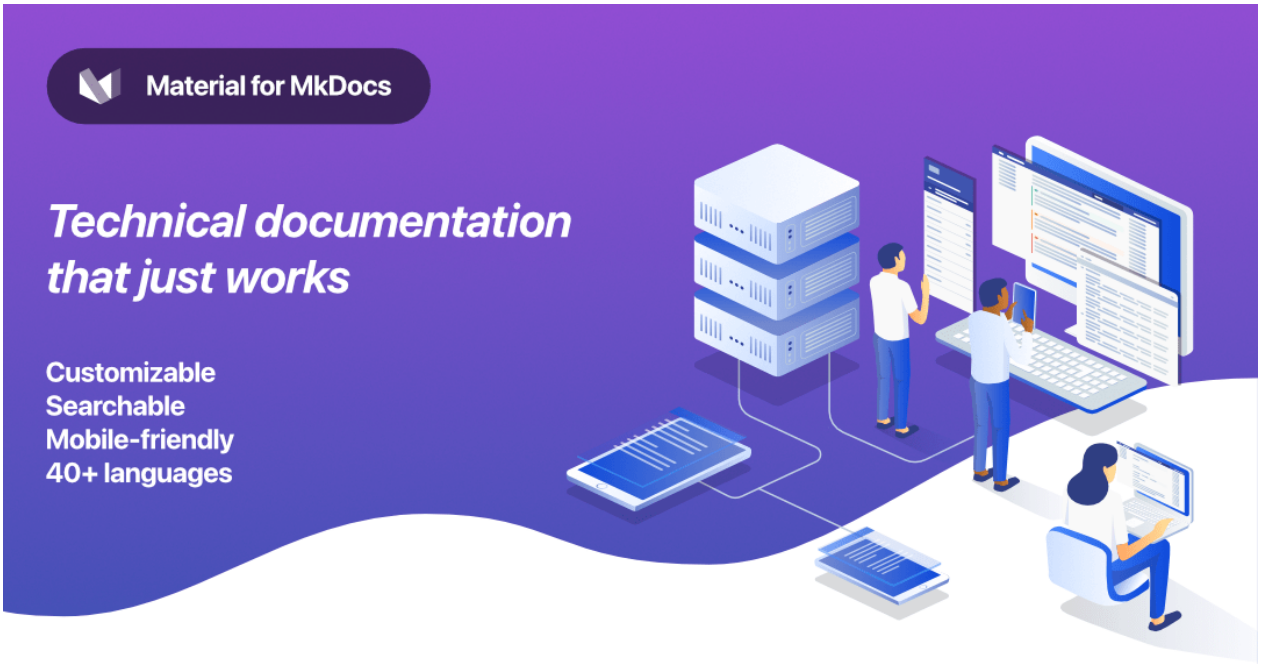


GSC STIG Manager

Installation Documentation

ZephyNayh

Table of contents

1. Welcome to GSC STIG Manager Installation Guide

GSC STIG Manager Docs is a guide for deploying STIG Manager on a RHEL 7/8 distributions system.

For full documentation, visit STIG Manager documentation here.

1.1 Project layout

```
.
├── CHANGELOG
├── CONTRIBUTING.md
├── LICENSE
├── Pipfile
├── Pipfile.lock
├── README.md
├── deployment
│   └── nginx
├── docs
│   ├── _downloads
│   ├── about
│   ├── appendix
│   ├── assets
│   ├── deployment
│   ├── index.md
│   ├── javascripts
│   └── stylesheets
├── extra_sass
│   └── style.css.scss
├── mkdocs.yml
├── pdf_event_hook
│   ├── __init__.py
│   └── __pycache__
├── site
│   ├── 404.html
│   ├── _downloads
│   ├── about
│   ├── appendix
│   ├── assets
│   ├── deployment
│   ├── index.html
│   ├── javascripts
│   ├── pdf
│   ├── search
│   ├── sitemap.xml
│   ├── sitemap.xml.gz
│   └── stylesheets
└── snippets
    └── definitions.md
```

2. Introduction

OpenSSL is a free and open-source cryptographic library that provides several command-line tools for handling digital certificates. Some of these tools can be used to act as a certificate authority.

A certificate authority (CA) is an entity that signs digital certificates. Many websites need to let their customers know that the connection is secure, so they pay an internationally trusted CA (eg, VeriSign, DigiCert) to sign a certificate for their domain.

In some cases it may make more sense to act as your own CA, rather than paying a CA like DigiCert. Common cases include securing an intranet website, or for issuing certificates to clients to allow them to authenticate to a server (eg, Apache, Nginx, OpenVPN).

2.1 SCRF Applications

- Stig-manager - Version 1.4.11
- Keycloak - Version 24.0.4

2.2 Dependencies

- OpenJDK 17
- Nginx
- MySQL

2.3 Changelog

STIG Manager Documentation-0.1.1 (2024-07-05)

- Fixed admonition issues by adding etra_sass plugin
- Updated Mkdocs Bower dependencies to most recent versions
- Changed footer/copyright link to Material theme to GitHub pages
- Made MkDocs building/serving in build process optional

STIG Manager Documentation-0.1.0 (2024-06-17)

- Initial release

3. Release Notes

3.1 Upgrading

To upgrade Zeph-OCA to the latest version, use git:

```
git clone https://github.com/Zephynyah/zeph-opensll-certificate-authority.git
```

You can determine your currently installed version using README.md:

```

```

3.2 Maintenance team

The current and past members of the MkDocs team.

- @zephynyah

3.3 Version 0.1.0 (2024-07-10)

3.3.1 Local preview

4. Contributing

5. License

The legal stuff.

5.1 Included projects

Themes used under license from the ReadTheDocs projects.

- ReadTheDocs theme - View license.

Many thanks to the authors and contributors of those wonderful projects.

6. Deployment Installation Steps

The recommended installation steps for deployment are:

1. Assumptions
2. Install Dependencies
3. Install & Configure MySQL
4. Install & Configure Keycloak
5. Install & Configure STIG Manager
6. Install & Configure NGINX
7. Deploy with TLS
8. Reboot

7. Assumptions

- The deployment directory is: `/opt/deployment/`
- The applications, config, and related files are staged in the deployment directory:

```
/opt/deployment/keycloak/
/opt/deployment/stigman/
/opt/deployment/mysql/
/opt/deployment/nginx/
/opt/deployment/docs/
```

- STIG Manager version is: `1.4.11`
- keycloak version is: `24.0.4`
- The server hostname is: `eh-stigman-1`
- Realm name is: `stigman`
- User is logged in as a user with root privileges on the system.

7.1 Summary

Summary

In these section, you have:

- Read all the Assumptions
- Read the Installation Steps

8. Dependencies

8.1 Steps

1. Installing Openjdk Using dnf
2. Open Ports for STIG Manager & keycloak (Temporary)

8.2 Installing Openjdk Using dnf

You can install Red Hat build of OpenJDK Java Runtime Environment (JRE) using the system package manager, dnf.

Run the yum command, specifying the package you want to install:

```
sudo dnf install java-17-openjdk
```

Check that the installation works:

```
java -version
```

Output:

```
openjdk version "17.0.11" 2024-04-16 LTS
OpenJDK Runtime Environment (Red_Hat-17.0.11.0.9-3) (build 17.0.11+9-LTS)
OpenJDK 64-Bit Server VM (Red_Hat-17.0.11.0.9-3) (build 17.0.11+9-LTS, mixed mode, sharing)
```

Note

If the output from the previous command shows that you have a different major version of Red Hat build of OpenJDK checked out on your system, you can enter the following command in your CLI to switch your system to use Red Hat build of OpenJDK 17:

```
sudo update-alternatives --config 'java'
```

```
There is 1 program that provides 'java'.

  Selection    Command
-----------------------------------------------
*+ 1           java-17-openjdk.x86_64 (/usr/lib/jvm/java-17-openjdk-17.0.11.0.9-2.el8.x86_64/bin/java)

Enter to keep the current selection[+], or type selection number:
```

8.3 Open Ports for STIG Manager and keycloak (Temporary)

Note

If the firewall is not running skip this step.

The `--permanent` flag is not used in these commands. Settings will go back to default after reboot.

By default, the port 443, 8080 & 54000 are filtered on Redhat 7 and 8 as you can only access this port from the actual localhost and not from any other public host. To open a port 80 on RHEL 7 and 8 Linux we need to add an iptables rule. For this RHEL uses `firewall-cmd`.

Execute the below commands to to check the firewall state.

```
firewall-cmd --state
```

Output:

```
running
[root@localhost opt]#
```

Execute the below commands to add port 8080 rule for keycloak:

```
firewall-cmd --zone=public --add-port=8080/tcp --permanent
```

Execute the below commands to add port 54000 rule for stig-manager:

```
firewall-cmd --zone=public --add-port=54000/tcp --permanent
```

Execute the below commands to reload the firewall service:

```
firewall-cmd --reload
```

Execute the below commands to check for open ports and services.

```
firewall-cmd --list-all
```

Info

Open ports are listed on line starting with ports:

The `--permanent` flag is used in this commands. Settings will persist after reboot.

Output:

```
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: ssh
  ports: 54000/tcp 8080/tcp
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

8.4 Summary

Summary

In these section, you have:

- Installed openjdk using dnf
- Opened ports Temporary for stig-manager & keycloak

9. Installing & Configuring MySQL

To install MySQL, use the following procedure.

9.1 Steps

1. Install the MySQL Server
2. Create MySQL and STIG Manager user
3. Configure MySQL for STIG Manager
4. Change the default MySQL data directory
5. Set SELinux security context for the new data directory
6. (Optional) Create a Database to Confirm Data Directory

9.2 Install the MySQL Server

Install MySQL server packages:

```
sudo dnf install mysql-server
```

Start the mysqld service:

```
systemctl start mysqld.service
```

Enable the mysqld service to start at boot:

```
sudo systemctl enable mysqld.service
```

> **Warning**
>
> Recommended: To improve security when installing MySQL, run the following command:

```
mysql_secure_installation
```

The command launches a fully interactive script, which prompts for each step in the process. The script enables you to improve security in the following ways:

- Setting a password for root accounts
- Removing anonymous users
- Disallowing remote root logins (outside the local host)

9.3 Create MySQL and STIG Manager user

Reference: 3.2.1. Configure MySQL

The STIG Manager API requires a dedicated MySQL database (equivalent to a schema in other RDBMS products). The API connects to MySQL with an account that must have a full grant to the dedicated database but does not require server administration privileges. On first bootstrap, all database tables, views, and static data will be created. Example commands to prepare MySQL for initial API execution:

> **Note**
>
> All database commands has end with a semi-colon `;` except logins in and exiting.

Login to the database:

```
mysql -u root -p
```

Create database:

```
CREATE DATABASE stigman;
```

Create API user account. Replace `new_password` with ypur password

```
CREATE USER 'stigman'@'%' IDENTIFIED BY 'new_password';
```

Grant API user account all privileges on created database

```
GRANT ALL ON stigman.* TO 'stigman';
```

Run `SHOW DATABASES` to lists the databases on the MySQL server host.

```
SHOW DATABASES
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| stigman            |
| sys                |
| tecmint            |
+--------------------+
6 rows in set (0.07 sec)
```

Execute the below commands to exit mysql.

```
exit
```

Info

Rather than using SET PASSWORD to assign passwords, ALTER USER is the preferred statement for account alterations, including assigning passwords. For example:

```
ALTER USER 'stigman'@'%' IDENTIFIED BY 'Password123!';
```

9.4 Configure MySQL for STIG Manager

Reference: 3.2.1. Configure MySQL

Warning

Suggested DB configuration options:

- `sort_buffer_size` - set to at least 2M (2097152), and perhaps up to 64M (Increasing the sort_buffer_size from the default of 256k may only be required if you have very large detail/comment text fields).
- `innodb_buffer_pool_size` - set to at least 256M (268435456), and perhaps up to 2GB (2147483648)

Login into mysql as the root user, enter password, press `enter`.

```
mysql -u root -p
```

Execute the below commands to and check the default values

```
SHOW VARIABLES LIKE 'sort_buffer_size';
```

sort_buffer_size

```
+------------------+--------+
| Variable_name    | Value  |
+------------------+--------+
| sort_buffer_size | 262144 |
+------------------+--------+
1 row in set (0.74 sec)
```

```
SHOW VARIABLES LIKE 'innodb_buffer_pool_size';
```

innodb_buffer_pool_size

```
+-------------------------+-----------+
| Variable_name           | Value     |
+-------------------------+-----------+
| innodb_buffer_pool_size | 134217728 |
+-------------------------+-----------+
1 row in set (0.01 sec)
```

Edit the config file and add the details in `/etc/my.cnf`.

```
sudo nano /etc/my.cnf
```

Within an option file, those variables are set like this:

```
[mysqld]
sort_buffer_size=64M
innodb_buffer_pool_size=2G
```

Example of /etc/my.cnf

```
#
# This group is read both both by the client and the server
# use it for options that affect everything
#
[client-server]


[mysqld]
sort_buffer_size=64M
innodb_buffer_pool_size=2G

#
# include all files from the config directory
#
!includedir /etc/my.cnf.d
```

Save and close the file when done, then restart MySQL

```
sudo systemctl restart mysqld.service
```

Perform the instructions in `Step 2` to validate the values have changed.

sort_buffer_size

```
+------------------+----------+
| Variable_name    | Value    |
+------------------+----------+
| sort_buffer_size | 67108864 |
+------------------+----------+
1 row in set (0.01 sec)
```

innodb_buffer_pool_size

```
+-------------------------+------------+
| Variable_name           | Value      |
+-------------------------+------------+
| innodb_buffer_pool_size | 2147483648 |
+-------------------------+------------+
1 row in set (0.00 sec)
```

9.5 Changing the default MySQL Data Directory

Danger

Changing the Default MySQL Data Directory in Linux has risk. Do a complete backup before performing this step.

Based on the expected use of the STIG Manager database server, we may want to change the default data directory (`/var/lib/mysql`) to a different location. This directory is expected to grow due to high usage.

Warning

GSC recommends changing the default for the following reasons:

The filesystem where `/var` is stored may collapse at one point causing the entire system to fail.

Changing the default directory to a dedicated local/network share that we want to use to store our actual data.

For these reason we will change the default MySQL data directory to a different path.

Note

We are going to assume that our new data directory is `/data/mysql` . It is important to note that this directory should be owned by `mysql:mysql` .

Execute the below commands to create the new directory if it does not exist:

```
sudo mkdir -p /data/mysql
sudo chown -R mysql:mysql /data/mysql
```

Note

To begin, it is worthy and well to identify the current data directory using the following command. Do not just assume it is still `/var/lib/mysql` since it could have been changed in the past.

Execute the below commands to identify the current MySQL Data directory:

```
sudo mysql -u root -p -e "SELECT @@datadir;"
```

After you enter the MySQL password, the output should be similar to.

Current MySQL Data Directory

```
+-----------------+
| @@datadir       |
+-----------------+
| /var/lib/mysql/ |
+-----------------+
```

To avoid data corruption, stop the service if it is currently running before proceeding. Use the systemd well-known commands to do so.

Execute the below commands to copy MySQL Data Directory to a new location:

```
sudo systemctl stop mysqld.service
sudo systemctl is-active mysqld
```

Info

If the service has been brought down, the output of the last command should be as follows:

Identify MySQL Data Directory

```
[root@localhost ~]# systemctl is-active mysqld.service
inactive
[root@localhost ~]#
```

Then copy recursively the contents of `/var/lib/mysql` to `/data/mysql` preserving original permissions and timestamps:

```
sudo cp -R -p /var/lib/mysql/* /data/mysql
```

or

```
sudo rsync -av /var/lib/mysql/* /data/mysql
```

Edit the configuration file (my.cnf) to indicate the new data directory (`/data/mysql in this case`).

Execute the below commands to configure the New MySQL data directory:

```
sudo nano /etc/my.cnf
```

Locate or add the `[client]` sections and make the following changes after the `[client-server]` block:

```
[client]
port=3306
socket=/data/mysql/mysql.sock
```

Save the changes and then proceed with the next step.

Configure New MySQL Data Directory

```
#
# This group is read both both by the client and the server
# use it for options that affect everything
#
[client-server]


[mysqld]
sort_buffer_size=64M
innodb_buffer_pool_size=2G


[client]
port=3306
socket=/data/mysql/mysql.sock


#
# include all files from the config directory
#
!includedir /etc/my.cnf.d
```

Now lets change the server settings in `/etc/my.cnf.d/mysql-server.cnf`

```
sudo nano /etc/my.cnf.d/mysql-server.cnf
```

Make the following changes then save and exit the editor:

- `datadir=/var/lib/mysql` -> `/data/mysql`
- `socket=/var/lib/mysql/mysql.sock` -> `/data/mysql/mysql.sock`

```
#
# This group are read by MySQL server.
# Use it for options that only the server (but not clients) should see
#
# For advice on how to change settings please see
# http://dev.mysql.com/doc/refman/en/server-configuration-defaults.html

# Settings user and group are ignored when systemd is used.
# If you need to run mysqld under a different user or group,
# customize your systemd unit file for mysqld according to the
# instructions in http://fedoraproject.org/wiki/Systemd

[mysqld]
datadir=/data/mysql
socket=/data/mysql/mysql.sock
log-error=/var/log/mysql/mysqld.log
pid-file=/run/mysqld/mysqld.pid
```

9.6 Set SELinux Security Context to Data Directory

> **Note**
>
> This step is only applicable to **RHEL/CentOS** and its derivatives.

Add the SELinux security context to `/data/mysql` before restarting MySQL by running the following commands:

```
sudo semanage fcontext -a -t mysqld_db_t "/data/mysql(/.*)?"
sudo restorecon -R /data/mysql
```

Next restart the MySQL service then check the status.

MySQL commands

```
sudo systemctl start mysqld.service
sudo systemctl is-active mysqld
```

Now, use the same command as in `Step 1` to verify the location of the new data directory:

Verify MySQL New Data Directory

```
sudo mysql -u root -p -e "SELECT @@datadir;"
```

New MySQL Data Directory

```
+-----------------+
| @@datadir       |
+-----------------+
| /data/mysql/    |
+-----------------+
```

9.7 (Optional) Create a Database to Confirm Data Directory

Run the following command to login to MySQL and create a new database in the new directory `/data/mysql` :

Check MySQL New Data Directory

```
mysql -u root -p -e "CREATE DATABASE gsctest;"
```

If you did not receive an error, Execute the below commands to remove the new Database you just created:

Drop MySQL gsctest Data Base

```
mysql -u root -p -e "DROP DATABASE gsctest;"
```

Both commands are expected to return an "ok" message.

9.8 Summary

> **Summary**
>
> In these procedures, on a RHEL 7/8 distributions, you have:
>
> • Install the MySQL Server
> • Created the STIG Manager MySQL Database.
> • Set the suggested DB configuration options in a MySQ.
> • Change the default MySQL data directory.
> • Set SELinux security context for the new data directory

10. Installing & Configuring Keycloak

10.1 Procedures

1. Extract Keycloak Server
2. Create a Systemd Unit File for Keycloak
3. Create a Symbolic Link (symlink) Folder
4. Enable Keycloak Service on System Startup
5. Create the First Administrator From Command Line
6. Login and Using the Admin Console
7. Import STIG Manager Realm Template
8. Start keycloak As a Service

10.2 Extract Keycloak Server

We are going to install Keycloak to `/opt/` directory, so we will copy/or extract the Keycloak package to that location.

Execute the below commands to move to the `/opt` directory:

```
cd /opt/
```

Execute the below commands to unzip the keycloak:

```
sudo unzip /opt/deployment/keycloak/keycloak-24.0.4.zip
```

10.3 Creating a systemd Unit File for Keycloak

Copy systemd unit file (keycloak.service) under `/opt/deployment/keycloak/scripts/systemd/` to `/etc/systemd/system/` directory.

> **Info**
>
> More information bout system D files can be found here. systemd Unit Files

Execute the below commands to move to `../systemd/` deployment directory:

```
cd /opt/deployment/keycloak/scripts/systemd/
```

Execute the below commands to copy the file:

```
sudo cp ./keycloak.service /etc/systemd/system/keycloak.service
```

Below is the content of the `keycloak.service` file

```
[Unit]
Description=The Keycloak Application Server
After=syslog.target network.target
Before=httpd.service

[Service]
Type=simple
WorkingDirectory=/opt/keycloak
Environment="KC_HTTP_ENABLED=true"
Environment="KC_HOSTNAME_STRICT=false"
AmbientCapabilities=CAP_SYS_ADMIN
LimitNOFILE=102642
PIDFile=/var/run/keycloak/keycloak.pid
ExecStart=/bin/bash -c "./bin/kc.sh start"

# Logging
StandardOutput=append:/var/log/keycloak.log
StandardError=append:/var/log/keycloak.log
```

```
# Restart
Restart=always
RestartSec=60
TimeoutStartSec=60
TimeoutStopSec=60

SuccessExitStatus=0 143

[Install]
WantedBy=multi-user.target
```

10.4 Create a symbolic link (symlink) folder.

A symlink is used hear to make the keycloak folder appear in the same directory on the filesystem but with different names.

> **Info**
>
> This is used to simply version changes without changing the main configuration files.
>
> For example: the systemd file "`keycloak.service`", will always point to Keycloak rather that a specific version.

Execute the below commands to create the symlink for keycloak:

```
sudo ln -s /opt/keycloak-24.0.4 /opt/keycloak
```

> **Note**
>
> Below is an example of what this directory would look like for an upgrade.
>
> To remove the `sudo unlink /opt/keycloak`
>
> To link `keycloak-24.0.5` run `sudo ln -s /opt/keycloak-24.0.5 /opt/keycloak`

Example:

```
...
lrwxrwxrwx. 1 root root   20 Jul  5 20:10 keycloak -> /opt/keycloak-24.0.5
drwxr-xr-x. 8 root root  157 Jul  5 20:09 keycloak-24.0.4
drwxr-xr-x. 8 root root  138 Jul  5 20:29 keycloak-24.0.5
...
```

10.5 Enable keycloak service on system startup

Reload systemd manager configuration

```
sudo systemctl daemon-reload
```

Execute the below commands to enable keycloak service on system startup:

```
sudo systemctl enable keycloak
```

Output:

```
Created symlink /etc/systemd/system/multi-user.target.wants/keycloak.service → /etc/systemd/system/keycloak.service.
[root@localhost opt]#
```

10.6 Creating the first administrator from command line.

> **Info**
>
> After installing Keycloak, you need an administrator account that can act as a super admin with full permissions to manage Keycloak. With this account, you can log in to the Keycloak Admin Console where you create or import realms and users and register applications that are secured by Keycloak.

If you cannot access the server from a localhost address or just want to start Keycloak from the command line, use the **KEYCLOAK_ADMIN** and **KEYCLOAK_ADMIN_PASSWORD** environment variables to create an initial admin account.

```
export KEYCLOAK_ADMIN=admin
export KEYCLOAK_ADMIN_PASSWORD=Password123!

/opt/keycloak/bin/kc.sh start-dev
```

Output:

```
UPDATE SUMMARY
Run:                        124
Previously run:               0
Filtered out:                 0
-------------------------------
Total change sets:          124

2024-07-08 20:39:30,834 INFO  [org.keycloak.services] (main) KC-SERVICES0050: Initializing master realm
2024-07-08 20:39:37,301 INFO  [org.keycloak.services] (main) KC-SERVICES0009: Added user 'admin' to realm 'master'
2024-07-08 20:39:37,526 INFO  [io.quarkus] (main) Keycloak 24.0.4 on JVM (powered by Quarkus 3.8.4) started in 26.614s. Listening on: http://0.0.0.0:8080
2024-07-08 20:39:37,527 INFO  [io.quarkus] (main) Profile dev activated.
2024-07-08 20:39:37,527 INFO  [io.quarkus] (main) Installed features: [agroal, cdi, hibernate-orm, jdbc-h2, keycloak, logging-gelf, narayana-jta, reactive-
routes, resteasy-reactive, resteasy-reactive-jackson, smallrye-context-propagation, vertx]
2024-07-08 20:39:37,532 WARN  [org.keycloak.quarkus.runtime.KeycloakMain] (main) Running the server in development mode. DO NOT use this configuration in
production.
```

> **Info**
>
> From a localhost, open the web browser, go to the http://localhost:8080 URL.
>
> From a different computer, open the web browser, go to the http://eh-stigman-1:8080 URL.

10.7 Start Keycloak as a service.

Execute the below commands to exit keycloak.

```
ctl + c
```

Execute the below commands to build keycloak:

```
sudo /opt/keycloak/bin/kc.sh build
```

Execute the below commands to restart service:

```
sudo systemctl restart keycloak.service
```

Execute the below commands to check the status of the service:

```
sudo systemctl status keycloak.service
```

Execute the below commands to look at the logs

```
tail -f /var/log/keycloak.log
```

10.8 Login and using the Admin Console

You configure realms and perform most administrative tasks in the Keycloak Admin Console. Before continuing this step make sure:

- The Keycloak Server is started and running.
- You have an administrator account you create in Step 7.

Go to the URL for the Admin Console.

- From localhost, use this URL: http://localhost/admin
- From remote host, use this URL: http://eh-stigman-1:8080/admin

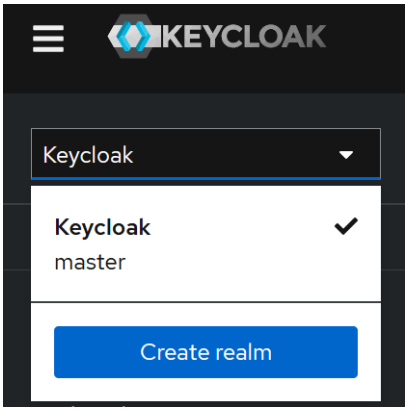
> **Info**
>
> This action should display the Admin Console.
>
> On the Keycloak Welcome Page, enter the Admin username and password you created in the initial previous guide.

10.9 Importing the STIG Manager Realm Template

A realm in Keycloak is equivalent to a tenant. Each realm allows an administrator to create isolated groups of applications and users. Initially, Keycloak includes a single realm, called master. Use this realm only for managing Keycloak and not for managing any applications.

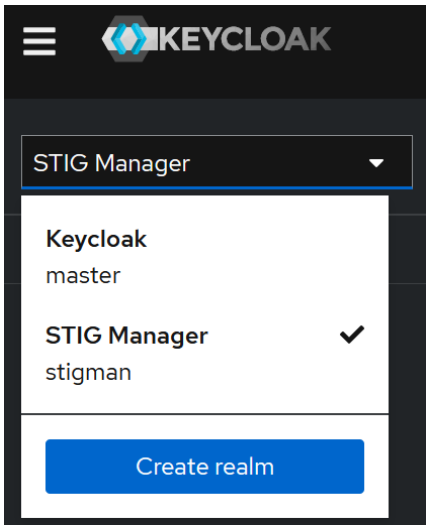
Use these steps to create the first realm form Admin console.

1. Open the Keycloak Admin Console http://eh-stigman-1:8080/admin.



1. Click Keycloak next to master realm, then click Create Realm.
2. You can drag and drop the `import_realm.json` file or press `browse` to to navigate to its location.
3. Click `Create`.

> **Info**
>
> You should receive a success message notification.
>
> The new realm should now be available for further configuration.



10.10 Summary

Summary

In these procedures, on a RHEL 7/8 distributions, you have:

- Installed the JRE requirement for keycloak.
- Created the Keycloak Server opt directory.
- Created a user and group for Keycloak.
- Change permission and ownership of keycloak directory.
- Created a systemd Unit file for keycloak
- Created a symbolic link (symlink) folder for keycloak.
- Enabled keycloak service on system startup.
- Created the first administrator from command line.
- Started keycloak as a Service.
- Log into the Admin Console and import a realm template.

11. Installing & Configuring STIG Manager

> **Info**
>
> STIG Manager binaries are made available with each release.

11.1 Steps

1. Extract STIG Manager Server
2. Create a systemd Unit File for stigman
3. Create a symbolic link (symlink) folder
4. Enable SIG Manager service on system startup
5. Configure STIG Manager

11.2 Extract STIG Manager Application

We are going to install STIG Manager to `/opt/` directory, so we will extract the STIG Manager package to that location.

Execute the below commands to move to the `/opt` directory:

```
cd /opt/
```

Execute the below commands to unzip the STIG Manager:

```
sudo tar -xf /opt/deployment/stigman/stig-manager-linux-1.4.11.tar.xz \
--one-top-level="stig-manager-1.4.11" \
--strip-components=1
```

11.3 Creating a systemd Unit File for stigman

Copy systemd unit file (`stigman.service`) under `/opt/deployment/stigman/scripts/systemd/` to `/etc/systemd/system/` directory.

> **Info**
>
> More information bout system D files can be found here. systemd Unit Files

Execute the below commands to move to `../systemd/` deployment directory:

```
cd /opt/deployment/stigman/scripts/systemd/
```

Execute the below commands to copy the file:

```
sudo cp ./stigman.service /etc/systemd/system/stigman.service
```

Execute the below commands to view contents of the file. Exit the editor when done.

```
nano /etc/systemd/system/stigman.service
```

Below is the content of the `stigman.service` file

```
[Unit]
Description=The STIG Manager Application Server
After=syslog.target network.target
Before=httpd.service

[Service]
Type=simple
LimitNOFILE=102642
```

```
AmbientCapabilities=CAP_SYS_ADMIN
PIDFile=/var/run/stigman/stigman.pid
WorkingDirectory=/opt/stig-manager
ExecStart=/bin/bash -c "./stig-manager.sh"

# Logging
StandardOutput=append:/var/log/stig-manager.log
StandardError=append:/var/log/stig-manager.log

SuccessExitStatus=0 143
Restart=always
RestartSec=60
TimeoutStartSec=60
TimeoutStopSec=60

[Install]
WantedBy=multi-user.target
```

Execute the below commands to set the proper permissions over a unit file:

```
chown root:root /etc/systemd/system/stigman.service
chmod 0644 /etc/systemd/system/stigman.service
```

11.4 Create a symbolic link (symlink) folder.

A symlink is used hear to make the stig-manager-1.4.11 folder appear in the same directory on the filesystem but with different names.

Info

This is used to simply version changes without changing the main configuration files.

For example: the systemd file "`stigman.service`", will always point to stig-manager rather that a specific version.

Execute the below commands to create the symlink for stig-manager:

```
sudo ln -s /opt/stig-manager-1.4.11 /opt/stig-manager
```

Note

Below is an example of what this directory would look like for an upgrade.

To remove the `sudo unlink /opt/stig-manager`

To link `stigman-1.4.12` run `sudo ln -s /opt/stigman-1.4.12 /opt/stig-manager`

```
```sh
...
lrwxrwxrwx. 1 root     root       21 Jul  6 01:45 keycloak -> /opt/keycloak-24.0.4/
drwxr-xr-x. 8 keycloak keycloak  138 Jul  6 11:00 keycloak-24.0.4
drwxr-xr-x. 8 root     root      163 Jul  6 02:12 keycloak-24.0.5
...
lrwxrwxrwx. 1 stigman  stigman    24 Jul  7 16:58 stig-manager -> /opt/stig-manager-1.4.11
drwxr-xr-x. 2 root     root       61 Jul  7 16:55 stig-manager-1.4.11
...
```

11.5 Enable stigman service on system startup

Reload systemd manager configuration

```
sudo systemctl daemon-reload
```

Execute the below commands to start the service:

```
sudo systemctl start stigman
```
```

Execute the below commands to check the status of the service:

```
sudo systemctl status stigman
```

Execute the below commands to enable the service on system startup:

```
sudo systemctl enable stigman
```

Execute the below commands to look at the logs

```
tail -f /var/log/stig-manager.log
```

Execute the below commands to stop the service:

```
sudo systemctl stop stigman
```

11.6 Configure STIG Manager

Reload systemd manager configuration

Execute the below commands to edit stig-manager environment variables file.

```
sudo nano /opt/stig-manager-1.4.11/stig-manager.sh
```

stig-manager.sh Before

```
# export STIGMAN_CLASSIFICATION=
...
# export STIGMAN_DB_PASSWORD=
...
# export STIGMAN_CLIENT_OIDC_PROVIDER=
...
# export STIGMAN_OIDC_PROVIDER=
```

Un-comment the following variable and update their values:

stig-manager.sh After

```
export STIGMAN_CLASSIFICATION=C
...
export STIGMAN_DB_PASSWORD=Password123!
...
export STIGMAN_CLIENT_OIDC_PROVIDER=http://192.168.100.119:8080/realms/stigman
...
export STIGMAN_OIDC_PROVIDER=http://localhost:8080/realms/stigman
```

Execute the below commands to restart service:

```
sudo systemctl restart stigman.service
```

11.7 Summary

Summary

In these procedures, on a RHEL 7/8 distributions, you have:

- Extracted STIG Manager Server
- Created a systemd Unit File for stigman
- Created a symbolic link (symlink) folder.
- Enabled SIG Manager service on system startup
- Configured STIG Manager

12. Installing & Configuring NGINX

The aim of this section is to get you started with basic Nginx web-server installation using the dnf install nginx command and configuration on RHEL 8 / CentOS 8. Nginx web server is an Apache alternative with a capability to be also used as reverse proxy, load balancer, mail proxy and HTTP cache.

12.1 Steps

1. Install Nginx
2. Enable HTTPS SSL support on Nginx
3. Create PKI directory for Keys and Certificates
4. Generate SSL CA Keys and Certificates
5. Generate SSL Server Keys and Certificates
6. Open HTTPS and HTTP ports (443 and 80)

12.2 Install Nginx

Install package nginx using the dnf command.

```
sudo dnf install nginx
```

Start the Nginx service:

```
sudo systemctl start nginx
```

To ensure that Nginx starts after the reboot enable systemd service the nginx:

```
sudo systemctl enable nginx
```

Execute the below commands to check the status of the service:

```
sudo systemctl status nginx.service
```

Output:

```
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
```

Open HTTP firewall port 80:

```
sudo firewall-cmd --zone=public --permanent --add-service=http
sudo firewall-cmd --reload
```

Backup the nginx configuration file `/etc/nginx/nginx.conf`

```
cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.distro
```

Info

All should now be ready to access Nginx from a remote host.

Access the Nginx welcome page.

- Open browser and navigate to `http://eh-stigman-1/` URL.

12.3 Enable HTTPS SSL support on Nginx

Copy the nginx sever configuration file to `/etc/nginx/conf.d`

Execute the below commands to copy the config file:

```
sudo cp /opt/deployment/nginx/etc/nginx/conf.d/stigman.nginx.conf /etc/nginx/conf.d
```

View this file as plain text

```
# generated 2024-06-05, Mozilla Guideline v5.7

server {
    listen                  443 ssl http2;
    listen                  [::]:443 ssl http2;
    server_name             eh-stigman-1;
    ssl_certificate         /etc/pki/nginx/eh-stigman-1.crt;
    ssl_certificate_key     /etc/pki/nginx/private/eh-stigman-1.key;

    ssl_session_timeout     1d;
    ssl_session_cache       shared:MozSSL:10m;  # about 40000 sessions
    ssl_session_tickets     off;

    # curl https://ssl-config.mozilla.org/ffdhe2048.txt > /path/to/dhparam
    # ssl_dhparam             /etc/pki/nginx/dhparam;

    # intermediate configuration
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-CHACHA20-POLY1305;
    ssl_prefer_server_ciphers off;

    # HSTS (ngx_http_headers_module is required) (63072000 seconds)
    add_header Strict-Transport-Security "max-age=63072000" always;


    # verify chain of trust of OCSP response using Root CA and Intermediate certs
    # ssl_trusted_certificate /etc/pki/nginx/ca-chain.cert.pem;


    location / {
        proxy_pass              http://eh-stigman-1:54000/;
    }

    location /auth/ {
        proxy_pass              http://eh-stigman-1:8080/auth/;
        proxy_set_header        Host               $host;
        proxy_set_header        X-Real-IP          $remote_addr;
        proxy_set_header        X-Forwarded-For    $proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Host   $host;
        proxy_set_header        X-Forwarded-Server $host;
        proxy_set_header        X-Forwarded-Port   $server_port;
        proxy_set_header        X-Forwarded-Proto  X$scheme;
        proxy_set_header        ssl-client-cert    $ssl_client_escaped_cert;
        proxy_buffer_size       128k;
        proxy_buffers           4 256k;
        proxy_busy_buffers_size 256k;
    }
}
```

12.4 Create PKI directory for Keys and Certificates

Create a directory to hold the SSL certificate and the private key for the Nginx server.

> **Info**
>
> The cert will be created and placed in `/etc/pki/nginx/server.crt`
>
> the key will be created and placed in `/etc/pki/nginx/private/server.key`

Execute the below commands to create the directories:

```
sudo mkdir -p /etc/pki/nginx/private/
```

```
sudo mkdir -p /opt/pki
```

```
cd /opt/pki
```

12.5 Generate SSL CA Keys and Certificates

> **Note**
>
> Skip the two step below if you already have a a signing certificate step if you

Execute the below command to generate CA's private key and self-signed certificate

```
openssl req -x509 -newkey rsa:4096 \
  -days 3650 \
  -nodes \
  -keyout ca-key.pem \
  -out ca-cert.pem \
  -subj "/C=US/ST=Connecticut/L=East Hartford/O=Alice Ltd/OU=Finance/CN=Alice Ltd Certificate Authority/emailAddress=alice.ca@support.com"
```

Execute the below command to create the CA's self-signed certificate:

```
openssl x509 -in ca-cert.pem -noout -text
```

12.6 Generate SSL Server Keys and Certificates

Execute the below command to generate web server's private key and certificate signing request (CSR)

```
openssl req -newkey rsa:4096 \
    -keyout server-key.pem \
    -out server-req.pem \
    -subj "/C=US/ST=Connecticut/L=East Hartford/O=Alice Ltd/OU=Finance/CN=*.staging.local/emailAddress=alice.finance@support.com"
```

Execute the below command to create the config file:

> **Note**
>
> Edit the DNS, IP, and wildcard information below before running the command.

```
echo "subjectAltName=DNS:*.monofinance.net,DNS:*.monofinance.com,DNS:*.monofinance.org,IP:0.0.0.0" \
 > server-ext.cnf
```

Execute the below command to use CA's private key to sign web server's CSR and get back the signed certificate

```
openssl x509 -req -in server-req.pem \
    -CA ca-cert.pem \
    -CAkey ca-key.pem \
    -CAcreateserial \
    -out server-cert.pem -days 3650 \
    -extfile server-ext.cnf
```

Execute the below command below to show the Server's signed certificate:

```
openssl x509 -in server-cert.pem -noout -text
```

Execute the below command below to verify Server and CA certificates:

```
openssl verify -CAfile ca-cert.pem server-cert.pem
```

Execute the below commands below to copy the file to the appropriate directories:

```
cp server-cert.pem /etc/pki/nginx/server.crt
cp server-cert.key  /etc/pki/nginx/private/server.key
```

Reload the Nginx configuration to pick load the new certs:

```
sudo systemctl restart nginx.service
```

12.7 Open HTTPS and HTTP ports (443 and 80)

Execute the below command below to open the port:

```
firewall-cmd --zone=public --permanent --add-service=https
firewall-cmd --zone=public --permanent --add-service=http
firewall-cmd --reload
```

Access the Nginx welcome page. All should now be ready to access Nginx from a remote host. Open the browser and paste the URL or navigate to https://eh-stigman-1.

12.8 Summary

Summary

In these procedures, on a RHEL 7/8 distributions, you have:

- Installed and enabled Nginx
- Enabled HTTPS SSL support on Nginx
- Created PKI directory for Keys and Certificates
- Generated SSL CA Keys and Certificates
- Generated SSL Server Keys and Certificates
- Opened HTTPS and HTTP ports (443 and 80)

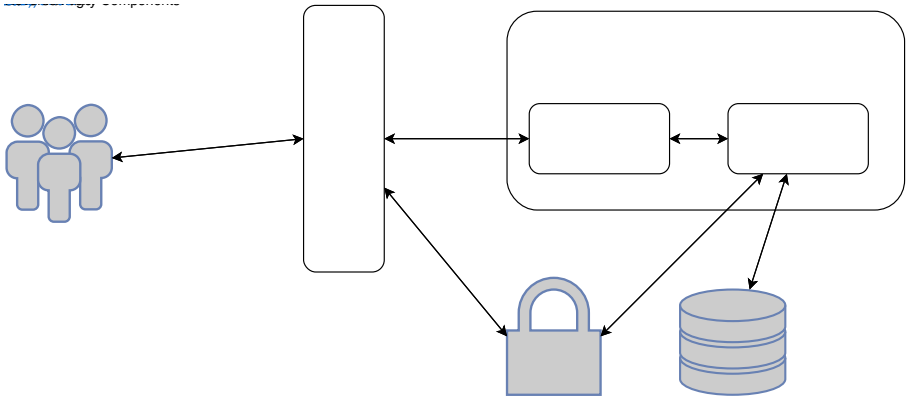
13. Deploy with TLS

13.1 Steps

1. Configure the Reverse Proxy
2. Allow Keycloak to Communicate Through the Proxy
3. Allow STIG Manager to Communicate with Keycloak

13.2 Configure the Reverse Proxy

To support HTTPS connections, STIG Manager components should be situated behind a reverse proxy. Configure the reverse proxy (such as nginx) in accordance with publisher documentation, local security requirements, and Keycloak documentation. In either case, you will have to set Keycloak environment variable `PROXY_ADDRESS_FORWARDING=true` and make sure appropriate headers are forwarded.



13.3 Allow Keycloak to Communicate Through the Proxy.

Execute the below command to edit the keycloak unit file.

```
sudo nano /opt/keycloak/conf/keycloak.conf
```

Add the below snippet to the bottom of the configuration file.

```
# Custom Configuration
proxy-address-forwarding=true
proxy=edge
```

keycloak.conf: after changes

```
# Basic settings for running in production. Change accordingly before deploying the server.

# Database

# The database vendor.
#db=postgres

# The username of the database user.
#db-username=keycloak

# The password of the database user.
#db-password=password

# The full database JDBC URL. If not provided, a default URL is set based on the selected database vendor.
#db-url=jdbc:postgresql://localhost/keycloak

# Observability

# If the server should expose healthcheck endpoints.
#health-enabled=true

# If the server should expose metrics endpoints.
#metrics-enabled=true
```

```
# HTTP

# The file path to a server certificate or certificate chain in PEM format.
#https-certificate-file=${kc.home.dir}conf/server.crt.pem

# The file path to a private key in PEM format.
#https-certificate-key-file=${kc.home.dir}conf/server.key.pem

# The proxy address forwarding mode if the server is behind a reverse proxy.
#proxy=reencrypt

# Do not attach route to cookies and rely on the session affinity capabilities from reverse proxy
#spi-sticky-session-encoder-infinispan-should-attach-route=false

# Hostname for the Keycloak server.
#hostname=myhostname

# Custom Configuration
proxy-address-forwarding=true
proxy=edge
```

Save and exit.

Execute the below command to restart keycloak:

```
sudo systemctl restart keycloak.service
```

Info

All should now be ready to access Keycloak from a remote host.

Access the Keycloak sign-in page.

- Open browser and paste the URL or navigate to https://eh-stigman-1 URL.
- Login with the Admin user name and password created earlier.
- All setting and Reals should still be accessible and unchanged.

13.4 Allow STIG Manager to Communicate with Keycloak.

Open browser and paste the url or navigate to https://eh-stigman-1/stigman URL.

Info

STIG Manger will not communicate with keycloak because nginx.

We need to:

- Change STIG Manager's client OIDC provider to now use `https://`.
- Remove the port `8080`

Execute the below commands to edit STIG Manager's environment file.

```
sudo nano /opt/stig-manager/stig-manager.sh
```

stig-manager.sh: after changes

```
...
#====================================================================
# STIGMAN_CLIENT_OIDC_PROVIDER
#
#  | Default: Value of "STIGMAN_OIDC_PROVIDER" | Client override of the base URL
#  of the OIDC provider issuing signed JWTs for the API.  The string "/.well-
#  known/openid-configuration" will be appended by the client when fetching
#  metadata.
#
#  Affects: Client
#====================================================================
export STIGMAN_CLIENT_OIDC_PROVIDER=https://eh-stigman-1/realms/stigman
...
```

Save and exit.

Execute the below commands restart STIG Manager:

```
sudo systemctl restart stigman.service
```

Open browser and paste the url or navigate to https://eh-stigman-1/stigman URL.

13.5 Summary

Summary

In these procedures, on a RHEL 7/8 distributions, you have:

- Configured the Reverse Proxy
- Allowed Keycloak to Communicate Through the Proxy
- Allowed STIG Manager to Communicate with Keycloak
- Fixed STIG Manager ServiceWorker.js Error

14. Reboot and validation

Perform the following procedure below to reboot the Host Server and validate previous configurations

14.1 Procedures

- Perform and Host Server reboot
- Validate Keycloak services is still running
- Validate STIG Manager services is still running
- Validate Nginx services is still running
- Validate Temporarily enabled port are not open
- Validate Permanently enabled Services are still open

14.2 Perform and Host Server reboot

Execute the below commands to reboot the computer.

```
reboot
```

Wait for the server to reboot the log back into the shell.

14.3 Validate Keycloak services is still running

Execute the below commands to check the status of the service:

```
sudo systemctl status keycloak.service
```

14.4 Validate STIG Manager services is still running

Execute the below commands to check the status of the service:

```
sudo systemctl status stigman.service
```

14.5 Validate Nginx services is still running

Execute the below commands to check the status of the service:

```
sudo systemctl status nginx.service
```

14.6 Validate Temporarily enabled port are not open

Execute the below commands to check for open ports and services.

```
sudo firewall-cmd --zone=public --permanent --list-ports
```

Info

Port `8080/tcp` and `54000/tcp` should not be present in this output.

Or using the list-all command line option

```
sudo firewall-cmd --list-all
```

Info

Open ports are listed on line starting with ports: Port `8080/tcp` and `54000/tcp` should not be present in this output.

Output:

```
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: http https ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

14.7 Validate Permanently enabled Services are still open

Execute the below commands to check for open ports and services.

```
sudo firewall-cmd --zone=public --permanent --list-services
```

Info

`http` and `https` service should be present in this output.

Output:

```
http https ssh
```

Or using the list-all command line option

```
sudo firewall-cmd --list-all
```

Info

Open services are listed on line starting with services: `http` and `https` Service should be present in this output.

Output:

```
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: http https ssh
  ports:
  protocols:
  forward: no
  masquerade: no
```

```
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@localhost opt]#
```

15. Index