



Protegrity Data Security Gateway (DSG) 3.1.0.5

Immutable Policy User Guide for Azure

Created on: Nov 19, 2024

Copyright

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

Protegity products are protected by and subject to patent protections;

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

The Protegity logo is the trademark of Protegity 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 Protegity Security Operations Cluster using Kubernetes.....	6
1.1 Business Problem.....	6
1.2 Protegity Solution.....	6
1.3 Architecture and Components.....	7
Chapter 2 Protegity Reference Deployment Model.....	9
2.1 Verifying the Prerequisites.....	9
2.1.1 Verifying the Prerequisites for Reference Solution Deployment.....	9
2.1.2 Verifying the Prerequisites for Reference Application.....	10
2.1.3 Verifying Prerequisites for Azure.....	10
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 Uploading the Images to the Container Repository.....	14
2.6 Creating and Uploading the CoP to Azure Storage Container.....	15
2.6.1 Creating and Exporting the Ruleset.....	15
2.7 Creating and Uploading the Policy to Azure Storage Container.....	19
2.7.1 Creating and Exporting the Policy.....	20
2.8 Creating the Cloud Runtime Environment.....	22
2.8.1 Creating the Azure Runtime Environment.....	22
2.8.1.1 Logging in to the Azure Environment.....	22
2.8.1.2 Registering an Application.....	25
2.8.1.3 Creating a Kubernetes Cluster.....	28
2.9 Setting up Splunk to Access Logs.....	29
2.9.1 Understanding the Logging Architecture.....	30
2.9.2 Setting Up Splunk.....	30
2.9.3 Configuring Splunk.....	31
2.9.4 Deploying the Splunk Connect for Kubernetes Helm Chart on the Kubernetes Cluster.....	35
2.9.5 Enabling Transaction Metric Logs for Splunk.....	37
2.10 Deploying the Containers.....	38
2.10.1 Applying an Azure Policy.....	38
2.10.2 Applying Role Based Access Control (RBAC).....	45
2.10.3 Setting up the Environment for Deploying the DSG Container.....	47
2.10.4 Deploying a Release on the Kubernetes Cluster with Azure Policy and RBAC.....	53
2.10.5 Deploying a Release on the Kubernetes Cluster without Azure Policy and RBAC.....	59
2.11 Upgrading the Helm Charts.....	70
Chapter 3 Protecting Data using the DSG Container Deployment.....	78
3.1 Running Security Operations.....	78
3.1.1 Running Security Operations with Static CoP.....	78
3.1.2 Running Security Operations with Dynamic CoP in HTTPS Mode.....	81
3.2 Viewing Logs in Splunk.....	82
3.3 Autoscaling the Protegity Reference Deployment.....	84
Chapter 4 Troubleshooting Issues.....	85
Chapter 5 Appendix A: DSG Container Helm Chart Details.....	86
5.1 Chart.yaml.....	86
5.2 Values.yaml.....	86
5.2.1 Image Pull Secrets.....	86
5.2.2 Name Override.....	87



5.2.3 Container Image Repository.....	87
5.2.4 Resources.....	87
5.2.5 Persistent Volume Claim.....	88
5.2.6 Pod Security Context.....	88
5.2.7 Container Security Context.....	88
5.2.8 Pod Service Account.....	89
5.2.9 Liveliness and Readiness Probe Settings.....	89
5.2.10 Password-Based Encryption (PBE) Secrets.....	89
5.2.11 Storage Access Secrets.....	89
5.2.12 Private Secret Key.....	89
5.2.13 CoP Secret.....	89
5.2.14 LDAP Secrets.....	90
5.2.15 Deployment.....	90
5.2.16 Autoscaling.....	90
5.2.17 Service Configuration.....	91
5.2.18 Ingress Configuration.....	92
5.3 credentials.json.....	94

Chapter 6 Appendix B: Using the Sample Application..... **95**

6.1 Overview.....	95
6.1.1 Policy Sample.....	95
6.1.2 CoP / RuleSet Sample.....	96
6.1.3 Autoscaling Script.....	96
6.1.4 PostMan Collection.....	97
6.2 Running the Sample Application.....	98
6.2.1 Generating Certificates and Keys for TLS Authentication.....	98
6.2.2 Importing Certificates on the ESA.....	99
6.2.3 Importing Certificates to the Postman Client.....	100
6.2.4 Enabling the Authentication Functionality.....	102
6.2.5 Importing the Policy Sample on the ESA.....	105
6.2.6 Importing the CoP Sample on the ESA.....	106
6.2.7 Creating Immutable Metadata Packages (IMP).....	107
6.2.8 Deploying Release 1.....	107
6.2.9 Running Sample Application (PostMan collection).....	107
6.2.10 Running Autoscaling Script.....	110

Chapter 1

Protegility Security Operations Cluster using Kubernetes

[1.1 Business Problem](#)

[1.2 Protegility Solution](#)

[1.3 Architecture and Components](#)

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

1.1 Business Problem

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

- Protegility customers are moving to the cloud. This includes data and workloads in support of transactional applications and analytical systems.
- It is impossible to keep up with the continual change in workloads by provisioning Protegility 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 Protegility data security components that can leverage the auto scaling capabilities of Kubernetes to scale.

1.2 Protegility Solution

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

- Cloud standard DSG form factor:
 - The delivery form factor for cloud deployments is a container. Protegility has developed a DSG Container form factor based on the DSG Appliance form factor that we have been delivering for several years.
 - The DSG Container is a standard image that is familiar and expected in cloud deployments.
 - The DSG Container form factor makes the container a lightweight deployment of the DSG.
- Immutable Deployment:
 - Cloud deployments or containers do not change dynamically – they are immutable. In other words, when there is a change in Policy, Configuration over Programming (CoP), or the actual DSG Container software, the change is carried out by terminating the existing DSG Container with Policy and CoP, and instantiating a new one.
 - Changes to Policy, CoP, or the DSG Container happen through special deployment strategies available through Kubernetes. For Protegility deployments, the recommended deployment strategy is a *Blue/Green* strategy.
- Auto Scalability:
 - Kubernetes can be setup to auto scale based on setting a configuration on CPU thresholds.



- Kubernetes will start with an initial set of pods. These will increase when the CPU threshold is passed, and will be shrunk when the CPU threshold is under the acceptable limits. This is automatic and hence gives the agility and scalability that is desired with cloud solutions. Similarly, the nodes on which the pods are run also auto scale when the node thresholds are reached.
- 3rd Party Integration:
 - The important implication is that the ESA is no longer required to be up and running when the Kubernetes runtime has all the required components and can auto scale, when needed.

1.3 Architecture and Components

This section describes the architecture, the individual components, and the workflow of the Protegility DSG Containerization solution.

The DSG Container deployment consists of the following two stages:

- Stage 1: Creating immutable policy and CoP and uploading them to an Azure Storage Container or a persistent volume.
- Stage 2: Deploying the DSG Container Application on a Kubernetes cluster, which uses the policy and CoP stored in stage 1.

The following figures illustrates the process for creating a CoP and uploading it to an Azure Storage Container or a persistent volume.

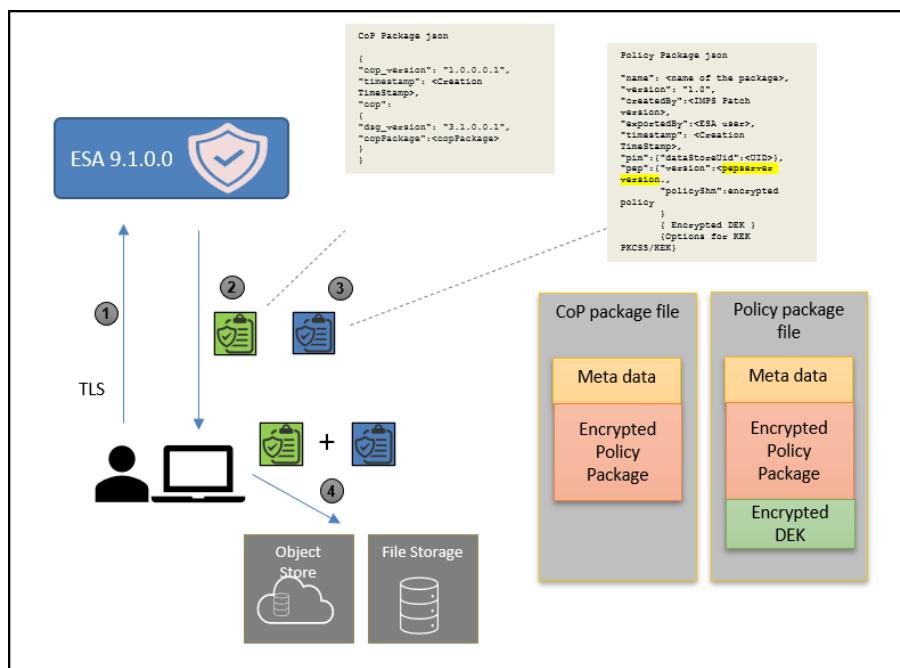


Figure 1-1: Creating CoP and uploading it to a persistent volume

Stage 1

1. The user first creates the CoP and policy on the ESA.
2. The user then exports the CoP using the export API. The CoP package is encrypted only using the Password-based encryption (PBE), where a passphrase and salt is used to encrypt the CoP package.

Note:

The CoP package cannot be encrypted using the Cloud Key Management System (KMS).

3. The user then exports the policy using the export API. This policy can be encrypted by one of the following methods:

- Using PBE where a passphrase and salt is used to encrypt the policy package.
 - Using Cloud KMS to encrypt the policy package.
4. The user copies the encrypted CoP package and policy to an Azure Storage Container or a persistent volume.

Stage 2

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

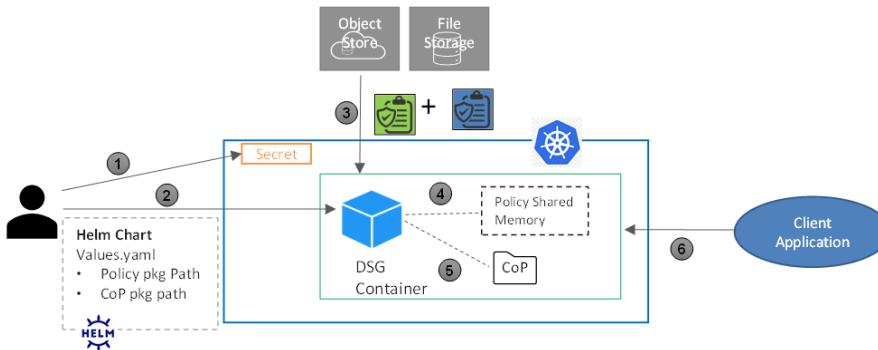


Figure 1-2: Deploying the DSG Container

1. The user creates a Kubernetes secret that is used to decrypt the CoP and the policy packages. This Kubernetes secret contains the following entries:
 - Passphrase and salt for the CoP package
 - Passphrase and salt/KMS key ID for the policy package
2. The user runs the Helm chart to deploy the DSG container to the Kubernetes cluster.
3. The DSG container retrieves the CoP and the policy packages from the Azure Storage Container or persistent volume, and decrypts the package using the secrets provided in the Kubernetes secrets.
4. The DSG container then loads the policy in the shared memory of the Pod.
5. The CoP package is extracted and the contents are stored in a specific directory on the Pod. The DSG service is started and it uses the policy and CoP that are available in the Pod.
6. The client Application sends a request to the DSG container for protecting, unprotecting, and reprotecting data.

Chapter 2

Protegility Reference 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 Uploading the Images to the Container Repository](#)
- [2.6 Creating and Uploading the CoP to Azure Storage Container](#)
- [2.7 Creating and Uploading the Policy to Azure Storage Container](#)
- [2.8 Creating the Cloud Runtime Environment](#)
- [2.9 Setting up Splunk to Access Logs](#)
- [2.10 Deploying the Containers](#)
- [2.11 Upgrading the Helm Charts](#)

This section describes the Protegility Reference Deployment model that customers can use to deploy the DSG application on a Kubernetes cluster.

2.1 Verifying the Prerequisites

Ensure that the following prerequisites are met:

1. Verify the reference solution deployment
2. Verify the prerequisites for the reference application
3. Verify the prerequisites that are required on Azure

2.1.1 Verifying the Prerequisites for Reference Solution Deployment

Reference Solution Deployment

- *ESA* – Protegility provides an Azure image *esa_pap-all-64_x86-64_azure_9.1.0.5.2242.vhd*. Use this image to launch an ESA instance on Azure.
 - After launching the ESA instance, you must install the *DSG_PAP-ALL-64_x86-64_3.1.0.5.6.HF-1* patch to extend the DSG Web UI on the ESA.

For more information about installing the DSG 3.1.0.5 patch, refer to section *Installing the DSG* in the *Data Security Gateway Guide 3.1.0.5*.

- *DSGI_RHUBI-ALL-64_x86-64_AZURE.K8S_3.1.0.5.x.tgz* – Installation package for the Protegility Reference Deployment. This package contains the files as described in the following table.

Table 2-1: Installation Package

Name	Description
<i>DSG_K8S-ALL-64_x86-64_3.1.0.5.x.tar.gz</i>	The DSG 3.1.0.5 image. This image is used to create the DSG Docker Container.
<i>DSG-HELM_ALL-ALL-ALL_x86-64_AZURE.K8S_3.1.0.5.x.tgz</i>	Package containing the Helm charts, which are used to deploy the DSG application on the Kubernetes cluster.
<i>DSG-Samples_Linux-ALL-ALL_x86-64_3.1.0.5.x.tgz</i>	Package containing the sample application for testing the DSG container with sample data.
<i>manifest.json</i>	File specifying the name of the product and its components, the version number of the protector, and the build number of product.

2.1.2 Verifying the Prerequisites for Reference Application

Reference Application

This section describes the prerequisites for using the reference application:

- Policy – Ensure that you have defined the security policy on the ESA. In addition, ensure that you fulfill either of the following requirements:
 - You must link the policy to the default data store in the ESA.
 - You must add the Kubernetes node or pod IP address ranges as allowed servers on the data store to which the policy is linked.
- Configuration over Programming (CoP) – The Protegility Reference Deployment uses the default **REST API Example** as the CoP.
- Client application – For example, the banking application, which contains the customer data that you want to protect using the DSG application.

2.1.3 Verifying Prerequisites for Azure

The following additional prerequisites for Azure are required:

Table 2-2: Azure Prerequisites

Prerequisite	Description
<i>Linux Instance</i>	This instance can be used to communicate with the Kubernetes cluster. This instance can be on-premise or on Azure. 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.
Azure Subscription and a valid Azure Active Directory User	Access to an Azure subscription and a valid Azure Active Directory User is required for creating the following Azure services: <ul style="list-style-type: none"> • Resource Group - Used to organize resources. All the remaining Azure services are created within a Resource Group. • Azure Storage Account - Used for creating the Azure Storage Container or the Azure File Share, where the immutable policy and COP 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 and COP package, instead of the Azure File Share. • Azure File Share - This prerequisite is optional, and is required only if you want to use the Azure File Share for uploading the immutable policy and COP package, instead of using an Azure Storage Container.



Prerequisite	Description
	<ul style="list-style-type: none"> Two service principals: <ul style="list-style-type: none"> Service principal 1 with <i>Read</i> permissions on the Azure Storage Account and <i>Decrypt</i> permission on Key Management of the Azure Key Vault. <p>Note: The <i>Decrypt</i> permissions are required if you want to use the Azure Key Vault for encrypting the policy and COP package, instead of using the passphrase-based encryption.</p> Service principal 2 is used to generate the access token for authenticating the requests that are sent from a REST API client to a DSG instance for performing security operations. <p>Note: A service principal is created when you register an application on the Azure portal.</p> <p>Note: Ensure that the user has permissions to get the Azure Key Vault Public Key and to upload to an Azure Storage Container.</p>
Permissions	<p>Ensure that the following permissions are available:</p> <ul style="list-style-type: none"> Azure Kubernetes Service (AKS) to create a Kubernetes cluster Azure Key Vault to create a key used in encryption and decryption operation Azure Container Registry to upload the IMP Updater and DSG Container images
KMS Key	Create KMS key if you want to use it for Policy encryption.

2.2 Downloading the Installation Package

After verifying the prerequisites, you must download the installation package for the Protegility Reference Deployment model.

► To download the installation package:

- Download the *DSGI_RHUBI-ALL-64_x86-64_AZURE.K8S_3.1.0.5.x.tgz* file on the Linux instance.
- Run the following command to extract the files from the *DSGI_RHUBI-ALL-64_x86-64_AZURE.K8S_3.1.0.5.x.tgz* file.

```
tar -xvf DSGI_RHUBI-ALL-64_x86-64_AZURE.K8S_3.1.0.5.x.tgz
```

The following files are extracted:

- DSG_K8S-ALL-64_x86-64_3.1.0.5.x.tar.gz*
- DSG-HELM_ALL-ALL-ALL_x86-64_AZURE.K8S_3.1.0.5.x.tgz*
- DSG-Samples_Linux-ALL-ALL_x86-64_3.1.0.5.x.tgz*
- manifest.json*



3. Run the following command to extract the *.tgz* package that carries the DSG container Helm charts.

```
tar -xvf DSG-HELM_ALL-ALL-ALL_x86-64_AZURE.K8S_<version_number>.tgz
```

The extracted package contains the *dsg* directory, which has the following contents:

- *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.
- *Chart.yaml* – A YAML file that provides information regarding the chart.
- *credentials.json* - This file is used to provide the credentials details for the service principal that has access to the storage object.
- *pty-rbac.yaml* – The values for configuring the Protegity Role Based Access Control (RBAC) for the pods.
- *values.yaml* – The default values for configuring the Helm chart.
- *pty-psp.yaml* – The Protegity Pod Security Policies (PSP) file that is used to manage the security policies for a pod.

Note: The *pty-psp.yaml* file is not applicable for the Azure release.

- *pty-rbac-op.yaml* - The values for configuring the Protegity Role Based Access Control (RBAC) for the pods in OpenShift.

Note: The *pty-rbac-op.yaml* file is not applicable for the Azure release.

- *pty-scc.yaml* – The Protegity Security Context Constraints (SCC) file that is used to manage the security policies for a pod.

Note: The *pty-scc.yaml* file is not applicable for the Azure release.

2.3 Initializing the Linux Instance

After downloading the installation packages, you must initialize the Linux instance. This involves installing the tools and utilities in the order as shown in the following table.

Table 2-3:

Order	Tools Utilities	Description	Notes	References
1	Azure CLI	Provides a set of command line tools for the Azure Cloud Platform.	Configure Azure CLI on your machine by running the following command. curl -sL https://aka.ms/InstallAzureCLIDeb sudo bash	For more information about installing Azure CLI on the Linux instance, refer to https://learn.microsoft.com/en-us/cli/azure/install-azure-cli .
2	<i>kubectl</i>	Command line interface for Kubernetes. Kubectl enables you to run commands from the Linux instance so that you can communicate with the Kubernetes cluster.	Install Kubectl by running the following command. az aks install-cli	For more information about installing <i>kubectl</i> , refer to the section Installing kubectl .
3	Helm Client		Run the following commands sequentially to install the Helm client on the Linux instance.	For more information about installing a Helm



Order	Tools Utilities	Description	Notes	References
			<p>1. curl -fSSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3</p> <p>2. chmod 700 get_helm.sh</p> <p>3. ./get_helm.sh</p>	<p>client, refer to https://helm.sh/docs/intro/install/.</p> <p>Important: Verify the version of the Helm client that must be used from the Readme provided with this build.</p>

2.4 Creating Certificates and Keys for TLS Authentication

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

Note:

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

Before you begin

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

► To initialize the Linux instance:

- 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 dsg-ca.key -out dsg-ca.crt -days 356  
-nodes -subj '/CN=DSG Certificate Authority'
```

Note:

If you have a CA certificate and a private key, then move to 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 dsg-wildcard.key -out dsg-wildcard.csr -nodes  
-subj '/CN=*.example.com'
```

```
openssl x509 -req -sha256 -days 365 -in dsg-wildcard.csr -CA dsg-ca.crt -CAkey dsg-ca.key  
-set_serial 04 -out dsg-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 dsg-client.key -out dsg-client.csr -nodes -subj '/CN=My DSG Client'
```

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

4. Copy all the certificates to a common directory.

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

2.5 Uploading the Images to the Container Repository

This section describes how you can upload the DSG container image to the Container Repository.

Before you begin

Ensure that you setup the Container Registry.

Note:

The steps listed in this section for uploading the container images to the Azure Elastic Container Repository (ECR) are for reference use. You can choose to use a different Container Registry for uploading the container images.

To upload the images to the Container Repository:

1. Install Docker on the Linux instance.

For more information about installing Docker on a Linux machine, refer to the [Docker documentation](#).

2. Run the following command to authenticate your Docker client with Azure.

```
az login --scope https://management.core.windows.net//.default  
sudo docker login sampleacr.azurecr.io --username <acr username> --password <password>
```

Note: For more information about authenticating your Docker client with Azure Container Registry, refer to the <https://learn.microsoft.com/en-us/azure/container-registry/container-registry-authentication> Azure Authentication documentation.

3. Login to the Azure Container Registry using the following command:

```
docker login <Container Registry Name>.azurecr.io
```

4. Extract the installation package.

The DSG container image is extracted.

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

- Run the following command to load the DSG container image on Docker.

```
docker load -i DSG_K8S-<Kubernetes_version>_x86-64_<version>.tgz
```

- Run the following command to list the DSG container image.

```
docker images
```

- Tag the image to the Azure Container Registry by running the following command.

```
docker tag <Container image>:<Tag> <Container registry path>/<Container image>:<Tag>
```

For example:

```
docker tag dsg-<version number>:Azure <azure container registry name>.azurecr.io/images/dsg_3.1.0.5.5:latest
```

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

- Push the tagged image to the Azure Container Registry by running the following command.

```
docker push <Container registry path>/<Container image>:<Tag>
```

For example:

```
docker push <azure container registry name>.azurecr.io/images/dsg_3.1.0.5.5:latest
```

- Navigate to the directory where you have extracted the Helm charts package.
- In the *values.yaml* file, update the required path for the *dsgImage* settings, with their tags.

2.6 Creating and Uploading the CoP to Azure Storage Container

This section describes how you can create a Ruleset on the ESA. In addition, this section describes how you can export a CoP from the ESA.

Important: Disable the Learn Mode on the DSG as it is not applicable to the containerized application.

2.6.1 Creating and Exporting the Ruleset

This section describes how you can create a Ruleset on the ESA.

► To create a Ruleset:

- On the ESA Web UI, create a Configuration Over Programming (CoP) Ruleset.

By default, the ESA provides you with default ruleset *REST API Examples* for testing the DSG functionality.

For more information about creating a CoP Ruleset, refer to the section *Ruleset Menu* in the *Data Security Gateway User Guide 3.1.0.5*.

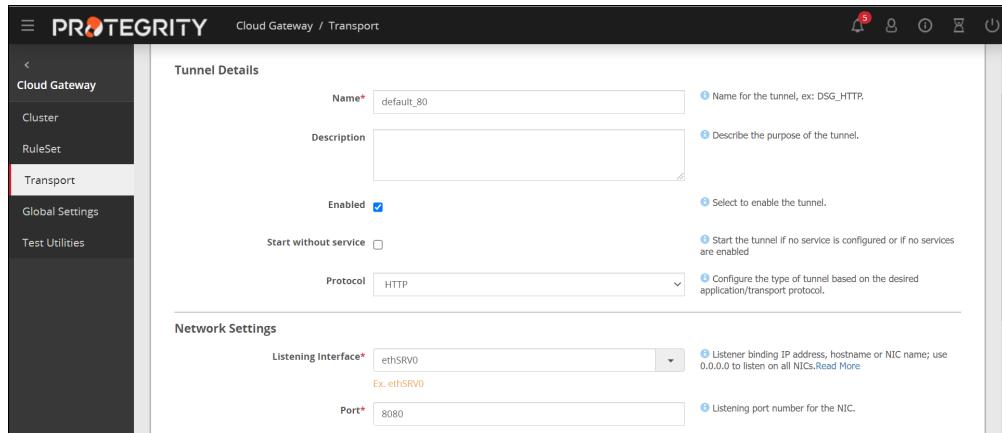
2. Navigate to **ESA Web UI > Cloud Gateway > Transport > Tunnels > Create Tunnel**.

For more information about creating a tunnel, refer to the section *Tunnels tab* in the *Data Security Gateway User Guide 3.1.0.5*.

Important:

Create a tunnel with port number higher than *1024* as the Docker containers do not allow any external communication on port numbers below *1024*.

The following figure shows a sample tunnel that uses the HTTP protocol on port *8080* using the service interface *ethSRV0*.



Note:

Ensure that any other tunnel that uses a port number lower than *1024* is disabled.

Note:

The NFS/CIFS tunnel and GPG transformations are not supported in the DSG containers.

Important:

Ensure that the port number that you specify in the **Port** field matches the value that you specify in the *targetPort* parameter in the *values.yaml* file under the **dsgService > tunnels** section.

For more information about the *values.yaml* file parameters, refer to the section *Values.yaml*.

3. If you want to use the transaction metrics, then in the **Advanced Settings** section, specify the following value to support the X-Forwarded-For (XFF) header in the REST API requests sent to the DSG instances.

```
{
  "xheaders": true
}
```



Figure 2-1: Advanced Settings

The XFF header is used to identify the client machine from which the REST API requests originate.

4. Click **Create** to create the tunnel.
5. On the ESA Web UI, navigate to **Cloud Gateway > RuleSet**.
By default, the DSG provides you default rulesets for testing the DSG functionality.
6. If you want to specify multiple hostnames for the same ruleset, then perform the following steps.
 - a. On the right-hand side screen, click **Edit**.
 - b. Click the icon next to the **Hostname** field.
 - c. Specify the required *hostname* in the **Hostname** fields.

Hostname	
Hostname*	restapi
Hostname*	restapi-prod
Hostname*	restapi-staging

You can use the multiple hostname option to run production URLs on one hostname and staging URLs on another hostname. If you want to use multiple hostnames, then you must add the same hostnames in the **hosts > host** field of the *values.yaml* file.

For more information about the *values.yaml* file parameters, refer to the section *Values.yaml*.

7. Navigate to **REST API Examples > Protect Rule > Data Protection**.
8. On the right-hand side screen, click **Edit** and specify the data element name in the **DataElement Name** field.

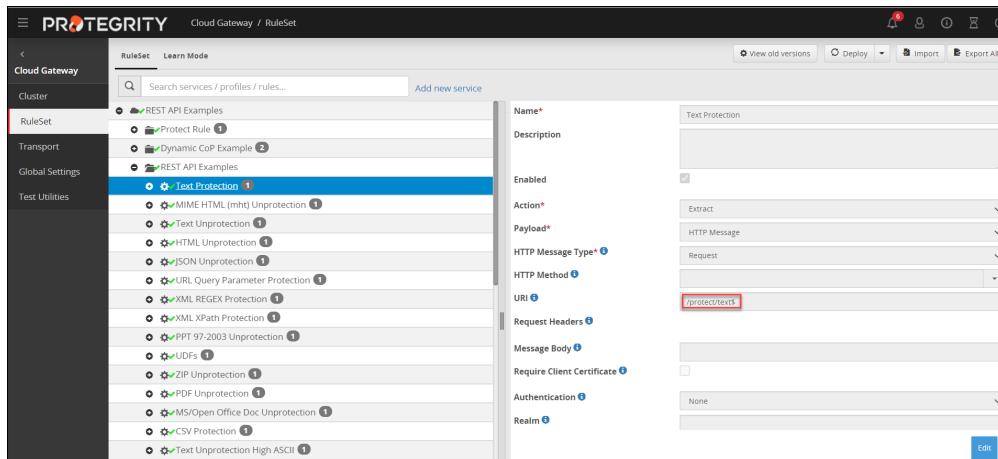
Note:

Ensure that this data element is part of the security policy that you have created.

9. Click **Save**.
10. Expand **REST API Examples** and navigate to the Text Protection ruleset. Note the URI.

The Postman client is used to send a request to the following location:

<http://<hostname>/protect/text>



Note: The Text Protection rule is used to check the health of the pods. Leave the **HTTP Method** text box blank or enter the **GET** method to perform this check.

11. Perform the following steps to enable the basic authentication functionality, which ensures that the user who is performing the security operations, is authenticated as part of an existing LDAP. The username and password of the user are sent as part of the REST API request to authenticate the user.

Important: Ensure that the user that is utilized for authentication is added to the Active Directory, and assigned a role that includes the *Cloud Gateway Auth* permissions.

You can also use the ESA LDAP for adding users.

For more information about managing users in the ESA, refer to the section *Managing Users* in the [Appliances Overview Guide 9.1.0.5](#).

For more information about assigning permissions to user roles in the ESA, refer to the section *Managing Roles* in the [Appliances Overview Guide 9.1.0.5](#).

For more information about the basic authentication functionality, refer to the section *Basic Authentication* in the [Data Security Gateway Guide 3.1.0.5](#).

- a. Navigate to the **Cloud Gateway > RuleSet** tab.

- b. Select the required rule.

- c. In the **Authorization** list, select **Basic**.

For more information about enabling the basic authentication functionality, refer to the section *Enabling REST API Authentication* in the [Data Security Gateway Guide 3.1.0.5](#).

- d. Save the changes.

- e. Click **Deploy** to push the configurations to all the DSG nodes.

Note: The user can either click **Deploy** or **Deploy to Node Groups** on the Ruleset page. For more information about the deploy functionality, refer to the section *Ruleset Tab* in the [Data Security Gateway User Guide 3.1.0.5](#).

12. To export the CoP package from the ESA and encrypt the policy package using a passphrase and a salt, run the following command.

CURL request syntax for exporting CoP

```
curl --location --request POST 'https://<ESA_IP_address>/exportGatewayConfigFiles' \
--header 'Authorization: Basic <Base64 encoded administrator credentials>' \
--header 'Content-Type: text/plain' \
```

```
--data-raw '{
"kek": {
"pkcs5": {
"passphrase": "<passphrase>",
"salt": "<salt>"
}
}
}' -k -o <Response file name in .json format>
```

Sample CURL request for exporting CoP

```
curl --location --request POST 'https://xxx.xxx.xxx.xxx/exportGatewayConfigFiles' \
--header 'Authorization: Basic <Base64 encoded administrator credentials>' \
--header 'Content-Type: text/plain' \
--data-raw '{
"kek": {
"pkcs5": {
"passphrase": "xxxxxxxxxxxx",
"salt": "salt"
}
}
}' -k -o cop_demo.json
```

Note: The response for the CURL request will be stored in *cop_demo.json* file.

Note: For more information about exporting a CoP, refer to the section *Appendix H: CoP Export API for deploying the CoP (Containers only)* in the [Data Security Gateway User Guide 3.1.0.5](#).

13. If you want to upload the CoP package to an Azure Storage Container, then perform the following steps.
 - a. Login to the Azure environment.
 - b. Create Azure Storage Container.
 - c. Upload the CoP package to an Azure Storage Container.

Important: Ensure that the user has the required permissions to upload an object to the Azure Storage Container.

14. If you want upload the CoP package to an Azure File Share, then perform the following steps.
 - a. Login to the Azure environment.
 - b. Create an Azure File Share instance.

For more information about creating an Azure File Share instance, refer to [Creating an Azure File Share](#).

After you have created a file system on the Azure File Share, you need to mount this file system to the Linux instance that has been created in the Azure environment.

- c. Navigate to the Azure File Share that you have created.
- d. Click **Upload**.
- e. In the **Files** field, browse to the location where you have downloaded the CoP package file, and select the file.
- f. Click **Upload** to upload the CoP package file to the Azure File Share.

2.7 Creating and Uploading the Policy to Azure Storage Container

This section describes how you can create a policy on the ESA. In addition, this section describes how you can export a Policy from the ESA.

2.7.1 Creating and Exporting the Policy

This section describes how you can create and export the policy on the ESA.

► To create a Ruleset:

1. On the ESA Web UI, create a policy.

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

2. To export the policy package from the ESA and encrypt the policy package using a passphrase and a salt, then run the following command.

CURL request syntax for exporting policy using PKCS5

```
curl -v -k -u "<username>:<ESA password>" https://<ESA_IP_address>/pty/v1/imp/export -H "Content-Type: application/json" --data '{"name": "<name>", "pepVersion": "<pepVersion>", "dataStoreName": "<dataStoreName>", "emptyString": "NULL", "caseSensitive": "YES", "kek": {"pkcs5": {"password": "<password>", "salt": "<salt>"}}}' -o <Response file name in .json format>
```

Note: Ensure that the **Export IMP** role is assigned to username. This role can be assigned from the **Roles** screen on the ESA Web UI.

Sample CURL request for exporting policy using PKCS5

```
curl -v -k -u "export_user:admin1234" https://xxx.xxx.xxx.xxx/pty/v1/imp/export -H "Content-Type: application/json" --data '{"name": "Policy1.json", "pepVersion": "1.2.0+16", "dataStoreName": "DS1", "emptyString": "NULL", "caseSensitive": "YES", "kek": {"pkcs5": {"password": "xxxxxxxxxx", "salt": "salt"}}}' -o Policy1.json
```

Note: The response for the CURL request will be stored in the *Policy1.json* file.

Note: For more information about the API to export the immutable policy, refer to the section *Appendix B: APIs for Immutable Protectors* in the [Protegility APIs, UDFs, Commands Reference Guide 9.1.0.0](#).

3. To export the policy package from the ESA and encrypt the policy package using a KMS, run the following command.

CURL request syntax for exporting policy using KMS

```
curl -v -k -u "<username>:<password>" https://<ESA IP>/pty/v1/imp/export -H "Content-Type: application/json" -o <Policy_package_name>.tgz --data '{"name": "<name>", "pepVersion": "<pepVersion>",
```

```
"dataStoreName": "<value>",
"emptyString": "<value>",
"caseSensitive": "<value> , "kek": { "publicKey": {
  "value": "Public Key of the
Azure Key Vault Key used to encrypt the policy" } } }'
-o Policy1.json
```

Note:

Ensure that the **Export IMP** role is assigned to username. This role can be assigned from the **Roles** screen on the ESA Web UI.

Note:

You can obtain the public key by running the following command.

```
az keyvault key download --id <keyid> -e PEM -f <keyFilePath>
```

Sample CURL request for exporting policy using KMS

```
curl -v -k -u "IMPUUser:<password>" https://xxx.xxx.xxx.pty/v1/imp/export -H "Content-Type: application/json" -o policy_package_key.tgz --data '{"name": "policypackage", "pepVersion": "1.2.0+16", "dataStoreName": "<value> , "emptyString": "<value> , "caseSensitive": "<value> , "kek": { "publicKey": { "value": "-----BEGIN PUBLIC KEY-----\nMIIBBojANBggkqhkiG9w0BAQEFAOCAY8AMIBigKCAYEAjwxqqrrNQ1kV84GEfYLR\\nVK/SyOkGTs7kxE\nImEYMGtugSal2G9m7hFdVlWhHH/OvLDHwdOYe6GLXHVwyfcfbM/\\nS6pzQPS5ujzyJEaLWYAfCrgAhI9g\n8GvazDiv34Z+VO5FRbJzP9u9/23J4hExc+e1\\nTezKsoBj6Ypx/zBkE4dGqdamfHJUF3ptB82nhJT/Pt\nh15ejL/SVRd/q8sORU838O\\negcIfQYmd5ziZJgbWLvzFaM9QZXH9hd/0JxAhqfHP3Fnhmcmc5MDVdg7D0\nSQkufIjf\\nu7djMy2I3ZRgnkGxjq8PuBUFaH2s6DvAa4e6K0ppGjFoByVVe2tumrUVEvJg2EM\\ndD/P\n115kzLelbDKjxrI8T1D8cWuGh43wf1xC+uRzsfoMSGwDWdpBmaOPF+q2k0d4\\nrQWKIgZw71mxYADPKB\nAUUwMgD/aREuvTD1pvpl0Vk2Y5i/4Xx/fK7GV5DCdy9TFM\\nZgzAA/09z5tuVzqcgodMu06+iqtXdTUo\nOafA+BTvSIXLAgMBAAE=\\n-----END PUBLIC KEY-----" , "algorithm": "RSA_OAEP_SHA256" } } }' -o Policy1.json
```

Note: For more information about the API to export the immutable policy, refer to the section *Appendix B: APIs for Immutable Protectors* in the *Protegrity APIs, UDFs, Commands Reference Guide 9.1.0.0*.

4. If you want to upload the Policy package to an Azure Storage Container, then perform the following steps.
 - a. Login to the Azure environment.
 - b. Create Azure Storage Container.
 - c. Upload the Policy package to an Azure Storage Container.

Important: Ensure that the user has the required permissions to upload an object to the Azure Storage Container.

5. If you want upload the Policy package to an Azure File Share, then perform the following steps.
 - a. Login to the Azure environment.
 - b. Create an Azure File Share instance.

For more information about creating an Azure file Share instance, refer to [Creating an Azure File Share](#).

After you have created a file system on the Azure File Share, you need to mount this file system to the Linux instance that has been created in the Azure environment.



- c. Navigate to the Azure File Share that you have created.
- d. Click **Upload**.
- e. In the **Files** field, browse to the location where you have downloaded the Policy package file, and select the file.
- f. Click **Upload** to upload the Policy package file to the Azure File Share.

2.8 Creating the Cloud Runtime Environment

This section describes how to create the Cloud runtime environment.

2.8.1 Creating the Azure Runtime Environment

This section describes how to create the Azure runtime environment.

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
- Introduction to Azure Kubernetes Service

You can follow the steps to setup the Azure cloud environment as described in the following table.

Table 2-4: Azure Cloud Environment Setup

Steps	References
Logging in to the Azure Environment	For more information about logging in to the Azure environment, refer to the section Logging in to the Azure Environment .
Registering an Application	For more information about registering an application in the Azure Active Directory, refer to the section Registering an Application .
Creating an Azure Storage Account	For more information about creating an Azure Storage Account, refer to Creating an Azure Storage Account .
Creating an Azure Storage Container	For more information about creating an Azure Storage Container, refer to Creating an Azure Storage Container .
Creating an Azure File Share	For more information about creating an Azure File Share, refer to Creating an Azure File Share .
Creating a Kubernetes Cluster	For more information about creating a Kubernetes Cluster, refer to the section Creating Kubernetes Cluster .

2.8.1.1 Logging in to the Azure Environment

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

► To login to the Azure environment:

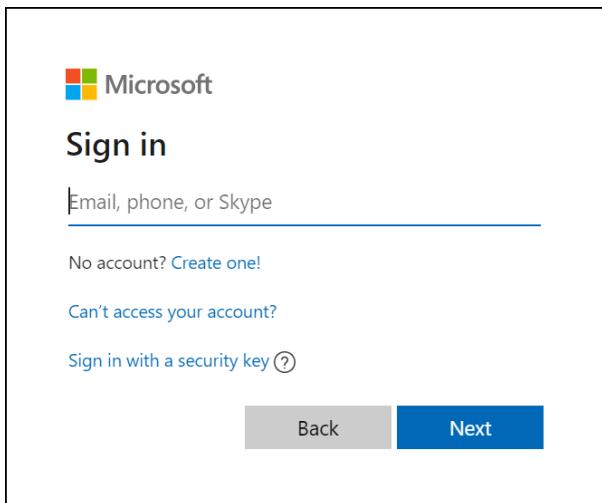
1. Access Azure using the following URL:

<https://azure.microsoft.com>

The **Microsoft Azure** home screen appears.

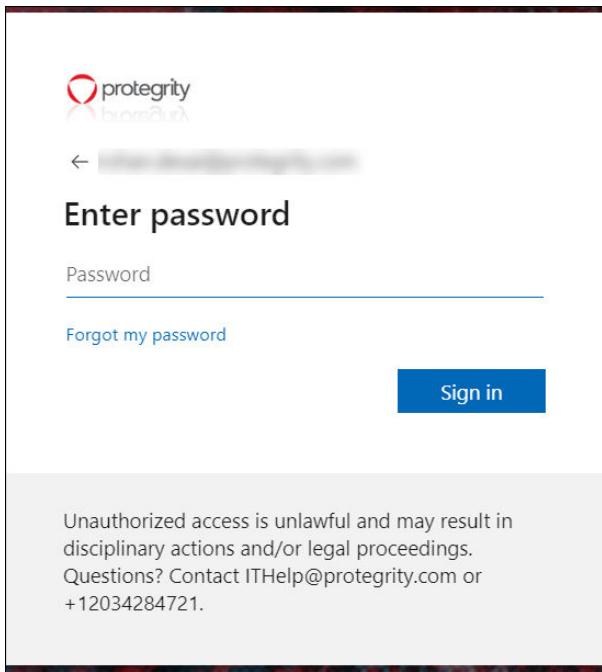
2. Click **Sign in**.

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.

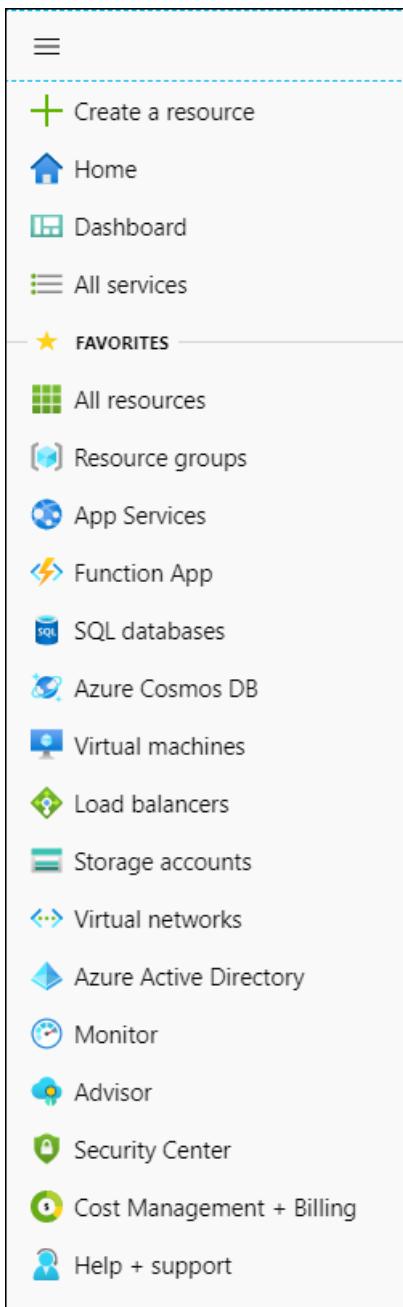
The Azure Portal home screen appears. By default, the home screen displays a list of frequently used Azure services.



Figure 2-2: Azure Portal Home Screen

6. Click the **Portal** menu icon (≡).

The **Portal** menu appears.



7. Click **Resource groups**.

The **Resource groups** screen appears.

Name	Subscription	Location
[REDACTED]	Azure Cloud Platform	East US
[REDACTED]	Azure Cloud Platform	East US 2
[REDACTED]	Azure Cloud Platform	East US 2
[REDACTED]	Azure Cloud Platform	East US 2
[REDACTED]	Azure Cloud Platform	East US 2

- 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.

2.8.1.2 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.

► To register an application:

- 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](#).

- Click the **Portal** menu icon (≡).

The **Portal** menu appears.

- Navigate to **Azure Active Directory**.

The **Overview** screen of the Azure Active Directory appears.

Name	Users	Tenant ID	Groups
Protegility USA, Inc.	589	REDACTED	1,278
Primary domain	Applications	Devices	
protegility.com	319	763	

Figure 2-3: Azure Active Directory Overview Screen

- Navigate to **App registrations**.

The **App Registrations** screen appears.

The screenshot shows the 'App registrations' section of the Azure Active Directory portal. On the left, there's a sidebar with options like Overview, Preview features, and Manage (Users, Groups, External Identities, Roles and administrators, Administrative units, Enterprise applications, Devices, App registrations). The main area shows a table with one row for 'service-principal'. The columns are Application (client) ID, Display name, Created on, and Certificates & secrets. A red box highlights the 'service-principal' entry.

Figure 2-4: App Registrations Screen

- Click **New registration**.

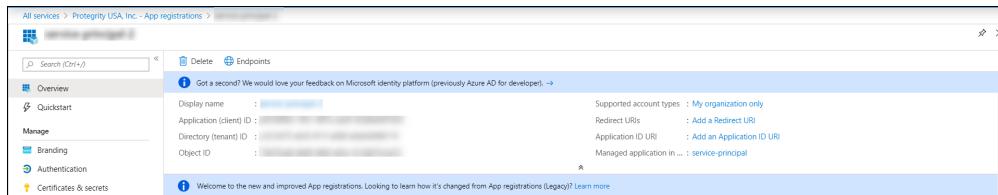
The **Register an application** screen appears.

The screenshot shows the 'Register an application' form. It includes fields for Name (with a placeholder 'My App'), Supported account types (set to 'Accounts in this organizational directory only'), Redirect URI (optional, with a dropdown set to 'Web' and a text input field for 'e.g. https://myapp.com/auth'), and a checkbox for agreeing to Microsoft Platform Policies. A 'Register' button is at the bottom.

Figure 2-5: Register an Application Screen

- In the **Name** field, specify a name for the service principal.
For example, specify **service-principal-1**.
- In the **Supported account types** field, retain the default option for single tenant.
- 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.

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.

This screenshot shows the 'Certificates & secrets' screen for a specific application. The left sidebar shows various management options. The main area displays sections for 'Certificates' (which are empty) and 'Client secrets'. A button labeled '+ New client secret' is visible. Below it, a table lists client secrets with columns for 'Thumbprint', 'Start Date', and 'Expires'.

Figure 2-6: Certificates & Secrets Screen

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

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

This is a modal dialog titled 'Add a client secret'. It contains two fields: 'Description' (with a placeholder text box) and 'Expires' (with three radio button options: 'In 1 year' (selected), 'In 2 years', and 'Never'). At the bottom are 'Add' and 'Cancel' buttons.

Figure 2-7: Add a Client Secret Screen

- c. In the **Description** field, specify a description for the client secret.
- d. In the **Expires** field, retain the default value.
- e. Click **Add**.

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

Description	Expires	Value
secret for service principal	11/19/2020	[Value]

- 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 file for the corresponding service principal.

10. Repeat *step 3* to *step 7* to register another application for creating service principal 2.

2.8.1.3 Creating a Kubernetes Cluster

This section describes how to create a Kubernetes Cluster on Azure.

Note:

The steps listed in this procedure for creating a Kubernetes cluster are for reference use. If you have an existing Kubernetes cluster or want to create a Kubernetes cluster based on your requirements, then you can directly navigate to [step 9](#) to connect your Kubernetes cluster and the Linux instance. However, you must ensure that your DSG service port is enabled on the Network Security group of your VPC.

► To create a Kubernetes cluster:

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 **All Services > Kubernetes services**.

The **Kubernetes Services** screen appears.

Type	Resource group	Kubernetes version	Location	Subscription
Kubernetes service	[redacted]	1.14.8	East US 2	Azure Cloud Platform
Kubernetes service	[redacted]	1.14.7	East US 2	Azure Cloud Platform
Kubernetes service	[redacted]	1.14.7	East US	Azure Cloud Platform
Kubernetes service	[redacted]	1.14.8	East US	Azure Cloud Platform
Kubernetes service	[redacted]	1.13.11	East US	Azure Cloud Platform
Kubernetes service	[redacted]	1.13.12	East US 2	Azure Cloud Platform

4. Click **Add**.

The **Create Kubernetes cluster** screen appears.

All services > Kubernetes services > Create Kubernetes cluster

Create Kubernetes cluster

Basics Scale Authentication Networking Monitoring Tags Review + create

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline. [Learn more about Azure Kubernetes Service](#)

Project details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ	Azure Cloud Platform
Resource group * ⓘ	Select existing... Create new

Cluster details

Kubernetes cluster name * ⓘ	
Region * ⓘ	(US) East US 2
Kubernetes version * ⓘ	1.13.12 (default)
DNS name prefix * ⓘ	

Primary node pool

The number and size of nodes in the primary node pool in your cluster. For production workloads, at least 3 nodes are required.

Review + create < Previous Next : Scale >

5. In the **Resource group** field, select the required resource group.
6. In the **Kubernetes cluster name** field, specify a name for your Kubernetes cluster.
Retain the default values for the remaining settings.
7. Click **Review + create** to validate the configuration.
8. Click **Create** to create the Kubernetes cluster.
The Kubernetes cluster is created.
9. Login to the Linux instance, and run the following command to connect your Linux instance to the Kubernetes cluster.

```
az aks get-credentials --resource-group <Name of Resource Group> --name <Name of Kubernetes Cluster>
```

The Linux instance is now connected with the Kubernetes cluster. You can now run commands using the Kubernetes command line interface (*kubectl*) to control the nodes on the Kubernetes cluster.
10. Validate whether the cluster is up by running the following command.

```
kubectl get nodes
```

The command lists the Kubernetes nodes available in your cluster.

2.9 Setting up Splunk to Access Logs

This section describes how to setup a third-party tool, such as Splunk, to access the DSG and Container logs.

Important:

Ensure that you have a valid license for using Splunk.

2.9.1 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 DSG 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.

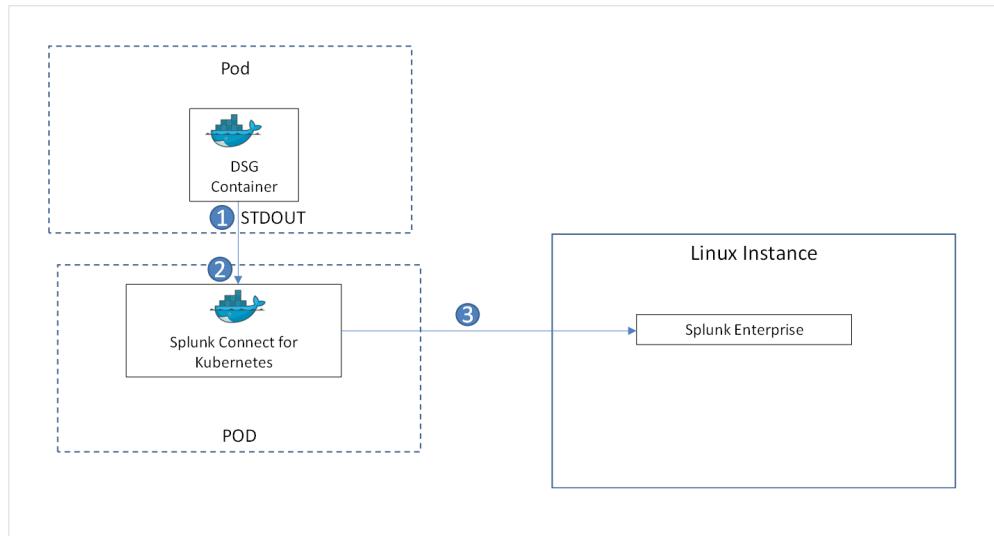


Figure 2-8: Sample Logging Workflow

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

1. The DSG 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.

2.9.2 Setting Up Splunk

This section describes how you can setup Splunk for accessing the DSG and Container logs.

► To setup Splunk:

1. Create a Linux instance, either 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.

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

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

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

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

- 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.

- 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 `xxx.xxx.xxx.xxx`, then you can access Splunk Web at `http://xxx.xxx.xxx.xxx:8000`.

2.9.3 Configuring Splunk

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

► To configure Splunk Enterprise:

- Login to the Splunk Web UI at `http://<IP of Linux instance>:8000`.
- 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.

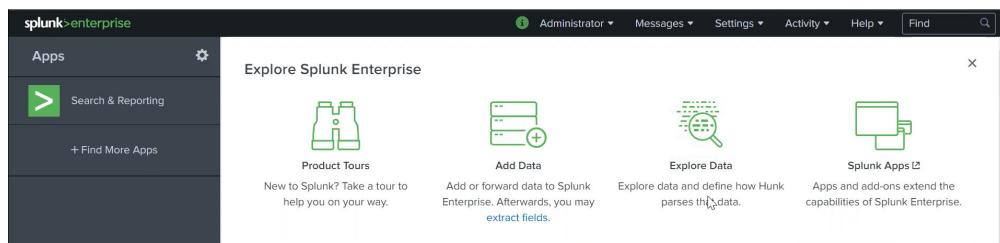


Figure 2-9: Splunk Enterprise Web UI

For more information about the user credentials, refer to the section [Setting Up Splunk](#).

3. Create an index, which is a repository for storing the Splunk Enterprise data, by performing the following steps.

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

The **Indexes** screen appears.

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

Figure 2-10: Indexes Screen

- b. Click **New Index**.

The **New Index** dialog box appears.

New Index

General Settings

Index Name

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

Index Data Type Events Metrics

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

Home Path

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

Cold Path

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

Thawed Path

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

Save **Cancel**

Figure 2-11: 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 DSG 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 setup HTTP Event Collector (HEC) on the Splunk Web UI. The HEC enables you to receive the DSG and Container logs sent from Kubernetes.

For more information about HEC, refer to the section [Setup and use HTTP Event Collector in Splunk Web](#) in the Splunk documentation.

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

The **Data inputs** screen appears.

The screenshot shows the Splunk Data inputs screen. At the top, there's a header with 'splunk>enterprise' and various navigation links like 'Administrator', 'Messages', 'Settings', 'Activity', 'Help', and a search bar. Below the header, the title 'Data inputs' is displayed with a sub-instruction: 'Set up data inputs from files and directories, network ports, and scripted inputs. If you want to set up forwarding and receiving between two Splunk instances, go to Forwarding and receiving.' A table titled 'Local inputs' lists five categories:

Type	Inputs	Actions
Files & Directories	9	+ Add new
HTTP Event Collector	1	+ Add new
TCP	0	+ Add new
UDP	0	+ Add new
Scripts	5	+ Add new

Figure 2-12: Data inputs Screen

b. Click **HTTP Event Collector**.

The **HTTP Event Collector** screen appears.

The screenshot shows the Splunk HTTP Event Collector screen. At the top, there's a header with 'splunk>enterprise' and various navigation links. Below the header, the title 'HTTP Event Collector' is displayed with a sub-instruction: 'Data Inputs > HTTP Event Collector'. A table lists existing tokens:

Name	Actions	Token Value	Source Type	Index	Status
k8s_token	Edit Disable Delete	c546efc9-14db-4cd5-adc8-b4cf185a6ebe	metrics		Enabled

Figure 2-13: HTTP Event Collector Screen

c. Click **New Token**.

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

The screenshot shows the Splunk Add Data > Select Source screen. At the top, it says 'Add Data' with a progress bar at step 1 ('Select Source'). Below the progress bar, there are four source types listed: 'Files & Directories', 'HTTP Event Collector', 'TCP / UDP', and 'Scripts'. The 'HTTP Event Collector' option is selected. On the right side, there's a configuration panel for 'Configure a new token for receiving data over HTTP' with fields for 'Name' (containing 'I'), 'Source name override' (optional), 'Description' (optional), 'Output Group (optional)' (None), and a checkbox for 'Enable indexer acknowledgement'.

Figure 2-14: 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.

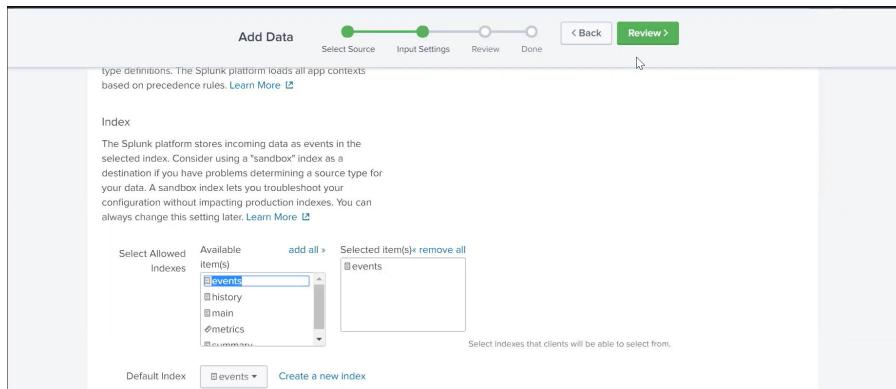


Figure 2-15: Input Settings Screen

- 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.

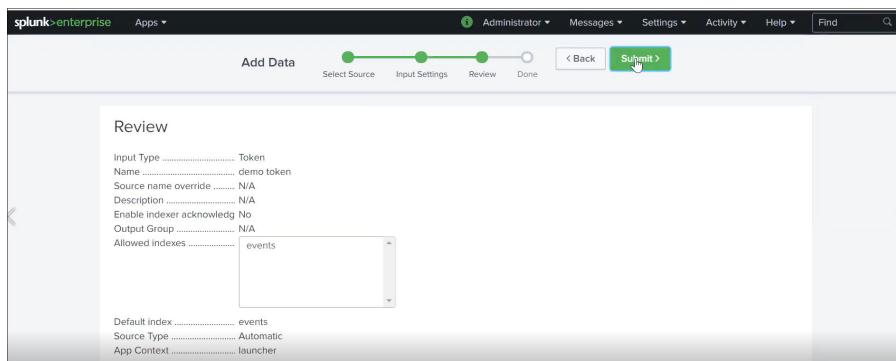


Figure 2-16: Review Screen

- 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.

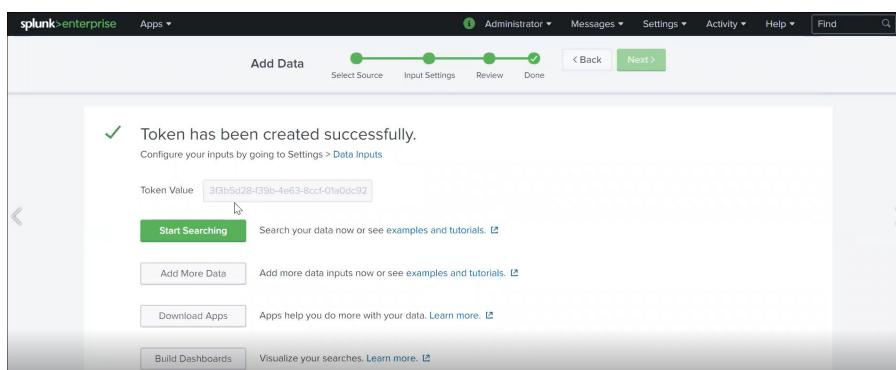


Figure 2-17: Done Screen

5. Configure the global settings for the HEC by performing the following steps.

- 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.

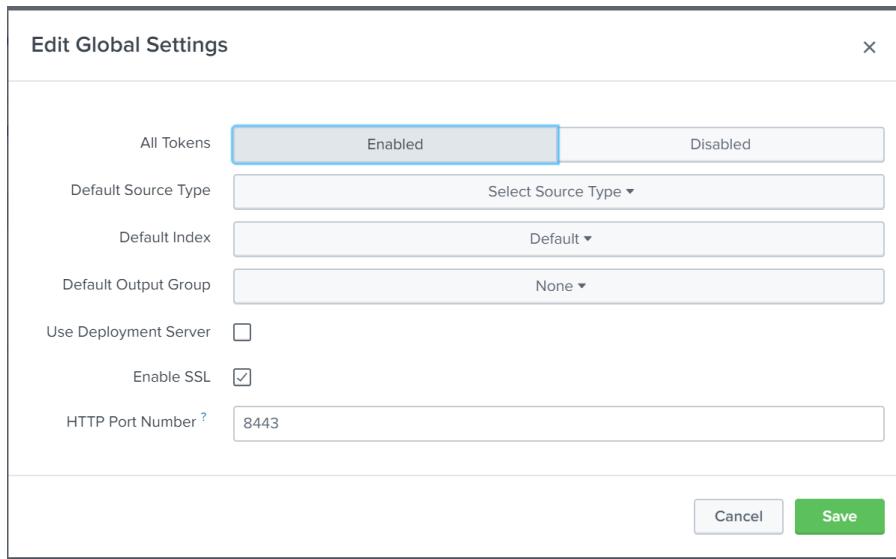


Figure 2-18: Edit Global Settings Dialog Box

- 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.

2.9.4 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 Kubernetes cluster where you have deployed the DSG container.

► To deploy the Splunk Connect for Kubernetes Helm chart:

1. 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: 10.0.0.100
      port: 8443
      indexName: <Index name>
  containers:
```

```
logFormatType: cri
logFormat: "%Y-%m-%dT%H:%M:%S.%N%:z"
```

Note:

If you use the *containerd* runtime, then the ***containers:*** annotation. If you use the *dockerd* runtime, then the ***containers:*** annotation.

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.

2. 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 <i>https</i> .
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 <i>true</i> .
global/splunk/hec/token	Specify the value of the token that you have created in <i>step 4d</i> of the section Configuring Splunk .
global/splunk/hec/host	Specify the IP address of the Linux instance where you have installed Splunk Enterprise.
global/splunk/hec/port	Specify the port number that you have used to configure the HTTP Event Collector of the Splunk Enterprise in <i>step 5e</i> of the section Configuring Splunk .
global/splunk/hec/indexName	Specify the name of the index that you have created in <i>step 3</i> of the section Configuring Splunk .

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.

3. If you want to forward only the Protegility 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 shown in the following snippet.

```
global:
  splunk:
    hec:
      protocol: https
      insecureSSL: true
      token: <Token value>
      host: <Splunk server IP>
      port: 8443
      indexName: <Index name>
  fluentd:
    path: /var/log/containers/*<Namespace where the DSG Container has been deployed>*.log
```

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



In the following example, the DSG Container has been deployed on the *protegity-dsg* namespace. You can then filter out the Protegity logs by specifying the value of the *path* parameter as shown in the following code snippet. This ensures that only the Protegity 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>
  fluentd:
    path: /var/log/containers/*protegity-dsg*.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 Protegity 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: ['.protegity-dsg.']}
```

The highlighted line in the snippet is used to exclude the Protegity 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
```

- Run the following command to deploy the Splunk Connect for Kubernetes Helm chart on the Kubernetes cluster.

```
wget https://github.com/splunk/splunk-connect-for-kubernetes/releases/download/1.5.0/
splunk-kubernetes-logging-1.5.0.tgz
tar xvf splunk-kubernetes-logging-1.5.0.tgz
helm install kube-splunk -n kube-system -f custom-values.yaml splunk-kubernetes-logging
```

- Run the following command to list all the pods that are deployed.

```
kubectl get pods -n kube-system
```

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

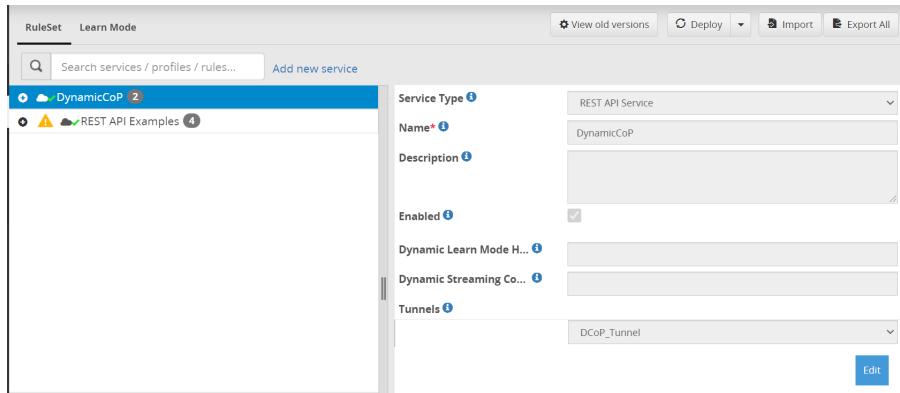
NAME	READY	STATUS	RESTARTS	AGE
kube-splunk-splunk-kubernetes-logging-	1/1	Running	0	15h

2.9.5 Enabling Transaction Metric Logs for Splunk

This section describes how you can enable the transaction metric logs so that the DSG can write these logs to the *STDOUT* stream. The Splunk for Kubernetes is then used to collect these logs and then send them to the Splunk Enterprise.

► To enable the transaction metric logs:

1. Login to the ESA as *admin*.
 2. Navigate to the **Cloud Gateway > RuleSet** tab.
- By default, the REST API Examples ruleset is selected.



3. Click **Edit** to edit the **DynamicCop**.
4. Under **Transaction Metrics Logging**, select the **Enabled** check box.
5. Click **Save** to save the changes.

2.10 Deploying the Containers

This section describes the different methods that you can use to deploy the DSG container. You can deploy the container as described in the following table.

Table 2-5: Deploying the DSG container

Methods	Steps
With Azure Policy and Role-Based Access Control (RBAC)	<ol style="list-style-type: none"> 1. <i>Apply Azure Policy</i> 2. <i>Apply RBAC</i> 3. <i>Setup the Environment for Deploying the DSG Container</i> 4. <i>Deploy a Release on the Kubernetes Cluster with Azure Policy and RBAC</i>
Without Azure Policy and RBAC	<i>Deploy a Release on the Kubernetes Cluster without Azure Policy and RBAC</i>

2.10.1 Applying an Azure Policy

This section describes how to apply an Azure Policy.

► To apply an Azure Policy:

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. Perform the following steps to enable the application of Azure Policies on the Kubernetes cluster.

- a. Navigate to **All Services > Kubernetes services**.

The **Kubernetes services** screen appears.

Name ↑	Type ↑↓	Resource group ↑↓	Kubernetes version ↑↓	Location ↑↓	Subscription ↑↓
[REDACTED]	Kubernetes service	[REDACTED]	1.14.8	East US 2	Azure Cloud Platform
[REDACTED]	Kubernetes service	[REDACTED]	1.14.7	East US 2	Azure Cloud Platform
[REDACTED]	Kubernetes service	[REDACTED]	1.14.7	East US	Azure Cloud Platform
[REDACTED]	Kubernetes service	[REDACTED]	1.14.8	East US	Azure Cloud Platform
[REDACTED]	Kubernetes service	[REDACTED]	1.13.11	East US	Azure Cloud Platform
[REDACTED]	Kubernetes service	[REDACTED]	1.13.12	East US 2	Azure Cloud Platform

- b. Select the Kubernetes cluster that you have created in the section [Creating Kubernetes Cluster](#).
- c. Navigate to **Policies**.

The **Onboard to Azure Policy for Azure Kubernetes Service (AKS)** screen appears.

Home > Kubernetes services > [REDACTED] | Policies

Kubernetes service

Search (Ctrl+ /)

- Configuration
- Scale
- Networking
- Dev Spaces
- Deployment center (preview)
- Policies**
- Properties

Onboard to Azure Policy for Azure Kubernetes Service (AKS)

With this integration, you can apply at-scale enforcements and safeguards for AKS clusters in a centralized, consistent manner through Azure Policy.

Learn more about Azure Policy for Kubernetes clusters. ↗

Enable add-on

- d. Click **Enable add-on**.

This allows you to assign Azure Policies to the Kubernetes cluster.

After you enable the policy application, the following pods are created in the Kubernetes cluster:

- *azure-policy-<ID>* - This pod is created in the kube-system namespace.
- *azure-policy-webhook-<ID>* - This pod is created in the kube-system namespace.
- *gatekeeper-audit-<ID>* - This pod is created in the gatekeeper-system namespace.
- *gatekeeper-controller-<ID>* - This pod is created in the gatekeeper-system namespace.

- e. To verify the new pods created, run the following commands.

```
kubectl get pods -nkube-system
```

```
kubectl get pods -ngatekeeper-system
```

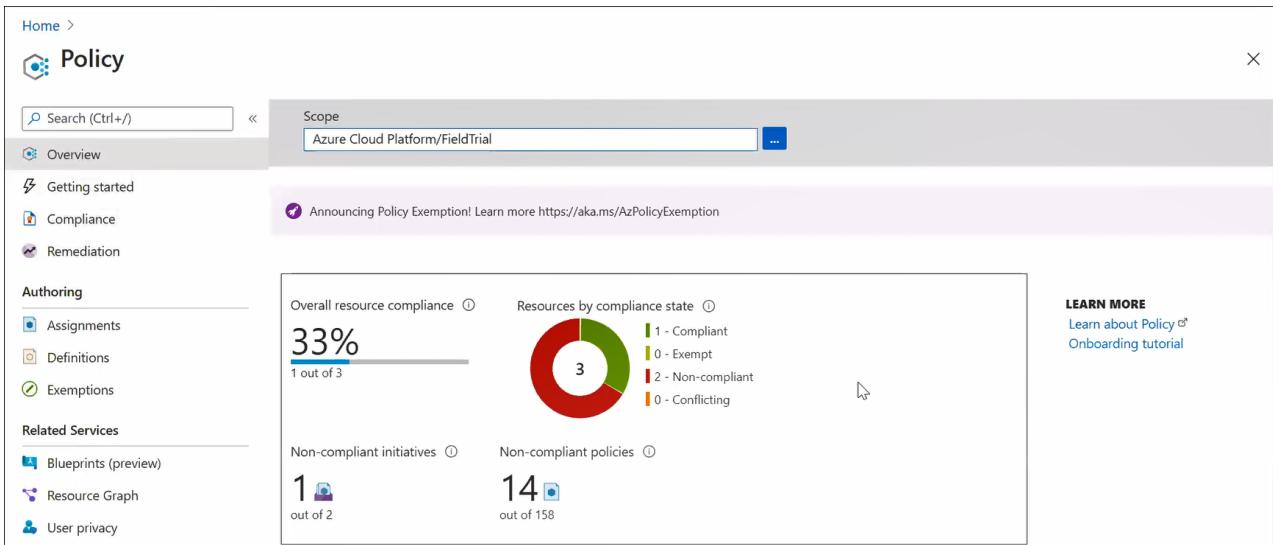
4. Perform the following steps to assign an existing Azure Policy to your Kubernetes cluster.

For more information about assigning a policy, refer to the section [Assign a built-in policy definition](#) in the Azure documentation.

- a. Navigate to **All Services > Policy**.

The **Policy** screen appears.





- On the left pane, click **Definitions**.

The **Definitions** screen appears.

The screenshot shows the Azure Policy Definitions screen. The left pane includes a search bar and a scope filter set to "Azure Cloud Platform". The main area displays a table of policy definitions:

Name	Definition location	Policies	Type	Definition type	Category
PtyAKSInitiative	Azure Cloud Platform	7	Custom	Initiative	Kubernetes
[Preview]: NIST SP 800-171 R2		78	Built-in	Initiative	Regulatory Compliant
Audit machines with insecu...		9	Built-in	Initiative	Guest Configuration
IRS1075 September 2016		62	Built-in	Initiative	Regulatory Compliant
[Preview]: Deploy prerequisi...		4	Built-in	Initiative	Guest Configuration
CIS Microsoft Azure Founda...		87	Built-in	Initiative	Regulatory Compliant
Enable Monitoring in Azure ...		151	Built-in	Initiative	Security Center
[Preview]: Australian Govern...		62	Built-in	Initiative	Regulatory Compliant

- From the **Category** drop-down list, clear the **Select all** check box, and then select the **Kubernetes** check box.
- Select the required policy definition.

The policy definition details screen appears.

The screenshot shows the 'Do not allow privileged containers in Kubernetes cluster' policy definition in the Azure portal. The 'Essentials' tab is selected, displaying the policy's name, description, available effects (audit, deny, disabled), category (Kubernetes), definition location, definition ID, type (Built-in), and mode (Microsoft.Kubernetes.Data). The 'Definition' tab is active, showing the JSON code for the policy:

```

1  {
2   "properties": {
3     "displayName": "Do not allow privileged containers in Kubernetes cluster",
4     "policyType": "BuiltIn",
5     "mode": "Microsoft.Kubernetes.Data",
6     "description": "This policy does not allow privileged containers creation in a Kubernetes cluster. This policy is generally available for Kubernetes clusters.",
7     "metadata": {
8       "version": "1.0",
9       "category": "Kubernetes"
10    },
}

```

You need to apply the following Azure Policies to the Kubernetes cluster:

- *Do not allow privileged containers in Kubernetes cluster*
- *Kubernetes cluster containers should not share host process ID or host IPC namespace*
- *Kubernetes cluster containers should only use allowed capabilities*
- *Kubernetes cluster pods and containers should only run with approved user and group IDs*
- *Kubernetes cluster pods should only use allowed volume types*
- *Kubernetes cluster pods should only use approved host network and port range*
- *Kubernetes clusters should not allow container privilege escalation*
- *Kubernetes cluster containers should run with a read-only root file system*

For more information about each of the Azure Policies, refer to the section [Azure Policy built-in definitions for Azure Kubernetes Service](#) in the Azure documentation.

Repeat [Step 4d](#) to [4m](#) for each of the in-built policies.

- e. Click **Assign**.

The **Basics** tab of the **Assign policy** screen appears.

All services > Policy > Do not allow privileged containers in Kubernetes cluster >

Do not allow privileged containers in Kubernetes cluster

Assign policy

Basics Parameters Remediation Review + create

Scope

Scope [Learn more about setting the scope *](#)

Exclusions

Optionally select resources to exclude from the policy assignment.

Basics

Policy definition

Do not allow privileged containers in Kubernetes cluster

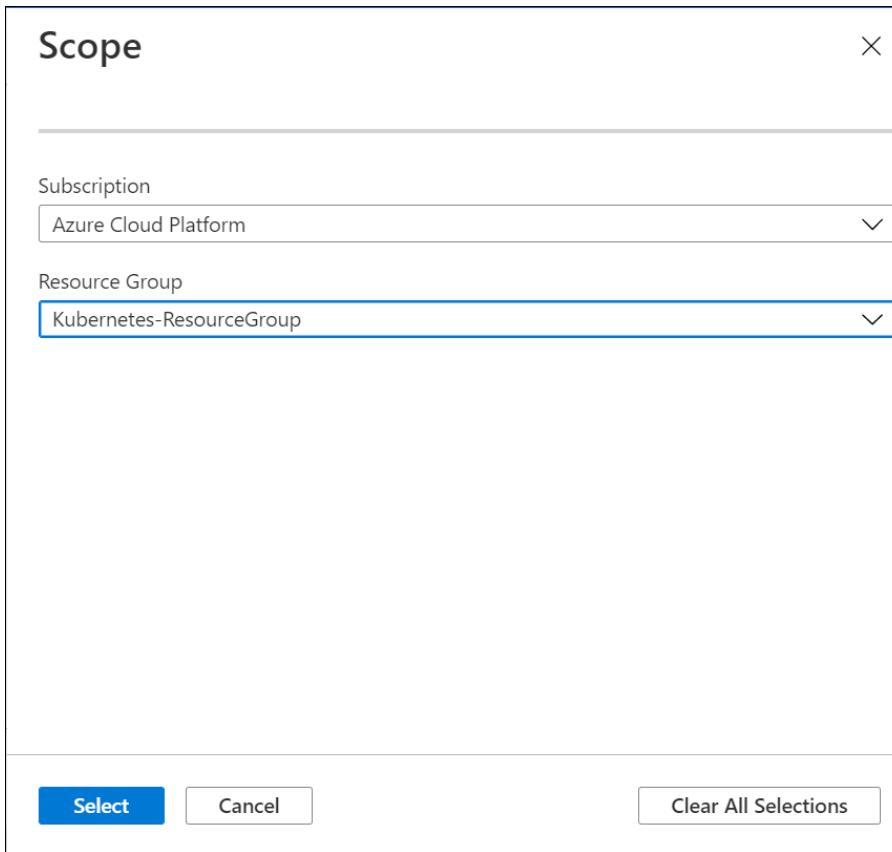
Assignment name * ⓘ

Do not allow privileged containers in Kubernetes cluster

Review + create **Cancel** **Previous** **Next**

- f. In the **Scope** section, click the **...** icon to launch the scope selector.

The **Scope** screen appears.



- g. In the **Subscription** list, select your subscription.
- h. In the **Resource Group** field, select the resource group of your Kubernetes cluster.
This ensures that the Azure Policy is applicable only to the resources that are a part of the resource group.
- i. Click **Select** to specify the scope.
- j. Navigate to the **Parameters** tab.
The **Parameters** tab appears.

All services > Policy > Do not allow privileged containers in Kubernetes cluster >

Do not allow privileged containers in Kubernetes cluster

Assign policy

Basics **Parameters** Remediation Review + create

Specify parameters for this policy assignment.

Effect * ⓘ
deny

Namespace exclusions * ⓘ
["kube-system", "gatekeeper-system", "azure-arc"]

Review + create **Cancel** **Previous** **Next**

The list of parameters depend on the specific policy that you have selected.

- k. In the **Effect** list, specify one of the following values:
- *audit* - The policy will allow any non-compliant resource to be created or updated. However, it will mark the resource as non-compliant.
 - *deny* - The policy will not allow any non-compliant resource to be created or updated.
 - *disabled* - Turns off the policy.

Specify the value as *deny* for all the in-built policies.

Note: This parameter is applicable to all the in-built policies.

- l. In the **Namespace** exclusions field, specify the namespaces that should not be assigned the Azure Policy.

Specify the following namespaces for which the Azure Policy is not applicable:

- *kube-system*
- *gatekeeper-system*
- *azure-arc*
- *nginx*

Enclose each namespace value in double quotes, and separate individual namespace values with a comma.

Note: This parameter is applicable to all the in-built policies.

- m. If you want to assign the *Kubernetes cluster containers should only use allowed capabilities* policy, then specify the values of the following parameters:

- **Allowed capabilities** - Specify the value as `[]`, to ensure that no capability can be added to a container.
 - **Required drop capabilities** - Specify the value as `["ALL"]` to ensure that the container drops all the capabilities.
- n. If you want to assign the *Kubernetes cluster pods and containers should only run with approved user and group IDs* policy, then specify the values of the following parameters:
- **Run as user rule**: Specify the value as `MustRunAsNonRoot`
 - **Allowed user ID ranges**: Specify the value as `{ "ranges": [] }`
 - **Run as group rule**: Specify the value as `MustRunAs`
 - **Allowed group ID ranges**: Specify the value as `{ "ranges": [{ "min": 1, "max": 65535 }] }`
 - **Supplemental group rule**: Specify the value as `MustRunAs`
 - **Allowed supplemental group ID ranges**: Specify the value as `{ "ranges": [{ "min": 1, "max": 65535 }] }`
 - **File system group rule**: Specify the value as `MustRunAs`
 - **Allowed file system group ID ranges**: `{ "ranges": [{ "min": 1, "max": 65535 }] }`
- o. If you want to assign the *Kubernetes cluster pods should only use allowed volume types* policy, then specify the values of the following parameter:
- **Allowed volume types**: Specify the value as `["configMap", "secret", "persistentVolumeClaim", "emptyDir"]`
- p. If you want to assign the *Kubernetes cluster pods should only use approved host network and port range* policy, then specify the values of the following parameters:
- **Allow host network usage**: Specify the value as `false`
 - **Min host port**: Specify the value as `0`
 - **Max host port**: Specify the value as `0`
- q. Click **Review + create** to create the policy.
- r. Repeat *step 4d* to *4q* for each Azure Policy.

2.10.2 Applying Role Based Access Control (RBAC)

This section describes how to apply Role Based Access Control (RBAC).

Important: You require the *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 DSG container.

Important:

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

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

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

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



For example:

```
kubectl create serviceaccount dsg-deployer --namespace dsg
```

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

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

```
kubectl get serviceaccount <Service account name> --namespace <Namespace> -o jsonpath='{.secrets[0].name}'
```

For example:

```
kubectl get serviceaccount dsg-deployer --namespace dsg -o jsonpath='{.secrets[0].name}'
```

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

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

For example:

```
kubectl get secret <token name> --context kubernetes-admin@kubernetes --namespace dsg -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:

```
kubectl config current-context
```

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

```
TOKEN=`echo -n <Token from step 1c> | base64 -d`
```

- Navigate to the directory where you have extracted the Helm charts for deploying the DSG container and run the following command to apply the Protegity RBAC to the Kubernetes cluster.

Important:

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

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

For example:

```
kubectl apply -f pty-rbac.yaml -n dsg
```

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

2.10.3 Setting up the Environment for Deploying the DSG Container

This section describes how to setup the environment for deploying the DSG container on the Kubernetes cluster.

Note:

The following steps are applicable only if Azure Policy and RBAC are applied.

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

► To setup the environment for deploying the DSG container:

1. Perform the following steps if you want to use NGINX as the Ingress Controller. Skip these steps if you want to use the default Ingress controller provided by Azure.

Note: Protegility recommends using the NGINX Ingress Controller.

- a. Run the following command to create a namespace where the NGINX Ingress Controller needs to be deployed.

```
kubectl create namespace <Namespace name>
```

For example:

```
kubectl create namespace nginx
```

- b. Run the following command to add the repository from where the Helm charts for installing the NGINX Ingress Controller need to be fetched.

```
helm repo add stable https://charts.helm.sh/stable
```

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

- c. Run the following command to install the NGINX Ingress Controller using Helm charts.

```
helm install nginx-ingress --namespace nginx ingress-nginx/ingress-inginx \
--set controller.replicaCount=2 \ --set controller.publishService.enabled=true \
\ --set controller.ingressClassResource.name=nginx-dsg \ --
set controller.ingressClassResource.controllerValue="k8s.io/nginx-dsg" \
--set controller.electionID="ingress-controller-leader-nginx-dsg" \ --
set-string controller.service.annotations."service\\.beta\\.kubernetes\\.io/azure-load-
balancer-internal"="true"
```

For example:

```
helm install nginx-ingress --namespace nginx ingress-nginx/ingress-inginx \
--set controller.replicaCount=2 \ --set controller.publishService.enabled=true \
\ --set controller.ingressClassResource.name=nginx-dsg \ --
set controller.ingressClassResource.controllerValue="k8s.io/nginx-dsg" \
--set controller.electionID="ingress-controller-leader-nginx-dsg" \ --
set-string controller.service.annotations."service\\.beta\\.kubernetes\\.io/azure-load-
balancer-internal"="true"
```

The NGINX ingress class name is specified in the `ingress/annotations/kubernetes.io/ingress.class` parameter in the `values.yaml` file.

For more information about the NGINX ingress class name, refer to the section [Ingress Configuration](#).

For more information regarding the configuration parameters for installing the NGINX ingress Helm charts, refer to the [Readme file](#) of the Helm charts.

- d. Check the status of the `nginx-ingress` release and verify that all the deployments are running accurately:

```
kubectl get pods -n <Namespace name>
```

For example:

```
kubectl get pods -n nginx
```

2. If you want to authenticate the communication between the Kubernetes cluster and the Azure Container Registry by using your existing Docker credentials, then perform the following steps.

- a. Run the following command to login to Docker.

```
docker login
```

The login process creates a `config.json` file, which contains an authorization token.

- b. Run the following command to specify a secret for pulling the DSG container images from the Azure Container Registry to Kubernetes.:

```
kubectl create secret generic <Secret_name> --from-file=.dockerconfigjson=/<Path to the config.json file> --type=kubernetes.io/dockerconfigjson --namespace dsg
```

For example:

```
kubectl create secret generic regcrd --from-file=.dockerconfigjson=/home/<user>/.docker/config.json --type=kubernetes.io/dockerconfigjson --namespace dsg
```

For more information about creating secrets, refer to <https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/>.

Note:

Ensure that you specify the name of the secret as the value of the `imagePullSecrets/name` parameter in the `values.yaml` file.

For more information about the `imagePullSecrets` field, refer to the section [Image Pull Secrets](#).

3. If you want to use an Azure Storage Container for storing the policy package instead of the Azure File Share, then modify the `credentials.json` file to update the credentials for the service principal 1.

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

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

4. If you want to use Azure Storage Container for storing the policy package, instead of the Azure File Share, then run the following command to create the *storageAccessSecret* secret, which is used by the DSG container to pull the Policy and CoP packages and use the KMS decryption.

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

```
kubectl create secret generic storageAccessSecret --from-file=get-esa-policy-cop/credentials.json --namespace dsg
```

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 *storageAccessSecretName* field in the *values.yaml* file.

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

5. If you want to use Azure File Share for storing the policy package, instead of the Azure Storage Container, then run the following command to create the *azure-secret* secret, which is used by the DSG container to pull the Policy and CoP packages.

```
kubectl create secret generic azure-secret --from-literal=azurerestorageaccountname='<Storage Account Name>' --from-literal=azurerestorageaccountkey='<Storage Account Key>' --namespace <Namespace>
```

```
kubectl create secret generic azure-secret --from-literal=azurerestorageaccountname='test' --from-literal=azurerestorageaccountkey='<Storage Account Key>' --namespace dsg
```

In the *azurerestorageaccountname* parameter, specify the value of the Azure Storage Account that you have created while creating the Azure File Share.

In the *azurerestorageaccountkey* parameter, specify the value of any one of the storage access keys of the Azure Storage Account.

Note:

Ensure that you specify the name of the secret as the value of the *azureFile/secretName* field in the *pv.yaml* file in [Step 7a](#).

6. Perform the following steps if you want to use an Azure File Share for storing the policy dump instead of the Azure Storage Container.

- Create a file named *pv.yaml* for creating a persistent volume resource.

The following snippet shows the contents of the *pv.yaml* file.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: azurefile
  labels:
    target: policy
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
```

```

storageClassName: azurefile
azureFile:
  secretName: azure-secret
  shareName: policy
  readOnly: false
mountOptions:
- dir_mode=0744
- file_mode=0644
- uid=1000
- gid=1000
#- mfsymlinks
#- nobrl

```

Important:

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

In the *storageClassName* parameter, ensure that you specify *azurefile*, for using the Azure File Share as the persistent volume.

In the *storageClassName* parameter, ensure that you specify *azurefile*, for using the Azure File Share as the persistent volume.

In the *azureFile/secretName* parameter, ensure that you specify name of the secret that you have created while creating the Azure File Share.

For example, specify the value of the *azureFile/secretName* parameter as *azure-secret*.

In the *azureFile/shareName* parameter, ensure that you specify name of the Azure File Share that you have created while creating the Azure File Share.

Ensure that the value specified in the *metadata/labels/target* is also specified as the value of the *selector/matchLabels/target* parameter in the *claim.yaml* file in [step 7c](#).

You can specify the access control settings for the files and directories on the Azure File Share by using the *mountOptions* parameter.

For more information about the mount option parameters, refer to the section [Use Azure Files with Linux](#) in the Azure documentation.

- Run the following command to create the persistent volume resource.

kubectl apply -f pv.yaml

A persistent volume resource is created.

- Create a file named *claim.yaml* for creating a claim on the persistent volume that you have created in [step 6b](#).

The following snippet shows the contents of the *claim.yaml* file.

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: azurefile
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: azurefile
  resources:
    requests:
      storage: 1Gi
  selector:

```

```
matchLabels:
  target: policy
```

Important:

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

In the *storageClassName* parameter, ensure that you specify *azurefile*, for using the Azure File Share as the persistent volume.

In the *selector/matchLabels/target* parameter, ensure that you specify the value that was defined in the *metadata/labels/target* parameter in the *pv.yaml* file in [step 7a](#).

- d. Run the following command to create the persistent volume claim.

```
kubectl apply -f claim.yaml -n <Namespace>
```

For example:

```
kubectl apply -f claim.yaml -n dsg
```

A persistent volume claim is created.

7. Run the following command to create the *cop-secret* secret for the CoP package.

```
kubectl create secret generic cop-secret --from-literal='passphrase=<COPIER_PASS>' --from-literal='salt=<COPIER_SALT>' -n <SAME_AS_OF_CHART_DEPLOYMENT>
```

For example,

```
kubectl create secret generic cop-secret --from-literal='passphrase=Passphrase123@' --from-literal='salt=salt' -n dsg
```

Note:

As with the values of the *<COPIER_PASS>* and *<COPIER_SALT>* parameters, you must specify the passwords that you created the section [Creating a Ruleset](#), when you export the CoP package from the ESA.

8. Run the following command to create the *policy-secret* secret for the policy package.

- a. If you are using PBE encryption, then run the following command to create the secret.

```
kubectl create secret generic cop-secret --from-literal=passphrase='<passphrase for decrypting the policy>' --from-literal='salt=<salt for encrypting the policy>' -n <SAME_AS_OF_CHART_DEPLOYMENT>
```

For example,

```
kubectl create secret generic policy-secret --from-literal='passphrase=Passphrase123@' --from-literal='salt=salt' -n dsg
```

For more information about creating and exporting policy, refer to [Creating and Exporting the Policy](#).

9. Run the following command to create the KMS secret.



- a. If you are using KMS encryption, then run the following command to create the secret.

```
kubectl create secret generic cop-secret --from-literal='privatekey='<Azure Key Identifier>' -n <SAME_AS_OF_CHART_DEPLOYMENT>
```

For example,

```
kubectl create secret generic key-secret --from-literal='privatekey=https://envelope.vault.azure.net/keys/envelope-rsa/dca679fe5e1e44bd8052c5046b9c9ac5' -n dsg
```

10. If you want to authenticate the user using basic authentication, then run the following command to create an LDAP secret, which is used by the DSG container to connect to the LDAP.

```
kubectl create secret generic <LDAP_secret_name> -n <Namespace> --from-literal=self.xml='<perldata> <hashref> <item key="LDAP_URI">ldap://<LDAP IP>:{port}</item> <item key="LDAP_BASEDN">{BASE DN}</item> <item key="LDAP_USERSDN">{USER DN}</item> <item key="LDAP_BINDDN">{BIND DN}</item> <item key="LDAP_BINDPW">{BIND Password}</item> <item key="LDAP_STARTTLS">1</item> </hashref> </perldata>'
```

For example, the following snippet shows the LDAP parameters that are used to connect to the ESA LDAP:

```
kubectl create secret generic cmdldap -n dsg
--from-literal=self.xml='<perldata> <hashref> <item key="LDAP_URI">ldap://10.120.0.58:389</item> <item key="LDAP_BASEDN">dc=esa,dc=protegility,dc=com</item> <item key="LDAP_USERSDN">ou=people,dc=esa,dc=protegility,dc=com</item> <item key="LDAP_BINDDN">cn=ldap_bind_user,ou=people,dc=esa,dc=protegility,dc=com</item> <item key="LDAP_BINDPW">{BIND_Password}</item> <item key="LDAP_STARTTLS">1</item> </hashref> </perldata>'
```

In this command, you need to specify the value for these LDAP parameters:

Table 2-6: LDAP Parameters

Parameter	Description
LDAP_URI	The IP address and port number used to connect to the LDAP server.
LDAP_BASEDN	Distinguished name, which uniquely identifies the base location that is used to search for a user in an LDAP directory. For example, if you are using the ESA LDAP and your domain name is <i>protegility.com</i> , then your Base DN might be: <code>dc=esa,dc=protegility,dc=com</code>
LDAP_USERSDN	Distinguished name, which uniquely identifies the user location that is used to search for a user in an LDAP directory. It is a subset of the Base DN. For example, if you are using the ESA LDAP, your domain name is <i>protegility.com</i> , and your users are part of an organizational unit named <i>people</i> , then your Users DN might be: <code>ou=people, dc=esa,dc=protegility,dc=com</code>
LDAP_BINDDN	Distinguished name of the user that is used to connect to the LDAP server.



Parameter	Description
	For example: <code>cn=ldap_bind_user,ou=people,dc=esa,dc=protegrity,dc=com</code>
LDAP_BINDPW	Password of the user that is used to connect to the LDAP server.
LDAP_STARTTLS	Set the value of this parameter to <i>1</i> to use the TLS protocol for encrypting the communication with the LDAP server.

You need to specify the LDAP secret name as the value of the *IdapXmlSecrets* parameter in the *values.yaml* file in [step 4](#) of the section [Deploying a Release on the Kubernetes Cluster with Azure Policy and RBAC](#).

Note: If you are using the ESA LDAP, then you can obtain the values for the LDAP parameters from the *self.xml* file.

To access the *self.xml* file, perform the following steps.

1. Login to the CLI Manager using the administrator credentials.
2. Navigate to **Administration > OS Console**.
3. Enter the *root* password, and press **OK**.
4. Navigate to the */etc/ksa/conf/* directory to access the *self.xml* file.

Important: If you are using a third-party LDAP server, then ensure that you add the following attribute to the user with the corresponding value in the LDAP server.

Attribute	Value
businessCategory	pty_role:cloud_gateway_auth

For more information about adding an attribute to the LDAP server, refer to the LDAP server documentation.

If you have connected the LDAP server to the ESA, and you add a user who has been assigned a role that includes the *Cloud Gateway Auth* permissions, then the *businessCategory* attribute is automatically added to the LDAP server.

For more information about connecting an LDAP server to the ESA using the CLI Manager, then refer to the section *Managing LDAP* in the [Appliances Overview Guide 9.1.0.0](#).

For more information about authenticating and authorizing external LDAP users using the ESA, refer to the section *Working with Proxy Authentication* in the [Enterprise Security Administrator Guide 9.1.0.0](#).

2.10.4 Deploying a Release on the Kubernetes Cluster with Azure Policy and RBAC

This section describes how to deploy a Release on the Kubernetes cluster by installing the Helm charts.

Important:

The following steps are applicable only when Azure Policy and RBAC are applied.

- To deploy a release on the Kubernetes cluster:

1. Login to the Linux instance, and run the following command to connect your Linux instance to the Kubernetes cluster.

```
az aks get-credentials --resource-group <Name of Resource Group> --name <Name of Kubernetes Cluster>
```

The Linux instance is now connected with the Kubernetes cluster. You can now run commands using the Kubernetes command line interface (*kubectl*) to control the nodes on the Kubernetes cluster.

2. Perform the following steps to setup a Kubernetes context for the *dsg-deployer* service account.

- a. Run the following command to create user credentials in the *kube-config* file, using the token created in *step 1d* of the section [Applying Role Based Access Control \(RBAC\)](#).

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

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

```
kubectl config set-context <Context Name> --cluster=<Name of the Kubernetes Cluster in the kube-config file> --user=<username>
```

- c. Run the following command to set the Kubernetes context to the newly created context in *step 4b*.

```
kubectl config use-context <Context name from step 4b>
```

3. On the Linux instance, navigate to the location where you have extracted the Helm charts.

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

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

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 [Appendix A: DSG Container 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 • <i>supplementalGroups</i> - 1000 <p>Note: If you want to use Azure Storage Container for storing the policy snapshot, then set the value of the <i>fsGroup</i> parameter to a non-root value. 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 three parameters must be <i>1000</i>.</p>

Field	Description
	<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>
pvcName	<p>Specify the name of the Persistent Volume Claim where you want to store the policy package.</p> <p>If you are storing the policy package in Azure File Share, then specify the name of the Persistent Volume Claim that you have specified in the <i>pv.yaml</i> file.</p> <p>Leave the value of this field blank if you want to use the Azure Storage Container for storing the policy package.</p> <p>Important: This field is required if you want to store the immutable policy package in the persistent volume.</p>
privateKeySource	<p>Specify one of the following methods to be used to decrypt the policy package:</p> <ul style="list-style-type: none"> • <i>PKCS5</i> - If you are using Passphrase-Based Encryption for encrypting the policy, then specify <i>PKCS5</i> in the <i>privateKeySource</i> parameter to decrypt the policy package. • <i>AZ_VAULT</i> - If you are using KMS for encrypting the policy, then specify <i>AZ_VAULT</i> in the <i>privateKeySource</i> parameter to decrypt the policy package.
storageAccessSecrets	<p>Specify the Kubernetes secret that contains the credentials for pulling the CoP and the Policy from the Azure Storage Container. The DSG Application container uses the value specified in the <i>storageAccessSecrets</i> parameter to decrypt the Policy package using the KMS.</p>
privateKeySecret	<p>Specify the Kubernetes secret that contains the private key ID for decrypting the policy.</p>
copSecrets	<p>Specify the Kubernetes secret that contains the passphrase and the salt that were used to generate key, which encrypted the CoP. The DSG Application container uses the value specified in the <i>CopSecrets</i> parameter to decrypt the CoP package.</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 DSG Application container uses the value specified in the <i>pbeSecrets</i> parameter to decrypt the policy.</p> <p>By default, the Passphrase-Based Encryption is used to encrypt the policy and CoP package.</p> <p>If you want to use the Azure KMS to decrypt the policy package, then you must comment out this entry or leave the value blank.</p> <p>Note: If you specify a value for <i>pbeSecrets</i> field, then Passphrase-Based Encryption is used instead of the Azure Key Vault.</p>

Field	Description
ldapXmlSecrets	Specify the value of the LDAP secret that you have created in step 8 of the section Setting up the Environment for Deploying the DSG Container .
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> . While staging Release 1, the <i>blue</i> deployment strategy is enabled.
blueDeployment/policy/filepath	<p>Specify the path in the persistent volume or the object store where you have stored the policy package.</p> <p>For example, if you are using Azure File Share as the persistent volume, then you can specify the file path as <i>file://var/data/ptypolicy/pathInVolumeToPolicy</i>.</p> <p>If you have stored the policy package in an Object Store, such as an Azure Storage Container, then you can specify the storage 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>https://<accountName>.blob.core.windows.net/<container name>/<pathToPolicy></i>.</p> <div data-bbox="687 798 816 827" style="background-color: #f0e6f6; padding: 5px;">Important:</div> <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> <div data-bbox="687 988 752 1015" style="background-color: #e0f2e0; padding: 5px;">Note:</div> <p>This value is case-sensitive.</p>
blueDeployment/cop/filepath	<p>Specify the path in the persistent volume or the object store where you have stored the CoP package.</p> <p>For example, if you are using Azure File Share as the persistent volume, then you can specify the file path as <i>file://var/data/ptypolicy/pathInVolumeToCop</i>.</p> <p>If you have stored the IMP in an Object Store, such as an Azure Storage Container, then you can specify the storage 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>https://<accountName>.blob.core.windows.net/<container name>/<pathToCop></i>.</p> <div data-bbox="687 1537 816 1567" style="background-color: #f0e6f6; padding: 5px;">Important:</div> <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> <div data-bbox="687 1719 752 1748" style="background-color: #e0f2e0; padding: 5px;">Note:</div> <p>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.
dsgService/type	Specify whether you want to use one of the following service types:

Field	Description
	<ul style="list-style-type: none"> • <i>LoadBalancer</i> - An external IP is created. You need to send a request to the IP address of the Azure Load Balancer for running the security operations. For more information about sending a request to the Load Balancer, refer to the section Using a load balancer in the section Running Security Operations with Static CoP. • <i>ClusterIP</i> - A cluster IP is created. You need to send a request to the cluster IP, which is a unique IP address assigned to the Kubernetes service, for running the security operations. For more information about sending a request to the cluster IP, refer to the section Using a cluster IP in the section Running Security Operations with Static CoP. <p>This parameter is only applicable if you have set the value of the <i>ingress/enabled</i> parameter to <i>false</i>.</p> <p>If you specify the service type, then you also need to uncomment the annotations for the corresponding service type.</p>
dsgService/healthCheckRestService/scheme	Specify the protocol used for sending a health check service request. By default, the value is set to <i>HTTP</i> .
dsgService/healthCheckRestService/host	Specify the host name that is used to send a health check service request. By default, this value is set to <i>restapi</i> . This default value corresponds to the value specified in the default REST API example ruleset available on the ESA.
dsgService/healthCheckRestService/port	Specify the port number for the health check service. By default, this value is set to <i>&healthCheckPort 8080</i> . This default value corresponds to the value specified in the default REST API example ruleset available on the ESA. This configuration is mandatory as the default port is used for running health checks on the container. Protegility recommends you not to modify the default port number.
dsgService/healthCheckRestService/path	<p>Specify the URI that is used to perform the health check service. By default, this value is set to <i>protect/text</i>.</p> <p>The health check is performed by sending a request to the <i>http://<Health_check_host>/protect/text</i> URI.</p> <p>This URI is defined as part of the <i>Text Protection</i> ruleset, which is the default ruleset available on the ESA. Ensure that you retain this default ruleset on the ESA so that the health check is performed successfully.</p>
dsgService/tunnels/name	Specify a name for the tunnel to distinguish between ports.
dsgService/tunnels/port	Specify the port number on which you want to expose the Kubernetes service externally.
dsgService/tunnels/targetPort	<p>Specify the port number configured while creating a tunnel on the ESA.</p> <p>By default, this value is set to <i>*healthCheckPort</i>, which indicates the value specified in the <i>dsgService/healthCheckRestService/port</i> field.</p> <p>For more information about creating a tunnel and a CoP ruleset, refer to the section Creating a Ruleset.</p>
ingress/enabled	Specify this value as <i>true</i> to enable the ingress resource in Kubernetes. The ingress resource acts as a load balancer. It enables the simultaneous deployment of <i>blue and green</i> environments.

Field	Description
	<p><i>green</i> strategy by exposing both the deployments on the same external IP address, but to different URL paths.</p> <p>Important: If you want to configure the DSG container for SMTP and SFTP services, then you must specify this value as <i>false</i> to disable the ingress resource in Kubernetes.</p>
ingress/annotations/kubernetes.io/ingress.class	<p>Specify the value of the external ingress controller, if any. This is the same value that you have specified for the controller.ingressClass parameter in step 2c of the section Setting up the Environment for Deploying the DSG Container.</p> <p>For example, specify, <i>nginx-dsg</i> if you want to use the NGINX Ingress Controller.</p> <p>Leave this field as blank if you want to use the default ingress controller given by the Cloud provider.</p>
ingress/annotations/nginx.ingress.kubernetes.io/ssl-passthrough	<p>Specify whether you want to use the NGINX Ingress Controller as an SSL passthrough, to ensure end-to-end encrypted communication between the REST API client and the DSG pods. Set the value of this field to <i>true</i>.</p>
ingress/hosts/host	<p>Specify the hostname of the ingress resource.</p> <p>If you are using the HTTPS channel for communication, then ensure that the hostname is the common name that you specified while creating the server certificate. Ensure that you specify separate subdomain names for the production and staging environment.</p> <p>For example, if you have specified the common name in the server certificates as <i>*.example.com</i>, where example.com is the domain name, then ensure that you use the following hostnames for the production and staging environments:</p> <ul style="list-style-type: none"> • For production environment: <i>prod.example.com</i> • For staging environment: <i>staging.example.com</i> <p>In this example, <i>prod</i> and <i>staging</i> are the subdomains under the example.com domain.</p> <p>For more information about creating a server certificate, refer to the section Creating Certificates and Keys for TLS Authentication.</p>
ingress/hosts/httpPaths/port	<p>Specify the same value that you have provided in the <i>dsgService/tunnels/port</i> field.</p>
ingress/hosts/httpPaths/prod	<p>If you are using the NGINX Ingress Controller as an SSL passthrough, then you must specify the value of this field as <i>/</i>. This ensures that the data traffic from the REST API client is redirected to the required production or staging URL based on the <i>hostname</i> value.</p>
ingress/hosts/httpPaths/staging	<p>If you are using the NGINX Ingress Controller as an SSL passthrough, then you must specify the value of this field as <i>/</i>. This ensures that the data traffic from the REST API client is redirected to the required production or staging URL based on the <i>hostname</i> value.</p>

- Run the following command to deploy the DSG on the Kubernetes cluster.

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

For example:

```
helm install dsg-demo --namespace dsg dsg
```

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.

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

For example:

```
kubectl get pods -n dsg
```

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

```
kubectl get ingress -n <namespace>
```

For example:

```
kubectl get ingress -n dsg
```

This command is used to obtain the IP address of the ingress.

2.10.5 Deploying a Release on the Kubernetes Cluster without Azure Policy and RBAC

This section describes how to deploy a Release on the Kubernetes cluster by installing the Helm charts.

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

► To deploy a release on the Kubernetes cluster:

1. Perform the following steps if you want to use NGINX as the Ingress Controller. Skip these steps if you want to use the default Ingress controller provided by Azure.

Note: Protegrity recommends using the NGINX Ingress Controller.

- a. Run the following command to create a namespace where the NGINX Ingress Controller needs to be deployed.

```
kubectl create namespace <Namespace name>
```

For example:

```
kubectl create namespace nginx
```

- b. Run the following command to add the repository from where the Helm charts for installing the NGINX Ingress Controller need to be fetched.

```
helm repo add stable https://charts.helm.sh/stable
```

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

- c. Run the following command to install the NGINX Ingress Controller using Helm charts.

```
helm install nginx-ingress --namespace nginx ingress-nginx/ingress-nginx \
--set controller.replicaCount=2 \ --set controller.publishService.enabled=true
\ --set controller.ingressClassResource.name=nginx-dsg \ --
set controller.ingressClassResource.controllerValue="k8s.io/nginx-dsg" \
--set controller.electionID="ingress-controller-leader-nginx-dsg" \ --
set-string controller.service.annotations."service\\.beta\\.kubernetes\\.io/azure-load-
balancer-internal"="true"
```

For example:

```
helm install nginx-ingress --namespace nginx ingress-nginx/ingress-nginx \
--set controller.replicaCount=2 \ --set controller.publishService.enabled=true
\ --set controller.ingressClassResource.name=nginx-dsg \ --
set controller.ingressClassResource.controllerValue="k8s.io/nginx-dsg" \
--set controller.electionID="ingress-controller-leader-nginx-dsg" \ --
set-string controller.service.annotations."service\\.beta\\.kubernetes\\.io/azure-load-
balancer-internal"="true"
```

The NGINX ingress class name is specified in the *ingress/annotations/kubernetes.io/ingress.class* parameter in the *values.yaml* file.

For more information about the NGINX ingress class name, refer to the section [Ingress Configuration](#).

For more information regarding the configuration parameters for installing the NGINX ingress Helm charts, refer to the [Readme file](#) of the Helm charts.

- d. Check the status of the ***nginx-ingress*** release and verify that all the deployments are running accurately.

```
kubectl get pods -n <Namespace name>
```

For example:

```
kubectl get pods -n nginx
```

2. Perform the following steps if you want to use an Azure File Share for storing the policy dump instead of the Azure Storage Container.

- a. Create a file named *pv.yaml* for creating a persistent volume resource.

The following snippet shows the contents of the *pv.yaml* file.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: azurefile
  labels:
    target: policy
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  storageClassName: azurefile
  azureFile:
    secretName: azure-secret
    shareName: policy
    readOnly: false
    mountOptions:
      - dir_mode=0744
      - file_mode=0644
      - uid=1000
      - gid=1000
```

```
#- mfsymlinks  
#- nobrl
```

Important:

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

In the *storageClassName* parameter, ensure that you specify *azurefile*, for using the Azure File Share as the persistent volume.

In the *storageClassName* parameter, ensure that you specify *azurefile*, for using the Azure File Share as the persistent volume.

In the *azureFile/secretName* parameter, ensure that you specify name of the secret that you have created in [step 6](#).

For example, specify the value of the *azureFile/secretName* parameter as *azure-secret*.

In the *azureFile/shareName* parameter, ensure that you specify name of the Azure File Share that you have created while creating the Azure File Share.

Ensure that the value specified in the *metadata/labels/target* is also specified as the value of the *selector/matchLabels/target* parameter in the *claim.yaml* file in [step 7c](#).

You can specify the access control settings for the files and directories on the Azure File Share by using the *mountOptions* parameter.

For more information about the mount option parameters, refer to the section [Use Azure Files with Linux](#) in the Azure documentation.

- b. Run the following command to create the persistent volume resource.

```
kubectl apply -f pv.yaml
```

A persistent volume resource is created.

- c. Create a file named *claim.yaml* for creating a claim on the persistent volume that you have created in [step 6b](#).

The following snippet shows the contents of the *claim.yaml* file.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: azurefile
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: azurefile
  resources:
    requests:
      storage: 1Gi
  selector:
    matchLabels:
      target: policy
```

Important:

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

In the *storageClassName* parameter, ensure that you specify *azurefile*, for using the Azure File Share as the persistent volume.

In the *selector/matchLabels/target* parameter, ensure that you specify the value that was defined in the *metadata/labels/target* parameter in the *pv.yaml* file in [step 7a](#).

- d. Run the following command to create the persistent volume claim.

```
kubectl apply -f claim.yaml -n <Namespace>
```

For example:

```
kubectl apply -f claim.yaml -n dsg
```

A persistent volume claim is created.

3. If you want to use an Azure Storage Container for storing the policy package instead of the Azure File Share, then modify the *credentials.json* file to update the credentials for the service principal 1.

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

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

4. If you want to use Azure Storage Container for storing the policy package, instead of the Azure File Share, then run the following command to create the *storageAccessSecret* secret, which is used by the DSG container to pull the Policy and CoP packages and use the KMS decryption.

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

```
kubectl create secret generic storageAccessSecret --from-file=get-esa-policy-cop/  
credentials.json --namespace dsg
```

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 *storageAccessSecretName* field in the *values.yaml* file.

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

5. If you want to use Azure File Share for storing the policy package, instead of the Azure Storage Container, then run the following command to create the *azure-secret* secret, which is used by the DSG container to pull the Policy and CoP packages.

```
kubectl create secret generic azure-secret --from-  
literal=azurerestorageaccountname='<Storage Account Name>' --from-  
literal=azurerestorageaccountkey='<Storage Account Key>' --namespace <Namespace>
```

```
kubectl create secret generic azure-  
secret --from-literal=azurerestorageaccountname='test' --from-  
literal=azurerestorageaccountkey='<Storage Account Key>' --namespace dsg
```

In the *azurerestorageaccountname* parameter, specify the value of the Azure Storage Account that you have created while creating the Azure File Share.

In the `azurorestorageaccountkey` parameter, specify the value of any one of the storage access keys for the Azure Storage Account.

Note:

Ensure that you specify the name of the secret as the value of the `azureFile/secretName` field in the `pv.yaml` file in [step 7a](#).

- Run the following command to create the `cop-secret` secret for the CoP package.

```
kubectl create secret generic cop-secret --from-literal='passphrase=<COP_PASS>' --from-literal='salt=<COP_SALT>' -n <SAME_AS_OF_CHART_DEPLOYMENT>
```

For example,

```
kubectl create secret generic cop-secret --from-literal='passphrase=Passphrase123@' --from-literal='salt=salt' -n dsg
```

Note:

As the value of the `<COP_PASS>` and `<COP_SALT>` parameters, you must specify the passwords that you created the section [Creating a Ruleset](#), when you export the CoP package from the ESA.

- Run the following command to create the `policy-secret` secret for the policy package.

- If you are using PBE encryption, then run the following command to create the secret.

```
kubectl create secret generic cop-secret --from-literal=passphrase='<Passphrase used to encrypt the policy>' --from-literal='salt=<salt for encrypting the policy>' -n <SAME_AS_OF_CHART_DEPLOYMENT>
```

For example,

```
kubectl create secret generic policy-secret --from-literal='passphrase=Passphrase123@' --from-literal='salt=salt' -n dsg
```

- If you are using KMS encryption, then run the following command to create the secret.

```
kubectl create secret generic cop-secret --from-literal='privatekey='<Azure Key Identifier>' -n <SAME_AS_OF_CHART_DEPLOYMENT>
```

For example,

```
kubectl create secret generic key-secret --from-literal='privatekey=https://envelope.vault.azure.net/keys/envelope-rsa/dca679fe5e1e44bd8052c5046b9c9ac5'
```

For more information about creating and exporting policy, refer to the section [Creating and Exporting the Policy](#).

- If you want to authenticate the communication between the Kubernetes cluster and the Azure Container Registry by using your existing Docker credentials, then perform the following steps.



- Run the following command to login to Docker.

```
docker login
```

The login process creates a *config.json* file, which contains an authorization token.

- Run the following command to specify a secret for pulling the DSG container images from the Azure Container Registry to Kubernetes.:

```
kubectl create secret generic <Secret_name> --from-file=.dockerconfigjson=/<Path to the config.json file> --type=kubernetes.io/dockerconfigjson --namespace dsg
```

For example:

```
kubectl create secret generic regcrd --from-file=.dockerconfigjson=/home/<user>/.docker/config.json --type=kubernetes.io/dockerconfigjson --namespace dsg
```

For more information about creating secrets, refer to <https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/>.

Note:

Ensure that you specify the name of the secret as the value of the *imagePullSecrets/name* parameter in the *values.yaml* file.

For more information about the *imagePullSecrets* field, refer to the section [Image Pull Secrets](#).

- If you want to authenticate the user using basic authentication, then run the following command to create an LDAP secret, which is used by the DSG container to connect to the LDAP.

```
kubectl create secret generic <LDAP_secret_name> -n <Namespace> --from-literal=self.xml='<perldata> <hashref> <item key="LDAP_URI">ldap://<LDAP IP>:{port}</item> <item key="LDAP_BASEDN">{BASE DN}</item> <item key="LDAP_USERSDN">{USER DN}</item> <item key="LDAP_BINDDN">{BIND DN}</item> <item key="LDAP_BINDPW">{BIND Password}</item> <item key="LDAP_STARTTLS">1</item> </hashref> </perldata>'
```

For example, the following snippet shows the LDAP parameters that are used to connect to the ESA LDAP:

```
kubectl create secret generic cmdldap -n dsg --from-literal=self.xml='<perldata> <hashref> <item key="LDAP_URI">ldap://10.120.0.58:389</item> <item key="LDAP_BASEDN">dc=esa,dc=protegility,dc=com</item> <item key="LDAP_USERSDN">ou=people,dc=esa,dc=protegility,dc=com</item> <item key="LDAP_BINDDN">cn=ldap_bind_user,ou=people,dc=esa,dc=protegility,dc=com</item> <item key="LDAP_BINDPW">{BIND_Password}</item> <item key="LDAP_STARTTLS">1</item> </hashref> </perldata>'
```

In this command, you need to specify the value for these LDAP parameters:

Table 2-7: LDAP Parameters

Parameter	Description
LDAP_URI	The IP address and port number used to connect to the LDAP server.
LDAP_BASEDN	Distinguished name, which uniquely identifies the base location that is used to search for a user in an LDAP directory.

Parameter	Description
	<p>For example, if you are using the ESA LDAP and your domain name is <i>protegility.com</i>, then your Base DN might be:</p> <pre>dc=esa,dc=protegility,dc=com</pre>
LDAP_USERSDN	<p>Distinguished name, which uniquely identifies the user location that is used to search for a user in an LDAP directory. It is a subset of the Base DN.</p> <p>For example, if you are using the ESA LDAP, your domain name is <i>protegility.com</i>, and your users are part of an organizational unit named <i>people</i>, then your Users DN might be:</p> <pre>ou=people, dc=esa,dc=protegility,dc=com</pre>
LDAP_BINDDN	<p>Distinguished name of the user that is used to connect to the LDAP server.</p> <p>For example:</p> <pre>cn=ldap_bind_user,ou=people,dc=esa,dc=protegility,dc=com</pre>
LDAP_BINDPW	Password of the user that is used to connect to the LDAP server.
LDAP_STARTTLS	Set the value of this parameter to <i>1</i> to use the TLS protocol for encrypting the communication with the LDAP server.

You need to specify the LDAP secret name as the value of the *ldapXmlSecrets* parameter in the *values.yaml* file in [step 6](#).

Note: If you are using the ESA LDAP, then you can obtain the values for the LDAP parameters from the *self.xml* file.

To access the *self.xml* file, perform the following steps.

1. Login to the CLI Manager using the administrator credentials.
2. Navigate to **Administration > OS Console**.
3. Enter the *root* password, and press **OK**.
4. Navigate to the */etc/ksa/conf/* directory to access the *self.xml* file.

Important: If you are using a third-party LDAP server, then ensure that you add the following attribute to the user with corresponding value in the LDAP server.

Attribute	Value
businessCategory	pty_role:cloud_gateway_auth

For more information about adding an attribute to the LDAP server, refer to the LDAP server documentation.

If you have connected the LDAP server to the ESA, and you add a user who has been assigned a role that includes the *Cloud Gateway Auth* permissions, then the *businessCategory* attribute is automatically added to the LDAP server.

For more information about connecting an LDAP server to the ESA using the CLI Manager, then refer to the section *Managing LDAP* in the [Appliance Overview Guide 9.1.0.5](#).

For more information about authenticating and authorizing external LDAP users from the ESA, refer to the section *Working with Proxy Authentication* in the [Enterprise Security Administrator Guide 9.1.0.05](#).

10. On the Linux instance, navigate to the location where you have extracted the Helm charts.

For more information about the extracted Helm charts, refer to the [step 10](#) of the section [Initializing the Linux Instance](#).

The `values.yaml` file contains the default configuration values for deploying the DSG application on the Kubernetes cluster.

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

11. 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 A: DSG Container 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"> • <code>runAsUser</code> - 1000 • <code>runAsGroup</code> - 1000 • <code>fsGroup</code> - 1000 • <code>supplementalGroups</code> - 1000 <p>Note: If you want to use Azure Storage Container for storing the policy snapshot, then set the value of the <code>fsGroup</code> parameter to a non-root value. For example, you can set the value of the <code>fsGroup</code> parameter to any value between <code>1</code> and <code>65534</code>.</p> <p>Note: Ensure that at any given point of time, the value of at least one of the three parameters must be <code>1000</code>. This ensures that the pod has the required permissions to access the <code>pty</code> 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>
pvcName	<p>Specify the name of the Persistent Volume Claim where you want to store the policy package.</p> <p>If you are storing the policy package in Azure File Share, then specify the name of the Persistent Volume Claim that you have specified in the <code>pv.yaml</code> file.</p> <p>Leave the value of this field blank if you want to use the Azure Storage Container for storing the policy package.</p> <p>Important:</p>

Field	Description
	This field is required if you want to store the immutable policy package in the persistent volume.
privateKeySource	<p>Specify one of the following methods to be used to decrypt the policy package:</p> <ul style="list-style-type: none"> • <i>PKCS5</i> - If you are using Passphrase-Based Encryption for encrypting the policy, then specify <i>PKCS5</i> in the <i>privateKeySource</i> parameter to decrypt the policy package. • <i>AZ_VAULT</i> - If you are using KMS for encrypting the policy, then specify <i>AZ_VAULT</i> in the <i>privateKeySource</i> parameter to decrypt the policy package.
storageAccessSecrets	Specify the Kubernetes secret that contains the credentials for pulling the CoP and the Policy from the Azure Storage Container. The DSG Application container uses the value specified in the <i>storageAccessSecrets</i> parameter to decrypt the Policy package using the KMS.
privateKeySecret	Specify the Kubernetes secret that contains the private key ID for decrypting the policy.
copSecrets	Specify the Kubernetes secret that contains the passphrase and the salt that were used to generate key, which encrypted the CoP. The DSG Application container uses the value specified in the <i>CopSecrets</i> parameter to decrypt the CoP package.
pbeSecrets	<p>Specify the Kubernetes secret that contains the passphrase and the salt that were used to generate key, which encrypted the policy. The DSG Application container uses the value specified in the <i>pbeSecrets</i> parameter to decrypt the policy.</p> <p>By default, the Passphrase-Based Encryption is used to encrypt the policy and CoP package.</p> <p>If you want to use the Azure KMS to decrypt the policy package, then you must comment out this entry or leave the value blank.</p> <p>Note: If you specify a value for <i>pbeSecrets</i> field, then Passphrase-Based Encryption is used instead of the Azure Key Vault.</p>
ldapXmlSecrets	Specify the value of the LDAP secret that you have created in step 8 of the section <i>Setting up the Environment for Deploying the DSG Container</i> .
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> . While staging Release 1, the <i>blue</i> deployment strategy is enabled.
blueDeployment/policy/filepath	<p>Specify the path in the persistent volume or the object store where you have stored the policy package.</p> <p>For example, if you are using Azure File Share as the persistent volume, then you can specify the file path as <i>file://var/data/ptypolicy/pathInVolumeToPolicy</i>.</p> <p>If you have stored the policy package in an Object Store, such as an Azure Storage Container, then you can specify the storage 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>https://<accountName>.blob.core.windows.net/<container name>/<pathToPolicy></i>.</p> <p>Important:</p>

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>
blueDeployment/cop/filepath	<p>Specify the path in the persistent volume or the object store where you have stored the CoP package.</p> <p>For example, if you are using Azure File Share as the persistent volume, then you can specify the file path as <i>file://var/data/ptypolicy/pathInVolumeToCop</i>.</p> <p>If you have stored the IMP in an Object Store, such as an Azure Storage Container, then you can specify the storage 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>https://<accountName>.blob.core.windows.net/<container name>/<pathToCop></i>.</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>
dsgService/type	<p>Specify whether you want to use one of the following service types:</p> <ul style="list-style-type: none"> • <i>LoadBalancer</i> - An external IP is created. You need to send a request to the IP address of the Azure Load Balancer for running the security operations. For more information about sending a request to the Load Balancer, refer to the section Using a load balancer in the section Running Security Operations with Static CoP. • <i>ClusterIP</i> - A cluster IP is created. You need to send a request to the cluster IP, which is a unique IP address assigned to the Kubernetes service, for running the security operations. For more information about sending a request to the cluster IP, refer to the section Using a cluster IP in the section Running Security Operations with Static CoP. <p>This parameter is only applicable if you have set the value of the <i>ingress/enabled</i> parameter to <i>false</i>.</p> <p>If you specify the service type, then you also need to uncomment the annotations for the corresponding service type.</p>
dsgService/healthCheckRestService/scheme	<p>Specify the protocol used for sending a health check service request. By default, the value is set to <i>HTTP</i>.</p>

Field	Description
dsgService/healthCheckRestService/host	Specify the host name that is used to send a health check service request. By default, this value is set to <i>restapi</i> . This default value corresponds to the value specified in the default REST API example ruleset available on the ESA.
dsgService/healthCheckRestService/port	Specify the port number for the health check service. By default, this value is set to <i>&healthCheckPort 8080</i> . This default value corresponds to the value specified in the default REST API example ruleset available on the ESA. This configuration is mandatory as the default port is used for running health checks on the container. Protegility recommends you not to modify the default port number.
dsgService/healthCheckRestService/path	<p>Specify the URI that is used to perform the health check service. By default, this value is set to <i>protect/text</i>.</p> <p>The health check is performed by sending a request to the <i>http://<Health_check_host>/protect/text</i> URI.</p> <p>This URI is defined as part of the <i>Text Protection</i> ruleset, which is the default ruleset available on the ESA. Ensure that you retain this default ruleset on the ESA so that the health check is performed successfully.</p>
dsgService/tunnels/name	Specify a name for the tunnel to distinguish between ports.
dsgService/tunnels/port	Specify the port number on which you want to expose the Kubernetes service externally.
dsgService/tunnels/targetPort	<p>Specify the port number configured while creating a tunnel on the ESA.</p> <p>By default, this value is set to <i>*healthCheckPort</i>, which indicates the value specified in the <i>dsgService/healthCheckRestService/port</i> field.</p> <p>For more information about creating a tunnel and a CoP ruleset, refer to the section Creating a Ruleset.</p>
ingress/enabled	<p>Specify this value as <i>true</i> to enable the ingress resource in Kubernetes. The ingress resource acts as a load balancer. It enables the simultaneous deployment of <i>blue and green</i> strategy by exposing both the deployments on the same external IP address, but to different URL paths.</p> <p>Important: If you want to configure the DSG container for SMTP and SFTP services, then you must specify this value as <i>false</i> to disable the ingress resource in Kubernetes.</p>
ingress/annotations/kubernetes.io/ingress.class	<p>Specify the value of the external ingress controller, if any. This is the same value that you have specified for the controller.ingressClass parameter in step 2c of the section Setting up the Environment for Deploying the DSG Container.</p> <p>For example, specify, <i>nginx-dsg</i> if you want to use the NGINX Ingress Controller.</p> <p>Leave this field as blank if you want to use the default ingress controller given by the Cloud provider.</p>
ingress/annotations/nginx.ingress.kubernetes.io/ssl-passthrough	Specify whether you want to use the NGINX Ingress Controller as an SSL passthrough, to ensure end-to-end encrypted communication between the REST API client and the DSG pods. Set the value of this field to <i>true</i> .
ingress/hosts/host	<p>Specify the hostname of the ingress resource.</p> <p>If you are using the HTTPS channel for communication, then ensure that the hostname is the common name that you specified while creating the server certificate. Ensure that you specify separate subdomain names for the production and staging environment.</p>

Field	Description
	<p>For example, if you have specified the common name in the server certificates as <code>*.example.com</code>, where <code>example.com</code> is the domain name, then ensure that you use the following hostnames for the production and staging environments:</p> <ul style="list-style-type: none"> • For production environment: <code>prod.example.com</code> • For staging environment: <code>staging.example.com</code> <p>In this example, <code>prod</code> and <code>staging</code> are the subdomains under the <code>example.com</code> domain.</p> <p>For more information about creating a server certificate, refer to the section Creating Certificates and Keys for TLS Authentication.</p>
<code>ingress/hosts/httpPaths/port</code>	Specify the same value that you have provided in the <code>dsgService/tunnels/port</code> field.
<code>ingress/hosts/httpPaths/prod</code>	If you are using the NGINX Ingress Controller as an SSL passthrough, then you must specify the value of this field as <code>/</code> . This ensures that the data traffic from the REST API client is redirected to the required production or staging URL based on the <code>hostname</code> value.
<code>ingress/hosts/httpPaths/staging</code>	If you are using the NGINX Ingress Controller as an SSL passthrough, then you must specify the value of this field as <code>/</code> . This ensures that the data traffic from the REST API client is redirected to the required production or staging URL based on the <code>hostname</code> value.

12. Run the following command to deploy DSG on the Kubernetes cluster:

```
helm install <Release Name> <Location of the directory that contains the Helm charts>
--namespace <Namespace>
```

For example:

```
helm install dsg-demo dsg --namespace dsg
```

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

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

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

For example:

```
kubectl get pods -n dsg
```

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

```
kubectl get ingress -n <namespace>
```

For example:

```
kubectl get ingress -n dsg
```

This command is used to obtain the IP address of the ingress.

2.11 Upgrading the Helm Charts

This section describes the steps for upgrading the Helm charts.



Protegility 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. Modify the policy or CoP as required.

Similarly, create a new CoP or modify an existing CoP on the ESA.

For more information about creating a Ruleset, refer to the section [Creating and Uploading the CoP to Azure Storage Container](#).

2. On the Linux instance, navigate to the location where you have extracted the Helm charts for installing the DSG application.

For more information about the extracted Helm charts, refer to the [step 3](#) of the section [Downloading the Installation Package](#).

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

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

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

For more information about Helm charts and their default values, refer to the section [Appendix A: DSG Container 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 • <i>supplementalGroups</i> - 1000 <p>Note: If you want to use Azure Storage Container for storing the policy snapshot, then set the value of the <i>fsGroup</i> parameter to a non-root value. 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 three parameters must be <i>1000</i>.</p>

Field	Description
	<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>
pvcName	<p>Specify the name of the Persistent Volume Claim where you want to store the policy package.</p> <p>If you are storing the policy package in Azure File Share, then specify the name of the Persistent Volume Claim that you have specified in the <i>pv.yaml</i> file.</p> <p>Leave the value of this field blank if you want to use the Azure Storage Container for storing the policy package.</p> <p>Important: This field is required if you want to store the immutable policy package in the persistent volume.</p>
privateKeySource	<p>Specify one of the following methods to be used to decrypt the policy package:</p> <ul style="list-style-type: none"> • <i>PKCS5</i> - If you are using Passphrase-Based Encryption for encrypting the policy, then specify <i>PKCS5</i> in the <i>privateKeySource</i> parameter to decrypt the policy package. • <i>AZ_VAULT</i> - If you are using KMS for encrypting the policy, then specify <i>AZ_VAULT</i> in the <i>privateKeySource</i> parameter to decrypt the policy package.
storageAccessSecrets	<p>Specify the Kubernetes secret that contains the credentials for pulling the CoP and the Policy from the Azure Storage Container. The DSG Application container uses the value specified in the <i>storageAccessSecrets</i> parameter to decrypt the Policy package using the KMS.</p>
privateKeySecret	<p>Specify the Kubernetes secret that contains the private key ID for decrypting the policy.</p>
copSecrets	<p>Specify the Kubernetes secret that contains the passphrase and the salt that were used to generate key, which encrypted the CoP. The DSG Application container uses the value specified in the <i>CopSecrets</i> parameter to decrypt the CoP package.</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 DSG Application container uses the value specified in the <i>pbeSecrets</i> parameter to decrypt the policy.</p> <p>By default, the Passphrase-Based Encryption is used to encrypt the policy and CoP package.</p> <p>If you want to use the Azure KMS to decrypt the policy package, then you must comment out this entry or leave the value blank.</p> <p>Note: If you specify a value for <i>pbeSecrets</i> field, then Passphrase-Based Encryption is used instead of the Azure Key Vault.</p>

Field	Description
ldapXmlSecrets	Specify the value of the LDAP secret that you have created in step 8 of the section Setting up the Environment for Deploying the DSG Container .
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> . While staging Release 1, the <i>blue</i> deployment strategy is enabled.
blueDeployment/policy/filepath	<p>Specify the path in the persistent volume or the object store where you have stored the policy package.</p> <p>For example, if you are using Azure File Share as the persistent volume, then you can specify the file path as <i>file://var/data/ptypolicy/pathInVolumeToPolicy</i>.</p> <p>If you have stored the policy package in an Object Store, such as an Azure Storage Container, then you can specify the storage 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>https://<accountName>.blob.core.windows.net/<container name>/<pathToPolicy></i>.</p> <div data-bbox="687 798 816 827" style="background-color: #f0e6e6; padding: 5px;">Important:</div> <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> <div data-bbox="687 988 752 1015" style="background-color: #d9e1f2; padding: 5px;">Note:</div> <p>This value is case-sensitive.</p>
blueDeployment/cop/filepath	<p>Specify the path in the persistent volume or the object store where you have stored the CoP package.</p> <p>For example, if you are using Azure File Share as the persistent volume, then you can specify the file path as <i>file://var/data/ptypolicy/pathInVolumeToCop</i>.</p> <p>If you have stored the IMP in an Object Store, such as an Azure Storage Container, then you can specify the storage 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>https://<accountName>.blob.core.windows.net/<container name>/<pathToCop></i>.</p> <div data-bbox="687 1537 816 1567" style="background-color: #f0e6e6; padding: 5px;">Important:</div> <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> <div data-bbox="687 1719 752 1748" style="background-color: #d9e1f2; padding: 5px;">Note:</div> <p>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.
dsgService/type	Specify whether you want to use one of the following service types:

Field	Description
	<ul style="list-style-type: none"> • <i>LoadBalancer</i> - An external IP is created. You need to send a request to the IP address of the Azure Load Balancer for running the security operations. For more information about sending a request to the Load Balancer, refer to the section Using a load balancer in the section Running Security Operations with Static CoP. • <i>ClusterIP</i> - A cluster IP is created. You need to send a request to the cluster IP, which is a unique IP address assigned to the Kubernetes service, for running the security operations. For more information about sending a request to the cluster IP, refer to the section Using a cluster IP in the section Running Security Operations with Static CoP. <p>This parameter is only applicable if you have set the value of the <i>ingress/enabled</i> parameter to <i>false</i>.</p> <p>If you specify the service type, then you also need to uncomment the annotations for the corresponding service type.</p>
dsgService/healthCheckRestService/scheme	Specify the protocol used for sending a health check service request. By default, the value is set to <i>HTTP</i> .
dsgService/healthCheckRestService/host	Specify the host name that is used to send a health check service request. By default, this value is set to <i>restapi</i> . This default value corresponds to the value specified in the default REST API example ruleset available on the ESA.
dsgService/healthCheckRestService/port	Specify the port number for the health check service. By default, this value is set to <i>&healthCheckPort 8080</i> . This default value corresponds to the value specified in the default REST API example ruleset available on the ESA. This configuration is mandatory as the default port is used for running health checks on the container. Protegility recommends you not to modify the default port number.
dsgService/healthCheckRestService/path	<p>Specify the URI that is used to perform the health check service. By default, this value is set to <i>protect/text</i>.</p> <p>The health check is performed by sending a request to the <i>http://<Health_check_host>/protect/text</i> URI.</p> <p>This URI is defined as part of the <i>Text Protection</i> ruleset, which is the default ruleset available on the ESA. Ensure that you retain this default ruleset on the ESA so that the health check is performed successfully.</p>
dsgService/tunnels/name	Specify a name for the tunnel to distinguish between ports.
dsgService/tunnels/port	Specify the port number on which you want to expose the Kubernetes service externally.
dsgService/tunnels/targetPort	<p>Specify the port number configured while creating a tunnel on the ESA.</p> <p>By default, this value is set to <i>*healthCheckPort</i>, which indicates the value specified in the <i>dsgService/healthCheckRestService/port</i> field.</p> <p>For more information about creating a tunnel and a CoP ruleset, refer to the section Creating a Ruleset.</p>
ingress/enabled	Specify this value as <i>true</i> to enable the ingress resource in Kubernetes. The ingress resource acts as a load balancer. It enables the simultaneous deployment of <i>blue and green</i> environments.

Field	Description
	<p><i>green</i> strategy by exposing both the deployments on the same external IP address, but to different URL paths.</p> <p>Important: If you want to configure the DSG container for SMTP and SFTP services, then you must specify this value as <i>false</i> to disable the ingress resource in Kubernetes.</p>
ingress/annotations/kubernetes.io/ingress.class	<p>Specify the value of the external ingress controller, if any. This is the same value that you have specified for the controller.ingressClass parameter in step 2c of the section Setting up the Environment for Deploying the DSG Container.</p> <p>For example, specify, <i>nginx-dsg</i> if you want to use the NGINX Ingress Controller.</p> <p>Leave this field as blank if you want to use the default ingress controller given by the Cloud provider.</p>
ingress/annotations/nginx.ingress.kubernetes.io/ssl-passthrough	<p>Specify whether you want to use the NGINX Ingress Controller as an SSL passthrough, to ensure end-to-end encrypted communication between the REST API client and the DSG pods. Set the value of this field to <i>true</i>.</p>
ingress/hosts/host	<p>Specify the hostname of the ingress resource.</p> <p>If you are using the HTTPS channel for communication, then ensure that the hostname is the common name that you specified while creating the server certificate. Ensure that you specify separate subdomain names for the production and staging environment.</p> <p>For example, if you have specified the common name in the server certificates as <i>*.example.com</i>, where example.com is the domain name, then ensure that you use the following hostnames for the production and staging environments:</p> <ul style="list-style-type: none"> • For production environment: <i>prod.example.com</i> • For staging environment: <i>staging.example.com</i> <p>In this example, <i>prod</i> and <i>staging</i> are the subdomains under the example.com domain.</p> <p>For more information about creating a server certificate, refer to the section Creating Certificates and Keys for TLS Authentication.</p>
ingress/hosts/httpPaths/port	<p>Specify the same value that you have provided in the <i>dsgService/tunnels/port</i> field.</p>
ingress/hosts/httpPaths/prod	<p>If you are using the NGINX Ingress Controller as an SSL passthrough, then you must specify the value of this field as <i>/</i>. This ensures that the data traffic from the REST API client is redirected to the required production or staging URL based on the <i>hostname</i> value.</p>
ingress/hosts/httpPaths/staging	<p>If you are using the NGINX Ingress Controller as an SSL passthrough, then you must specify the value of this field as <i>/</i>. This ensures that the data traffic from the REST API client is redirected to the required production or staging URL based on the <i>hostname</i> value.</p>

- Run the following command to upgrade the Helm charts:

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

For example:

```
helm upgrade dsg-demo dsg --namespace dsg
```

Using this configuration ensures that the DSG application is deployed with the updated policy and CoP snapshot on the staging environment.

5. 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 DSG containers with the updated configuration.

6. 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.

7. Run the following command to upgrade the Helm charts:

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

For example:

```
helm upgrade dsg-demo dsg --namespace dsg
```

Using this configuration ensures that the DSG application is deployed with the updated policy and CoP snapshot on the staging environment.

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

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

This will shutdown 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 consuming resources.

9. Run the following command to upgrade the Helm charts:

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

For example:

```
helm upgrade dsg-demo dsg --namespace dsg
```

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

If you want to modify any policy or CoP for subsequent releases, you need to repeat steps 1 to 9. 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

Protecting Data using the DSG Container Deployment

[3.1 Running Security Operations](#)

[3.2 Viewing Logs in Splunk](#)

[3.3 Autoscaling the Protegity Reference Deployment](#)

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

Protegity provides you a sample application that you can use to test the data protection capabilities of the DSG Container deployment.

For more information about using the sample application, refer to the section [Appendix B: Using the Sample Application](#).

3.1 Running Security Operations

This section describes how you can use the DSG instances running on the Kubernetes cluster to protect the data that is sent by a REST API client, by using static and dynamic CoP.

3.1.1 Running Security Operations with Static CoP

This section describes how you can use the DSG instances running on the Kubernetes cluster to protect the data that is sent by a REST API client, by using static CoP. In this example, the default *Text Protection* ruleset is used for protecting the data.

► To run security operations with static CoP:

Using Ingress:

1. Add an entry in the *hosts* file of your node from where you are making the REST API calls, to map the external IP address of the Ingress object to the hostname.

Note:

The external IP address is the IP address of the ingress object that you obtain after you run the *kubectl get ingress* command after deploying the application on the Kubernetes cluster.

For more information about the external IP address associated with the ingress object, refer to [step 10](#) of the section [Deploying a Release on the Kubernetes Cluster](#).

For Windows:



- a. Navigate to the `C:\Windows\System32\drivers\etc` directory.
- b. Edit the `hosts` file in any text editor, such as Notepad, as an administrator, and add the following entry.
`<External_IP> <Host name in your Ruleset>`

For Linux:

- a. Run the following command.

```
vi /etc/hosts
```

- b. Add the following entry.

```
<External_IP> <Host name in your Ruleset>
```

2. In the URL field of the Postman client, enter the URL of the DSG Kubernetes cluster that you want to send the request for protecting data.

For example, `http://restapi/prod/protect/text`

This example uses the default `Text Protection` ruleset that is a part of the REST API Examples available on the ESA.

Header	Value
Postman-Token	<calculated when request is sent>
Content-Type	text/plain
Content-Length	<calculated when request is sent>
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.26.8
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Host	restapi
Content-Type	text/plain
Key	Value

3. In the **Header** tab, specify the value for the Host key.

For example, enter `restapi`.

4. Specify the IP address of the Linux instances, from where you are sending the REST API request, as the value for the **X-Forwarded-For** key in the REST API client.

This is used by the Transaction Metrics.

5. In the **Authorization** tab, select the authorization type as **Basic Auth** in the **Type** list, and then specify the values of the **Username** and **Password** fields respectively.

Note: Ensure that the Basic Authentication functionality has been enabled in the ESA.

For more information about enabling the Basic Authentication functionality in the ESA, refer to [step 11](#) of the section [Creating and Exporting the Ruleset](#).

6. In the **Body** tab, specify the data that you want to protect.

7. Click **Send**.

The DSG instance protects the data and returns the protected data.

Using a load balancer:

8. Run the following command to retrieve the load balancer IP address.

```
kubectl get svc -ndsg
```

9. Specify the details in the Postman client as shown in the following figure.

The screenshot shows the 'Headers' tab in Postman with the following configuration:

Header	Description
Content-Type	text/plain
Content-Length	<calculated when request is sent>
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.26.8
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Host	restapi
Content-Type	text/plain

- In the **Authorization** tab, select the authorization type as **Basic Auth** in the **Type** list, and then specify the values of the username and password in the **Username** and **Password** fields respectively.

Note: Ensure that the Basic Authentication functionality has been enabled in the ESA.

For more information about enabling the Basic Authentication functionality in the ESA, refer to [step 11](#) of the section *Creating and Exporting the Ruleset*.

- In the **Body** tab, specify the data that you want to protect.

- Click **Send**.

The DSG instance protects the data and returns the protected data.

Using a cluster IP:

- Ensure that the clusterIP service is enabled and the Ingress is disabled in the *values.yaml* file.

- Add the hostname in the CoP in the following format.

<Kubernetes service name of DSG container>.<Namespace of DSG container>.svc.cluster.local

The screenshot shows the Kubernetes Cluster IP interface with the following details for the 'restapi' service:

- Hostname:** restapi
- Streaming:**
 - Resource:** /passwordMaskingTest
 - Pattern:** (?<=(bpw=)(.-?)(?=\$|&))
 - Mask:** *m*a*t*k*#d*
- Learn Mode Settings:** Enabled
- Exclude Resource:** \css\image\video\svg\b
- Exclude Content-Type:** \css\image\video\svg\b
- Include Resource:**

- From a pod in the same cluster as the DSG container deployment, run the following cURL request.

```
curl --location --request POST 'http:// <Kubernetes service name of DSG container>.<Namespace of DSG container>.svc.cluster.local:8080/v1/restapi/protect/text'
--header 'Host: <Kubernetes service name of DSG container>.<Namespace of DSG container>.svc.cluster.local' --header 'Content-Type: text/plain' --data
'DataToBeProtected'
```

If you are using the Basic Authentication functionality, then you need to include the Base64-encoded value of the username and password as part of the authorization header, as shown in the following snippet.

```
curl --location --request POST 'http:// <Kubernetes service name of DSG container>.<Namespace of DSG container>.svc.cluster.local:8080/v1/restapi/protect/text'
```

```
--header 'Host: <Kubernetes service name of DSG container>.<Namespace of
DSG container>.svc.cluster.local' --header 'Content-Type: text/plain' --data
'DataToBeProtected' -H 'Authorization: Basic <Base64 Encoding{Username:Password}>'
```

Note: Ensure that the Basic Authentication functionality has been enabled in the ESA.

For more information about enabling the Basic Authentication functionality in the ESA, refer to [step 11](#) of the section [Creating and Exporting the Ruleset](#).

3.1.2 Running Security Operations with Dynamic CoP in HTTPS Mode

This section describes how you can use the DSG instances running on the Kubernetes cluster to protect the data that is sent by a REST API client, by using dynamic CoP in HTTPS mode.

► To run security operations with dynamic CoP:

1. In the URL field of the Postman client, enter the URL of the DSG Kubernetes cluster that you want to send the request for protecting data.

For example, <https://prod.example.com/protect>

In this example:

- *protect* is the endpoint URI that you have specified while creating your ruleset.
- *prod.example.com* is the hostname that you have configured in your ruleset. This value is also specified as the hostname of the Ingress service in the *values.yaml* file. In addition, ensure that you add this value in the *hosts* file of your node from where you are making the REST API calls, to map the external IP address of the Ingress object to the hostname.

For more information regarding the external IP address associated with the ingress object, refer to [step 10](#) of the section [Deploying a Release on the Kubernetes Cluster](#).

2. In the */etc/resolv.conf* file in Linux, resolve the IP address.
3. In the **Header** tab, specify the value for the **X-Protegrity-DCoP-Rules** key.

For example, specify the following dynamic CoP as the header value:

```
[{"action": {"payload": {"encoding": {"type": "NO_ENCODING"}, "hasToMatch": "", "pattern": "\w+", "type": "TEXT"}, "type": "EXTRACT"}, "children": [{"action": {"method": {"dataElementName": "TE_Unicode_S23", "encoding": {"type": "NO_ENCODING"}, "method": "Protect", "type": "PROTEGRITY_DATA_PROTECTION"}, "type": "TRANSFORM"}, "enabled": true, "text": "transformrule"}], "enabled": true, "text": "extracttext"}]
```

Note:

Ensure that the data element included in the header value is present in the ESA policy.

For example, the data element *TE_Unicode_S23*, which is included in the header value, must be present in your ESA policy.

4. Specify the IP address of the Linux instances, from where you are sending the REST API request, as the value for the **X-Forwarded-For** key in the REST API client.



- This is used by the transaction metrics.
5. In the **Authorization** tab, select the authorization type as **Basic Auth** in the **Type** list, and then specify the values of the username and password in the **Username** and **Password** fields respectively.

Note: Ensure that the Basic Authentication functionality has been enabled in the ESA.

For more information about enabling the Basic Authentication functionality in the ESA, refer to [step 11](#) of the section [Creating and Exporting the Ruleset](#).

6. In the **Body** tab, specify the data that you want to protect.
7. Import the CA certificate, client certificate and the private key of the client certificate to the Postman client.
For more information about how to create the CA certificate and client certificate and keys, refer to the section [Creating Certificates and Keys for TLS Authentication](#).

For more information about how to import the certificates to the Postman client, refer to the section [Importing Certificates to the Postman Client](#).

8. Click **Send**.
The DSG instance protects the data and returns the protected data.

3.2 Viewing Logs in Splunk

This section describes how you can view the DSG and Container logs in Splunk Enterprise.

► To view logs:

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

The Splunk Enterprise screen appears.

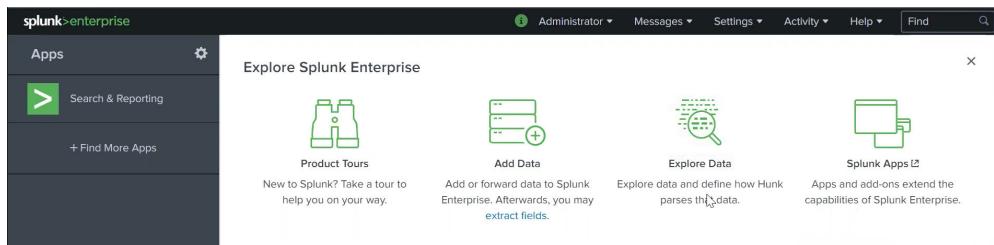


Figure 3-1: Splunk Enterprise Web UI

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.

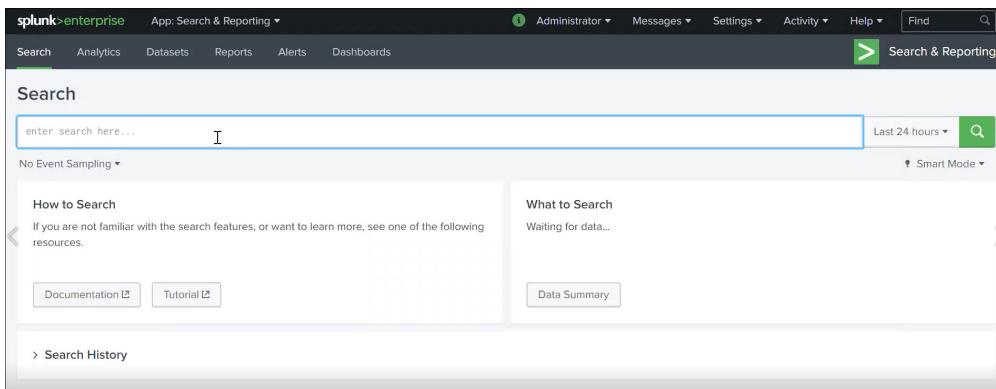


Figure 3-2: Search Screen

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

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

For example:

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

- Press **Enter**.

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

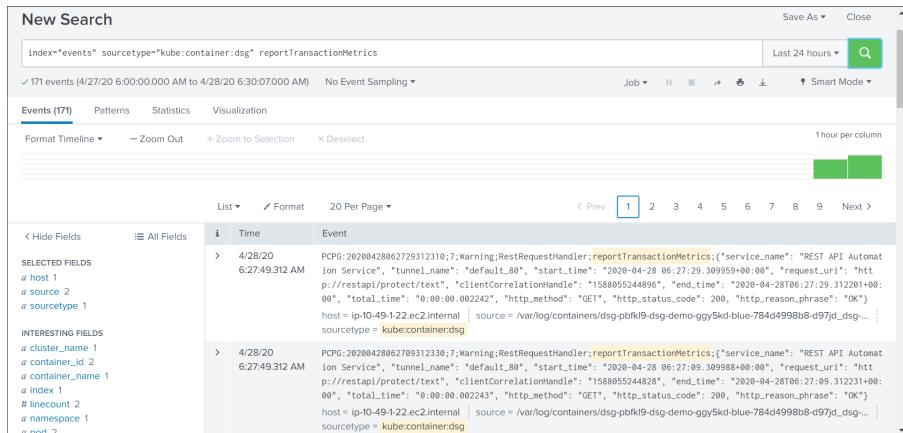
Time	Event
2020-04-28 05:43:20.198	host = ip-10-49-1-22.ec2.internal source = /var/log/containers/dsg-pbfkl9-dsg-demo-ggy5kd-blue-784d4998b8-d97jd_dsg... sourcetype = kube:container:dsg-policy-provider
2020-04-28 05:43:20.085	host = ip-10-49-1-22.ec2.internal source = /var/log/containers/dsg-pbfkl9-dsg-demo-ggy5kd-blue-784d4998b8-d97jd_dsg... sourcetype = kube:container:dsg-policy-provider
2020-04-28 05:43:17.105	host = ip-10-49-1-22.ec2.internal source = /var/log/containers/dsg-pbfkl9-dsg-demo-ggy5kd-blue-784d4998b8-d97jd_dsg... sourcetype = kube:container:dsg-policy-provider

- If you want to view only the transaction metric logs that are related to the DSG, 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:dsg" reportTransactionMetrics
```

- Press **Enter**.

The search results display only the logs that are related to the DSG.



3.3 Autoscaling the Protegity Reference Deployment

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

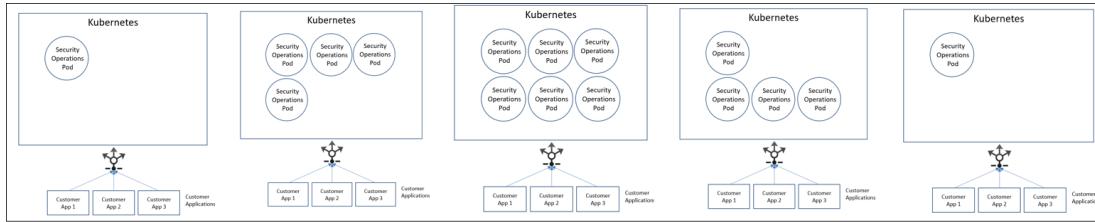


Figure 3-3: Autoscaling Kubernetes Cluster

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

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

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

Chapter 4

Troubleshooting Issues

This section includes the issues that you might encounter and the recovery actions to troubleshoot these issues.

Table 4-1: Troubleshooting Issues

Issue	Recovery Action
When you run the <code>kubectl get pods</code> command, the <code>ImagePullBackOff</code> status appears.	<p>This issue occurs if the container images have not been pulled accurately from the container repository.</p> <p>Run the <code>kubectl describe pod</code> command to retrieve the pod details. This can help you in debugging the error.</p>
When you run the <code>kubectl get pods</code> command, the <code>Pending</code> status appears.	<p>This issue occurs if adequate memory or CPU is not available for creating a pod.</p> <p>As a workaround, you can try to increase the memory or CPU of the existing nodes where the pod is being deployed, or create a new node where you can deploy the pod.</p>
<p>The DSG container log displays the following error:</p> <pre data-bbox="110 1157 780 1214">in bind_sockets sock.bind(sockaddr) PermissionError: [Errno 13] Permission denied</pre>	<p>This error can occur if you create a tunnel with port lesser than <code>1024</code>, while creating a ruleset on the DSG Web UI.</p> <p>To avoid this error, ensure that you create a tunnel with port greater than <code>1024</code> as the Docker containers do not allow any external communication on ports below <code>1024</code>.</p>
<p>If you send a request to the DSG container with the Basic Authentication functionality enabled, then the response takes more than 1 minute and returns the following error:</p> <pre data-bbox="110 1410 355 1438">401 Unauthorized</pre> <p>In addition, the state of the serving DSG pod changes to <i>non-ready</i>.</p>	<p>Check the DSG pod logs. If the logs display the following error, then it indicates that the LDAP server is unavailable.</p> <pre data-bbox="833 1383 1465 1488">Error;LDAPConnection;CheckUserRole;LDAP Error: {u'info': 'Transport endpoint is not connected', 'errno': 107, 'desc': u"Can't contact LDAP server"}</pre>

Chapter 5

Appendix A: DSG Container Helm Chart Details

[5.1 Chart.yaml](#)

[5.2 Values.yaml](#)

[5.3 credentials.json](#)

The DSG containers are bundled with Helm charts, which are included in the package. This section explains the details of the Helm chart structure and the entities that the user needs to modify for adapting the charts for their environments.

5.1 Chart.yaml

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

```
annotations:
  pepVersion: 1.2.2+42
  apiVersion: v1
  description: A DSG chart for Kubernetes
  name: dsg
  version: 1.0.0
```

5.2 Values.yaml

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

5.2.1 Image Pull Secrets

This section provides information for the credentials 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:

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

5.2.2 Name Override

These values indicate how naming for releases are managed for deployment. Protegity 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 combination of the `nameOverride` parameter value and the Helm chart release name.

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

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

5.2.3 Container Image Repository

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

```
dsgImage:
  repository: REPOSITORY_DSG_IMAGE
  tag: "DSG_IMAGE_TAG"
  pullPolicy: Always
  debug: false # change debug to true to get container debug logs on STDOUT.
```

The following list provides details for the `dsgImage` parameter, which refers to the details for the DSG container image:

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

Note:

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

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

5.2.4 Resources

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

```
dsgContainerResources:
  ## Below allocation covers largest policy that can be imported.
  ## Based on the policy size/application requirement update below limits.
  limits:
    cpu: 1000m
    memory: 3500Mi
  requests:
    cpu: 200m
    memory: 200Mi
```

You can specify the following parameters:

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

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

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

5.2.5 Persistent Volume Claim

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

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

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

5.2.6 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
```

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

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

5.2.7 Container Security Context

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

```
# Security Context object for DSG container
## leave the field empty if not applicable
capabilities:
  drop:
    - ALL
allowPrivilegeEscalation: false
privileged : false
runAsNonRoot : true
readOnlyRootFilesystem: true
seccompProfile:
  type: RuntimeDefault
```

Leave the field blank if it is not applicable.

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

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

5.2.8 Pod Service Account

This section provides information for the name of the service account that must be created for the pod in the Kubernetes Cluster.

Note: The *Pod Service Account* parameter is not applicable for the Azure release.

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

5.2.9 Liveliness and Readiness Probe Settings

This section specifies the intervals required to run health checks for the DSG container images. The users must not change these settings.

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

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

The health check is performed by sending a request to the <http://<hostname>/protect/text> URI. This URI is defined as part of the *Text Protection* ruleset, which is the default ruleset available on the ESA. Ensure that you retain this default ruleset on the ESA so that the health check is performed successfully.

5.2.10 Password-Based Encryption (PBE) Secrets

This section provides a brief overview for the *pbeSecrets* parameter. If you are using PBE encryption to encrypt the policy package, then the DSG container uses the value specified in the *pbeSecrets* parameter to decrypt the policy.

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

5.2.11 Storage Access Secrets

This section provides a brief overview for the *storageAccessSecrets* parameter. This secret contains the credentials for pulling the CoP and the Policy from the Azure Storage Container. The DSG Application container uses the value specified in the *storageAccessSecrets* parameter to decrypt the Policy package using the KMS.

```
# k8s secret for storing credentials with read access to azure storage account
# keep empty for GCP and AWS
storageAccessSecrets: STORAGE_ACCESS_SECRET_NAME
```

5.2.12 Private Secret Key

This section provides a brief overview for the *privateKeySecret* parameter. If you are using KMS to encrypt the policy package, then the DSG container uses the value specified in the *privateKeySecret* parameter to decrypt the policy package.

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

5.2.13 CoP Secret

This section provides a brief overview for the *copSecrets* parameter. The DSG container uses the value specified in the *copSecrets* parameter to decrypt the CoP package.

```
## k8s secret containing passphrase for decrypting cop
copSecrets: COP_SECRETE_NAME
```

5.2.14 LDAP Secrets

This section provides a brief overview for the *ldapXmlSecrets* parameter. The DSG container uses the value specified in the *ldapXmlSecrets* parameter to connect to the LDAP.

```
# LDAP configuration secret name
ldapXmlSecrets: LDAP_SECRET_NAME
```

5.2.15 Deployment

These configurations refer to the DSG package-related information for the current release. The first release is considered to be *blueDeployment*.

When, *blueDeployment* is enabled, parameter is set to *true*, the configurations mentioned under the *blueDeployment* section are applied.

```
## start with blue deployment first
deploymentInUse: blue

blueDeployment:
  enabled: true

blueDeployment:
  enabled: true
  policy:
    # absolute path in ObjectStore from where the policy dump file will be fetched.
    # <S3> "s3://bucketName/pathToPolicy"
    # <GS> "gs://bucketName/pathToPolicy"
    # <Azure> "https://<accountName>.blob.core.windows.net/<container name>/<pathToPolicy>"
    # absolute path in PV mount from where the policy dump file will be fetched.
    # <VOLUME> "file://var/data/ptypolicy/pathInVolumeToPolicy"
    # relative path in PV mount from where the policy dump file will be fetched.
    # it will prepend the '/var/data/ptypolicy' to below path
    # <VOLUME> "file:///pathInVolumeToPolicy"
    # <VOLUME> "pathInVolumeToPolicy" # without any leading '/'
    filePath: ABSOLUTE_POLICY_PATH
  cop:
    # absolute path in ObjectStore from where the cop dump file will be fetched.
    # <S3> "s3://bucketName/pathToCop"
    # <GS> "gs://bucketName/pathToCop"
    # <Azure> "https://<accountName>.blob.core.windows.net/<container name>/<pathToCop>"
    # absolute path in PV mount from where the cop dump file will be fetched.
    # <VOLUME> "file://var/data/ptypolicy/pathInVolumeToCop"
    # relative path in PV mount from where the cop dump file will be fetched.
    # it will prepend the '/var/data/ptypolicy' to below path
    # <VOLUME> "file:///pathInVolumeToCop"
    # <VOLUME> "pathInVolumeToCop" # without any leading '/'
    filePath: ABSOLUTE_COP_PATH
```

The user is required to update the highlighted values as per the details where you have stored the CoP and policy package. Note that all entries are case sensitive.

5.2.16 Autoscaling

This section specifies the settings for the autoscaling of pods on the cluster.

```
## specify the initial no. of DSG 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 when the deployment starts.
- *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* parameter.

5.2.17 Service Configuration

This section has configurations for the REST service to be used on the cluster running the DSG containers.

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

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

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

healthCheckRestService:
  scheme: HTTP
  host: restapi
  port: &healthCheckPort 8080
  path: /protect/text
tunnels:
  - name: "restapi"
    port: 8080
    targetPort: *healthCheckPort
    # Add the below object for new tunnel,
    # specify the port used while creating TLS enabled HTTP Tunnel in "targetPort"
    # - name: "servicel"
    #   port: 9443
    #   targetPort: 9443
```

You can specify the value of the *dsgService/type* parameter as *LoadBalancer* or *ClusterIP*. This parameter is only applicable if you have set the value of the *ingress/enabled* parameter to *false*.

The value of the *healthCheckRestService/port* parameter must be set to the same port value as that of the tunnel used in the *REST API Examples* service on the ESA. This port is used to perform the Readiness probe.

Update the health check-related section with the following details:

- *scheme* - The protocol used to send a request for checking the health of the DSG service.
- *host* - The host name of the heath check rule defined in the CoP.
- *port* - The port number configured in the DSG CoP ruleset for the health check rule.
- *path* - The URI of the health check rule configured in the DSG CoP ruleset.

If the user requires to add a new tunnel for the DSG Containers, then the following points must be ensured:

- The required CoP must be built on the ESA.
- The CoP and policy package should be made available on the Azure Storage Container or the Azure File Share.



- Add the tunnel-related section:
 - *name* - Distinguishes between different ports used in the DSG tunnels. The name must contain only lowercase alphanumeric characters, which is based on standard Kubernetes naming conventions.
 - *port* - Specifies the port on which you want to expose the service externally.
 - *targetPort* - Specifies the port number configured while creating the Tunnel.

5.2.18 Ingress Configuration

This section explains the Ingress configuration for deployment.

```
## Ingress enables testing of Green (V2/Staging) Deployments alongside Blue ones
## simultaneously.
## This is realized by exposing both deployments on the same External IP but on different URL
## Paths.
## If you don't want expose both the deployments at the same time, keep 'ingress.enabled' as
## false.
## This means that only Blue deployment gets exposed on the Load Balancer's External IP via
## prod service.
ingress:
  enabled: true
  # specify ingressClass cluster resource. It associates which controller will implement the
  ## resource.
  ingressClassName: nginx-dsg
  ## else keep the 'ingress.class' field empty
  ## to use cloud native Ingress Controller.
  annotations:
    nginx.ingress.kubernetes.io/ssl-passthrough: "true"
    ## specify the host names and paths for Ingress rules
    ## prod and staging http paths can be used as optional fields.
  hosts:
    - host: ""
      httpPaths:
        - port: 8080
          prod: "/v1/restapi"
          staging: "/v2/restapi"
```

In this case, we are referring to NGINX as the ingress. The following configurations can be edited by the user as per the CoP and deployment requirements.

```
ingress:
  enabled: true
```

When the *enabled* parameter is set to *true*, the ingress controller will be based on the value used in the annotation *ingressClassName*. If the parameter *enabled* is set to *false*, then the ingress controller will not be used. Instead, the default load balancer of the Kubernetes service will be used to expose the DSG Container services.

```
annotations:
  nginx.ingress.kubernetes.io/ssl-passthrough: "true"
```

If the *ingressClassName* value is set to *nginx-dsg*, then the NGINX Ingress Controller will be used. If the *ingress.ClassName* value is empty, then the native ingress controller will be used.

If the *nginx.ingress.kubernetes.io/ssl-passthrough* value parameter is set to *true*, then the NGINX Ingress Controller works as an SSL passthrough. This ensures an end-to-end encrypted communication channel between the REST API client and the DSG pods.

The following snippet refers to URLs and ports referred to for *blue green* deployments.

```
hosts:
  - host: ""
    httpPaths:
      - port: 8080
        prod: "/v1/restapi"
```

```

staging: "/v2/restapi"
## Add host section with the hostname used as CN while creating server certificates.
## For accessing the staging end-points also, we need hostname as that of CN in server
certs.
## While creating the certificates you can use *.example.com as CN for the below example
# - host: "prod.example.com"
#   httpPaths:
#     - port: 9443
#       prod: "/"
#     - host: "staging.example.com"
#       httpPaths:
#         - port: 9443
#           staging: "/"
#
## If TLS is terminated on the Ingress Controller Load Balancer,
## K8s TLS Secret containing the certificate and key must also be provided
# tls:
#   - secretName: chart-example-tls
#   hosts:
#     - chart-example.local

```

The user must ensure the related COPs are made available on the storage before applying the changes. Ensure that you specify the values of the HTTP paths parameters as shown in the following snippet.

```

httpPaths:
- port: 8080
  prod: "/v1/restapi"
  staging: "/v2/restapi"

```

- *port* – Refers to the same value that you have provided in the *dsgService/tunnels/port* field.
- *prod* – Refers to the URI used for pointing to the current production deployment. This is the URI on which the production data traffic is diverted.
- *staging* – Refers to the URI used for pointing to the current deployment under test.

```

# - host: "somehostname"
#   httpPaths:
#     - port: 8080
#       staging: "/v3/restapi"
# - host: "randomhostname"
#   httpPaths:
#     - port: 8080
#       prod: "/v4/restapi"

```

If you specify hostname in the *host* field, then ensure that you have the same hostname value configured in your ruleset. In this case, the ingress maps the value of the provided hostname to the external IP address, and you also need to configure the hostname in your DNS.

Note:

The hostname must be specified in the Fully Qualified Domain Name (FQDN) format, such as *abc.def.xyz*.

Using the hostname option, you have *prod* URLs running on one hostname and *staging* URLs running on another hostname. If you specify multiple hostnames in the *values.yaml* file, then you must ensure that you have added these hosts names in the DSG ruleset.

For more information about adding multiple hostnames to the DSG rule set, refer to the section [Creating a Ruleset](#).

If you do not specify hostname in the *host* field, then the ingress can be accessed on the external IP directly.

If you are using the NGINX Ingress Controller as an SSL passthrough, then you must specify the value of the *prod* and *staging* URL as /. This ensures that the data traffic from the REST API client is redirected to the required production or staging URL based on the *hostname* value.

5.3 credentials.json

The *credentials.json* file contains information about the credentials for the service principal 2 that are required to access the Azure storage account. The credentials are required to enable the service principal 2 to retrieve the IMP Metadata Package that has been uploaded to the Azure storage container. The following snippet is shown for Azure. Users are required to edit the file and replace the values for the service principal 2 credentials.

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

Chapter 6

Appendix B: Using the Sample Application

6.1 Overview

6.2 Running the Sample Application

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

6.1 Overview

Protegility delivers a sample application as part of the DSG Container installation package. The sample application constitutes a canned policy (to get imported to the ESA), canned CoP, sample Postman collection (to test the DSG Container Pods serving deployed IMP) and an auto-scaling script (to push more load to the Kubernetes cluster to force auto-scaling of the DSG Container Pods).

On consuming the deliverables in the DSG Container installation package, the users must run this sample application end-to-end, as a sanity test, to confirm that the installation was completed accurately. In this section, we are providing details on the exact steps that the user needs to follow to run the sample application end-to-end. This section explains details and usage of the components included in the *DSG-Samples_Linux-ALL-ALL_x86-64_<DSG_Containerization_App_version>.tgz* archive.

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

6.1.1 Policy Sample

Package – *Sample_App_Policy_V5.tgz*

This component consists of the sample policy that can be imported on the ESA 9.1.0.5 for getting started with the DSG Containers use case. The following table contains the policy details.

Policy Name - Sample_policy

Token Type	Data Element Name
Alphanumeric	deFirstName
Alphanumeric	deLastName
Alphanumeric	deStreet
Alphanumeric	deCity
Alpha	deState
Alphanumeric	deZipcode
Numeric	deSSN

Token Type	Data Element Name
Unicode	PTY_UNICODE

6.1.2 CoP / RuleSet Sample

Package – *Sample_App_COP_V5.zip*

This component consists of sample ruleset for REST for getting started with the DSG Containers use case. The following are the details for the ruleset.

Ruleset 1

- RuleSet Name - REST API Example
- Port for REST – 8080

Caution:

This ruleset is only applicable for performing a health check. Ensure that you do not modify this ruleset.

Ruleset 2

- RuleSet Name - DynamicCoP
- Port for REST (with TLS) - 9443
- Format Supported for Dynamic CoP – ALL
- Path for Dynamic CoP - /protect

Name	Description	Protocol	Enabled	Interface	Port	Certificate	Action
CIFS	The Common Internet File System (CIFS) is a file sharing protocol.	CIFS	false	\\\			edit trash
DCoP_Tunnel		HTTP	true	ethSRV0	9443	dsg.wildcard	edit trash
default_25		SMTP	false	ethSRV0	25	service	edit trash
default_80		HTTP	true	ethSRV0	8080		edit trash
default_81		HTTP	false	ethSRV0	81		edit trash
NFS	The Network File System (NFS) enables users to store files on remote servers.	NFS	false	10.10.98.222:/var/nfsshare			edit trash
S3		S3	false	CHANGE_ME			edit trash

Note:

In the sample, if you are using the TLS for communication between the Postman client and the DSG instance on the Kubernetes pod, then ensure that the port number is set to **9443** in the Tunnel.

6.1.3 Autoscaling Script

Package – *Sample_App_autoscale.sh*

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

6.1.4 PostMan Collection

Package – *Sample_App_PostMan_Collection_V4.json*

This collection can be used to make REST calls to the DSG for protecting the data set provided in the *Sample App Test Data.csv* file. The collection has two entries (Release 1 and Release 2).

Release 1

Post Request Path - <https://prod.example.com/protect>

Body (Data to be Protected)

The request is built using the *Sample App Test Data.csv*. The contents of the .csv file are added to the *Body* of the request.

Header - DCOP request

The header is populated with the DCOP request, which protects data using the following Data Elements.

Field	Column Name	Token Type	Data Element Name
First Release			
First Name	first name	Alphanumeric	deFirstName
Last Name	last name	Alphanumeric	deLastName
Street	street address	Alphanumeric	deStreet
City	City	Alphanumeric	deCity
State	State Name	Alpha	deState
Zip Code	zip code	Alphanumeric	deZipcode
SSN	SSN	Numeric	deSSN

Release 2

Post Request Path - <https://staging.example.com/protect>

Body (Data to be Protected)

The request is built using the *Sample App Test Data.csv* file. The contents of the .csv file are added to the *Body* of the request.

Header - DCOP request

The header is populated with the DCOP request, which protects data using the following Data Elements. Note that the *Customer ID* is added as a new data element to protect the *Customer ID* column.

Field	Column Name	Token Type	Data Element Name
First Release			
First Name	first name	Alphanumeric	deFirstName
Last Name	last name	Alphanumeric	deLastName
Street	street address	Alphanumeric	deStreet
City	City	Alphanumeric	deCity
State	State Name	Alpha	deState
Zip Code	zip code	Alphanumeric	deZipcode



Field	Column Name	Token Type	Data Element Name
SSN	SSN	Numeric	deSSN
Customer ID	Customer ID	Customer ID	Customer ID

6.2 Running the Sample Application

1. The Azure Storage Container is created.
2. The cluster is created. For more information about creating a Kubernetes Cluster, refer to the section [Creating a Kubernetes Cluster](#).

Ensure that the [Prerequisites](#) are met before performing the following steps.

The user must perform the following steps.

1. Generate Certificates and Keys for TLS Authentication
2. Import Certificates on the ESA
3. Import certificates to the Postman Client
4. Enable the Authentication functionality
5. Import the Policy sample on ESA
6. Import the CoP sample on the ESA
7. Create Immutable Metadata Packages (IMP)
8. Deploy Release 1
9. Run Sample Application (PostMan collection)
10. Run Autoscaling Script

6.2.1 Generating Certificates and Keys for TLS Authentication

This section describes how you can create certificates and keys for establishing a secure communication between the Postman client and the server.

Before you begin

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

► To initialize the Linux instance:

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 dsg-ca.key -out dsg-ca.crt -days 356 -nodes -subj '/CN=DSG Certificate Authority'
```
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 dsg-wildcard.key -out dsg-wildcard.csr -nodes -subj '/CN=*.example.com'
```



```
openssl x509 -req -sha256 -days 365 -in dsg-wildcard.csr -CA dsg-ca.crt -CAkey dsg-ca.key -set_serial 04 -out dsg-wildcard.crt
```
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 dsg-client.key -out dsg-client.csr -nodes
-subj '/CN=My DSG Client'
```

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

4. Copy all the certificates to a common directory.

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

6.2.2 Importing Certificates on the ESA

The user needs to run the following steps to import CA and server certificates, and keys on the ESA, if you want to ensure secure communication between the Postman client and the DSG pods using TLS.

► To import CA and server certificates on the ESA:

1. Login to the ESA as *admin*.
2. Navigate to the **Cloud Gateway > Transport > Certificate/Key Material** tab.
3. Click **Choose File** to browse for the *dsg-ca.crt* certificate in the *dsg-certs* directory that you have created in the section [Creating Certificates and Keys for TLS Authentication](#).
4. Select the certificate.
5. Click **Upload certificate**.
6. Repeat steps **3** to **5** to upload the following files:
 - *dsg-ca.key* - Private key for the CA certificate
 - *dsg-wildcard.crt* - Server certificate
 - *dsg-wildcard.key* - Private key for the server certificate

Note: If you are uploading a key, then you are prompted to specify a password in the **Password** field for encrypting the key. You also have to re-type the password in the **Confirm Password** field.

However, you can choose to directly upload the key without specifying the password.

7. Perform the following steps to create a DSG tunnel for the TLS traffic between the REST API client and the DSG pods on the Azure.
 - a. Navigate to the **Tunnels** tab.
 - b. Click **Create Tunnel** to create a new tunnel for the TLS traffic.

The **Create Tunnel** screen appears.

 - c. Enter the required details to create a new tunnel.

For more information about creating a tunnel, refer to the section *Tunnels tab* in the [Data Security Gateway User Guide 3.1.0.5](#).

Figure 6-1: Create Tunnel Screen

- d. In the **TLS/SSL Security Settings** section, select the **TLS Enabled** check box to enable TLS for secure tunnel communication.
- e. In the **Certificate** list, select the server certificate that you created in *step 2* of the section *Creating Certificates and Keys for TLS Authentication*.
- f. In the **TLS Mutual Authentication** list, select the value as *CERT_REQUIRED* to ensure mutual TLS authentication between the Postman client and the DSG instance.
- g. In the **CA Certificates** field, specify the path where you have uploaded the CA certificate.
For example, specify the path as */opt/protegility/alliance/config/resources/dsg-ca.crt*.
- h. Click **Reload** to refresh the list of tunnels on the **Tunnels** tab.
Use this newly created tunnel while creating your ruleset.

6.2.3 Importing Certificates to the Postman Client

The user needs to perform the following steps to import the CA certificate, the client certificates, and keys to the Postman client, if you want to ensure secure communication between the Postman client and the DSG pods using TLS.

► To import certificates to the Postman client:

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

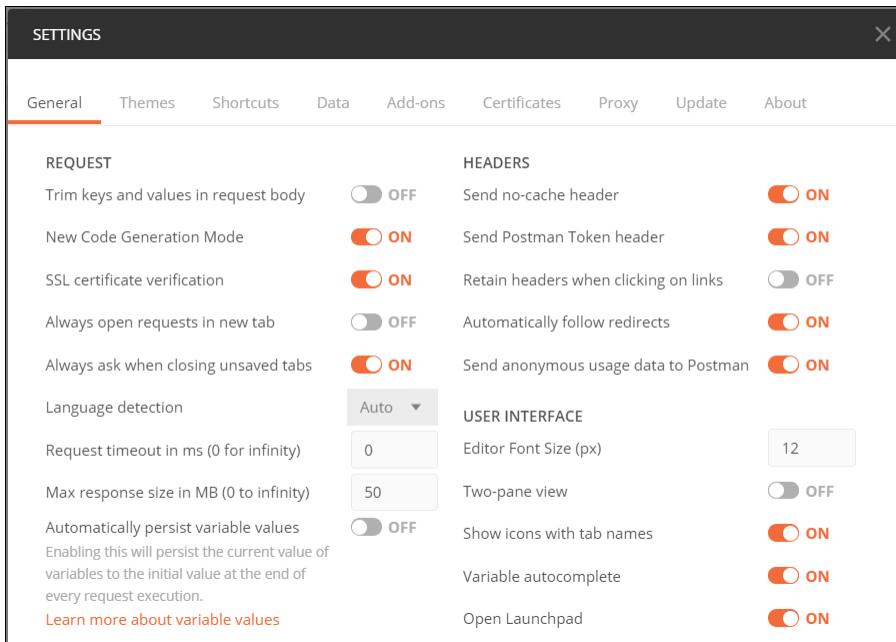


Figure 6-2: SETTING Dialog Box

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

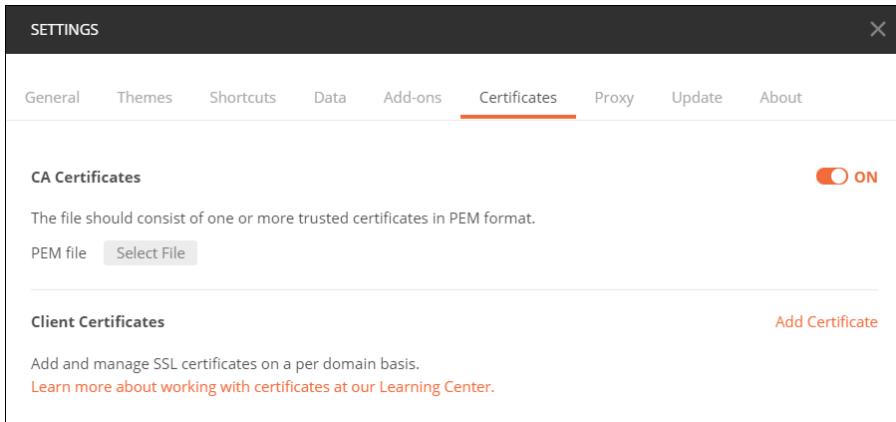


Figure 6-3: Certificates Tab

4. In the **CA Certificates** section, click **Select File** and browse for the *dsg-ca.crt* file in the *dsg-certs* directory that you have created in the section [Creating Certificates and Keys for TLS Authentication](#).

5. In the **Client Certificates** section, click **Add Certificate**.

The **Add Certificate** screen appears.

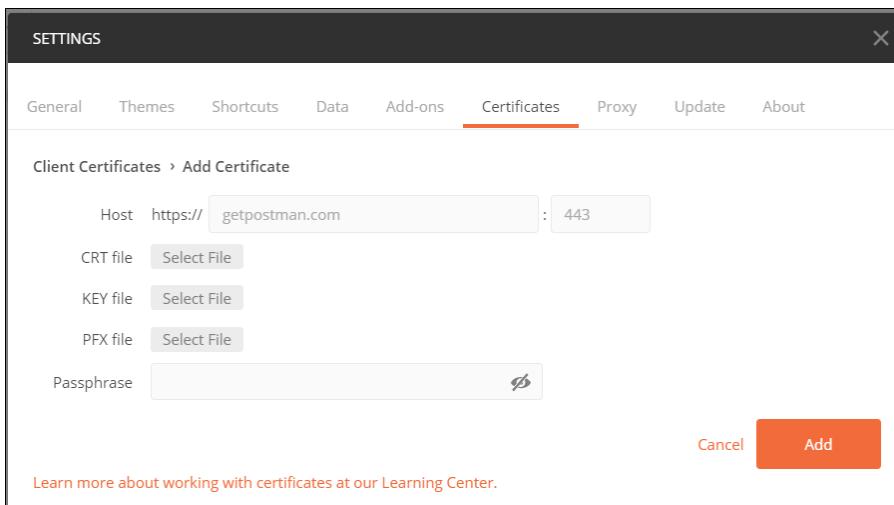


Figure 6-4: Add Certificate Screen

6. In the **Host** field, specify the value as *prod.example.com*.
You need to specify the ingress port number, which is *443* by default.
7. In the **CRT file** field, click **Select File** to browse for the *dsg-client.crt* file in the *dsg-certs* directory.
8. In the **KEY file** field, click **Select File** to browse for the *dsg-client.key* file in the *dsg-certs* directory.
9. Click **Add** to add the client certificate and key to the Postman client.
10. Repeat step 5 to step 9 for adding client certificates for the host *staging.example.com*.

6.2.4 Enabling the Authentication Functionality

The sample application provides the OAuth UDF that is used to implement the authentication functionality for the DSG instance. This functionality uses an access token, which is a JSON Web Token (JWT), to authenticate the requests that are sent from a REST API client to a DSG instance for performing security operations. The access token is digitally signed by Microsoft and is then included in the authorization header of the request that is sent to the DSG instance. The OAuth UDF is used to authenticate this access token.

For more information about JWT, refer to the section *Using JSON Web Token (JWT)* in the *Appliances Overview Guide 9.1.0.5*.

► To enable the authentication functionality:

1. Generate a token for the application by performing the following steps.
 - a. Navigate to the **App Registrations** page in Azure AD.
 - b. Select the **New registration** option to create an application.

The **Register an application** page appears.

 - c. Specify the name of the application using the **Name** input box. Leave the other fields empty.
 - d. Click **Register** to create a new application.
 - e. Navigate to the new application using the **App registrations** page by selecting the new application by name.
For example: app //dsg-demo
 - f. Note the application (client) id from the **Overview** tab.
 - g. Navigate to the **Certificates & secrets** page and create a new secret for the application. Note the newly created certificates.
 - h. Navigate to the **API permissions** tab to remove any permission if present. It should have zero permissions.

- i. Navigate to the **Expose an API** tab and create **Application ID URI**.

Note the application ID URI.

- j. Create a client credentials *openid* flow based request using the following URL.

<https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-client-creds-grant-flow>

- k. Send the following CURL request to obtain the access token.

```
curl -X POST -H "Content-Type: application/x-www-form-urlencoded" -d
'client_id=<client id from step 1f.>&scope=<url encoded application
id URI from step 1i>.default&client_secret=<client secret from
step 1g>&grant_type=client_credentials' 'https://login.microsoftonline.com/
<TENANT_ID>/oauth2/v2.0/token'
```

For example:

```
curl -X POST -H "Content-Type: application/x-www-form-
urlencoded" -d 'client_id=<client id from step
1f.>&scope=api%3A%2F%2Fdsg-demo%2F.default&client_secret=<client secret from
step 1g>&grant_type=client_credentials' 'https://login.microsoftonline.com/
<TENANT_ID>/oauth2/v2.0/token'
```

You need to copy this token and paste it in the *Token* field of the authorization header in the REST API request that you send to the DSG instance.

Important:

Ensure that the Application ID URI should end with a / (forward slash).

For example: *api://test-auth/*

Important:

The generated token has a default timeout of 60 minutes, and you need to regenerate the token after it times out.

2. Perform the following steps to enable the OAuth UDF in the Dynamic CoP.

- a. Login to the ESA as *admin*.
- b. Navigate to the **Cloud Gateway > RuleSet** tab.
- c. Expand **DynamicCoP > DynamicCoP > Extract HTTP request message body**.
- d. Select the **OAuth** UDF.

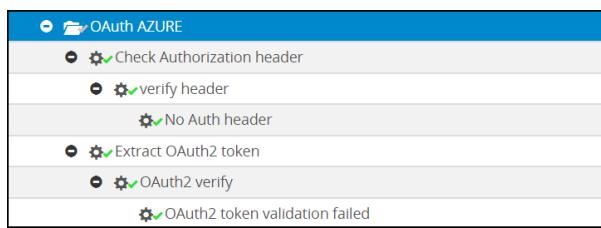


Figure 6-5: OAuth UDF

The OAuth ruleset consists of the following rules:

- *Check Authorization header* - Used to verify whether you have included a token in the authorization header of the REST API request.
- *Extract Authorization OAuth Token* - Used to verify whether a valid token has been included in the authorization header.

By default, the OAuth UDF is not enabled.

- e. Click **Edit** to edit the UDF.
- f. Select the **Enabled** check box.
- g. Expand the **Extract Authorization OAuth2 Token > OAuth2 verify** rule.

The **Initialization Arguments** section appears on the right side panel.

- h. Specify the following initialization arguments.

Argument Name	Description	Value	Required / Optional
Debug logging	Specify whether you want to run the UDF in debugging mode The default value is set to <i>true</i> .	<ul style="list-style-type: none"> • true • false 	Required
JSON Web Key Set (JWKS) certificates	<p>Specify the public keys that are used to digitally sign the ID token. You can obtain the latest JWKS from Microsoft by accessing the following URL:</p> <pre><code>https:// login.microsoftonli ne.com/TENANT_ID/ discovery/v2.0/keys</code></pre> <p>The <i>TENANT_ID</i> is the tenant ID of the application that you have created in step 1.</p> <p>Ensure that you specify valid keys. If you specify keys that are already expired, then the OAuth UDF automatically retrieves the latest keys by accessing the URL specified in the <i>JWKS URI</i> argument. However, the automatic retrieving of keys from the JWKS URL can impact the performance of the REST API request.</p>		Required
JWKS URI	<p>Specify the URL used to obtain the JWKS from Microsoft.</p> <p>For example, specify the following value as the URL:</p> <pre><code>https:// login.microsoftonli ne.com/TENANT_ID/ discovery/v2.0/keys</code></pre> <p>The <i>TENANT_ID</i> is the tenant ID of the application that you have created in step 1.</p>		Required

- i. Click **Save** to save the changes.



- j. Click **Reload**.

6.2.5 Importing the Policy Sample on the ESA

The user needs to perform the following steps to import the *Sample_App_Policy_V4.tgz* file on the ESA.

► To import policy sample on the ESA:

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

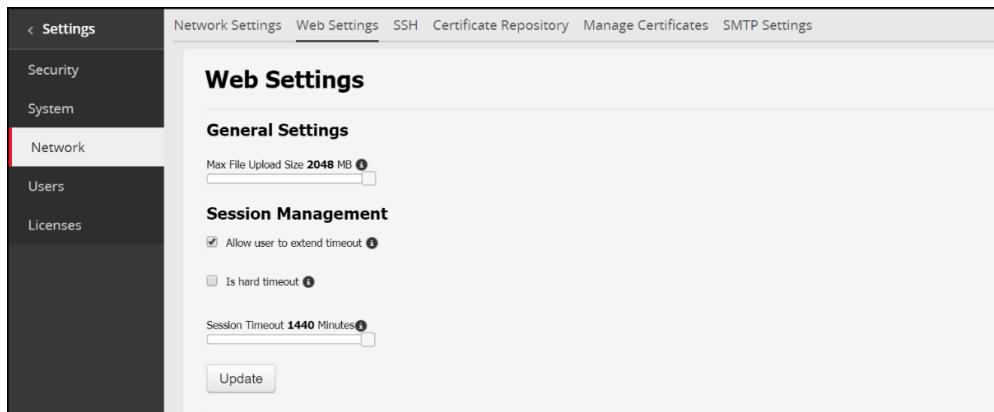


Figure 6-6: ESA Web Settings

3. In the **General Settings** section, change the **Max File Upload Size** value to the maximum value.
4. In the **Session Management** section, change the **Session Timeout** value to the maximum value.
5. Click **Update**.
6. Navigate to **Settings > System > File Upload**.
7. Click **Choose File** to select the *Sample_App_Policy_V4.tgz* file that you want to upload.
8. Navigate to **System > Backup and Restore > Import**, and select the *Sample_App_Policy_V4.tgz* file from the drop down and click **Import**.
9. Provide the password as **admin1234** and click **Import**.

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

10. Run the following command to export the Policy package.

```
curl -v -k -u "export_user:admin1234"
https://xxx.xxx.xxx.pty/v1/imp/export
-H "Content-Type: application/json"
--data '{"name": "Policy1.json",
"pepVersion": "1.2.0+16",
"dataStoreName": "DS1",
"emptyString": "NULL",
"caseSensitive": "YES", "kek": {"pkcs5": {
"password": "Passphrase123@",
"salt": "salt"}}}' -o Policy1.json
```

For more information about exporting the Policy package, refer to the [step 2](#) or [step 3](#) in the section [Creating and Exporting the Policy](#).

11. Upload the Policy package to an Azure Storage Container or Azure File Share.

For more information about uploading the Policy package to an Azure Storage Container, refer to the [step 4](#) in the section [Creating and Exporting the Policy](#).

For more information about uploading the Policy package to an Azure File Share, refer to the [step 5](#) in the section [Creating and Exporting the Policy](#).

Important:

Ensure that the user has the required permissions to upload an object to the Azure Storage Container or persistent volume.

6.2.6 Importing the CoP Sample on the ESA

The user needs to run the following steps to import the *Sample_App_COP_V4.zip* file on the ESA.

► To import CoP sample on ESA:

1. Login to the ESA as *admin*.
2. Navigate to the **Cloud Gateway > Ruleset** tab.
3. Click **Import**.
4. Click **Choose File** and select *Sample_App_COP_V4.zip*.
5. Select the **Override existing configuration in case of conflicts** check box.
6. Click **Import Configuration**.

After successful import, the *REST API Example* and *DynamicCop* rulesets should be available in the section RuleSet.

7. Run the following command to export the CoP package.

```
curl --location --request POST 'https://xxx.xxx.xxx.xxx/exportGatewayConfigFiles' \
--header 'Authorization: Basic YWRtaW46YWRtaW4xMjM0' \
--header 'Content-Type: text/plain' \
--data-raw '{
  "nodeGroup": "LOB1",
  "kek": {
    "pkcs5": {
      "passphrase": "Passphrase1#",
      "salt": "salt"
    }
  }
}' -k -o cop_demo.json
```

For more information about exporting the CoP package, refer to the [step 12](#) in the section [Creating and Exporting the Ruleset](#).

8. Upload the CoP package to an Azure Storage Container or Azure File Share.

For more information about uploading the CoP package to an Azure Storage Container, refer to the [step 13](#) in the section [Creating and Exporting the Ruleset](#).

For more information about uploading the CoP package to an Azure File Share, refer to the [step 14](#) in the section [Creating and Exporting the Ruleset](#).

Important:

Ensure that the user has the required permissions to upload an object to the Azure Storage Container or persistent volume.

6.2.7 Creating Immutable Metadata Packages (IMP)

For Release 1

Set the following values for creating the IMP packages for release 1:

- policy/filepath: `https://<accountName>.blob.core.windows.net/<container name>/<pathToPolicy>`
- cop/filepath: `https://<accountName>.blob.core.windows.net/<container name>/<pathToCop>`

6.2.8 Deploying Release 1

1. Ensure that the `Values.yaml` file has the following configuration set on the Linux node.

```
blueDeployment:
  enabled: true
  policy:
    filepath: https://<accountName>.blob.core.windows.net/<container name>/<pathToPolicy>
    cop:
      filePath: https://<accountName>.blob.core.windows.net/<container name>/<pathToCop>
...
...
dsgService:
  tunnels:
    - name: "service1"
      port: 9443
      targetPort: 9443

ingress:
...
...
  nginx.ingress.kubernetes.io/ssl-passthrough: "true"
...
...
hosts:
...
...
  - host: "prod.example.com"
    httpPaths:
      - port: 9443
        prod: "/"
  - host: "staging.example.com"
    httpPaths:
      - port: 9443
        staging: "/"
```

2. Deploy Release 1 on the cluster.

For more information about deploying a release on the Kubernetes Cluster, refer to the section [Deploying a Release on the Kubernetes Cluster](#).

6.2.9 Running Sample Application (PostMan collection)

The component consists of the PostMan JSON file to generate the REST request for protecting data.

Post Request Path - <https://prod.example.com/protect>

Note:

If you want to test the sample application with the ESA, then you must specify the post request path as <http://restapi:9090/protect>.

Body (Data to be Protected)

The request is built using the *Sample App Test Data.csv* file. The contents of the .csv file are added to the *Body* of the request.

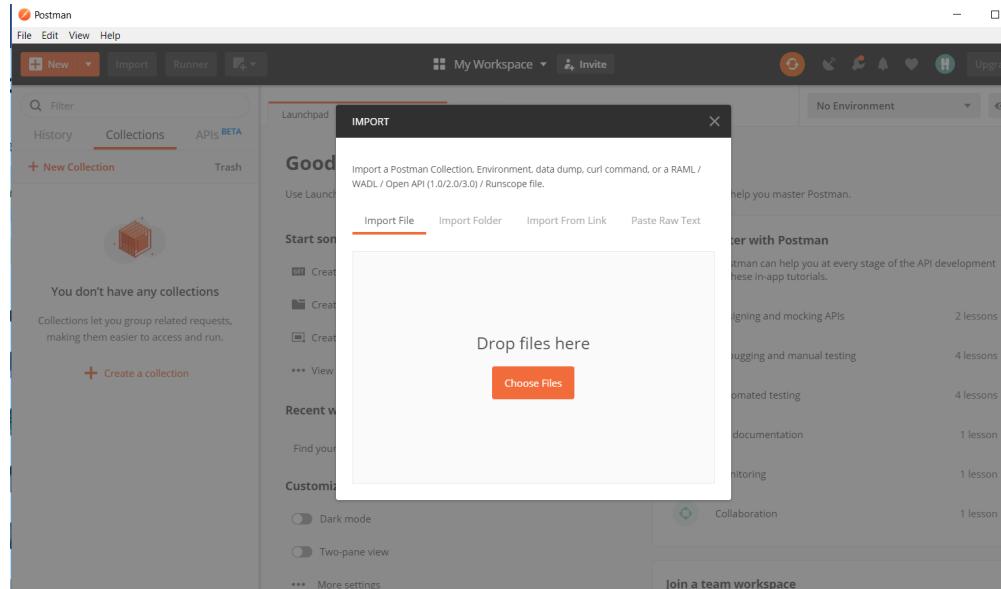
Header - DCOP request

The header is populated with the DCOP request, which protects data using the following Data Elements.

Field	Column Name	Token Type	Data Element Name
First Release			
First Name	first name	Alphanumeric	deFirstName
Last Name	last name	Alphanumeric	deLastName
Street	street address	Alphanumeric	deStreet
City	City	Alphanumeric	deCity
State	State Name	Alpha	deState
Zip Code	zip code	Alphanumeric	deZipcode
SSN	SSN	Numeric	deSSN

For protecting data with Release 1

1. Import the Postman collection.



2. After completing the import, the following two collections should be available.

The screenshot shows the Postman interface with a collection named 'DCoP_Sample'. It contains two POST requests:

- POST** https://prod.example.com/protect ...
- POST** https://staging.example.com/prote...

3. Navigate to the **Authorization** tab.

The screenshot shows the Postman Authorization tab. The 'Type' dropdown is set to 'Bearer Token'. A tooltip message reads: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)'.

Figure 6-7: Authorization Tab

4. Select **Bearer Token** as the type of authorization header from the **Type** drop-down list.

The **Token** field appears.

- In the **Token** field, specify the *access_token* that you have generated in [step 3](#) of the section *Enabling the Authentication Functionality*.
- Navigate to the **Headers** tab and specify the IP address of the Linux instances, from where you are sending the REST API request, as the value for the **X-Forwarded-For** key.

The screenshot shows the Postman Headers tab. A new header 'X-Forwarded-For' has been added, indicated by a checked checkbox next to it.

Figure 6-8: X-Forwarded-For Key

7. Select **Release 1** in DCoP_Sample and click **Send**.

User should get response as successful *200 OK* and receive protected Data.

For protecting data with Release 2

- Add the data element for *Customer ID* on the ESA.
- Create the IMP package R2.

For Release 2

Set the following values for creating the IMP package for release 2:

```
# policy and cop metadata configuration
job:
  ...
  ...
  securityConfigStorage: ObjectStore
  policy:
    filepath: https://<accountName>.blob.core.windows.net/<container name>/
    <pathToRelease2Policy>
```

```
cop:
  filepath: https://<accountName>.blob.core.windows.net/<container name>/<pathToRelease2Cop>
```

- Run helm upgrade for switching to Release 2.

For more information about upgrading the Helm Charts, refer to the section [Upgrading the Helm Charts](#).

- Ensure that the `values.yaml` file has the following configurations set on the Linux node.

```
greenDeployment:
  enabled: true
  securityConfigSource: ObjectStore
  policy:
    filepath: https://<accountName>.blob.core.windows.net/<container name>/<pathToRelease2Policy>
    cop:
      filepath: https://<accountName>.blob.core.windows.net/<container name>/<pathToRelease2Cop>
    ...
    ...

dsgService:
  tunnels:
    - name: "service1"
      port: 9443
      targetPort: 9443

ingress:
  ...
  ...
    nginx.ingress.kubernetes.io/ssl-passthrough: "true"
  ...
  ...
  hosts:
    ...
    ...
      - host: "prod.example.com"
        httpPaths:
          - port: 9443
            prod: "/"
      - host: "staging.example.com"
        httpPaths:
          - port: 9443
            staging: "/"
```

- Select **Release 2** in DCoP_Sample and click **Send**.

The user should get the response as successful *200 OK* and receive protected data.

6.2.10 Running Autoscaling Script

The Autoscaling script is used to issue continuous requests on the DSG containers. When the number of REST requests hitting the container are increased, with Horizontal Pod Autoscaling, Kubernetes automatically scales the number of pods based on the CPU utilization observed.

The user can run the autoscaling script from any Linux node, which can connect the external IP for the deployment.

Usage

```
./Sample_App_autoscale.sh <Ingress IP address> <CA certificate path> <Client certificate path> <Client certificate key path> [OAuth token] [IP address of the Linux instance]
```

After running this script, the user can observe that new pods are created to handle to the incoming traffic.

