IBM ECM System Monitor
Field Guides

Deploying ESM as Container

September 20st, 2024
CENIT AG
Michael Wohland

Content

Introduction

This guide will provide a step-by-step deployment information. All information will use basic tools and if we have, we provide examples as well.

You may need to adopt the commands to fit your container environment. As a requirement you need to know how to work within your container environment e.g.

- Authenticating to the API of your private cluster
- Providing container images through a private registry
- Configuring DNS and load balancers for accessing the ESM UI from outside the cluster.

The deployment is performed using a Helm Chart. An ESM instance installed with Helm (a “Helm Release”) can consist of:

- 0 or 1 ESM Servers
- 0 or 1 ESM Agents

For most scenarios, a single release consisting of 1 ESM Server and 1 ESM Agent is sufficient for typical environments to be monitored within a cluster.

Typically, we recommend installing the ESM Server in a VMWare or on a bare metal system as this provides additional functionality which cannot be used within a container ESM Server. Nevertheless, the deployment is also described here.

Prerequisites

ECM System Monitor

This guide is for customers running IBM ECM System Monitor (ESM). The container images and the Helm Chart are continuously improved on each release of System Monitor. If required, this guide will be updated to reflect these changes.

OpenShift

ECM System Monitor is supported on OpenShift (OCP) 4.x only. OpenShift 3.x versions are not supported.

Client Tools

- **`helm`** needs to be available (v3+). Helm 2.x is not supported.
- Command line tool compatible with your cluster's version is recommended. E.g. oc or az …
- To push the required container images to a Container Registry (such as the "IBM Container Registry"), you may use your preferred container runtime (such as **`docker`** or **`podman`**) or other tools (**`buildah`** or **`skopeo`**). In this document we will provide instructions for **`docker`**.
- The IBM Cloud CLI[1] (**`ibmcloud`**) will be required to interact with the IBM Container Registry. If you are using another registry (i.e. a private registry within the OCP cluster or another private registry on another cloud platform), the IBM Cloud CLI is not required.

[1] How to setup the IBM Cloud CLI: https://cloud.ibm.com/docs/cli?topic=cli-getting-started

First Step - Download ESM Container Images

Unfortunately, the ESM Containers are not part of any container registry. Therefore, the container files must be downloaded and imported manually to your “private” container registry.

The containers are available via:

- IBM Password Advantage (major and minor release versions)
- IBM Fix Central (feature pack and interim fix versions)

Although IBM Fix Central only provides ESM feature pack and interim fix versions, these containers are fully functional and do not require a container of a major or minor release version. Nevertheless, for the ESM Server it may be required to use certain versions due to Database updates that have to be performed when updating the system.

For better readability and since the IF packages also use this naming convention, please rename the tar.gz files after download to esmserver.tar.gz and esmagent.tar.gz.

Once done please extract the gz archive so you receive a tar file:

- ESM Server: **`esmserver.tar.gz -> esmserver.tar`**
- ESM Agent: **`esmagent.tar.gz -> esmagent.tar`**

If the extracted tar file does not have this name – please rename again.

Second Step - Import Container Images in (IBM) Container Registry

Example using docker for Openshift:

1) Setup access to the registry. This may require authentication with e.g. an IBM Cloud or Azure Cloud account .
   Notice: Your account needs to be member of a Resource Group which has permissions to use the “Container Registry Service”.
2) Import both images separately to your local container runtime:

```
docker load --input esmserver.tar
docker load --input esmagent.tar
```

3) This step is optional as you might already have a namespace in a registry. If not - Create a namespace for the images in the container. In this example we will use a private namespace called **`cenit`** in the **`de.icr.io`** registry (this registry/namespace will not be accessible to you).

4) Tag the images for the registry (replace registry/namespace “de.icr.io/cenit” with your registry/namespace – also use the correct version number of ESM for the tag – W = major version, X = minor version, Y = feature pack version and Z = interims fix version e.g. 5.6.0.0-000) :

```
docker tag esmserver:5.W.X.Y-Z de.icr.io/cenit/esmserver:5.W.X.Y-Z
docker tag esmagent:5.W.X.Y-Z de.icr.io/cenit/esmagent:5.W.X.Y-Z
```

5) Login to Container Registry.
6) Push the tagged images:

```
docker push de.icr.io/cenit/esmserver:5.W.X.Y-Z
docker push de.icr.io/cenit/esmagent:5.W.X.Y-Z
```

7) List the pushed images:

```
ibmcloud cr image-list
```

Output (truncated):

```
Repository                     Tag           Digest         Size
de.icr.io/cenit/esmagent       5.W.X.Y-Z     a7941079a4c7   367 MB
de.icr.io/cenit/esmserver      5.W.X.Y-Z     6d1b52f06af3   281 MB
```

The images are available in your registry namespace now.

Third Step - Add Helm Chart Repository

You can get the chart by adding the Helm repository:

`cenit-ag.github.io/helm-charts`

Run the following commands to add the repository to your Helm environment:

```
# Add repository
helm repo add cenit https://cenit-ag.github.io/helm-charts

# Update the repository index
helm repo update
```

Available versions of the chart can be listed with a search:

```
helm search repo cenit
```

Output:

```
NAME           CHART VERSION   […]
cenit/sm        1.2.0          […]
```

If the client you will be sending your commands from does not have an internet connection, you can pull the chart:

```
helm pull cenit/sm --version 1.2.0
```

This will download the chart as a .tgz file which can be used for an offline installation.

Notice: The `--version` flag is optional. If no version is stated, always the latest version will be downloaded.

The Helm chart repository also included extensive chart documentation. See following as an example for a specific chart version (`v1.2.0`):

https://github.com/cenit-ag/helm-charts/blob/main/sm/docs/sm-1.2.0.md

Fourth Step – Deploying Containers / Installation

Example deployment using helm

The deployment can be done using a simple helm command:

```
helm install mysm cenit/sm -f values.yaml -n esm
```

- **`mysm`** is the release name and will also be used for the agent name for the connection to the ESM server.
- **`cenit/sm`** points to the chart in the repository we added in Step 3 in this guide. This can also be replaced with the path to a **`.tgz`** archive containing the chart or a directory with the chart sources from within the **`.tgz`** archive.
- **`values.yaml`** is the values file that contains adjustments values for the helm chart deployment – examples follow below.
- **-n esm** namespace in which the deployment should be done – make sure to deploy the agent into the namespace where it should monitor information.
- Optionally the flag **`--dry-run`** can be added in order to simulate the Helm templating procedure on the client side without committing any changes to the targeted cluster. This is helpful in case you want to validate if your values YAML file (**`values.yaml`** in the example) is syntactically correct.

The command **`helm list`** will show all deployed releases.

Command **`helm status mysm`** will give the current status of the chart including the post-installation notes, which include some helpful commands for further troubleshooting.

Notice: After a successful helm installation attempt, the “NOTES” passage in the output features several **`kubectl`** commands to validate the installation. You can replace **`kubectl`** with **`oc`** to run these commands without having **`kubectl`** available on your command line – also see next chapter.

Example values.yaml files

Minimum values.yaml Server and Agent deployment:

For deploying ESM a minimum set of parameters (image repository and image tag) in the values.yaml is required – again, please make sure to replace repository and tag with the correct information.

Content of values.yaml:

```
server:
  image:
    repository: de/icr/io/cenit/esmserver
    tag: 5.W.X.Y-Z
agent:
  image:
    repository: de/icr/io/cenit/esmagent
    tag: 5.W.X.Y-Z
```

Minimum values.yaml Agent deployment only:

If you don´t want to deploy the ESM Server you have to disable it via the parameters:

Content of values.yaml:

```
server:
  enabled: false
agent:
  image:
    repository: de/icr/io/cenit/esmagent
    tag: 5.W.X.Y-Z
```

values.yaml Agent deployment connect to specific ESM server host:

If the agent should connect to an ESM Server that is not deployed within the same release name, you need to add the serverHostname parameter to the agent section. Make sure the required firewall ports are open. Specify the hostname optionally full qualified or IP-address:

Content of values.yaml:

```
server:
  […]
agent:
  […]
  serverHostname: myESMServerName.myDomain
```

values.yaml Agent adding secure (SSL) CPE certs to deployment:

For monitoring CPE information via SSL URL it is required to import certificates in the cacerts truststore of the agent jre. As we don´t want to do that within the container we want to mount the cacerts file as a secret during deployment and therefore we need to specify some additional parameters:

First: Get a copy of the cacerts and add all required certificates (root/inter/host).

Second: Create a secret e.g. with kubectl command (remember to redo that if certificates change):

```
kubectl create secret generic sm-cacerts --from-file=cacerts
```

Content of values.yaml:

```
server:
  […]
agent:
  […]
  extraVolumes:
   - name: cacerts
     secret:
       secretName: sm-cacerts
  extraVolumeMounts:
   - name: cacerts
     mountPath: /opt/sm/agent/jre/lib/security/cacerts
     subPath: cacerts
imagePullSecrets:
  - name: artifactorykey
```

values.yaml Server with external DB:

For running an ESM server container that uses an external database server like DB2 or MSSQL.

First: Create secret(s) for the Database Credentials:

```
kubectl create secret generic creds-conf-db --from-literal=user='YourUsr' --from-literal=password='YourPW'

kubectl create secret generic creds-mon-db --from-literal=user='YourUsr' --from-literal=password='YourPW'
```

Content of values.yaml for DB2:

```
server:
  […]
  externalConfigDatabase:
    enabled: true
    jdbcUrl: "jdbc:db2://mydb2:50000/smconfdb"
    jdbcDriverClass: "com.ibm.db2.jcc.DB2Driver"
    jdbcSecret: creds-conf-db
  externalMonitoringDatabase:
    enabled: true
    jdbcUrl: "jdbc:db2://mydb2:50000/smmondb"
    jdbcDriverClass: "com.ibm.db2.jcc.DB2Driver"
    jdbcSecret: creds-mon-db
agent:
  […]
```

Content of values.yaml for MSSQL:

```
server:
  […]
  externalConfigDatabase:
    enabled: true
    jdbcUrl: "jdbc:sqlserver://mysqlserver:1433;databaseName=smconfdb"
    jdbcDriverClass: "com.microsoft.sqlserver.jdbc.SQLServerDriver"
    jdbcSecret: creds-conf-db

  externalMonitoringDatabase:
    enabled: true
    jdbcUrl: "jdbc:sqlserver://mysqlserver:1433;databaseName=smmondb"
    jdbcDriverClass: "com.microsoft.sqlserver.jdbc.SQLServerDriver"
    jdbcSecret: creds-mon-db
agent:
  […]
```

Preparations for Logfile Monitoring

The following shows the steps that are necessary to monitor CPE and ICN log files by an ESM Agent pod.

The basic principles apply to all log files that should be monitored directly by an ESM Agent pod. The log files to be monitored must be accessible by a Persistent Volume Claim (PVC) and the agent’s deployment must then reference that PVC to be able to mount it in its own filesystem hierarchy.

Newer versions of ICN and CPE deployments are providing PVCs and Persistent Volumes (PV) for the log folders of CPE and ICN pods. All ICN and CPE pods in the deployment share the PVs for the log files.

The following table shows examples of PVCs and PVs for CPE and ICN deployments.

Product	Type	Name	PV Mount Path
CPE	PVC	cpe-logstore	N/A
CPE	PV	cpe-log-store	/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides
ICN	PVC	cpe-fnlogstore	N/A
ICN	PV	fnlog-store	/opt/ibm/wlp/usr/servers/defaultServer/FileNet

The agent deployment must include volume definitions in the **`volumes`** section so the PVCs can be referenced in the **`volumeMounts`** section for mounting. Here an example of the **`volumes`** section in the agent’s deployment YAML. The entries for these **`volumes`** are highlighted in bold for better readability only.

```
[…]
volumes:
  - name: data
    emptyDir: {}
  - name: p8lib
    persistentVolumeClaim:
      claimName: esm-vol-ext
  - name: my-cpe-logs
    persistentVolumeClaim:
      claimName: cpe-logstore
  - name: my-cpe-fnlogs
    persistentVolumeClaim:
      claimName: cpe-fnlogstore
[…]
```

The agent’s deployment template of the helm-chart already contains an entry for these logfile PVCs. You can find the entries in the deployment YAML below **`volumeMounts`**. The following shows an example of that section in an agent deployment YAML. The entries for these **`volumeMounts`** are highlighted in bold for better readability only.

```
[…]
imagePullPolicy: Always
      volumeMounts:
        - name: data
          mountPath: /opt/sm/agent/karaf/db
        - name: p8lib
          mountPath: /opt/sm/agent/p8lib
        - name: my-cpe-logs
          mountPath: /opt/sm/agent/karaf/log-volumes/my-cpe-logs
        - name: my-cpe-fnlogs
          mountPath: /opt/sm/agent/karaf/log-volumes/my-cpe-fnlogs
        - name: kube-api-access-7gcgg
          readOnly: true
          mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      terminationMessagePolicy: File
      image: 'de.icr.io/cenit/esmagent:5.6.0.0-000'
[…]
```

For the log monitoring itself you can use the given mount paths and enhance that with logfile names. As containers use specific log names, you probably have to use wild cards. The log file mechanism in ESM Version 5.6 was enhanced, to only read logs which have changed within the last 72 hours.

Additional Information for values.yaml

The helm chart contains templates for OpenShift installations below the “templates” folder and examples below the “examples” folder.

Fifth Step - Validation

Check Resource States

Check ESM Server for readiness (takes 120 seconds or more):

```
kubectl rollout status statefulset mysm-smserver
```

Check ESM Agent for readiness (takes 120 seconds or more):

```
kubectl rollout status deployment mysm-smagent
```

Check Logs

Check ESM Server logs:

```
kubectl logs mysm-smserver-0 -f
```

Check ESM Agent logs:

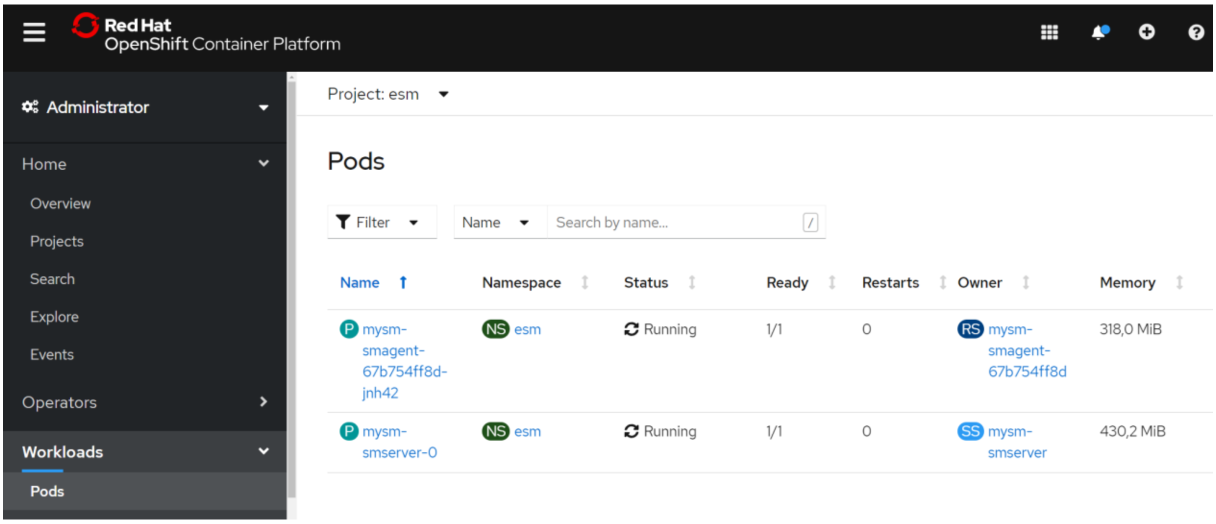
```
kubectl logs \
    $(kubectl get pod -n esm -l smagent-instance=mysm \
    -o=jsonpath='{.items[0].metadata.name}') -f
```

Show Pods

After a successful deployment the command **`oc get pod -n esm`** should show you the ESM Server and the ESM Agent Pods running and ready:

```
NAME                             READY    STATUS     RESTARTS    AGE
mysm-smagent-67b754ff8d-jnh42    1/1      Running    0           5m22s
mysm-smserver-0                  1/1      Running    0           5m22s
```

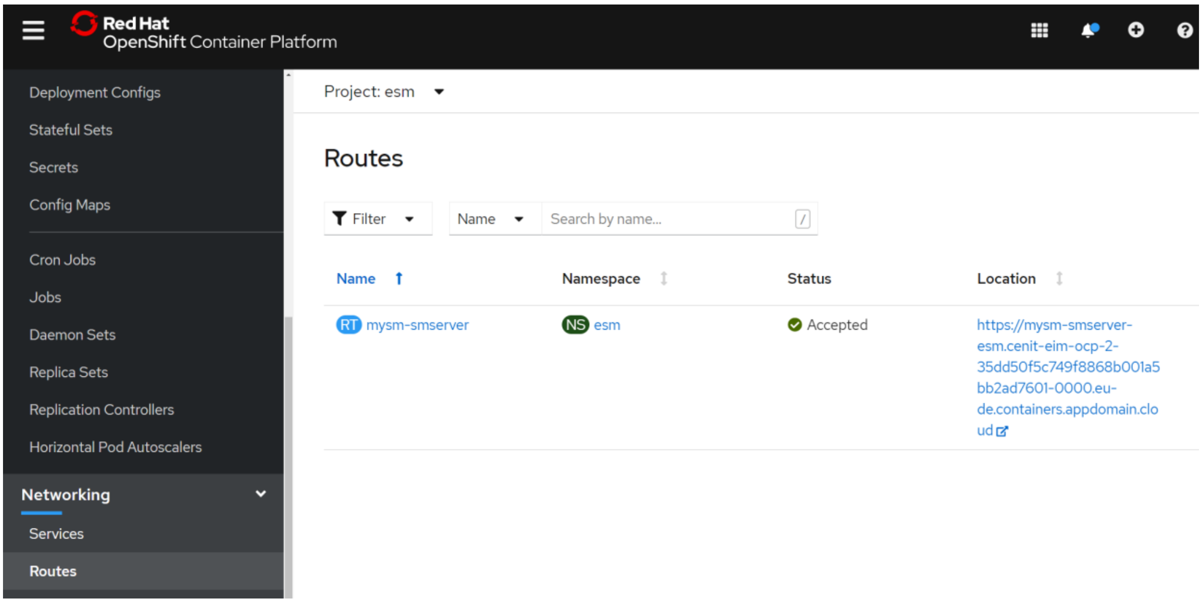
The running Pods can also be shown in the OpenShift Admin Console:



Name	Namespace	Status	Ready	Restarts	Owner	Memory
P mysm-smagent-67b754ff8d-jnh42	NS esm	Running	1/1	0	RS mysm-smagent-67b754ff8d	318,0 MiB
P mysm-smserver-0	NS esm	Running	1/1	0	SS mysm-smserver	430,2 MiB

Access ESM Server UI Using Route

The example configuration **`roks_persistent.yaml`** creates a Route that exposes the ESM Server UI through an external HTTPS endpoint:



Routes

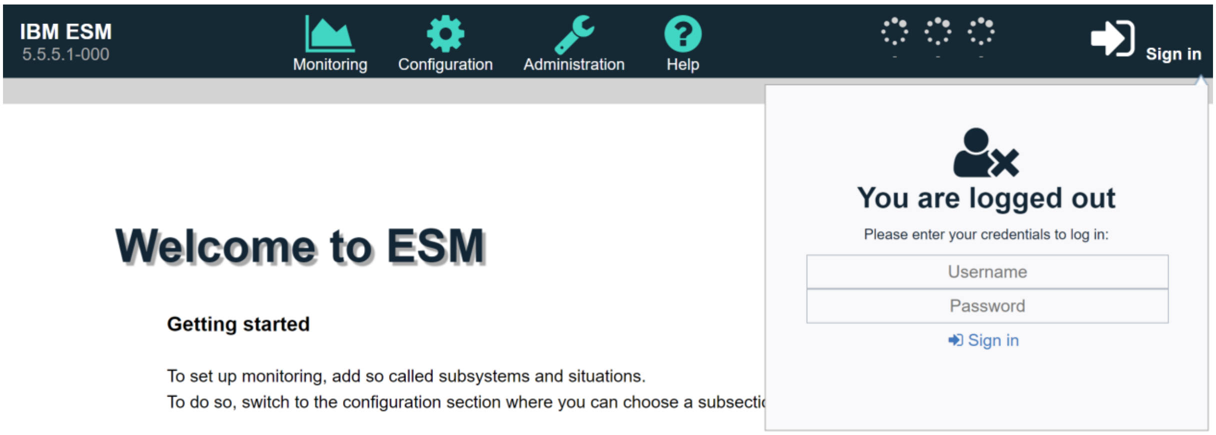
Name	Namespace	Status	Location
RT mysm-smserver	NS esm	Accepted	https://mysm-smserver-esm.cenit-eim-ocp-2-35dd50f5c749f8868b001a5bb2ad7601-0000.eu-de.containers.appdomain.cloud

Figure 1: Routes in OpenShift Console

The URL pointing to the service can also be obtained through the **`oc`** command:

```
RELEASE=mysm
echo "https://$(oc get route $RELEASE-smserver -o=jsonpath='{.spec.host}')"
```

This will load the ESM UI's login interface:



Welcome to ESM

Getting started

To set up monitoring, add so called subsystems and situations.
To do so, switch to the configuration section where you can choose a subsecti

You are logged out

Please enter your credentials to log in:

Figure 2: ESM UI Login Interface

To login to the UI, utilize user **`admin`** and the password you defined earlier when creating the Secret **`my-sm-admin-pw`**.

Access ESM Server UI Using Port Forwarding

If your cluster is configured in a way that external access is not permitted or technically impossible, you can alternatively tunnel the access to the ESM Server UI through **`kubectl`** or **`oc`** to your local client machine:

```
oc port-forward -n esm mysm-smserver-0 8443:8443
```

As long as the shell of this running command is kept open, you will be able to access the UI from the machine you executed the command using URL: https://localhost:8443

If no browser is available on the local system, use curl command for a connection test (should return **`200`**):

```
curl -skL -o /dev/null https://localhost:8443 -w "%{http_code}\n"
```

Uninstallation

Uninstall the Helm release using:

```
helm uninstall mysm
```

Before running this command, make sure to have the correct namespace context configured (using **`oc project`**). A safer approach is to explicitly state the namespace in the uninstall command:

```
helm uninstall mysm -n esm
```

This will remove all resources managed by the Helm chart. This does not include:

- Manually created secrets using oc (like the access to the container registry or for the ESM admin password).
- The Persistent Volume of ESM Server.

Although the creation of the ESM Server pods's Persistent Volume is initiated by the Helm Chart, the chart does not manage the volume, as this is accomplished using Dynamic Volume Provisioning. The positive side-effect of this behavior is that after an accidental uninstallation, all persisted data (i.e. configuration done in the UI or collected monitoring data) is not automatically deleted, but still retained.

To completely erase all resources related to ESM from the cluster, run the following commands:

```
# Delete PVC (PV should be deleted automatically then)
RELEASE=mysm
oc delete pvc -n esm data-$RELEASE-smserver-0

# Delete admin password secret
oc delete secret -n esm my-sm-admin-pw

# Delete registry access secret
oc delete secret -n esm all-icr-io
```

Contact Information

If you have any questions, please contact us at ECM.SystemMonitor@cenit.com.

CENIT AG
Phone: +49 711 7825 30
Email: ECM.SystemMonitor@cenit.com