

IBM Security Verify Information Queue Deployment Guide

Contents

Summary of Installation & Configuration Steps.....	3
Steps for Docker Swarm.....	4
Steps for OpenShift Container Platform (OCP).....	6
Common Final Steps	7
System Requirements	7
Port Usage	9
Docker Swarm Management	10
Docker Configs.....	10
Stateful Services	11
Configuring the YAML Files	12
Deploying the ISIQ stacks.....	12
Verifying the stacks are deployed	13
Removing the ISIQ stacks.....	13
Using the isiq shell script.....	13
Implementing configuration file changes.....	14
Upgrading to a new ISIQ release	14
Rolling Updates	16
Backups.....	16
High Availability	18
High Availability Options for InfluxDB.....	18
Tuning Requirements	18
Docker Log Rotation	19
Running setup.sh	20
Generating SSL Certificates	20
Other cfg directory contents.....	21

Product Password Encryption	22
Secure Communication to Applications	22
OpenShift Container Platform (OCP) instructions	23
OpenID	23
Shortcuts for Rapid ISIQ Setup.....	29
Deploying with Limited Internet Access	30
Proxy Considerations	30
ISIQ Event Auditing	31
Docker Cleanup	32
Appendix A. YAML Files	33
broker-stack.yml	33
connect-stack.yml	38
app-stack.yml	42
logs-stack.yml.....	45

IBM® Security Verify Information Queue, which uses the acronym “ISIQ”, is a cross-product integrator that leverages Kafka technology and a publish/subscribe model to integrate data between IBM Security products.

This Deployment Guide helps System Administrators install, configure, and secure their ISIQ environments.

© Copyright IBM Corporation 2019-2024

Summary of Installation & Configuration Steps

Here's a summary of the steps required to install and configure ISIQ. These steps will be elaborated on later in this guide. The cases where different instructions apply for Docker Swarm vs. Red Hat OpenShift Container Platform (OCP) have been marked as such:

1. Extract the starter kit on your docker host machine:

- The latest version of ISIQ can be found on its starter kit page at

<https://www.ibm.com/support/pages/ibm-security-information-queue-starter-kit>

2. Create the nginx certificate:

- After you extract the contents of the starter kit .zip file, go to the <starterKitDir>/cfg/nginx directory.
- Update the "[alternate_names]" list in cert.conf to secure access to the nginx web server that runs inside ISIQ. Specify for "DNS.1", "DNS.2", and so on, the fully qualified hostnames where ISIQ is installed, and any hostname aliases. Be sure to include the hostname that you will enter in your browser when logging in to ISIQ.

The "[alternate_names]" list is used to generate the nginx certificate. If you're deploying a cluster, "[alternate_names]" should also include the hostnames of your cluster nodes and, if you're deploying ISIQ on OCP, the infrastructure node. Save the cert.conf changes.

- Run <starterKitDir>/isiq setup`. The isiq shell script simplifies and consolidates common ISIQ configuration and management functions. For example, `isiq setup` combines this nginx certificate Step 2 with Step 6. Refer to [Using the isiq shell script](#) for details.

OR

If you prefer a more granular approach to security configuration,

- Run setup.sh in the <starterKitDir>/cfg directory. It calls other scripts to create the nginx certificate and the JSON Web Token (JWT) files that ISIQ requires for application data protection. To successfully run either `isiq setup` or setup.sh, Java must be installed. Refer to [Running setup.sh](#).

Notes:

- If you're deploying ISIQ on OCP, you don't have to run `isiq setup` or setup.sh. The createConfig.sh script referenced in OCP Step 7 handles the creation of the nginx certificate.
- If you're deploying ISIQ on Docker Swarm, you can temporarily circumvent Step 2. See the section, "Shortcuts for Rapid ISIQ Setup". The documented shortcuts are intended for proof-of-concept (PoC) demo situations and should **not** be your regular mode of operation.

3. Set up an OpenID Connect (OIDC) provider to authenticate ISIQ logins:

- Follow the instructions in the "OpenID" section for IBM Security Verify Governance (previously known as IBM Security Identity Governance and Intelligence or "IGI") or the instructions for using IBM Security Verify as an OIDC provider (see <https://docs.verify.ibm.com/gateway/docs/tasks-oidc-rp-verify>). You can also use IBM Security Verify Access (previously known as "ISAM") or any other OIDC provider.
- Edit ISIQ's cfg/oidc/oidcSettings.json file. Update the credentials and URLs as needed to work with your OIDC provider. Save the changes.

Note: As a temporary measure to get ISIQ up-and-running quickly, you can bypass OIDC authentication and log in directly to ISIQ. See the section, "Shortcuts for Rapid ISIQ Setup". But bypassing OIDC should **not** be your regular mode of operation.

4. Update the kernel to support Elasticsearch (only required if you deploy the optional logs stack):

- Edit /etc/sysctl.conf to set vm.max_map_count to 262144

Editing `sysctl.conf` doesn't take effect until you reboot the host machine. To implement the change immediately, run the command: `sysctl -w vm.max_map_count=262144`

Note: Throughout this guide, the ISIQ-supplied logs stack is referred to as optional. You can decide to not run the logs stack, or to implement monitoring and logging with different component parts. If you choose to deploy the ISIQ-supplied logs stack (by setting `ENABLE_LOG_STACK=true` in the `isiq` shell script), we have the following recommendations:

- (a) Treat the out-of-the-box `logs-stack.yml` as a starting point. It's an example to help you set up an ISIQ self-monitoring solution. You will likely want to customize the example for your needs.
- (b) Follow the advice of the various third-party vendors who provide the logs stack docker images for Elasticsearch, Logstash, Kibana, Grafana, and so on.
- (c) Adhere to each vendor's licensing and usage rules.
- (d) Regularly update your `logs-stack.yml` to pull the vendor's latest docker images in order to obtain fixes and remediate security vulnerabilities.

5. Prepare for application-level security and customization:

- If you will be setting up SSL connections to your endpoints such as IGI databases and LDAP directories, refer to the "Secure Communication to Applications" section. If you will be customizing how your data is integrated, refer to Appendix E in the [ISIQ User's Guide](#), which has information about modifying `<starterKitDir>/cfg/connect/txdef.json` and `isimConfiguration.json`. Note: These customizations can be implemented later if necessary.

At this point, the installation & configuration steps diverge depending on whether you're using Docker Swarm or Red Hat OpenShift Container Platform as your container orchestrator:

Steps for Docker Swarm

6. Pull the required docker images:

Waiting for dynamic docker image downloads during the initial ISIQ stack deployment can take a while. Therefore, we recommend manually pulling the required images ahead of time, which is accomplished with the following commands:

- `docker login`

- Run `<starterKitDir>/isiq setup`` (note: if you did not already run ``isiq setup`` in Step 2)

The ``docker login`` command is not necessary to pull the ISIQ images. However, some of the third-party component images used by ISIQ such as Kafka, nginx, vault, and Redis might require an initial login before downloading. If you do not already have a Docker account, you can obtain one at <https://hub.docker.com/signup>.

Alternatively, you can run `<starterKitDir>/isiq images``. This script option reads `*.yml` files located under the `<starterKitDir>/yml` directory and calls ``docker pull $image`` for each "image: xxxx" tag.

If your ISIQ host is isolated with limited Internet access, there is a procedure for pulling the docker images on a separate host system that does have Internet access (even if it's temporary). You then transmit the images over your internal network to the ISIQ host. See the section, "Deploying with Limited Internet Access".

You will need approximately 8 - 10 Gigabytes of free space on the host where you download all of the docker images.

7. Decide whether to run ISIQ in a single-node or cluster configuration (you can change your decision later):

- Under <starterKitDir>/yml, there are “single_node” and “cluster” subdirectories with corresponding .yml files. The advantages of a multi-node cluster include fault tolerance, workload balancing, and rolling updates. (The Kafka technology that ISIQ uses needs an odd number of nodes to perform its automatic failure recovery. That’s why the .yml files in <starterKitDir>/yml/cluster specify a replication factor of 3.) But it might not be feasible for you to allocate three or five systems for ISIQ, in which case a single-node configuration is more suitable.

ISIQ runs perfectly well in a single-node configuration and can handle high-volume traffic. However, the tradeoff is that you lose the HA benefits of a cluster. One strategy would be to start your ISIQ deployment on a single node during the testing and familiarization phases, and then set up a multi-node cluster for your production environment.

8. Create the Docker Swarm:

- On the system you’ve designated as the manager of your cluster, or on your single-node system, run ``docker swarm init``.

Note: This command is automatically run for you, with default parameters, when you use the ``isiq setup`` script. However, if your system has more than one network interface, the command will fail with an error stating that the docker daemon “could not choose an IP address to advertise since this system has multiple addresses - specify one with `--advertise-addr``”. In that circumstance, you need to choose an IP address that will serve as the external listener of your swarm. Then, run:

- `docker swarm init --advertise-addr x.x.x.x:port#``

Replace x.x.x.x with the address you’ve chosen to use. The default port# for a swarm is 2377. Once ``docker swarm init`` completes successfully, rerun ``isiq setup`` to finish the other initialization steps.

- If you’re deploying a cluster, run ``docker swarm join --token xxx`` on each additional node, where “xxx” is the worker token returned when running ``docker swarm init`` on the manager node. For more information, see https://docs.docker.com/engine/reference/commandline/swarm_join/

9. Add metadata to each node to allow pinning tasks to them, for example:

- `docker node update --label-add isiq=node1 alpha``
- `docker node update --label-add isiq=node2 bravo``
- `docker node update --label-add isiq=node3 charlie``

In this example, the hostnames of your nodes are “alpha”, “bravo”, and “charlie”.

Note: This step is only applicable if you’re deploying a multi-node cluster. In a single-node configuration, labeling is unnecessary.

10. Edit the .yml files (also known as YAML files) if needed. The out-of-the-box .yml files are usually adequate as is, but there might be environment variables you want to modify or other customizations to make. If you’re running a Docker Swarm, the ISIQ .yml files follow the Docker Compose specification. For more information about the syntax and parameters of Docker Compose files, see <https://docs.docker.com/compose/compose-file/>

11. Deploy the ISIQ stacks:

- `<starterKitDir>/isiq start``

OR

If you prefer a more granular approach to managing ISIQ,

- docker stack deploy --compose-file broker-stack.yml broker
- docker stack deploy --compose-file connect-stack.yml connect
- docker stack deploy --compose-file app-stack.yml app

And optionally

- docker stack deploy --compose-file logs-stack.yml logs

12. Verify the stacks are deployed:

Run the `docker service ls` command. Refer to the "Verifying the stacks are deployed" section to help you interpret the command output.

Steps for OpenShift Container Platform (OCP)

These next steps assume you're deploying ISIQ on an existing OCP with Helm installed:

6. Edit openshift/values.yaml. Replace SPECIFY_ISIQ_HOSTNAME_USED_FOR_URL with the hostname of your ISIQ server. Under "storage:", set "className:" to the OCP storage class you're using.

Review the rest of the values.yaml file. It has ISIQ environment variable settings such as oidcBypass, oidcSelfSignedCerts, and autoImportSslCerts that you might want to override.

7. Define ConfigMaps for the various ISIQ components. The createConfig.sh script creates the nginx certificate that ISIQ requires, and it populates all of the YAML files that define the ISIQ application: openshift/createConfig.sh all

8. For an initial install of ISIQ on OCP, run the following oc command:

```
oc create -f openshift/000-isiq.yaml
```

9. Manage the security context constraints:

```
oc adm policy add-scc-to-user privileged -z default -n isiq
```

```
oc adm policy add-scc-to-user privileged -z filebeat -n isiq
```

```
oc adm policy add-scc-to-user anyuid -z default -n isiq
```

Note: Like Step 8, Step 9 only needs to be run during first-time ISIQ setup on OCP.

10. If migrating from a previous ISIQ release, run the following commands to upgrade your ISIQ OCP installation:

```
cd <new_starter_dir>/cfg
```

```
cp -r <old_starter_dir>/cfg/* .
```

```
cd ../openshift
```

```
cp <old_starter_dir>/openshift/values.yaml .
```

```
./createConfig.sh all
```

```
oc apply -f yaml
```

The `createConfig.sh all` command will bring over your existing config files, turn them into new ConfigMaps (which should not have changed as a result of the upgrade), and also process all the files in the template directory into the yaml directory. When the files are applied with `oc apply -f yaml`, the changes will take effect in OCP.

11. In the OCP admin console, navigate to Networking -> Routes. Be sure to change the Project selection from "default" to "isiq". Click "Create Route" and specify the following details:

Create Route

Name: login

Hostname: <infrastructure node hostname>

Path: /

Service: nginx

Target Port: 443

Secure Route (checked)

TLS Termination: Passthrough

Insecure Traffic: Redirect

12. The initial ISIQ startup on OCP will take a while (as much as 15-30 minutes is possible) depending on the speed at which the containerized images can be pulled.

Common Final Steps

13. Log in to the ISIQ console to validate that the prior steps were successfully completed. The ISIQ login URL is:

<https://<your-ISIQ-hostname>/#!/login>

For more information about connecting to the ISIQ console, see the “Logging In” section of the [ISIQ User’s Guide](#).

14. (Optional) If you enabled the ISIQ-supplied logs stack, perform the following actions:

A) Update logs-stack.yml. Under the “kibana:” service definition, edit the SERVER_PUBLICBASEURL environment variable. Replace [ISIQ_SERVER_URL] with your ISIQ hostname or IP address. You must update logs-stack.yml or the logs_kibana service will fail during startup.

B) Load the Grafana dashboard using these steps:

- Log in to Grafana at <https://<your-ISIQ-hostname>/grafana>. The default credentials are admin/admin.
- Optionally change the password or click Skip.
- Click the “Add data source” icon. This may require first clicking the “cog” icon on the left sidebar, which displays the configuration options, including “Data Sources”.
- Select “InfluxDB” as the data source type.
- Provide a name (“ISIQ Engine Metrics”).
- For the URL, specify the InfluxDB server you are using, <http://influxdb:8086>.
- Leave Access as “Server”. Finally, provide the database name of “isiq_broker”.
- Nothing else needs to be specified on this page.
- Click “Save & Test” to save and validate this entry.
- On the left sidebar, click the “Dashboards” icon to view its menu.
- Select “Import” from the Dashboards menu.
- Click the “Upload .json File” button.
- Select the “isiq_broker_dashboard.json” file in `cfg/grafana`
- Set the “Influx” line to the data source you created.
- After you click “Import”, you see graphs of all the data being collected.

System Requirements

ISIQ is packaged as a set of docker images and requires Docker Swarm or OCP to be configured. Here are the system requirements:

Minimum Software/Hardware: Docker Community Edition (CE) 19.03 running on Debian, Fedora, Ubuntu, or CentOS with x86_64/amd64, 16 GB RAM, 2 vCPU, 25 GB free disk space per node. Docker CE is also referred to in the docker documentation as “Docker Engine - Community”.

Note that these are the minimum settings. The Kafka broker in particular benefits from additional RAM when processing tens of thousands of entities. Refer to the [Confluent System Requirements](#) web page for details.

Recommended: Docker CE 19.03 or higher running on a Linux OS with x86_64/amd64, 32 GB RAM, 4 vCPU, and 100 GB free disk space per node.

Instructions for installing docker can be found here: <https://docs.docker.com/engine/install/>. In the navigation sidebar, look under “Installation per distro” to find instructions for your Linux OS. The rule of thumb is: ISIQ can be installed on any Linux distro (with the exception of the s390x platforms) where Docker CE is available.

Running ISIQ on Windows or MacOS is not currently supported.

The ISIQ web interface requires a minimum browser version of Mozilla Firefox 78, Google Chrome 88, Microsoft Edge 90, or Apple Safari 14.

ISIQ supports cross-product integration for IBM Identity Manager (referred to as “IM” but previously known as IBM Security Identity Manager with the “ISIM” acronym), IGI, and IBM Cloud Identity Analyze:

- ISIQ supports IBM Security Verify Governance (ISVG) v10.0.0 or higher. For backward compatibility, ISIQ also supports IGI v5.2.5 FP1 or higher.
- ISIQ supports the IM that’s part of ISVG v10.0.0 or higher. For backward compatibility, ISIQ also supports ISIM v6.0 FP18 or higher, and ISIM v7.0 FP7 or higher. ISIQ supports all database types in both IM/ISIM and ISVG/IGI.

Notes:

1. ISVG/IGI **must** be customized in several ways to allow for correct interoperability with ISIQ. For the list of required customizations, see “Appendix A: IGI Customizations for ISIQ” in the [ISIQ User’s Guide](#). If you will be using ISIM as a data source, you **must** create an ISIM LDAP index to optimize retrievals by ISIQ. For information about creating the index, see the “Tuning Requirements” section later in this guide.
2. In order for ISIQ to process deleted ISIM entities such as users, accounts, and roles, the “change log” option in the IBM Security Directory Server for ISIM **must** be configured. The details for configuring it can be found [here](#) and in “Appendix B: ISIM Customizations for ISIQ” in the [ISIQ User’s Guide](#).
3. The Docker website <https://docs.docker.com/engine/install/#server> does not list Red Hat as a supported Linux distribution (other than RHEL s390). Although some ISIQ customers have had success running Docker CE on RHEL 7—by pulling Docker CE from the CentOS 7 repository—there is risk involved since the Docker CE-on-Red Hat combination is not officially supported by Docker or Red Hat. By the time RHEL 8 was released, the method of using CentOS did not work any longer. At that point, the only viable options were to either install Docker CE and ISIQ on a different Linux distribution, or to run ISIQ on Red Hat but to use a Kubernetes-based implementation such as OpenShift in place of Docker.
4. For customers who wish to run ISIQ on a non-OpenShift Kubernetes distribution such as K3S or MicroK8S, the starter kit also includes a /kubernetes directory with scripts and YAML files for deploying ISIQ in a Kubernetes cluster.

- Customers deploying ISIQ on Amazon Web Services have been able to use AWS Linux 2 as their OS platform. They installed Docker CE via the “Extras” mechanism of AWS Linux 2.
- Root access is generally not required to administer ISIQ. For example, you can run docker commands and the isiq shell script with a non-root ID. But root access is required if you need to stop or start the docker daemon or access docker system files under the /var/lib/docker folder. Docker Engine 19.03 introduced a “rootless” mode for running the docker daemon (for details, see <https://docs.docker.com/engine/security/rootless/>). This mode is **not** feasible for ISIQ, which relies on an overlay network for docker containers to communicate with each other. Overlay networks are one of the features that rootless mode does not support. For ISIQ, the docker daemon must run as root so that overlay networks will work.

Port Usage

To help plan your ISIQ deployment, the following table lists the ports used in a full ISIQ implementation that includes ISIM and IGI.

Product	Component	Server	Port/Protocol	Notes
docker	Docker Swarm	swarm manager/worker	22/tcp 2376/tcp 2377/tcp 7946/tcp, udp 4789/udp	https://www.digitalocean.com/community/tutorials/how-to-configure-the-linux-firewall-for-docker-swarm-on-ubuntu-16-04
ISIQ	ISIQ	ISIQ Management Web Interface & REST API	80/http 443/https	All ISIQ accesses, both GUI and REST API calls, go through nginx port 443. Any requests to 80/http are redirected to 443/https.
ISIM	ISIM	ISIM App Server	9080/tcp (or as configured)	
ISIM	ISIM	ISIM LDAP Server	389/ldap 636/ldaps (or as configured)	
ISIM	ISIM	ISIM Database	50000/tcp or 1521/tcp (or as configured)	
IGI	IGI	IGI Database	50000/tcp or 1521/tcp (or as configured)	
IGI	IGI Virtual Appliance	IGI OIDC Provider	9443/tcp 10443/tcp 11443/tcp	Required for Admin authentications and ISIQ communication with IGI and ISIM
IGI	IGI Service Center	IGI App Server	9343/tcp	

	and Admin Console			
--	-------------------	--	--	--

Docker Swarm Management

Docker Swarm can manage a cluster of docker nodes with the benefits of redundancy, failover, and scaling. After you perform initial ISIQ configuration, you need to do only minimal swarm management. But if you find later that changes are necessary, docker supports adding and removing nodes, altering whether a node acts as a manager or not, and modifying the metadata of a node.

The docker documentation provides all the information required to work with a swarm over its lifecycle. It is highly recommended that you familiarize yourself with the topics under "Scale your app", starting with the swarm mode overview at <https://docs.docker.com/engine/swarm/>

The Docker Swarm Administration Guide can be found here:

https://docs.docker.com/engine/swarm/admin_guide/

To set up a multi-node cluster, the basic steps are:

1. On the first node, known as the manager node or leader node, run ``docker swarm init`` (note: this command is automatically run if you use the ``isiq setup`` script)
2. On each additional node, known as a worker node, run ``docker swarm join --token xxx``

In a multi-node cluster, verify that there's network connectivity between the nodes, and that the Docker Swarm ports are reachable. It's helpful to run a port check on each of the nodes using a network connection tool like netcat. You can also run ``sudo iptables -L INPUT`` to ensure there are no firewall rules blocking inbound traffic over ports needed for intra-cluster communication. Worker nodes must be permitted to listen on ports 7946 (TCP/UDP) and 4789 (UDP). Manager nodes must also listen on port 2377 (TCP). ZooKeeper connections between nodes use TCP ports 2181 and 3888.

Docker Swarm requires a functioning Domain Name System (DNS) to resolve the addresses of nodes in a cluster. You cannot configure a cluster's network by simply updating the `/etc/hosts` file on each docker node, nor can you specify dotted decimal IP addresses in place of symbolic hostnames. For more information about internal docker networking, its use of DNS, and how it overlays network definition files inside the docker container, see <https://docs.docker.com/config/containers/container-networking/>.

Docker Configs

Docker Configs is an optional feature that allows small (<500k) text and binary files to be shared among services in a Docker Swarm without the need to create a separate volume or build a new image. This feature allows docker images to be kept as generic as possible for easier management.

When created or managed manually, a config is immutable, and the name of the config is tied to the contents of the file, not to the file itself. In other words, simply replacing the file will not update any of the containers in your ISIQ swarm. However, when defined in a YAML file and deployed as a stack, this constraint does not present an issue. The config definition exists only while the stack is running, which means that updating the file, and then redeploying the stack, refreshes the config with the updated contents of the file. In most cases, docker config files are read-only during startup, and so a refresh of the stack is necessary to use a new or changed config.

For more information about configs, see <https://docs.docker.com/engine/swarm/configs/>.

Stateful Services

ISIQ relies on two types of containers: stateful and stateless. The majority are stateless, either because they are entirely self-contained (rest, web-app, and nginx), or they store their data elsewhere (kafka-rest1, connect, and products). Among ISIQ's stateful containers, there are two types:

(1) The first type of stateful container needs to be pinned to a specific node. By default, docker attempts to balance the workload across the swarm and for stateless containers, this balancing performs well. When a container must access external data to maintain state, the data must exist on the machine where the container is running. Docker Swarm does not currently replicate its volumes across a cluster, and so we must guarantee that the data a container requires is present on the node running it. We can force a container to run on a specific node by using a "constraint" during service creation. More details about docker's service constraint mechanism can be found [here](#).

In the YAML files in <starterKitDir>/yaml/cluster, the containers are configured for a sample three-node cluster. Before you start the cluster, you must ensure the configuration values are updated to reflect your actual environment. These values are **required** for ISIQ to function correctly in a cluster. In the following example, `docker node ls` was used to list the members, and the hostnames found were: alpha, bravo, and charlie. Those hostname values are then leveraged to provide a hook for the constraint, as shown in the following `docker node update` commands:

```
docker node update --label-add isiq=node1 alpha
docker node update --label-add isiq=node2 bravo
docker node update --label-add isiq=node3 charlie
```

In ISIQ's broker-stack.yml file, each of the three Apache ZooKeeper services is linked to a specific node. ZooKeeper replicates its data to every container. By running one instance on each node, coupled with ZooKeeper's internal replication, you can achieve high availability. If you have greater than three nodes in your cluster, create more ZooKeeper services, one per node (note: you cannot use the "global" deployment option since every ZooKeeper instance requires a unique hostname for its clustering feature to work). Similar logic applies to ISIQ's three Kafka containers. Kafka handles its own data replication. Therefore, each node has its own instance and volume. If a node goes down, Kafka still has access to all of its data.

(2) ISIQ's second type of stateful container is one that might benefit from a network-mounted storage solution such as NFS or Samba. Grafana is an obvious candidate for this method. The optional logs stack includes Grafana, though you might already have a logging/monitoring stack you prefer to use instead. Grafana can run anywhere, but it does need to persist details about its database configuration and any dashboards or other created elements. Because Grafana does not replicate data, if one node goes down and Grafana restarts elsewhere, it cannot locate its data. To address this situation, the recommendation is to create an NFS mount point that is available to all nodes in the swarm, and configure a bind mount to the Grafana container. The supplied logs-stack.yml has the volume mount defined with a target of /var/lib/grafana. Changing the type to "bind" and specifying the NFS mount point as the source, as well as removing the default deployment constraint, allows Grafana to run on any node.

The other potential candidate for NFS is the InfluxDB container. Here, you have several options. InfluxDB is both a required part of ISIQ's connect stack and an optional part of the logging stack. In the latter stack, JMX statistics are pulled from Kafka by "jmxtrans" and deposited in InfluxDB for use by Grafana. There is no reason the YAML files cannot be configured to leverage the connect stack InfluxDB for both use cases. On the other hand, you might already have a monitoring stack that you would like to hook into. There is also the issue of whether to configure influxdb-relay or to upgrade to InfluxDB Enterprise,

which can provide clustering capability. More information on this subject can be found in the “High Availability” section.

If InfluxDB runs in stand-alone mode, we recommend you configure an NFS mount point that is accessible from all nodes in the swarm. Also, make sure the service is not constrained to a specific node and that the volume is defined as a bind mount. Example definition:

```
influxdb:
  image: influxdb:1.8.10-alpine
  volumes:
    - type: bind
      source: /var/lib/influxdb
      target: /var/lib/influxdb
```

By contrast, if either of the HA solutions is implemented, then each service must have a unique name and be pinned to a particular node with a named volume. Example definition:

```
Influxdb1:
  image: influxdb:1.8.10-alpine
  volumes:
    - isiq_influxdata:/var/lib/influxdb
  deploy:
    placement:
      constraints: node.labels.isiq == node2
```

Configuring the YAML Files

The ISIQ-supplied YAML files assume either a single-node configuration or a three-node cluster. If you intend to have five or seven nodes in your cluster, the files would need to be adjusted.

The supplied YAML files offer usable defaults wherever possible. However, there are several areas, such as specifying constraints in broker-stack.yml, and specifying secrets in app-stack.yml, where it might be necessary to review and modify the defaults.

See [Appendix A. YAML Files](#) for a detailed description of the contents of ISIQ’s YAML files.

Deploying the ISIQ stacks

To deploy the ISIQ stacks, run the script, <starterKitDir>/isiq start

The script issues a set of `docker stack deploy` commands, one for each stack. If for any reason, you need to deploy the stacks individually, this is what the commands look like:

```
docker stack deploy --compose-file broker-stack.yml broker
docker stack deploy --compose-file connect-stack.yml connect
docker stack deploy --compose-file app-stack.yml app
And optionally
docker stack deploy --compose-file logs-stack.yml logs
```

The "--compose-file" flag can be abbreviated to "-c".

Verifying the stacks are deployed

To confirm that ISIQ images started correctly, use the ``docker service ls`` command. In the replicas column of the command output, the numerator and denominator should match (for example, 1/1 or 3/3). If that's the case, then docker believes everything that should be started is started. In other words, all replicas for all services in your xxxx-stack.yml files are initialized.

In addition to ``docker service ls`` command output, you can also check for particular log messages that indicate successful initialization. If you deployed the optional logs stack, the messages may or may not exist in Elasticsearch and Kibana. As an alternative, you can search the native docker service logs for the following messages:

```
docker logs $(docker ps -q -f name=broker_kafka1) | grep "started (kafka.server.KafkaServer)"
docker logs $(docker ps -q -f name=connect_connect) | grep "Herder started
(org.apache.kafka.connect.runtime.distributed.DistributedHerder)"
docker logs $(docker ps -q -f name=connect_connect) | grep "Finished starting connectors and tasks
(org.apache.kafka.connect.runtime.distributed.DistributedHerder)"
docker logs $(docker ps -q -f name=connect_vault) | grep "core: vault is unsealed"
docker logs $(docker ps -q -f name=app_products) | grep "Jetty Server and the registry started"
```

Removing the ISIQ stacks

To stop the ISIQ stacks, run the script, `<starterKitDir>/isiq stop`

The script issues a set of ``docker stack rm`` commands, one for each stack. If for any reason, you need to stop the stacks individually, this is what the commands look like:

```
docker stack rm logs (only if the logs stack was started)
docker stack rm app
docker stack rm connect
docker stack rm broker
```

Due to dependencies between stacks, you need to remove them in the reverse order from how the stacks were deployed. Since ISIQ is configured with Docker Swarm, running ``docker stack rm`` from the manager node removes the stack whether you are using a multi-node cluster or a single-node configuration.

The ``docker stack rm`` command removes all services in a stack and stops the containers for each service. Before you restart ISIQ, you can verify that services were removed, and containers stopped, by running the ``docker service ls`` and ``docker ps`` commands.

If you are stopping ISIQ with the intent to immediately restart it, you can perform all the steps with just one script invocation:

```
<starterKitDir>/isiq restart
```

Stopping ISIQ does not delete associated data in Kafka topics. That data persists on the ISIQ volumes.

Using the isiq shell script

As mentioned earlier, you can simplify common management functions by using the "isiq" shell script in the starter kit root directory. For example, instead of (a) running `<starterKitDir>/cfg/setup.sh` to configure your certificate for nginx, and (b) running a series of ``docker pull`` commands to download

ISIQ's images, you can run ``isiq setup``. Instead of deploying and removing individual ISIQ stacks, you can run ``isiq start``, ``isiq stop``, or ``isiq restart``, which consolidate multiple docker commands into one script invocation. To list the script's options, run ``isiq help``.

Usage Notes:

- The `isiq` script assumes you are running ISIQ from a directory tree that was extracted from the starter kit .zip, with its relative pathing to `/cfg`, `/yml`, etc. If you choose to run ISIQ out of a different directory structure, then you need to either not use the script, or modify it to correspond to your path layout.
- If you aren't deploying the optional logs stack, keep the default `ENABLE_LOG_STACK=false` setting in the "User Configuration" section of the script file. With this setting in effect, the `isiq` script does not attempt to start or stop the logs stack.
- ``isiq support`` is a useful command if you're collecting documentation to report an ISIQ problem. The command obtains your current docker and OS levels, along with copies of your YAML and config files, and zips up this data into a file that you can transmit to IBM Customer Support.

Implementing configuration file changes

After you install and run ISIQ, you might need to make a change to a configuration file such as `<starterKitDir>/cfg/oidc/oidcSettings.json` or `<starterKitDir>/cfg/connect/txdef.json`. The process for implementing configuration file changes will vary depending on whether you're using Docker Swarm or OpenShift Container Platform (OCP).

For Docker Swarm, you must restart the stack(s) that reference the modified configuration file. For example, `oidcSettings.json` appears in `app-stack.yml`, and so the app stack must be restarted after modifying `oidcSettings.json`. The `txdef.json` file appears in both `connect-stack.yml` and `app-stack.yml`. Therefore, the connect and app stacks must both be restarted after modifying `txdef.json`. When using Docker Swarm, it's generally simpler to run the ``isiq restart`` command to implement a configuration file change, rather than attempt to individually restart each affected stack.

For OCP, the implementation process is as follows:

1. Modify your configuration file, e.g., `txdef.json`
2. Run `createConfig.sh` against the modified configuration file, e.g.:
`openshift/createConfig.sh txdef.json`
3. Run ``oc apply`` against the YAML representation of the configuration file, e.g.:
`oc apply -f yaml/txdef.yaml`
4. Restart the pod(s) the configuration file applies to, so that the change takes effect.

Upgrading to a new ISIQ release

To upgrade to a new ISIQ release, extract the .zip file attached to the [starter kit page](#) into a directory location that's separate from where your production ISIQ is installed. Then, perform the following steps:

(1) In the separate directory, pull the new docker images:

```
<newStarterKitDir>/isiq images
```

If you have a multi-node cluster, you must run ``isiq images`` on each node. After the images are pulled, run the ``docker image ls`` command. Look for an extra set of images that include the new ISIQ release in

the TAG column, and “/isiq-*” in the REPOSITORY column. For example, if you are running ISIQ v10.0.5 in production, and you download the v10.0.6 images, your `docker image ls` output might look like this:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
icr.io/isiq/isiq-metrics	10.0.6	e53381befcfe	3 days ago	114MB
icr.io/isiq/isiq-products	10.0.6	9bea0823a986	3 days ago	334MB
icr.io/isiq/isiq-connect	10.0.6	fe1bd9b171ae	3 days ago	1.95GB
icr.io/isiq/isiq-web-app	10.0.6	eae3684f8971	3 days ago	1.26GB
icr.io/isiq/isiq-rest	10.0.6	ad40443cf1f1	3 days ago	1.19GB
influxdb	1.8.10-alpine	3ab4fc5517d7	12 days ago	164MB
icr.io/isiq/isiq-metrics	10.0.5	0de7d79a1e29	3 months ago	115MB
icr.io/isiq/isiq-products	10.0.5	bb354ea8f3d4	3 months ago	331MB
icr.io/isiq/isiq-connect	10.0.5	bcfd924ce93a	3 months ago	1.94GB
icr.io/isiq/isiq-web-app	10.0.5	fd03edc0a62e	3 months ago	1.41GB
icr.io/isiq/isiq-rest	10.0.5	6ce636ec4268	3 months ago	1.19GB
zookeeper	3.6.4	530baa007e0d	5 months ago	299MB
redis	7.0.5-alpine	ffd287e43d20	8 months ago	29.9MB
vault	1.12.0	7193130a202b	10 months ago	221MB
nginx	1.23.1-alpine	568998804441	10 months ago	23.5MB
confluentinc/cp-kafka	7.2.2	dfd6b1e2848a	11 months ago	825MB
confluentinc/cp-kafka-rest	7.2.2	4b1bc157202b	11 months ago	1.81GB
zookeeper	3.4.14	4b03fe5b3f64	2 years ago	260MB

For purposes of illustration, this example assumes that you installed v10.0.6 on the same system where your production ISIQ v10.0.5 is running. But of course, you can install v10.0.6 on a different test system. In that case, the `docker image ls` output does not display the v10.0.5 images if the older release isn't present in the test system's docker repository.

The point to remember is that downloading a new set of ISIQ images does not prevent you from continuing to use older images on the same system, if you keep the directory trees separate. By doing so, you can still run `isiq start` from your v10.0.5 directory tree. When you want to test v10.0.6, run `isiq start` from the new directory tree and the v10.0.6 .yml files and associated v10.0.6 images will get used.

(2) In the Readme.txt file of the new ISIQ release, review the “Upgrade Considerations” information. Depending on the ISIQ release you're upgrading from, there might be special actions to take.

(3) When you're ready to upgrade your production environment to the new ISIQ release, back up any production configuration files that have modifications you want to preserve. There are two possible options for accomplishing the backup:

(a) Run `isiq upgrade` - the “upgrade” parameter backs up your current YAML and configuration files and copies them to a compressed tar file in the starter kit root directory. You must specify the location of the prior ISIQ release's starter kit (in other words, the ISIQ installation that is being upgraded) by setting the ORIGINAL_STARTER_KIT_DIR variable in the “User Configuration” section of the script file.

The upgrade script merges your previous YAML file changes into the YAML files that come with the new ISIQ release. Running `isiq upgrade` requires Python3. If you don't have Python3 installed, you need to use Option (b).

(b) Manually save a copy of files you've changed. The ones most frequently changed are:

```
-- cfg/connect/txdef.json, cfg/connect/isiConfiguration.json
-- cfg/crypto/isiq.jwt, cfg/crypto/isiq-ext.jwt
-- cfg/nginx/nginx.conf, cfg/nginx/site.crt, cfg/nginx/site.key
-- cfg/oidc/oidcSettings.json
```

- configuration files pertaining to the optional logs stack
- .yaml files under yaml/cluster or yaml/single-node

(4) Stop the production ISIQ by running ``isiq stop``.

(5) Copy the new directory tree to your production directory tree.

(6) Run ``isiq images`` to download the new images to your production system, or systems if you have a cluster. This step is only necessary if your testing occurred on a separate test system and you did not already pull the new images into the docker repository of the production system or systems.

(7) Copy the backed-up configuration files to their proper locations in the production directory tree.

Note: Before copying, check if retrofitting is required. ISIQ releases can include txdef.json extensions in support of new connectors, or YAML file updates that specify updated component release numbers and environment variables. Review ISIQ's newly installed configuration files to determine whether you need to merge your previous changes. The advantage of using the ``isiq upgrade`` script is that the extra step of retrofitting YAML file changes is handled for you.

(8) Restart the production ISIQ by running ``isiq start``.

Other than extra disk space utilization, there's no harm in keeping older ISIQ images in your docker repository. They do not interfere with your ability to run a newer ISIQ release. When you decide to delete the older images, run the ``docker rmi -f xxxxxx`` command for each image, where you specify the "IMAGE ID" value, obtained from ``docker image ls`` output, as the xxxxxx value in the command.

Rolling Updates

When planning upgrades, another feature to consider is Docker Swarm's rolling updates, which allow replicated containers in a cluster to be migrated to a new version in a relatively safe manner. Instead of stopping and restarting all instances of the container at once and hoping for the best, the set of containers are incrementally moved to the new image, and if the image fails to function correctly, the change can be rolled back. A tutorial on rolling updates can be found at:

<https://docs.docker.com/engine/swarm/swarm-tutorial/rolling-update/>

This feature applies only to a clustered service that has a "replicas:" value > 1, or that is deployed globally on a swarm with more than one node. Specifically, the feature does NOT apply to the ZooKeeper and Kafka services because they each have unique names. While their overall effect is to support replication, from the swarm's perspective they are separate services that must be updated individually.

To enable rolling updates, the "update_config" setting must be specified when the service is created. The setting appears under the "deploy:" section of YAML files. To implement the change, you would use the ``docker service update`` command. For more information about "update_config", see

https://docs.docker.com/compose/compose-file/#update_config

Backups

There are three categories of ISIQ data that you should back up periodically in order to recover your environment if disaster occurs, or if something goes wrong operationally with ISIQ that cannot be readily repaired:

1. Configuration Data: You must have a snapshot of your current ISIQ configuration files. This category of data includes the files listed under "secrets:" and "configs:" in your ISIQ YAML files, and the YAML files themselves. Ideally, the configuration files reside in the same directory tree to make them easier to back

up, although you are not required to store the files in a common location (note: in a cluster, ISIQ config files need to exist on only one node). Files such as `oidcSettings.json` and `txdef.json` tend to be modified early in your ISIQ implementation, and then they stabilize, which means a weekly production backup cycle should be sufficient.

2. Volumes Data: Docker maintains its named volumes in `/var/lib/docker/volumes`. This is the data you should back up most frequently since it contains state information about your ISIQ publishers and subscribers. The subdirectories under `/var/lib/docker/volumes` include the latest offsets in the various Kafka topics, which change throughout the day as data flows between your sources and sinks. A best practice would be to perform a nightly backup of your production ISIQ volumes. This means that ISIQ would have at most 23 hours of updates to catch up on in the event of a disaster recovery.

ISIQ's named volumes are listed under "volumes:" in your YAML files. They begin with an "isiq_" identifier and are prefixed in a `/var/lib/docker/volumes` subdirectory with the stack name. For example, in the "kafka1:", "kafka2:", and "kafka3:" services in `broker-stack.yml`, you find these lines:

```
volumes:
  - isiq_kdata:/var/lib/kafka/data
```

This tells you that on each of your three nodes, there is a `/var/lib/docker/volumes/broker_isiq_kdata` directory that you should back up. If you have network-mounted volumes such as `/var/lib/grafana`, you should back those up as well. Each node only stores data for the containers it runs, and so you must back up volumes from all nodes. Because `/var/lib/docker` is a root-owned area of the filesystem, any script or cron job that backs up subdirectories under `/var/lib/docker/` must have root-level access.

To help you with volumes backup, consider using the "backup" option of the `isiq` shell script. This option requires that you specify a `BACKUP_DIR` variable in the "User Configuration" section of the script. If you later need to restore the ISIQ volumes, rerun the script with the "restore" option. For more information about the ISIQ-provided "backup" and "restore", see the `<starterKitDir>/docs/backup_readme.txt` file.

We recommend you stop ISIQ before backing up volumes. Otherwise, you run the risk that one or more containers will write to `/var/lib/docker/volumes` while the backup's in progress, and you'll miss updates and won't have a fully consistent backup. In the interests of a highly available cluster, you can back up volumes one node at a time. However, don't run the ``isiq stop`` command since it stops the ISIQ stacks on all nodes in the cluster. Instead, stop just the docker daemon on node1 while leaving its stacks running. While the node1 daemon is down, all updates to `/var/lib/docker/volumes` cease, but the ISIQ stacks on nodes 2, 3, etc., continue operating. After the node1 backup completes, restart its docker daemon and repeat the process on node2 and so on until you've backed up all your nodes.

3. Swarm Data: This is metadata about the Docker Swarm itself: the nodes in your cluster, which IP addresses the nodes are listening on, etc. The data is stored in `/var/lib/docker/swarm` on the manager node of a cluster, or on the one node of a single-node configuration. Before you back up swarm data, run ``isiq stop`` on the manager node so that data isn't changing during the backup. To restore from the backup, ensure that ISIQ is stopped on the manager node, remove the existing `/var/lib/docker/swarm` directory, replace it with the copy you made earlier, run ``docker swarm init --force-new-cluster`` to create a new swarm using the backed-up data, and run ``isiq start``. Worker nodes can then connect to the manager node and transfer the swarm information they need. For more details about this process, see https://docs.docker.com/engine/swarm/admin_guide/#back-up-the-swarm.

Because swarm data doesn't change very often, and it's generally not difficult to rebuild a swarm from scratch with ``docker swarm init`` and ``docker swarm join`` commands should that become necessary, swarm backups can be done less frequently than volumes backups.

High Availability

Docker Swarm has built-in load balancing to help achieve high availability (HA). The swarm keeps track of all of its containers. If any of them crash, they're automatically restarted. If an entire node crashes, the swarm tries to move that node's containers to other nodes. (Note: if you use default volumes on the localhost and the node-pinning approach, the swarm is limited in its ability to dynamically relocate containers. NFS or other network file storage is required to be fully fluid on container placement.)

From an HA standpoint, the key component in an ISIQ swarm is nginx. It is the gateway to all other components. A copy of the nginx web server runs on every node in a cluster, and port 443 is exposed on all of them. Whichever nginx instance you access, your request gets directed to the appropriate ISIQ container. If you access nginx on multiple nodes, make sure that each of the node's hostnames is listed in the alternate DNS section of cert.conf so that your browsers trust the connection.

If you're setting up a load balancer as part of your HA architecture, the load balancer can periodically send a request to nginx on the primary node, or to any node where nginx is available. The TCP keepalive parameter is often used in an HTTP header as a means of testing for failure.

High Availability Options for InfluxDB

The ISIQ connect stack relies on InfluxDB to store alert messages as well as metrics data related to Kafka consumer groups. A second instance of InfluxDB is included in the sample logging stack to hold JMX data from Kafka. In general, the information held in InfluxDB is performance-related rather than mission critical. After an outage, any alert messages that are still a problem will be re-created. Thus, there isn't an urgent need to invest in HA for the InfluxDB component.

But if you decide to do so, your first option is InfluxDB-Relay. It is an open source project that acts as a proxy to the InfluxDB instances. It requires setting up two or more InfluxDB instances and a load balancer that can route "/query" requests to InfluxDB, and "/write" requests to InfluxDB Relay. More information is available at: <https://github.com/influxdata/influxdb-relay>

The second option is to enable clustering with InfluxDB Enterprise. This is a paid upgrade for InfluxDB, and it provides management of users, queries, and cluster members. It also includes a performance monitoring dashboard for data visualization. InfluxDB Enterprise offers a richer and more robust HA solution than a simple load balancer can provide. For details, visit: https://docs.influxdata.com/enterprise_influxdb/v1.9/

Tuning Requirements

If you plan to consume ISIM LDAP data to integrate into IGI, you must define an ISIM LDAP index that optimizes retrievals by ISIQ's directory connector. The retrievals use the MODIFY_TIMESTAMP attribute, which is a column in the IBM Security Directory Server's LDAP_ENTRY table. After you connect to your LDAP DB2 database, create the following index for MODIFY_TIMESTAMP to greatly speed up directory connector lookups:

db2 "create index ldap_entry_modify_timestamp on ldap_entry(modify_timestamp, eid) collect detailed statistics"

To optimize database inserts performed by ISIQ's IGI sink connector, you must define the following IGI DB indexes:

```
CREATE INDEX ENTITLEMENT_IDX ON IGACORE.ENTITLEMENT(LOWER(EXT_REF), PROFILE_TYPE)
COLLECT detailed statistics ALLOW REVERSE SCANS;
```

```
CREATE INDEX IGACORE.PERSON_LOWER_DN ON IGACORE.PERSON (LOWER(DN) ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;  
CREATE INDEX IGACORE.PWDCFG_LOWER_VALUE ON IGACORE.PWDCFG (LOWER(VALUE) ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
```

Additional IGI DB indexes might be recommended by IBM Customer Support if you encounter throughput problems. For instance, customers who reported slowdowns during large-scale account processing were helped by the following indexes (note: these SQL statements are tailored for Oracle):

```
CREATE INDEX IGA_CORE.PWDMANGEMENT_CODE_IDX ON IGA_CORE.PWDMANAGEMENT (CODE)  
NOLOGGING NOPARALLEL tablespace IGA_CORE_INDX compute statistics;  
CREATE INDEX IGA_CORE.PWDMANGEMENT_DN_IDX ON IGA_CORE.PWDMANAGEMENT (LOWER(DN))  
NOLOGGING NOPARALLEL tablespace IGA_CORE_INDX compute statistics;
```

Other IGI tuning tips can be found in the [IBM Security Verify Governance v10.0 Performance Tuning Guide](#). In particular, see the section entitled “Tuning IGI in preparation for ISIM -> ISIQ -> ISIG scenario”.

If you have an ISIQ application that consumes data from the AUDIT_EVENT table in the ISIM database, the following ISIM DB indexes have been found to significantly reduce CPU utilization (note: these SQL statements are tailored for DB2):

```
CREATE INDEX ITIMUSER.T1_TIMESTAMP_ITIM_EVENT_CATEGORY ON ITIMUSER.AUDIT_EVENT  
("ITIM_EVENT_CATEGORY" ASC, "TIMESTAMP" ASC) ALLOW REVERSE SCANS COLLECT DETAILED  
STATISTICS;  
CREATE INDEX ITIMUSER.COMPLETED ON ITIMUSER.PROCESS ("COMPLETED" ASC) ALLOW REVERSE  
SCANS COLLECT DETAILED STATISTICS;  
CREATE INDEX ITIMUSER.EVENTTYPE ON ITIMUSER.PROCESSLOG ("EVENTTYPE" ASC) ALLOW REVERSE  
SCANS COLLECT DETAILED STATISTICS;
```

The equivalent CREATE INDEX statements for an Oracle ISIM database are as follows:

```
CREATE INDEX ITIMUSER.T1_TS_EVENT_CAT ON ITIMUSER.AUDIT_EVENT ("ITIM_EVENT_CATEGORY"  
ASC, "TIMESTAMP" ASC);  
CREATE INDEX ITIMUSER.COMPLETED ON ITIMUSER.PROCESS ("COMPLETED" ASC);  
CREATE INDEX ITIMUSER.EVENTTYPE ON ITIMUSER.PROCESSLOG ("EVENTTYPE" ASC);
```

If you aren’t currently using data from the ISIM database, you can bypass the need for these indexes by simply limiting the default queries issued against the DB. In Appendix E of the [ISIQ User’s Guide](#) under “Additional ISIM Customization Options”, see the section entitled “Reduce ISIM Database Queries”.

Other valuable tuning recommendations can be found in the [ISIQ Performance Tuning Guide](#). It contains a set of performance best practices for ISIQ, ISIM, and IGI. Although the title page of the guide says it’s for ISIQ v1.0.5, the documented recommendations are still pertinent for newer ISIQ versions.

Docker Log Rotation

The ISIQ-supplied YAML files include “logging:” definitions to control the number and size of docker log files per ISIQ stack. These definitions take priority over any system-wide docker log rotation settings that you may have configured (refer to <https://success.docker.com/article/how-to-setup-log-rotation-post-installation>). For example, the ISIQ connect-stack.yml specifies:

```
logging:  
  driver: "json-file"  
  options:
```

```
max-size: "50m"  
max-file: "40"
```

This means that the connect stack can write up to 40 docker logs of 50 megabytes each, or a maximum of 2 Gigabytes. After that point, the connect stack logs will roll over. See [Appendix A. YAML Files](#) for a detailed description of the contents of ISIQ's YAML files.

Running setup.sh

The ISIQ installation process requires cryptographic configuration to generate ISIQ's SSL certificates and JSON Web Token (JWT) secrets. These configuration tasks are handled by the ``isiq setup`` script.

Alternatively, if you prefer a more granular approach to security configuration, you can run the `<starterKitDir/cfg/setup.sh` script. This script is a wrapper that invokes `"gencert.sh"`, `"genstore.sh"`, and `"gentoken.sh"`, which generate the certificates and JWT files that ISIQ requires for its nginx service and for application data protection. You can individually run the scripts later if necessary.

Note that `"genstore.sh"` invokes the Java keytool utility and so your ISIQ system must have Java installed. If you see the error, `"keytool not found in path"`, after the status message `"Generating SSL truststore"`, it likely means you need to install Java before proceeding with the ISIQ installation.

Refer to the subsequent `"Generating SSL Certificates"` and `"Product Password Encryption"` sections for more details about ISIQ's certificate and encryption setup.

Generating SSL Certificates

All user interaction with ISIQ occurs via HTTPS using Transport Layer Security (TLS). TLS is the successor to Secure Socket Layer (SSL), and the terms are often used interchangeably. In this document, we specify `"SSL"` because the commands, certificates, and settings still refer to SSL. But as a matter of security, SSL is disabled in ISIQ's nginx web server, forcing connections to use the newer TLS framework.

Setting up SSL can be a challenge, depending on your needs. The certificates involved can provide encryption as well as authentication, and there are many ways to generate certificates. ISIQ's nginx web server needs to be configured to reference the key and certificate that you want to use to enable SSL. Review the `"secrets:"` at the end of `app-stack.yml` to ensure the `"site.key:"` and `"site.crt:"` declarations point to the correct location of your key and certificate files.

In the `cfg/nginx` directory, the `"gencert.sh"` script and its configuration files `"ca.conf"` and `"cert.conf"` are provided to help you generate the required certificate using OpenSSL (note: `cfg/nginx` directory also has an `nginx.conf` file, but it does not need to be modified to enable SSL). The `"gencert.sh"` script can take one of two options:

1 (ca)

With this option, a new root CA is generated. A private key and a Certificate Signing Request (CSR) are generated for nginx, and the new CA certificate signs it. This provides most of the benefits of using a CA without having to track one down. The only drawback is that clients won't know about your new CA, and thus cannot automatically trust it. You can either distribute the CA certificate to your clients, or they can allow an exception in their browsers.

2 (request)

With this option, a `.csr` file is created. It must then be provided to a CA, whether a known third party like Verisign, or a CA internal to your company. The CA will return a signed certificate, which can then be

used with nginx. **Note:** If you pick Option 2 (request), you **MUST** take the .csr file to your CA and get it signed. Otherwise, you will not have a site.crt certificate file to store in <starterKitDir>/cfg/nginx, which means that nginx and ISIQ won't be able to start. Also, the file must conform to nginx specifications. Only the PEM (Base64 ASCII) format is allowed; nginx does not support the DER format.

Before you run gencert.sh, you **MUST** edit the "[alternate_names]" list in the cert.conf file to specify the DNS name of the ISIQ host where nginx runs. In a single-node configuration, "DNS.1" will contain the fully qualified domain name (FQDN) of your one ISIQ system. If you have a three-node ISIQ cluster, and a load balancer with a single DNS name is not being employed, then the FQDN of each node in the cluster must be specified in DNS.1, DNS.2, and DNS.3.

The principle to remember is: Any hostnames, whether fully qualified or not, that users will enter in a browser when connecting to ISIQ should be listed under "[alternate_names]". Current browsers do not allow an SSL connection to a website if the site is not defined in a certificate accepted by the browser. If your ISIQ host has aliases, include them in the list too, for example:

```
[alternate_names]
```

```
DNS.1 = isiqHost
```

```
DNS.2 = isiqHost.my-network.com
```

```
DNS.3 = isiq-alias.my-network.com
```

The gencert.sh script uses the list of hosts under [alternate_names] when creating the Subject Alternative Name field in your SSL certificate. Later, if you need to change your ISIQ hostname or add an alias, you can edit the cert.conf file and rerun gencert.sh.

There's also a `gencert.sh renew` command whose purpose is to reset a certificate's expiration date. Note: If you've made any changes in the "alternate_names" list since the last time you generated the certificate, those changes will be reflected in the new certificate after running `gencert.sh renew`.

For clients to trust an HTTPS connection, they must possess the full chain of CA certificates that signed your server certificate. If you use Option 1 of gencert.sh to create a CA-signed certificate, the relevant CA certificate can be found with the name "rootCA.crt" in the cfg/nginx directory. If you use Option 2, your CA should provide the necessary CA certificate(s) when returning the signed certificate to you.

Other cfg directory contents

In addition to /nginx, there are other directories under <starterKitDir>/cfg that contain artifacts to help with your ISIQ deployment. Here are some of the important ones to be aware of:

- /connect

Includes the "txdef.json" file, which defines mapping rules for transformations performed during ISIQ data integration. Refer to the "Custom Transformations" section of the [ISIQ User's Guide](#) for an explanation of txdef.json.

- /crypto

Includes "gentoken.sh" and the JSON Web Token files described in the "Running setup.sh" section.

- /metricsdb

The "influxdb.conf" provided here is used by the "metricsdb" InfluxDB server in the connect stack. It is usually fine as is.

- /oidc

OpenID Connect (OIDC) must be configured before anyone can log in to ISIQ. You specify details about your OIDC provider by updating the “oidcSettings.json” file. A sample “oidcSettings.json” is available in the /oidc directory. For more information, refer to the “OpenID” section later in this guide.

Product Password Encryption

During product configuration, you typically must enter one or more password values for connecting to the product’s directory server or database. As a security precaution, each time you edit a product configuration in the ISIQ UI, you must re-enter the product’s passwords. To protect these password strings, ISIQ encrypts them when they’re saved, and decrypts them only when they’re needed to connect to a product’s data store.

Before ISIQ 10.0.0, the encryption and decryption were done with a secret key defined in the /cfg/crypto/isiq.key file. Beginning with ISIQ 10.0.0, every new product configuration, or change to an existing configuration, assigns the encryption and decryption functions to a vault service. The service runs in its own container in the connect stack. Products configured prior to ISIQ 10.0.0 continue to use isiq.key. But eventually, as all of your products are created or updated post-10.0.0, the isiq.key file will become obsolete.

When you start ISIQ 10.0.0 (or higher) for the first time, the vault is initialized. This allows it to generate its data keys. The output from the initialization is a token used for subsequent access, and a key that unseals the vault. The vault token is stored in the ISIQ metricsdb in the connect stack. For each password encryption and decryption request, ISIQ retrieves the token from the metricsdb and issues a request to the vault service to perform the operation.

If there is a failure during vault initialization or when accessing the metricsdb, it’s a serious problem that must be dealt with right away since it prevents ISIQ from connecting to your data stores. In that case, you will see one or more alerts on the System Health dashboard. There will likely also be errors in the “connect_vault” docker service log. For help with diagnosing and resolving vault-related problems, refer to “Diagnosing Encryption and Vault Problems” in the [ISIQ Troubleshooting Guide](#).

Secure Communication to Applications

ISIQ’s connect service, which has the docker image name “connect_connect”, reads from and writes to target applications. The list of supported applications includes:

- ISIM LDAP directory server
- ISIM DB2 or Oracle database
- ISIM Server (for the ISIM Web Services API)
- IGI DB2, Oracle, or PostgreSQL database

If ISIQ needs to communicate with any of these applications over a secure TLS channel, you must import the application certificate to ISIQ’s truststore.

In the extracted contents of the ISIQ starter kit, there are two files to be aware of in the <starterKitDir>/cfg/connect/ssl directory:

1. **isiq.truststore.jks** – The ISIQ truststore is copied to the connect service’s container. Note that if you supply your own certificates, you should create your own truststore.
2. **ssl.client.props** – The client security properties file that specifies the truststore password and file path. To change the password, you must update com.ibm.ssl.trustStorePassword. Do not modify the truststore file path defined in the com.ibm.ssl.trustStore property.

In the `connect-stack.yml` file, the `ISIQ_AUTOMATICALLY_IMPORT_CERTIFICATE` environment variable determines how an application certificate is installed for ISIQ's connect service:

- If the environment variable is set to false (the default), you must manually import application certificate(s) to `<starterKitDir>/cfg/connect/ssl/isiq.truststore.jks` before you deploy the connect service.
- If the environment variable is set to true, a generic application certificate is installed automatically into the truststore copy located at `/etc/isiq/ssl/isiq.truststore.jks` in the connect service's container.

As an example, suppose you want ISIQ's directory connector to communicate over a secure channel to your ISIM LDAP server at `isimLdap6.austin.myCorp.com:636`. The `connect-stack.yml` file has `ISIQ_AUTOMATICALLY_IMPORT_CERTIFICATE=false`, but the LDAP server certificate is not yet imported into `<starterKitDir>/cfg/connect/ssl/isiq.truststore.jks`. In that scenario, the directory connector fails to start. The following error message appears in the docker log for the `connect_connect` image:

```
connect_connect| [2021-11-15 14:43:15,807] ERROR Failed to start ISIM LDAP source connector due to the following error: The certificate from isimLdap6.austin.myCorp.com:636 is not trusted. The certificate must be imported to isiq.truststore.jks file. (com.ibm.kafka.connect.directory.util.RuntimeContext)
```

To resolve the error, you must:

- Stop ISIQ's app and connect stacks.
- Perform one of the following tasks:
 - a. Import the LDAP server certificate to `isiq.truststore.jks` in the starter kit directory, OR
 - b. Set `ISIQ_AUTOMATICALLY_IMPORT_CERTIFICATE=true` in `connect-stack.yml`.
- Restart the connect and app stacks.

Note: Do not run with `ISIQ_AUTOMATICALLY_IMPORT_CERTIFICATE=true` on a long-term basis. It's an insecure mode of operation that's intended to give you a functioning connection until you have a proper application certificate imported into your truststore.

OpenShift Container Platform (OCP) instructions

If you need to connect to endpoints using SSL in an OCP environment, you must decide how to handle the CA certificates. There are two options. The first is simple but less secure because it trusts whatever certificate is presented. The second option is the recommended approach because only the certificates you provide will be trusted, but it requires more work up front to obtain them:

Option 1. Edit the `openshift/values.yml` file. Find the line named `"autoImportSslCerts"`. Set this to `"true"` (without quotes).

Option 2. Manually obtain certificates and then do the following:

- Run `cfg/connect/ssl/genstore.sh`
- Collect the CA certificates for all of your SSL-protected endpoints (e.g., LDAP, IGI DB, WebSphere).
- Place the certificates in `cfg/connect/ssl/isiq.truststore.jks`, using the password found in `cfg/connect/ssl/ssl.client.props`

OpenID

ISIQ login requires authentication that depends on the OpenID Connect (OIDC) standard. Administrators can use IGI's OIDC provider or any other provider such as the one offered by IBM Cloud App ID, or by IBM Security Verify Access (formerly IBM Security Access Manager/ISAM), or by Azure AD or Google.

To allow users to authenticate to ISIQ using your preferred OIDC provider, you need to collect the following information from the respective provider:

- a. Client ID
- b. Client Secret
- c. Issuer URL
- d. Authorization URL
- e. Token URL
- f. UserInfo Endpoint/UserInfo URL
- g. Logout URL

If you are using IBM Cloud App ID, the following data items are necessary (note: they can be found under the Service Credentials tab of your App ID service):

- a. Client ID
- b. Secret
- c. OAuth Server URL
- d. Profiles URL
- e. Tenant ID

In addition to these data items, when you register your client with an OIDC provider, you must also supply a redirect/callback URL in the format: <https://<YourISIQhostName>/api/oidc>

The purpose of collecting this provider information is to edit app-stack.yml along with ISIQ's configuration file, oidcSettings.json, which is located in your <starterKitDir>/cfg/oidc directory. The default oidcSettings.json **must** be updated because the file contains placeholder strings, such as "Enter your issuer URL" or "Enter your callback URL", for its required values.

To illustrate the OIDC setup process, here are the steps if you plan to use IGI's OIDC provider:

Step 1: Log in to your IGI Virtual Appliance.

Step 2: Select Configure -> OpenID Connect Provider Configuration



Note: When you configure the OpenID Connect Provider in IGI, if you accept the default certificate for the keystore, you must update an ISIQ environment variable in order to log in. Specifically, in app-stack.yml, set ISIQ_OIDC_ALLOW_SELF_SIGNED=true. Otherwise, ISIQ login attempts fail with a 403 error and a "Missing session" browser message because IGI's self-signed certificate breaks the chain of trust. If you don't want to set this environment variable to allow self-signed certificates, then you will need to export a CA-signed certificate to IGI and add it to the OpenID Connect Provider keystore. For details, see <https://www.ibm.com/docs/en/sig-and-i/10.0.1?topic=certificates-updating-personal-certificate-openid-connect-provider>

Step 3: On the subsequent page, make a note of the Issuer Identifier, Authorization URL and Token URL. Those values are needed when you edit oidcSettings.json.

Note: The Issuer Identifier, Authorization URL, and Token URL are different for IGI Administrator Console Users versus Service Center Console Users. Depending on which type of user you are configuring for the IGI OIDC provider, note the URLs that correspond to the particular type.

Then, select Manage -> External Client Configuration

OpenID Connect Provider Configuration

Start Stop Restart Refresh Disable Manage

Enabled: True
Status: Started

- Administrator User Registry
- Service Center User Registry
- External client configuration
- Reverse Proxy

	Administrator Console	Service Center
User Registry	Internal User Registry	Internal User Registry
Scope	openid	openid

Step 4: On the “Manage external client configuration” page, you must choose either the Admin Provider or the ServiceCenter Provider. If you prefer IGI’s Admin Console Users to be able to log in to ISIQ, select the Edit option for AdminConsoleRestClient. Otherwise, select the Edit option for ServiceCenterRestClient to support IGI’s Service Center Users.

Manage external client configuration

Add Edit Delete Refresh

Name	Client ID	Provider
AdminConsoleRestClient	Zjk2OGY5Y2M3YTM3MmJj	Admin
ServiceCenterRestClient	MDEzNmNmMDAyYWQzMmlw	ServiceCenter

1 - 2 of 2 items 10 | 25 | 50

Although this example Edits an existing client configuration, you can also Add a new client configuration for the AdminConsoleRestClient or ServiceCenterRestClient.

Step 5: On the Edit page, update or add the Redirect URL field. This is a REST API callback location on your ISIQ host that gets invoked during login authentication.

Admin Console Example:

Edit external client configuration

Name: AdminConsoleRestClient

Provider: Admin

Client ID: YzhjNTUxMzk3YTJyY2Zk Regenerate

Client Secret: NWJkZDExOWQ4ZTA0ZjQ5 Regenerate

Redirect URL: https://IsiqHostA.ibm.com/api/oidc

Save Configuration Cancel

Service Center Example:

Manage external client configuration		
 Add  Edit  Delete  Refresh		
Name	Client ID	Provider
 AdminConsoleRestClient	YzhjNTUxMzk3YTVjY2Zk	Admin
 ServiceCenterRestClient	NWE0YTM4ODAxZDIyZTll	ServiceCenter
1 - 2 of 2 items		
10 25 50		

Edit external client configuration	
Name:	ServiceCenterRestClient
Provider:	Service Center
Client ID:	NWE0YTM4ODAxZDIyZTll Regenerate
Client Secret:	NDA2NDlmNzE2MjQzMWY0 Regenerate
Redirect URL:	https://IsiqHostA.ibm.com/api/oidc
Save Configuration Cancel	

Step 6: Note down the Client ID and Client Secret, and then click Save Configuration. After you make changes in the OpenID Connect Provider Configuration, you normally have to restart the IGI server. Check the Notifications dashboard widget on the IGI Appliance Dashboard to see whether a restart is required.

Step 7: You now have all OIDC fields needed to configure ISIQ login. Enter your Client ID and Client Secret values into the `ISIQ_OIDC_CLIENT` and `ISIQ_OIDC_SECRET` environment variables in `app-stack.yml`. Next, edit the `oidcSettings.json` file located in `<starterKitDir>/cfg/oidc`

Note: The following example, which lists symbolic hostnames in the URL strings, assumes your docker environment has a DNS server that can resolve the URL hostnames. While internal docker networking requires DNS, you can if necessary specify dotted decimal IP addresses in your OIDC endpoint URL strings.

Step 8: Update the fields in `oidcSettings.json` under the “Generic” stanza. Since you are using IGI as your provider, you can ignore the fields under the other two stanzas and leave those placeholder values as is. If you are allowing IGI’s Admin Console users to log in, the `oidcSettings.json` file might look like this:

```
{
  "Generic":{
    "issuer":"https://myIGIva.ibm.com:10443/oidc/endpoint/openidProviderAdmin",
    "authorizationURL":"https://myIGIapp.ibm.com:10443/oidc/endpoint/openidProviderAdmin/authorize",
    "tokenURL":"https://myIGIapp.ibm.com:10443/oidc/endpoint/openidProviderAdmin/token",
    "userInfoURL":"https://myIGIapp.ibm.com:10443/oidc/endpoint/openidProviderAdmin/userinfo",
    "callbackURL":"https://IsiqHostA.ibm.com/api/oidc",
    "logoutURL":"https://myIGIapp.ibm.com:10443/logout"
  },
  "AppID":{
    "oauthServerUrl":"Enter your AppID OAuth Server URL",
    "profilesUrl":"Enter your AppID profiles URL",
    "tenantId":"Enter your AppID tenant ID",
    "redirectUri":"https://<YourISIQhostName>/api/oidc"
  },
  "Azure":{
    "identityMetadata":"Enter your Azure AD metadata URL",
    "redirectUri":"https://<YourISIQhostName>/api/oidc"
  }
}
```

If you are allowing IGI's Service Center users to log in, the oidcSettings.json file might look like this:

```
{
  "Generic":{
    "issuer":"https://myIGIva.ibm.com:11443/oidc/endpoint/openidProvidersC",
    "authorizationURL":"https://myIGIapp.ibm.com:11443/oidc/endpoint/openidProviderSC/authorize",
    "tokenURL":"https://myIGIapp.ibm.com:11443/oidc/endpoint/openidProviderSC/token",
    "userInfoURL":"https://myIGIapp.ibm.com:11443/oidc/endpoint/openidProviderSC/userinfo",
    "callbackURL":"https://IsiqHostA.ibm.com/api/oidc",
    "logoutURL":"https://myIGIapp.ibm.com:11443/logout"
  },
  "AppID":{
    "oauthServerUrl":"Enter your AppID OAuth Server URL",
    "profilesUrl":"Enter your AppID profiles URL",
    "tenantId":"Enter your AppID tenant ID",
    "redirectUri":"https://<YourISIQhostName>/api/oidc"
  },
  "Azure":{
    "identityMetadata":"Enter your Azure AD metadata URL",
    "redirectUri":"https://<YourISIQhostName>/api/oidc"
  }
}
```

Notes:

- a) The issuer field is a case-sensitive URL that must match what the OIDC provider sends. For example, if you update oidcSettings.json to specify the issuer's hostname as a dotted decimal IP address, but the OIDC provider sends its URL in symbolic hostname format, the ISIQ login will fail. In debugging scenarios such as this, it's helpful to set `ISIQ_LOG_OIDC=true` in `app-stack.yml`. In the "app_rest" docker service log, you will see an error stating that the issuer URL returned by the provider did not match the expected value. To correct the problem, change the issuer URL in `oidcSettings.json` to use the exact same symbolic hostname format as was returned by the OIDC provider, and then restart the app stack.
- b) The `userInfoURL` field is optional. Leave it empty if you do not have a `userInfoURL`.
- c) *There must be two-way network connectivity between your ISIQ node (or nodes in a cluster) and your OIDC provider.* If there are firewall rules blocking traffic between the OIDC provider system and the ISIQ system, the OIDC protocol (which relies on credential passing and callbacks) fails, and consequently all ISIQ logins fail. As a quick test, run a `cURL` command from the ISIQ system to port 443 on the OIDC provider system. If you don't get a response back, it tells you there's a connectivity problem which must be fixed before proceeding.
- d) If your ISIQ host requires a proxy to access external websites, and your OIDC provider is located externally, you might need to add `HTTP_PROXY` and `HTTPS_PROXY` environment variables to your `app-stack.yml` file. In the "environment:" block under the "rest:" service, insert these statements:

- HTTP_PROXY=<your-proxy-URL>
- HTTPS_PROXY=<your-proxy-URL>

For more information, refer to “Proxy Considerations” later in this guide.

e) The logOutURL field is necessary to log out a user completely, and to ensure that credentials are requested at the next ISIQ login. The value for this field is usually obtainable from the OIDC provider. But if the logOutURL isn’t known, you can leave the field empty. However, it means that ISIQ users need to manually clear browser cookies after logout in order to force a prompt for credentials at the next login. Here are examples of logout URLs for some common OIDC providers:

IGI (Admin Console REST Client) – https://<IGI_Application_Interface_Hostname>:10443/logout

IGI (Service Center REST Client) – https://<IGI_Application_Interface_Hostname>:11443/logout

ISAM – <https://<Hostname>:<port>/pkmslogout>

IBM Security Verify – <https://<Hostname>/pkmslogout>

Google – <https://accounts.google.com/Logout>

f) As a further logout security measure, you need to update your <starterKitDir>/cfg/nginx/nginx.conf file. The default file includes a Content-Security-Policy header that restricts access to only the domain hosting ISIQ:

```
add_header Content-Security-Policy "frame-src *.ibm.com/docs/; script-src 'self' 'unsafe-inline' 'unsafe-eval'; default-src 'self' 'unsafe-inline' blob: data;";
```

In this default nginx header, your OIDC logout URL will be blocked, which means a logout will **not** automatically invalidate the authentication token with the OIDC provider. As a result, an ISIQ user will be able to log in again, without providing credentials, until the token expires.

To enforce authentication at each login, you must grant access to the domain of your OIDC logout URL. The domain is added in the ‘default-src’ section, immediately after the ‘self’ indicator. Using the Google logout URL as an example, your Content-Security-Policy header would look like this:

```
add_header Content-Security-Policy "frame-src *.ibm.com/docs/; script-src 'self' 'unsafe-inline' 'unsafe-eval'; default-src 'self' *.google.com 'unsafe-inline' blob: data;";
```

Using the logout URL for an IGI OIDC provider, your Content-Security-Policy header might look like this:

```
add_header Content-Security-Policy "frame-src *.ibm.com/docs/; script-src 'self' 'unsafe-inline' 'unsafe-eval'; default-src 'self' https://myIGIapp.ibm.com:11443/logout 'unsafe-inline' blob: data;";
```

With this change in effect, ISIQ users must authenticate at each login. For information about the Content-Security-Policy header, see <https://content-security-policy.com/>.

Step 9: In the app-stack.yml file, the ISIQ_OIDC_PROVIDER environment variable must be set to the appropriate stanza identifier. In the screen captures above of oidcSettings.json, you will notice that each stanza has an identifier: “Generic”, “AppID”, or “Azure”. Most OIDC providers, such as the ones for IGI and ISAM, use the “Generic” stanza. That means you should (a) update the fields under the “Generic” stanza, and (b) keep the default **ISIQ_OIDC_PROVIDER=Generic** setting in app-stack.yml.

However, if you are using IBM Cloud App ID or Azure as your OIDC provider, you must change the ISIQ_OIDC_PROVIDER=xxx value accordingly. In addition, update the corresponding fields under the “AppID” or “Azure” stanza. For Azure, you must also set ISIQ_OIDC_POST_METHOD=true and so your first few environment settings in app-stack.yml would look like this:

```
environment:
- ISIQ_CONNECT_BASE_URL=http://connect:8083
- ISIQ_OIDC_PROVIDER=Azure
- ISIQ_SKIP_OIDC=false
- ISIQ_SKIP_OIDC_USER=isiquiser
- ISIQ_LOG_OIDC=false
```

- ISIQ_OIDC_ALLOW_SELF_SIGNED=false
- ISIQ_OIDC_POST_METHOD=true
- . . .

For Azure, there are two other actions that need to be done in your Azure application portal. Under the “Manage” section of the left navigation bar, select “Authentication”. On the Authentication page, (a) Add your ISIQ callback URL (<https://<YourISIQHostName>/api/oidc>) to the “Redirect URIs” list. (b) Farther down that same page, check the “ID tokens” box.

Step 10: After you make your oidcSettings.json changes, and you configure and deploy the app stack, ISIQ will be ready to accept logins.

Shortcuts for Rapid ISIQ Setup

If you need to accelerate ISIQ setup, for example, because you’re preparing a proof-of-concept (PoC) demo in which security is less important at the moment than getting ISIQ operational, consider the following two shortcuts:

1. Set ISIQ_AUTOMATICALLY_IMPORT_CERTIFICATE=true in the connect-stack.yml file located under <starterKitDir>/yml. This environment variable automatically installs application certificates into the truststore copy in the connect service’s container, and it eliminates the need to manually import certificates before you deploy the connect service.
2. Set ISIQ_SKIP_OIDC=true in the app-stack.yml file located under <starterKitDir>/yml. This environment variable bypasses OIDC authentication checking. You might consider using it if configuring an OIDC provider will require a day or two to set up, and meanwhile you’d like ISIQ logins to occur so that product familiarization can begin. With ISIQ_SKIP_OIDC=true in effect, everyone gets logged in with a generic ID of “isiquser” by default. You can override “isiquser” by adding an ISIQ_SKIP_OIDC_USER=xxxxxx environment variable. The generic ID has authority to configure products, create subscriptions, and otherwise exercise ISIQ. Bear in mind that skipping OIDC should be a temporary arrangement because you’re bypassing authentication.

Alternatively, you can run `isiq setup poc`. This mode of invoking the isiq shell script combines the previous two shortcuts and also disables the optional logs stack. During script execution, you are prompted to configure an SSL certificate for nginx. If you supply ISIQ’s hostname and hit the Enter key through the remaining prompts, the script generates a certificate, <starterKitDir>/cfg/nginx/rootCA.crt. To summarize, here are the steps for installing and running ISIQ using this streamlined method:

1. Install Docker CE 19.03 or higher on a Linux system.
2. Download and extract the .zip file attached to the [ISIQ starter kit page](#).
3. Run the `isiq setup poc` script from the directory where you extracted the starter kit.
4. Reply to the script’s prompts and then wait for the ISIQ stacks to start.
5. Log in to ISIQ.

To emphasize, **these are temporary shortcuts**. After the PoC when you run ISIQ in a more secure, long-term mode, you need to:

- Change the ISIQ_SKIP_OIDC environment variable from true to false.
- Change the ISIQ_AUTOMATICALLY_IMPORT_CERTIFICATE environment variable from true to false, and import real certificates.
- Reconfigure products and subscriptions. Note: This last step might not be required. If you used the generic “isiquser” during the PoC, you will likely need to reconfigure, assuming that “isiquser” is not a valid ID to your OIDC provider. But if you set ISIQ_SKIP_OIDC_USER=xxxxxx to an ID that you intend to

use with OIDC, then you can continue to work with the products and subscriptions created during the PoC, without reconfiguring, since the ID that owns those ISIQ entities is already set correctly.

Deploying with Limited Internet Access

If you intend to deploy ISIQ on a node with limited or no Internet access, here's what you can do:

- (a) Find a system—let's call it the “download system”—somewhere in your environment that has Internet access, even if the access is temporary. Without Internet access, you cannot retrieve the ISIQ starter kit from the public IBM website, nor can you pull ISIQ's docker images from the IBM Cloud Container Registry, nor can you install Docker CE itself.
- (b) For information about installing Docker CE on an isolated system, search Stack Overflow for articles pertaining to your Linux distribution. Try the search argument, “How to install docker-ce without internet and intranet yum repository?”
- (c) To obtain the latest ISIQ, go to the [ISIQ starter kit page](#) and extract the attached .zip file.
- (d) On the download system where you extracted the .zip file, run the command:
- `<starterKitDir>/isq images``
- (e) Review the Stack Overflow article entitled, [How to copy Docker images from one host to another without using a repository](#). It gives an overview of the procedure to be followed. The two main steps described in the article are:
 - (1) Save each downloaded docker image in a compressed format, such as a tar file. You then use a file transfer tool such as scp or rsync to copy the compressed images from the download system to the ISIQ node (or nodes if you're running a cluster).
 - (2) Once the images have been copied and uncompressed, make sure they have the correct names. This means that the REPOSITORY and TAG names listed in the `docker image ls`` output should match what is specified in your ISIQ YAML files.

You have control over the image-naming process. For instance, if you're running ISIQ 10.0.5, the connect-stack.yml file searches by default for an image named “icr.io/isq/isq-connect:10.0.5”. Assuming the connect image that you copied from the download system has a REPOSITORY name of “icr.io/isq/isq-connect” and a TAG of “10.0.5”, when your ISIQ swarm starts up, docker will see that it already has a local copy of the required connect image and will not attempt to access the IBM Cloud Container Registry website or any external registry. But if you copied the images into a private docker registry that you named, for example, “package-repo” on port 5000, then you would update the “image:” statements in your ISIQ YAML files accordingly:

image: [package-repo:5000/isq-connect:10.0.5](#)

In this way, you can deploy and run ISIQ in an isolated environment with no public Internet access.

Proxy Considerations

If you're deploying ISIQ in a cloud computing environment such as Amazon EC2, you might need to configure a web proxy to manage any external connections that ISIQ requires. For example, if you're using Azure AD as your OIDC provider, your proxy must allow ISIQ to establish a connection with the Microsoft Azure URL.

In these types of proxy-based scenarios, there are extra configuration tasks to be aware of. First, you must tell Docker Swarm where the proxy is located. You do this in two places:

(1) Edit (or create if the file doesn't exist) `/etc/systemd/system/docker.service.d/http-proxy.conf`. If we assume your proxy URL is reachable at `http://myproxy.acme.com`, port 8080, then you would add the following statements to the file (note: updating the file requires root-level access):

```
[Service]
Environment="HTTP_PROXY=http://myproxy.acme.com:8080"
Environment="HTTPS_PROXY=http://myproxy.acme.com:8080"
```

For more information on `http-proxy.conf`, see <https://docs.docker.com/config/daemon/systemd/>.

(2) Add the same environment variables to the ISIQ service that uses the external connection. For an OIDC provider, it's ISIQ's `app_rest` service that implements the OIDC handshake protocol. Therefore, in `app-stack.yml`, under `"rest:"` -> `"environment:"`, insert these two variables:

- `HTTP_PROXY=http://myproxy.acme.com:8080`
- `HTTPS_PROXY=http://myproxy.acme.com:8080`

Next, determine which connections to exclude. Rather than perform a lot of proxy configuration, it can be simpler to specify a `NO_PROXY` environment variable that lists inter-container traffic which should stay on the local system and avoid the proxy. If we continue the example of the `app_rest` service that talks to an external OIDC provider, you would add a statement such as this to `http-proxy.conf`:

```
Environment="NO_PROXY=localhost,127.0.0.1,*.acme.com,connect,products,kafka-rest1,metricsdb,redis,vault"
```

Also, add the same environment setting to `app-stack.yml` under `"rest:"` -> `"environment:"`:

- `NO_PROXY=localhost,127.0.0.1,*.acme.com,connect,products,kafka-rest1,metricsdb,redis,vault`

The last six entries in the comma-separated list are ISIQ containers which `app_rest` communicates with.

Putting the pieces together, you see how the process operates: First, you add the `HTTP(S)_PROXY` environment variables so that an ISIQ service running in Docker Swarm can connect via a proxy to an external destination. But since you don't want the ISIQ service's other network connections to go through the proxy, you construct a `NO_PROXY` exclude list.

If you're uncertain which entries to specify in `NO_PROXY`, look at an `"app_nginx"` docker service log when the proxy is active. If you notice HTTP GET requests to an ISIQ container are failing with a 403/Forbidden status code, then you know the connection attempted to use the proxy and was blocked. In that case, add the blocked ISIQ container to the `NO_PROXY` exclude list.

ISIQ Event Auditing

Key ISIQ lifecycle events, such as configuring a product, creating a group, deleting a subscription, or reprocessing topics, are recorded in the Audit table. The table resides in the ISIQ metrics database, which uses InfluxDB and is part of the ISIQ connect stack.

To view the contents of the Audit table, click the "Audit Events" link in the Navigation sidebar. The table includes the following columns to help answer who/what/when questions about each significant event:

- (1) The time the event took place.
- (2) The user who initiated it.
- (3) The event type, such as "Create Subscription" or "Pause Product".
- (4) The ID of the object acted upon, such as a configured product name or subscription ID.
- (5) The type of object acted upon, such as "IGI", "Group", "Subscription", or "EXTAPP".
- (6) Additional details, such as the LDAP and Web Services URLs when configuring an ISIM product, or the names of the two products that are subscribed to each other.

When you first navigate to the Audit Events page, it shows you all events for all products and users. You can limit the search results by specifying particular event types, products, users, or IDs as filter criteria, and then clicking the “Submit Query” button in the upper right.

InfluxDB only allows the Time column to be sorted. If you need to do other sorting or manipulations of audit event data, click the “Download CSV File” button to extract the contents of the Audit table to a spreadsheet for offline analysis.

By default, audit events are preserved indefinitely. If you want to purge the oldest audit events to reduce the size of the Audit table, run the `<starterKitDir>/util/auditPrune.sh` script.

Docker Cleanup

As a rule, docker does not clean up unused or “dangling” objects. Hence, after you start and stop your ISIQ swarm multiple times, you should issue commands such as ``docker image ls``, ``docker container ls``, and ``docker ps -a``. These commands will tell you if a long list of duplicate, obsolete objects have accumulated. If so, you can run ``docker image prune`` and ``docker container prune`` to remove the objects and reclaim disk space on the ISIQ host. The ``docker network ls`` and ``docker network prune`` commands can help you clean up prior network configurations.

More information on docker pruning can be found at <https://docs.docker.com/config/pruning/>.

There is one special cleanup scenario to be mindful of. ISIQ stores its persistent data on docker volumes. As the ISIQ Administrator, you might be asked by a user to remove problematic ISIQ definitions in order to recover from a system outage, or to reset an ISIQ environment for some other reason. You can do so with the ``docker volume prune --all`` command. However, **you should seriously consider the consequences of running this command!** It removes **all** persistent ISIQ data—configured products, subscriptions, group membership, Kafka topic data, audit events, monitoring metrics—and therefore impacts every user of that ISIQ system (note: it does NOT clear any data on your product endpoints, for example, in ISIM or IGI; that endpoint data must be cleared manually).

For a lightly used ISIQ test or dev system, the ``docker volume prune --all`` command offers a quick reset method. But you should avoid using the command in shared or production ISIQ environments.

If you decide it’s appropriate to run ``docker volume prune --all``, be aware that you might need to wait a few seconds after you stop the swarm. A volume cannot be removed until all stack networks are no longer holding on to the volume. To determine whether volumes were successfully pruned, run ``docker volume ls``. It should display an empty list. If not, rerun ``docker volume prune --all``, or run ``docker volume rm xxx`` where xxx is an individual volume name returned by ``docker volume ls``.

Here are two considerations if you have a multi-node ISIQ configuration:

- (1) Volumes cannot be removed if any existing containers reference them. To make sure all containers are gone, first run on each node: ``docker container rm $(docker container ls -a -q)``
- (2) The ``docker volume prune --all`` command must be run on each node as well.

Appendix A. YAML Files

Of the four YAML files included in the ISIQ starter kit, three of them (broker, connect, app) are required for ISIQ to function. The fourth (logs) is included as an unsupported example of one approach to monitoring ISIQ. It is provided as a convenience so that you can quickly define monitoring and enhanced logging. But you are free to omit the logs stack from your swarm (by leaving `ENABLE_LOG_STACK=false` in the `isiq` shell script), or to replace the contents of `logs-stack.yml` with a different solution that better satisfies your monitoring requirements.

Note: If you decide to implement your own ISIQ monitoring and logging solution, you must still name your YAML file, `logs-stack.yml`.

broker-stack.yml

The broker stack is the base layer of ISIQ. It includes the ZooKeeper and Kafka services. In docker YAML file syntax, the “services:” label starts in column 1. It is followed by one or more services listed at the next indentation level. Each service begins with a label that names the service, followed by a block of associated settings and parameters.

Example ZooKeeper service

```
zoo1:
  image: zookeeper:3.9.1
  hostname: zoo1
  deploy:
    placement:
      constraints:
        - node.labels.isiq == node1
  logging:
    driver: "json-file"
    options:
      max-size: "2m"
      max-file: "5"
  volumes:
    - isiq_zdata:/data
    - isiq_zdatalog:/datalog
  environment:
    ZOO_MY_ID: 1
    ZOO_SERVERS: server.1=zoo1:2888:3888 server.2=zoo2:2888:3888 server.3=zoo3:2888:3888;2181
    JVMFLAGS: -Dzookeeper.snapshot.trust.empty=true
```

```
zoo2:
  image: zookeeper:3.9.1
  hostname: zoo2
  deploy:
    placement:
      constraints:
        - node.labels.isiq == node2
  logging:
```

```

driver: "json-file"
options:
  max-size: "2m"
  max-file: "5"
volumes:
  - isiq_zdata:/data
  - isiq_zdatalog:/datalog
environment:
  ZOO_MY_ID: 2
  ZOO_SERVERS: server.1=zoo1:2888:3888 server.2=zoo2:2888:3888 server.3=zoo3:2888: 3888;2181
  JVMFLAGS: -Dzookeeper.snapshot.trust.empty=true

```

```

zoo3:
  image: zookeeper:3.9.1
  hostname: zoo3
  deploy:
    placement:
      constraints:
        - node.labels.isiq == node3
  logging:
    driver: "json-file"
    options:
      max-size: "2m"
      max-file: "5"
  volumes:
    - isiq_zdata:/data
    - isiq_zdatalog:/datalog
  environment:
    ZOO_MY_ID: 3
    ZOO_SERVERS: server.1=zoo1:2888:3888 server.2=zoo2:2888:3888 server.3=zoo3:2888: 3888;2181
    JVMFLAGS: -Dzookeeper.snapshot.trust.empty=true

```

The data maintained by ZooKeeper is replicated to all ZooKeeper servers in the ZooKeeper cluster. Each ZooKeeper service has a unique name (zoo1, zoo2, etc.) because each instance must have a unique ID (ZOO_MY_ID). Also, the volumes (isiq_zdata and isiq_zdatalog) must be unique per host. By specifying the constraint (node.labels.isiq == node1), the zoo1 service is forced to run on whichever node is defined as node1. Since each ZooKeeper service runs on a separate node, each service can specify the same volume name, and the data is preserved in case a host goes down.

If you need to run ZooKeeper on a single host, there are two options. Either remove the "deploy" section and make the volume names unique for all of the ZooKeeper services, or use a single ZooKeeper service, removing zoo2 and zoo3. However, with only one machine, you can't benefit from data replication, and so there's no compelling reason to run more than one ZooKeeper instance on a single host. If you add or remove any ZooKeeper service entries, be sure to update ZOO_SERVERS to reflect which ZooKeeper services exist.

The JVMFLAGS setting was added when ZooKeeper was previously upgraded to 3.6.4, which requires a snapshot file. The file is automatically created as transactions occur, but on a new or lightly used ISIQ

system, the snapshot file might not yet exist. This results in instability in ZooKeeper and in the Kafka services that depend on it. To circumvent the problem and stabilize those services, the JVMFLAGS environment variable allows ZooKeeper to run without a snapshot:

```
JVMFLAGS: -Dzookeeper.snapshot.trust.empty=true
```

As a best practice, the next time you need to recycle ISIQ, you should set the value to 'false' since (a) the snapshot will presumably exist by then and (b) it's the more secure setting for the environment variable.

To specify which node in your environment is node1, you need to know which hosts are part of your docker cluster. The `docker node ls` command displays the nodes in the cluster. By referencing the HOSTNAME field from the output, you can run the following commands to assign labels:

```
docker node update --label-add isiq=node1 machineA
```

```
docker node update --label-add isiq=node2 machineB
```

This example assumes that machineA and machineB are the hostnames of the two nodes listed in the `docker node ls` output.

Example Kafka service

kafka1:

image: confluentinc/cp-kafka:7.6.0

hostname: kafka1

deploy:

placement:

constraints:

- node.labels.isiq == node1

logging:

driver: "json-file"

options:

max-size: "10m"

max-file: "5"

volumes:

- isiq_kdata:/var/lib/kafka/data

environment:

KAFKA_ZOOKEEPER_CONNECT: zoo1:2181,zoo2:2181,zoo3:2181

KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka1:9092

KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT

KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT

KAFKA_BROKER_ID: 1

KAFKA_NUM_PARTITIONS: 9

KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 3

KAFKA_DEFAULT_REPLICATION_FACTOR: 3

KAFKA_LOG_CLEANUP_POLICY: compact

KAFKA_DELETE_TOPIC_ENABLE: "true"

KAFKA_JMX_PORT: 50123

KAFKA_JMX_HOSTNAME: kafka1

KAFKA_JMX_OPTS: -Djava.rmi.server.hostname=kafka1 -Dcom.sun.management.jmxremote=true -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.rmi.port=50123

KAFKA_HEAP_OPTS: -Xmx1G -Xms1G

kafka2:

image: confluentinc/cp-kafka:7.6.0

hostname: kafka2

deploy:

placement:

constraints:

- node.labels.isiq == node2

logging:

driver: "json-file"

options:

max-size: "10m"

max-file: "5"

volumes:

- isiq_kdata:/var/lib/kafka/data

environment:

KAFKA_ZOOKEEPER_CONNECT: zoo1:2181,zoo2:2181,zoo3:2181

KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka2:9092

KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT

KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT

KAFKA_BROKER_ID: 2

KAFKA_NUM_PARTITIONS: 9

KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 3

KAFKA_DEFAULT_REPLICATION_FACTOR: 3

KAFKA_LOG_CLEANUP_POLICY: compact

KAFKA_DELETE_TOPIC_ENABLE: "true"

KAFKA_JMX_PORT: 50123

KAFKA_JMX_HOSTNAME: kafka2

KAFKA_JMX_OPTS: -Djava.rmi.server.hostname=kafka2 -Dcom.sun.management.jmxremote=true -
Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -
Dcom.sun.management.jmxremote.rmi.port=50123

KAFKA_HEAP_OPTS: -Xmx1G -Xms1G

kafka3:

image: confluentinc/cp-kafka:7.6.0

hostname: kafka3

deploy:

placement:

constraints:

- node.labels.isiq == node3

logging:

driver: "json-file"

options:

max-size: "10m"

max-file: "5"

volumes:

```
- isiq_kdata:/var/lib/kafka/data
environment:
  KAFKA_ZOOKEEPER_CONNECT: zoo1:2181,zoo2:2181,zoo3:2181
  KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka3:9092
  KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
  KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT
  KAFKA_BROKER_ID: 3
  KAFKA_NUM_PARTITIONS: 9
  KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 3
  KAFKA_DEFAULT_REPLICATION_FACTOR: 3
  KAFKA_LOG_CLEANUP_POLICY: compact
  KAFKA_DELETE_TOPIC_ENABLE: "true"
  KAFKA_JMX_PORT: 50123
  KAFKA_JMX_HOSTNAME: kafka3
  KAFKA_JMX_OPTS: -Djava.rmi.server.hostname=kafka3 -Dcom.sun.management.jmxremote=true -
Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -
Dcom.sun.management.jmxremote.rmi.port=50123
  KAFKA_HEAP_OPTS: -Xmx1G -Xms1G
```

Similar to the ZooKeeper services, each Kafka service also has a unique ID (KAFKA_BROKER_ID), and thus must be configured with a unique service name (kafka1, kafka2, etc.). And like ZooKeeper, the volume name must be unique per node, so each service is configured to run on a specific node. If you changed the number of ZooKeeper services, be sure to update KAFKA_ZOOKEEPER_CONNECT.

The KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR specifies the minimum number of Kafka services that must be running in the cluster for it to operate correctly. The replication factor generally does not need to be increased. If you are running only one Kafka service on a single node, it can be set to 1.

The KAFKA_DEFAULT_REPLICATION_FACTOR is a high availability option to replicate topics so that their data isn't lost if a Kafka broker goes down. The value must not be greater than the number of brokers you're running in your cluster. In a single-node configuration, this option doesn't need to be specified because the Kafka-supplied default value is 1.

The KAFKA_JMX_* settings allow the jmxtrans service in the logs stack to harvest JMX metrics data for use in Grafana.

The KAFKA_HEAP_OPTS settings are the default for Kafka brokers and are sufficient in most cases. However, if you experience problems with brokers shutting down, and see "Cannot allocate memory" errors in your broker logs, then update KAFKA_HEAP_OPTS to increase the heap size.

Example Kafka-REST service

```
kafka-rest1:
  image: confluentinc/cp-kafka-rest:7.6.0
  hostname: kafka-rest1
  logging:
    driver: "json-file"
  options:
    max-size: "5m"
    max-file: "5"
```

```
environment:
  KAFKA_REST_HOST_NAME: kafka-rest1
  KAFKA_REST_ZOOKEEPER_CONNECT: zoo1:2181
  KAFKA_REST_LISTENERS: http://0.0.0.0:7090
  KAFKA_REST_BOOTSTRAP_SERVERS:
PLAINTEXT://kafka1:9092,PLAINTEXT://kafka2:9092,PLAINTEXT://kafka3:9092
```

```
volumes:
  isiq_kdata:
  isiq_zdata:
  isiq_zdatalog:
```

The Kafka-REST service needs to connect with only one ZooKeeper instance. If you have multiple ZooKeeper services, you can include them in the KAFKA_REST_ZOOKEEPER_CONNECT section (for example, zoo1:2181,zoo2:2181,zoo3:2181). Also, if you change the number of Kafka services, be sure to update the KAFKA_REST_BOOTSTRAP_SERVERS field.

connect-stack.yml

The connect stack includes the Kafka Connect service, the consumer metrics services, and the vault service that manages encryption and decryption of product credentials.

Example connect service

```
connect:
  image: icr.io/isiq/isiq-connect:10.0.8
  deploy:
    replicas: 3
  networks:
    - default
    - broker_default
  configs:
    - source: txdef.json
      target: /etc/connect/txdef.json
    - source: isimConfiguration.json
      target: /etc/connect/isimConfiguration.json
  logging:
    driver: "json-file"
    options:
      max-size: "50m"
      max-file: "40"
  environment:
    - CONNECT_REST_PORT=8083
    - CONNECT_BOOTSTRAP_SERVERS=kafka1:9092,kafka2:9092,kafka3:9092
    - CONNECT_GROUP_ID=isiq-connect
    - CONNECT_CONFIG_STORAGE_TOPIC=isiq-connect-config
    - CONNECT_OFFSET_STORAGE_TOPIC=isiq-connect-offsets
    - CONNECT_STATUS_STORAGE_TOPIC=isiq-connect-status
    - CONNECT_CONFIG_STORAGE_REPLICATION_FACTOR=3
```

- CONNECT_OFFSET_STORAGE_REPLICATION_FACTOR=3
- CONNECT_STATUS_STORAGE_REPLICATION_FACTOR=3
- CONNECT_OFFSET_FLUSH_INTERVAL_MS=30000
- CONNECT_OFFSET_FLUSH_TIMEOUT_MS=65000
- CONNECT_REQUEST_TIMEOUT_MS=70000
- CONNECT_KEY_CONVERTER=org.apache.kafka.connect.json.JsonConverter
- CONNECT_VALUE_CONVERTER=org.apache.kafka.connect.json.JsonConverter
- CONNECT_KEY_CONVERTER_SCHEMAS_ENABLE=false
- CONNECT_VALUE_CONVERTER_SCHEMAS_ENABLE=false
- CONNECT_INTERNAL_KEY_CONVERTER=org.apache.kafka.connect.json.JsonConverter
- CONNECT_INTERNAL_VALUE_CONVERTER=org.apache.kafka.connect.json.JsonConverter
- CONNECT_LOG4J_ROOT_LOGLEVEL=INFO

CONNECT_LOG4J_LOGGERS=org.reflections=ERROR,org.apache=ERROR,org.eclipse=ERROR,org.apache.kafka.connect.runtime.distributed.DistributedHerder=INFO

- CONNECT_DEPENDENCY_WAIT_TIMEOUT=600000
- CONNECT_MAX_PARTITIONS=9
- KAFKA_HEAP_OPTS=-Xms256M -Xmx2G
- ISIQ_AUTOMATICALLY_IMPORT_CERTIFICATE=false
- ISIQ_IGI_ROLE_DESCRIPTION_SIZE=255
- ISIQ_IGItoISIM_FULFILL_USER_EVENTS=false
- ISIQ_IGISINK_ACCOUNT_CHECK_DEPENDENCY_FOR_OWNER=true
- ISIQ_ISIM_PROFILE_CHECK_INTERVAL_IN_MINUTES=10
- ISIQ_ENABLE_CIA_DATA=false
- ISIQ_METRICS_DB=http://metricsdb:8086
- ISIQ_VAULT_DB=http://vault:8200

secrets:

- source: isiq.truststore.jks
target: /etc/isiq/ssl/isiq.truststore.jks
- source: ssl.client.props
target: /etc/isiq/ssl/ssl.client.props

The connect service includes all the connectors that communicate with endpoints like ISIM and IGI. If you added or removed Kafka services, be sure to update the CONNECT_BOOTSTRAP_SERVERS environment variable.

The *_STORAGE_REPLICATION_FACTOR settings do not need to change unless you are running more than three Kafka services in a multi-node cluster. If so, adjust these values to the number of Kafka services that you defined.

If you are working with IBM Customer Support to diagnose an ISIQ data integration problem, you might be asked to change CONNECT_LOG4J_ROOT_LOGLEVEL from 'INFO' to 'DEBUG' or 'TRACE'.

The CONNECT_DEPENDENCY_WAIT_TIMEOUT value (in milliseconds) is used for both ISIM and IGI and helps address the asynchronous nature of data integration. For instance, if an ISIM account is being loaded into IGI, but the owning person for the account is not yet in IGI's PERSON table, ISIQ detects a missing dependency and caches the account while waiting for the person entry to become available. Once the person entry is found in the PERSON table, ISIQ loads the account as normal. However, if the

wait timeout interval elapses and the person is still not found, ISIQ loads an Unmatched account into IGI. To reduce or expand the timeout interval, modify `CONNECT_DEPENDENCY_WAIT_TIMEOUT`.

If you updated the `KAFKA_NUM_PARTITIONS` value in `broker-stack.yml`, you must update `CONNECT_MAX_PARTITIONS` with the same value.

`ISIQ_ISIM_PROFILE_CHECK_INTERVAL_IN_MINUTES` controls how often ISIM is queried for new object profiles. For example, if an identity adapter is imported into ISIM, you would want the ISIQ directory source connector to recognize that there's new topic data. The default check interval is 10 minutes. You can decrease the value to a minimum of 5 minutes or increase it to a maximum of 60 minutes.

By default, the IGISource connector only collects Account events to send to ISIM. If you want the connector to also collect "Create User" events, set `ISIQ_IGItoISIM_FULFILL_USER_EVENTS=true`. In addition, you need to ensure that the IGI trigger is enabled for the `EVENT_OUT_USER` table. Refer to the section entitled "Enable the `EVENT_OUT_USER` table" in Appendix A of the [ISIQ User's Guide](#).

Example metricsdb service

```
metricsdb:
  image: influxdb:1.8.10-alpine
  volumes:
    - isiq_metricsdb:/var/lib/influxdb
  deploy:
    placement:
      constraints:
        - node.labels.isiq == node1
  configs:
    - source: metricsdb.conf
      target: /etc/influxdb/influxdb.conf
  logging:
    driver: "json-file"
    options:
      max-size: "5m"
      max-file: "3"
```

The metricsdb service is an InfluxDB instance that stores details about messages consumed, consumer lag, audit events, and any alerts. Only one instance is needed. Because the metricsdb service requires a volume, a deployment constraint is used to direct it to always run on the same node. If it was started on a different node, all of its data would be missing. As mentioned in the "Docker Configs" section, the "configs" option is a way to pass in relatively small configuration data (<500KB) without creating a volume dependency or rebuilding the docker image. The actual file used for the "config" is defined at the end of the YAML file:

```
configs:
  metricsdb.conf:
    file: cfg/metricsdb/influxdb.conf
```

The path can either be absolute, or relative to where the YAML file was executed.

Example metrics service

```

metrics:
  image: icr.io/isiq/isiq-metrics: 10.0.8
  networks:
    - default
    - broker_default
  logging:
    driver: "json-file"
    options:
      max-size: "20m"
      max-file: "5"
  environment:
    KAFKA_VERSION: 0.11.0
    KAFKA_BROKERS: "\"kafka1:9092\", \"kafka2:9092\", \"kafka3:9092\""
    ZOOKEEPER_SERVERS: "\"zoo1:2181\", \"zoo2:2181\", \"zoo3:2181\""
    POLL_FREQUENCY: 30
    METRIC_DB_SERVER: 'metricsdb'
    METRIC_DB_PORT: 8086
    SCRIPT_DEBUG: 'False'
    ALERT_LAG_MIN: 1

```

The metrics service runs Burrow (<https://github.com/linkedin/Burrow>), which monitors the consumer groups active in Kafka. This information is extracted and pushed to the metricsdb service for use by ISIQ. If you changed the number of Kafka images running, update the KAFKA_BROKERS field. If you changed the number of ZooKeeper images running, update the ZOOKEEPER_SERVERS field.

The POLL_FREQUENCY value specifies the number of seconds to wait between iterations of the connector that moves data from Burrow to metricsdb. As Burrow refreshes its data every 30 seconds, setting POLL_FREQUENCY lower than 30 wastes resources. If diagnostic information is needed about Burrow data retrieval, set SCRIPT_DEBUG: 'True' and then restart the app and connect stacks.

ALERT_LAG_MIN lets you configure the threshold at which warnings about delayed updates are posted on the ISIQ System Health Dashboard. For example, if ISIM is rapidly sending updates to IGI, and IGI is lagging behind in processing the updates, you will see warning messages on the dashboard for each IGI sink topic that has an increasing lag. If you don't want to see these warnings unless the lag increase exceeds a certain threshold, set ALERT_LAG_MIN to a value greater than 1.

Example vault service

```

vault:
  image: hashicorp/vault:1.15.6
  command:
    - "server"
  environment:
    VAULT_ADDR: 'http://127.0.0.1:8200'
    VAULT_LOCAL_CONFIG: '{"backend": {"file": {"path": "/vault/file"}},
"listener":{"tcp":{"address":"0.0.0.0:8200","tls_disable":1}}, "ui":false, "disable_mlock":true}'
  options:
    volumes:
      - isiq_vault:/vault/file:
  deploy:

```

```
placement:
  constraints:
    - node.labels.isiq == node1
```

The vault service maintains the key that ISIQ uses to encrypt and decrypt credentials for connecting to your ISIM, IGI, and other data stores. As described in the “Product Password Encryption” section, the encryption mechanism relies on the vault and metricsdb services in the connect stack. In addition, there is ISIQ code running in the app stack that must reach those two services. By default, the metricsdb service is contacted via `http://metricsdb:8086`, and the vault service via `http://vault:8200`. To override the defaults, you must insert new environment variables in the connect and app stack YAML files as shown in this example:

```
ISIQ_METRICS_DB=http://mymetricsdb:18000
ISIQ_VAULT_DB=http://myvault:30000
```

app-stack.yml

The app stack contains all user interface (UI) components of ISIQ, as well as the logic for managing your product, subscription, and group definitions.

Example rest service

```
rest:
  image: icr.io/isiq/isiq-rest: 10.0.8
  deploy:
    replicas: 3
    update_config:
      parallelism: 1
      delay: 10s
      failure_action: rollback
  configs:
    - source: teamlist.json
      target: /usr/local/etc/teamlist.json
    - source: oidcSettings.json
      target: /usr/src/app/openID/oidcSettings.json
  secrets:
    - source: isiq.key
      target: /etc/crypto/isiq.key
    - source: isiq.jwt
      target: /etc/crypto/isiq.jwt
    - source: isiq-ext.jwt
      target: /etc/crypto/isiq-ext.jwt
  networks:
    - default
    - connect_default
    - broker_default
  logging:
    driver: "json-file"
  options:
```

```

    max-size: "10m"
    max-file: "5"
environment:
  - ISIQ_CONNECT_BASE_URL=http://connect:8083
  - ISIQ_OIDC_PROVIDER=Generic
  - ISIQ_SKIP_OIDC=false
  - ISIQ_SKIP_OIDC_USER=isiquser
  - ISIQ_LOG_OIDC=false
  - ISIQ_OIDC_ALLOW_SELF_SIGNED=false
  - ISIQ_OIDC_POST_METHOD=false
  - ISIQ_OIDC_CLIENT=clientId
  - ISIQ_OIDC_SECRET=clientSecret
  - ISIQ_PRODUCTS_BASE_URL=http://products:7081
  - KAFKA_REST_PROXY=http://kafka-rest1:7090
  - METRICS_DB=http://metricsdb:8086
  - BOOTSTRAP_SERVERS=kafka1:9092,kafka2:9092,kafka3:9092
  - ISIQ_SESSION_INACTIVITY_TIMEOUT_IN_MINUTES=60

```

The rest service provides backend functions for the UI. Because it doesn't need persistent storage, it is not constrained to run on any particular node. Depending on the size of your cluster, you might want to change the number of replicas.

If you set `ISIQ_SKIP_OIDC=true`, the default user ID is “isiquser”. You can override that by adding an `ISIQ_SKIP_OIDC_USER=xxx` environment variable, where “xxx” is the user ID to use in place of “isiquser”.

If you change the number of Kafka services, be sure to update the `BOOTSTRAP_SERVERS` setting. Adjust the `ISIQ_SESSION_INACTIVITY_TIMEOUT_IN_MINUTES=60` value if you prefer a shorter or longer duration for automatically logging out idle ISIQ UI sessions. Note that the value must be a positive integer no greater than 1440 (or 24 hours).

Example web-app service

```

web-app:
  image: icr.io/isiq/isiq-web-app: 10.0.8
  deploy:
    replicas: 3
    update_config:
      parallelism: 1
      delay: 10s
      failure_action: rollback
  logging:
    driver: "json-file"
    options:
      max-size: "10m"
      max-file: "5"
  environment:
    - ISIQ_REST_BASE_URL=http://rest:7060

```

The web-app service implements the ISIQ front end. Adjust the replicas setting as needed.

Example nginx service

```
nginx:
  image: nginx:1.25.4-alpine
  ports:
    - 443:443
    - 80:80
  deploy:
    mode: global
    update_config:
      parallelism: 1
      delay: 10s
      failure_action: rollback
  logging:
    driver: "json-file"
    options:
      max-size: "5m"
      max-file: "5"
  secrets:
    - source: site.key
      target: /etc/nginx/ssl/site.key
    - source: site.crt
      target: /etc/nginx/ssl/site.crt
  configs:
    - source: nginx.conf
      target: /etc/nginx/nginx.conf
```

The nginx service in the app stack functions as a web server to handle all requests to ISIQ. The service is configured for "global" deployment, which means a nginx instance runs in each node in the cluster. The `/etc/nginx/site.key` (secret key) and `/etc/nginx/site.crt` (certificate) are artifacts you must generate. Refer to the "Generating SSL Certificates" section for more details.

Example redis service

```
redis:
  image: redis:7.2.4-alpine
```

The redis service stores ISIQ user session information so that idle session timeout rules can be enforced.

Example products service

```
products:
  image: icr.io/isiq/isiq-products: 10.0.8
  networks:
    - broker_default
  logging:
    driver: "json-file"
```

```
options:
  max-size: "5m"
  max-file: "5"
environment:
  - ISIQ_PRODUCTS_REST_PORT=7081
  - BOOTSTRAP_SERVERS=kafka1:9092,kafka2:9092,kafka3:9092
```

The products service is a registry that manages the products, subscriptions, and groups defined in ISIQ. If you changed the number of Kafka servers, be sure to update the BOOTSTRAP_SERVERS setting.

logs-stack.yml

The logs stack is an example of how to monitor the ISIQ server. The monitoring has two main elements, logs and metrics:

1. The **logs** are harvested from the docker containers by Logspout, which sends them to Logstash for processing before Elasticsearch stores them. You can view the logs via Kibana.
2. The **metrics** are collected from Kafka and its JVM by Jmxtrans, which stores the data in InfluxDB. Grafana can then be used to access the metrics.

Note: While the out-of-the-box logs stack is functional, and ISIQ documentation includes numerous references to this stack, remember that it is just an example and NOT an officially supported ISIQ component. There is no requirement for you to run the logs stack when you deploy ISIQ, and any problems encountered with the constituent parts (Elasticsearch, Grafana, etc.) must be addressed with the third-party owners of those docker images. You are free to use other logging or performance monitoring methods that better meet your needs.

Example Elasticsearch service

```
elasticsearch:
  image: docker.elastic.co/elasticsearch/elasticsearch:8.12.2
  volumes:
    - isiq_esdata:/usr/share/elasticsearch/data
  networks:
    - default
    - app_default
  deploy:
    placement:
      constraints:
        - node.labels.isiq == node1
  logging:
    driver: "json-file"
    options:
      max-size: "5m"
      max-file: "5"
  environment:
    - LOGSPOUT=ignore
    - discovery.type=single-node
    - LOG4J_FORMAT_MSG_NO_LOOKUPS=true
```

Only one instance is needed, so it is constrained to run on a specific node. For high availability, you can switch to a global deployment (see “Example Logstash service”). In that circumstance, see the online doc for Elasticsearch to learn how to configure it to find other members of its cluster. The LOGSPOUT=ignore environment variable tells Logspout to not capture the docker logs from this container.

On the machines where Elasticsearch runs, you must update the vm.max_map_count setting in /etc/sysctl.conf to 262144. See this web page for details:

<https://www.elastic.co/guide/en/elasticsearch/reference/current/vm-max-map-count.html>

Example InfluxDB service

```
influxdb:
  image: influxdb:1.8.10-alpine
  volumes:
    - isiq_influxdata:/var/lib/influxdb
  deploy:
    placement:
      constraints:
        - node.labels.isiq == node2
  logging:
    driver: "json-file"
    options:
      max-size: "5m"
      max-file: "5"
  environment:
    - LOGSPOUT=ignore
```

This influxdb instance stores the JMX metrics provided by the Kafka server. The example logging stack is self-contained by design so that it can be reconfigured to use your existing logging environment if that option is preferred. Thus, you can remove this service and point jmxtrans and Grafana at the InfluxDB metrics service in the connect stack. To do so, you must add those two services to the connect stack network, as well as update the URL in the –DinfluxUrl setting on jmxtrans. Example:

```
grafana:
  ...
  networks:
    - default
    - connect_default
```

And at the end of the logs-stack.yml file, make sure there are entries for the broker and app networks:

```
networks:
  broker_default:
    external: true
  app_default:
    external: true
```

In most cases, leaving the influxdb service in the logs stack is the best strategy.

Example Logstash service

```
logstash:
  image: docker.elastic.co/logstash/logstash-oss:8.12.2
  deploy:
    mode: global
    update_config:
      parallelism: 1
      delay: 10s
      failure_action: rollback
  configs:
    - source: logstash.conf
      target: /usr/share/logstash/pipeline/logstash.conf
  logging:
    driver: "json-file"
    options:
      max-size: "5m"
      max-file: "5"
  environment:
    - LOGSPOUT=ignore
    - LOG4J_FORMAT_MSG_NO_LOOKUPS=true
```

The log messages are routed through Logstash, which parses the messages to extract various fields that Elasticsearch can index. It is configured to run one instance per node. This service **MUST** be excluded from Logspout (per the “LOGSPOUT=ignore” environment variable) or an exponentially growing set of repeated messages will be created. Make sure the “configs:” declaration points to the path where you placed the provided logstash.conf file.

Example Logspout service

```
logspout:
  image: gliderlabs/logspout:v3.2.14
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
    - /etc/hostname:/etc/host_hostname:ro
  deploy:
    mode: global
    restart_policy:
      condition: on-failure
      delay: 30s
    update_config:
      parallelism: 1
      delay: 10s
      failure_action: rollback
  logging:
    driver: "json-file"
    options:
```

```
max-size: "5m"
max-file: "5"
environment:
- ROUTE_URI=syslog+tcp://logstash:5044
```

Logspout runs on each node and listens to all the docker logs produced by containers running on that node. This is handled via the `/var/run/docker.sock` volume. The other volume, `/etc/hostname`, is mounted read-only so that Logspout can add the hostname to the log message. Its job is simply to transfer the logs from docker to Logstash where they can be processed.

Logspout will likely be the last container to start because it must first connect to Logstash, and Logstash can take a few minutes to initialize. Also, Logstash does not start unless it can successfully connect to Elasticsearch. This dependency chain of Elasticsearch -> Logstash -> Logspout can delay Logspout startup for several minutes.

If you run ISIQ along with other docker containers on the same nodes, it is highly recommended that you exclude them from Logspout with the environment variable `LOGSPOUT=ignore`. More information can be found here: <https://github.com/gliderlabs/logspout>

Example jmxtrans service

```
jmxtrans:
  image: jmxtrans/jmxtrans:latest
  deploy:
    mode: global
    update_config:
      parallelism: 1
      delay: 10s
      failure_action: rollback
  networks:
    - default
    - broker_default
  configs:
    - source: kafka_jmx
      target: /var/lib/jmxtrans/kafka.json
    - source: jvm_jmx
      target: /var/lib/jmxtrans/jvm.json
  logging:
    driver: "json-file"
    options:
      max-size: "5m"
      max-file: "5"
  environment:
    - JMXTRANS_OPTS=-Dport1=50123 -DinfluxUrl=http://influxdb:8086/ -DinfluxDb=isiq_broker
    - LOGSPOUT=ignore
```

The jmxtrans service retrieves JMX metrics from Kafka and moves the data to InfluxDB. Since there should be at least one Kafka service per node, this service is configured for global deployment. The two “configs:” declarations contain details about which JMX counters to retrieve.

In the “environment:” section, if you define a user in InfluxDB, be sure to specify the details here so that jmxtrans can connect. By default, the admin user is not used, and neither the user nor the password needs to be included in JMXTRANS_OPTS. But if you decide to define a user in InfluxDB for jmxtrans to connect as, you would do so as follows:

```
- JMXTRANS_OPTS=-Dport1=50123 -DinfluxUrl=http://influxdb:8086/ -DinfluxDb=isiq_broker -DinfluxUser=admin  
-DinfluxPwd=adminPswd
```

Example Kibana service

kibana:

image: docker.elastic.co/kibana/kibana:8.12.2

deploy:

placement:

constraints:

- node.labels.isiq == node2

networks:

- default
- app_default

configs:

- source: kibana.yml

target: /usr/share/kibana/config/kibana.yml

logging:

driver: "json-file"

options:

max-size: "5m"

max-file: "5"

environment:

- ELASTICSEARCH_URL=http://elasticsearch:9200

- SERVER_PUBLICBASEURL=https://[ISIQ_SERVER_URL]/kibana

- LOGSPOUT=ignore

Kibana provides the UI for the ELK (Elasticsearch, Logstash, Kibana) stack. By using Kibana, you can view the log messages produced by ISIQ and its dependencies. Kibana does not require a volume (its data is stored in Elasticsearch), but you should constrain Kibana to a single node so that it is easier to locate. Without a constraint, Kibana can run anywhere, and might switch between nodes whenever the logs stack is restarted. Be sure to replace “[ISIQ_SERVER_URL]” with your ISIQ hostname or IP address.

Example Grafana service

grafana:

image: grafana/grafana:9.5.16

volumes:

- type: volume

source: isiq_grafana

target: /var/lib/grafana

deploy:

placement:

constraints:

```
- node.labels.isiq == node3
networks:
- default
- app_default
configs:
- source: grafana.ini
  target: /etc/grafana/grafana.ini
logging:
driver: "json-file"
options:
  max-size: "5m"
  max-file: "5"
environment:
- LOGSPOUT=ignore
```

Grafana is used to display graphs of the Kafka metrics data. Because Grafana stores its data in a volume, it must be pinned to a particular node, or an NFS/Samba share needs to be defined. If you are using a network share, change the volume type to “bind” and set the source to be the mount point.