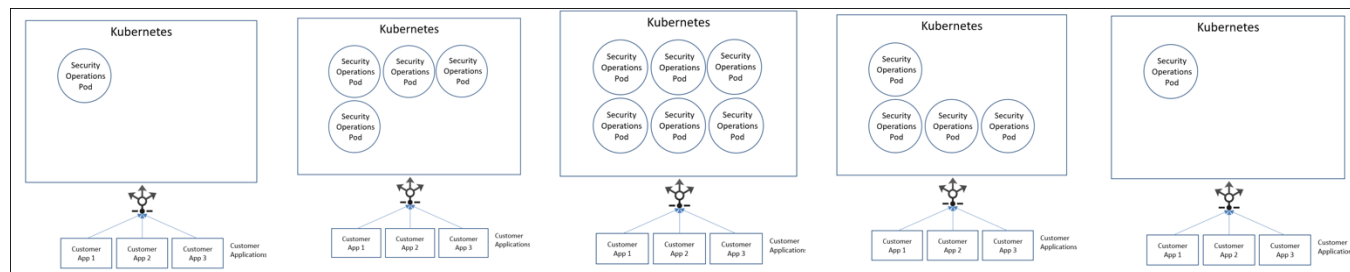


Figure 4-3: Autoscaling Kubernetes Cluster

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 A: Get ESA Policy Helm Chart Details

[5.1 Chart.yaml](#)

[5.2 pty-scc.yaml](#)

[5.3 pty-rbac.yaml](#)

[5.4 nfs-pv.yaml](#)

[5.5 nfs-pvc.yaml](#)

[5.6 Credentials.json](#)

[5.7 Values.yaml](#)

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

5.1 Chart.yaml

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

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

5.2 pty-scc.yaml

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

Important: Do not modify the values.

```
kind: SecurityContextConstraints
apiVersion: security.openshift.io/v1
metadata:
  annotations:
    kubernetes.io/description: This is a restrictive SCC.
  name: pty-scc
allowHostDirVolumePlugin: false
allowHostIPC: false
allowHostNetwork: false
allowHostPorts: false
allowHostPID: false
allowPrivilegeEscalation: false
allowPrivilegedContainer: false
defaultAddCapabilities: []
allowedCapabilities: []
readOnlyRootFilesystem: true
```

```

groups:
- system:serviceaccounts:<NAMESPACE>
requiredDropCapabilities:
- ALL
runAsUser:
  type: MustRunAsNonRoot
seLinuxContext:
  type: RunAsAny
supplementalGroups:
  type: MustRunAs
  ranges:
    - min: 1
      max: 65535
fsGroup:
  type: MustRunAs
  ranges:
    - min: 1
      max: 65535
volumes:
- persistentVolumeClaim
- secret
- emptyDir

```

5.3 pty-rbac.yaml

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

Important: Do not modify the values.

```

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: iap-install-upgrade
rules:
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["create", "get", "list", "patch", "watch", "delete"]
- apiGroups: ["autoscaling"] # "" indicates the core API group
  resources: ["horizontalpodautoscalers"]
  verbs: ["create", "get", "list", "patch", "watch", "delete"]
- apiGroups: ["route.openshift.io"] # "" indicates the core API group
  resources: ["routes", "routes/custom-host"]
  verbs: ["create", "get", "list", "patch", "watch", "delete"]
- apiGroups: [""] # "" indicates the core API group
  resources: ["services"]
  verbs: ["create", "get", "list", "patch", "watch", "delete"]
- apiGroups: [""] # "" indicates the core API group
  resources: ["persistentvolumeclaims"]
  verbs: ["create", "get", "list", "watch"]
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods", "pods/log"]
  verbs: ["get", "list", "watch"]
- apiGroups: [""] # "" indicates the core API group
  resources: ["events"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["metrics.k8s.io"] # Top pods permission
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: iap-install-upgrade
subjects:
- kind: ServiceAccount
  name: iap-deployer
roleRef:
  kind: Role
  name: iap-install-upgrade

```



```

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

```

5.4 nfs-pv.yaml

If you want to use a persistent volume for storing the policy package, instead of the Azure Storage Container, then 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: MOUNT_PATH
      # The host name or IP address of the NFS server.
      server: SERVER_NAME_OR_IP
      readOnly: false

```

5.5 nfs-pvc.yaml

If you want to use a persistent volume for storing the policy package, instead of the Azure Storage Container, then modify the *nfs-pvc.yaml* file to update the persistent volume claim details, by specifying the name of the persistent volume claim and the storage amount

```

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

```

5.6 Credentials.json

The *credentials.json* file contains information about the credentials for the service principal that are required to access the Azure storage account. The credentials are required to enable the Get ESA Policy container to upload the immutable policy package to the Azure storage container. The following snippet below is shown for Azure. Users are required to edit the file and replace the values for the service principal credentials.

```

{
  "storage_account_name" : "AZURE_STORAGE_ACCOUNT_NAME",
  "tenant_id" : "AZURE_TENANT_ID",
  "client_id" : "AZURE_CLIENT_ID",
  "client_secret": "AZURE_CLIENT_SECRET"
}

```

5.7 Values.yaml

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

5.7.1 Image Pull Secrets

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

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

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

For example:

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

5.7.2 Name Override

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

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

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

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

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

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

5.7.3 Proxy Config

This section specifies the IP address and port number of the server that will be used as a proxy server between the ESA and the pod. This can be used in case the pod cannot communicate with the ESA directly due to network segmentation.

```
## add proxy configuration as env variables
## leave the field empty if not applicable.
proxyConfig:
# - name: "HTTP_PROXY"
#   value: "http://<user>:<password>@<ip_addr>:<port>"
```

5.7.4 Container Image Repository

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

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

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

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

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

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

5.7.5 Resources

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

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

You can specify the following parameters:

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

For more about the resource parameters, refer to the section [Managing Resources for Containers](#) in the Kubernetes documentation.

5.7.6 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.

5.7.7 Container Security Context

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

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

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

Note: It is recommended to not modify 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.

5.7.8 ESACred Secrets

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

```
esaCredSecrets: esa-creds
```

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

Important: You require a *cluster-admin* role to create the ESACred secrets.

For example:

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

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

5.7.9 PBE Secrets

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

```
pbeSecrets: PBE_SECRET_NAME
```

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

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: You require a *cluster-admin* role to create the PBE secrets.

For example:

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

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

Important: Ensure that the passphrase has a minimum length of 12 characters, with at least 1 uppercase letter, 2 digits, 2 special characters, and 2 non-letters (such as, digits or special characters). The passphrase must also have a minimum value of entropybits as *30* and a minimum strength of *0.66*.

Entropybits defines the randomness of the password and the strength defines the complexity of the password.

For more information about the password strength, refer to the section [Password Strength](#).

5.7.10 Security Config Destination

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

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

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

5.7.11 Persistent Volume Claim

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

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

For more information about persistent volumes and persistent volume claims, refer to the section [Persistent Volumes](#) in the Kubernetes documentation.

5.7.12 Storage Provider

This section describes the specific Cloud provider that is used to provide the Object Storage for uploading the policy package.

```
## If securityConfigDestination is ObjectStore
## specify the respective cloud provider ['AWS', 'AZURE', 'GCP']
storageProvider:
```

5.7.13 Storage Access Secrets

This section describes the credentials required to access the Azure storage container for uploading the policy package. The user is required to create a Kubernetes secret for writing to the cloud storage, such as an Azure storage container.

```
## If securityConfigDestination is ObjectStore
## specify k8s secret for storing credentials with access to cloud storage.
storageAccessSecrets:
```

5.7.14 Pod Service Account

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

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

Do not edit the field if not applicable.

5.7.15 Policy

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

```
policy:
  esaIP: ESA_IP

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

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

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

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

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

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

Chapter 6

Appendix B: Sample Application Helm Chart Details

[6.1 Chart.yaml](#)

[6.2 Credentials.json](#)

[6.3 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.

6.1 Chart.yaml

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

```
apiVersion: v1
description: A Spring boot application chart for Kubernetes
name: spring-apjava
version: 1.0.0
```

6.2 Credentials.json

The *credentials.json* file contains information about the credentials for service principal that are required to access the Azure storage account. The credentials are required to enable the Sample Application container to download the immutable policy package that has been uploaded to the Azure storage container. The following snippet displays the content for Azure. Users are required to edit the file and replace the values for the service principal credentials.

```
{
  "storage_account_name" : "AZURE_STORAGE_ACCOUNT_NAME",
  "tenant_id" : "AZURE_TENANT_ID",
  "client_id" : "AZURE_CLIENT_ID",
  "client_secret": "AZURE_CLIENT_SECRET"
}
```

6.3 Values.yaml

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

6.3.1 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>
```

6.3.2 Name Override

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

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

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

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

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

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

6.3.3 Container Image Repository

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

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

The following list provides an overview of the `initImage` parameter, which refers to the details for the Init container image:

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

Note: Ensure that you only add the Container registry path as the repository value. Do not include the tag name in this value. You need to add 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 Init container. By default, the value of this field is set to *false*.
- *springappImage* – Refers to the details for the Sample Application container image.

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

6.3.4 Resources

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

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

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

You can specify the following parameters:

- *limits* - Specify the resource limits for the containers in a pod. These limits ensure that the running container does not use 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.

6.3.5 Pod Service Account

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

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

6.3.6 Liveness 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
```

6.3.7 Pod Security Context

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

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

For more information about the Pod Security Context, refer to the section [Configure a Security Context for a Pod or Container](#) in the Kubernetes documentation.

For more information about the parameters, refer to the section [PodSecurityContext v1 Core](#) in the Kubernetes API documentation.

6.3.8 Container Security Context

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

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

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

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

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

For more information about the Container Security Context, refer to the section [Configure a Security Context for a Pod or Container](#) in the Kubernetes documentation.

For more information about the parameter, refer to the section [SecurityContext v1 core](#) in the Kubernetes API documentation.

6.3.9 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*

6.3.10 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.

6.3.11 Autoscaling

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

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

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

The following list provides a description for the parameters:

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

6.3.12 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 springapp configurations where,
## name - distinguishes between different ports.
## port - the port on which you want to expose the service externally.
## targetPort - the port no. configured while creating Tunnel.
springappService:
  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.

6.3.13 Routes Configuration

This section explains the Routes configuration for deployment.

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

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

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

Chapter 7

Appendix C: Using the Dockerfile to Build Custom Images

7.1 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 images for the Get ESA Policy, Init, and Sample Application containers. You can use the custom image instead of the default images that are provided by Protegrity as part of the installation package. This enables you to use a base image of your choice, instead of using the default RHEL Universal Base Image, which is used as the default base image for the Protegrity images.

7.1 Creating Custom Images Using Dockerfile

This section describes the steps for creating custom images for the Get ESA Policy, Init, and Sample Application containers, using the Dockerfile provided with the installation package.

Before you begin

Ensure that Maven 3.2 or later is installed on the machine on which you are creating the custom Docker image.

► To create custom images:

1. Download the installation package.

For more information about downloading the installation package, refer to the section [Downloading the Installation Package](#).

Important: The dependency packages required for building the Docker images are specified in the *HOW-TO-BUILD-DOCKER-IMAGES*, which is a part of the installation package. You must ensure that these dependency packages can be downloaded either from the Internet or from your internal repository.

For more information about the *HOW-TO-BUILD-DOCKER-IMAGES* file, refer to the section [Verifying the Prerequisites](#).

2. Perform the following steps to build a Docker image for the Get ESA Policy container.

- a. Run the following command to extract the files from the *IAP-GET-ESA-POLICY_OPENSIFT_SRC_<version_number>.tgz* file.

```
tar -xvf IAP-GET-ESA-POLICY_OPENSIFT_SRC_<version_number>.tgz
```

The following files are extracted:

- *GET-ESA-POLICY_DOCKERFILE_<version_number>*
- *PepServerSetup_Linux_x64_<version_number>.iap.tgz*
- *HealthCheckSetup_Linux_x64_<version_number>.tgz*
- *PtyShmCatSetup_Linux_x64_<version_number>.tgz*

- b. Edit the *GET-ESA-POLICY_DOCKERFILE_<version_number>* file and in the following code snippet, replace the placeholder *RHEL-MINIMAL-UBI* with the name of the repository from where you want to download the custom RHEL UBI.

```
FROM RHEL-MINIMAL-UBI
```

- c. Run the following command in the directory where you have extracted the contents of the *IAP-GET-ESA-POLICY_OPENSIFT_SRC_<version_number>.tgz* file.

```
docker build -t <image-name>:<image-tag> -f GET-ESA-POLICY_DOCKERFILE_<version_number> .
```

For more information the Docker build command, refer to the [Docker](#) documentation.

For more information about tagging an image, refer to the [OpenShift documentation](#).

- d. Run the following command to list the Get ESA Policy container image.

```
docker images
```

- e. Push the Get ESA Policy container image to your preferred Container Repository.

For more information about pushing an image to the repository, refer to the [OpenShift documentation](#).

3. Perform the following steps to build a Docker image for the Init container.

- a. Run the following command to extract the files from the *IAP-INIT-CONTAINER_OPENSIFT_SRC_<version_number>.tgz* file.

```
tar -xvf IAP-INIT-CONTAINER_OPENSIFT_SRC_<version_number>.tgz
```

The following files are extracted:

- *INIT-CONTAINER_DOCKERFILE_<version_number>*
- *PtyShmPutSetup_Linux_x64_<version_number>.tgz*

- b. Edit the *INIT-CONTAINER_DOCKERFILE_<version_number>* file and in the following code snippet, replace the placeholder *RHEL-MINIMAL-UBI* with the name of the repository from where you want to download the custom RHEL UBI.

```
FROM RHEL-MINIMAL-UBI
```

- c. Run the following command in the directory where you have extracted the contents of the *IAP-INIT-CONTAINER_OPENSIFT_SRC_<version_number>.tgz* file.

```
docker build -t <image-name>:<image-tag> -f INIT-CONTAINER_DOCKERFILE_<version_number> .
```

For more information the Docker build command, refer to the [Docker](#) documentation.

For more information about tagging an image, refer to the [OpenShift documentation](#).

- d. Run the following command to list the Init container image.

```
docker images
```

- e. Push the Init container image to your preferred Container Repository.

For more information about pushing an image to the repository, refer to the [OpenShift documentation](#).

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

- a. Run the following command to extract the files from the *IAP-SAMPLE-APP_OPENSIFT_SRC_<version_number>.tgz* file to a directory.

```
tar -C <Directory_name> -xvf IAP-SAMPLE-APP_OPENSIFT_SRC_<version_number>.tgz
```

The following files and directories are extracted:

- *IAP_RHUBI_SAMPLE-APP_DOCKERFILE_<version_number>*
- *libs/*

- *src/*
 - *HealthCheckSetup_Linux_x64_<version_number>.tgz*
 - *docker-entrypoint.sh*
 - *passwd.template*
 - *pom.xml*
- b. Switch to the directory where you have extracted the *IAP-SAMPLE-APP_OPENSHIFT_SRC_<version_number>.tgz* package.
- c. Execute the following command in the directory.
- ```
mvn clean install
```
- The *apjava-springboot-<version\_number>.jar* file appears in the *./target* directory.
- d. Copy the *APJavaIMSetup\_<OS>\_x64\_<App\_Protector\_version>.tgz* file from the installation package to the directory where you have extracted the *IAP-SAMPLE-APP\_OPENSHIFT\_SRC\_<version\_number>.tgz* package.
- e. 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
```
- f. Run the following command in the directory where you have extracted the contents of the *IAP-SAMPLE-APP_OPENSHIFT_SRC_<version_number>.tgz* 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](#).
- g. Run the following command to list the Sample Application container image.
- ```
docker images
```
- h. 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 8

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/passwd` file does not contain an entry for the user ID. As a result, the Application Protector Java 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
export NSS_WRAPPER_PASSWD=/tmp/passwd
export NSS_WRAPPER_GROUP=/etc/group

exec java \
-cp <APPLICATION_LIBS>:/opt/protectrity/applicationprotector/java/lib/* \
<APPLICATION_NAME>
```

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
ptyituser:x:${USER_ID}:${GROUP_ID}:ptyituser:/home/ptyituser:/bin/bash

```

You need to replace the *ptyituser* 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" ]
...

```