Shibboleth-IdP Standalone Windows Container Release 17070

Introduction

This packaged TIER Shibboleth-IdP release is a standalone Docker container (Windows-based) implementation of the Shibboleth IdP.

What is the TIER Shibboleth IdP release?

- A specifically packaged, distributed, instrumented, and operated implementation of the standard Shibboleth IdP.

What is a container?

- A lightweight, executable package of an application, which typically includes everything needed to run it. Though similar, they are not VMs as there is no hypervisor involved. Therefore, they are much leaner and more scalable.

What is Docker?

- An ecosystem for building, packaging, deploying, and running applications using containers.

Why Docker?

- Docker is the leader in container development and operations and is supported across a number of platforms!

Things to think about for deploying the TIER Shib IdP

- Config type: burned, mounted, or hybrid (see explanation below)
- HA/Container Orchestration needs (there are instructions for Docker Swarm below)
- Image lifecycle (know what is in your containers and what happens to old containers)

Building/Operating the Windows-based Shibboleth IdP

Getting Started

- Using the instructions below, you can learn how to build a docker image, store it somewhere else, then setup a swarm service to pull this image and run it in an HA manner.
- You'll need (at least) 2 VMs - call one 'manager1' and the other 'worker1'
- It is best to assign appropriate hostnames to your VMs for their respective roles (manager/worker).

Prepare for building an image

- The needed files can be obtained from the TIER github repository
  - It's best to be in a new empty directory. You can create a new one with a name you like and/or the command below will create a new child directory in your current directory named shib-idp_noVM
  - *git clone https://github.internet2.edu/docker/shib-idp_noVM*
- Decide on a config type
  - **Burned**
    - All configuration is supplied when the image is built and is "burned" into the resulting image. The image stands alone and has no external dependencies, but contains sensitive information in the images (like passwords, private keys, etc.). To update the configuration, new images are built and rotated into production. This image cannot take advantage of the IdP's automatic configuration reloading features.
  - **Mounted**
    - All configuration is dynamically mounted at run-time. The resulting image will be dependent upon the presence of local configuration files on all docker nodes that run the container (so, as a result, the image does not stand alone). These types of images do not contain sensitive information. Further, these images are able to take advantage of the IdP's automatic configuration reloading features. To update the configuration, it is necessary to sync the updated file(s) to each docker node running the container.
  - **Hybrid**
    - This type of configuration looks mostly like the 'burned' type detailed above, with the exception that all sensitive information, along with the most dynamic config files for the IdP, has been removed from the built image and will instead be supplied at run-time using docker secrets. Therefore, these images do not contain sensitive information. They can use the IdP's automatic configuration reloading, but only if the changed configuration file was previously defined as a docket secret (and the secret had been updated in the docker swarm). Unlike the mounted type of configuration, this config type does not require manually syncing config files to all docker nodes (or the creation of user accounts to enable such syncing). For the default hybrid configuration, the following IdP files are defined as secrets:
      - *idp-signing.key*
      - *idp-signing.crt*
      - *idp-encryption.key*
      - *idp-encryption.crt*
      - *keystore.jks (tomcat/SSL)*

        - *sealer.jks*
        - *sealer.kver*
        - *idp.properties*
        - *ldap.properties*
        - *relying-party.xml*
        - *attribute-filter.xml*
        - *attribute-resolver.xml*
        - *metadata-providers.xml*
- Build your configuration
    - Use the TIER IdP Config Builder image in interactive mode to dynamically build your config. (Note: This runs fine on Windows10, Windows Server 2016 may need updates)
        - TIER Shibboleth-IdP Config Builder Docker image
        - You'll be asked the usual questions:
            - FQDN of your IdP
            - Attribute scope value for your IdP (typically your main domain name)
            - LDAP
                - LDAP URL
                - LDAP Base DN
                - LDAP service account DN for the IdP
                - Password on the above account
            - Config type (see descriptions above)
    - Unzip the file produced by the above process

BUILD the image

- Do this work from the same directory where the Dockerfile is located
- **Mounted**
    - It will be necessary to first transfer your built config to your manager VM if you built the config elsewhere
    - *docker build --rm -t my/shibb-idp-tier .*
- **Burned**
    - If your config is located in the default directories generated by the Config Builder image (and resulting zipfile)
        - *docker build --rm -t my/shibb-idp-tier .*
- **Hybrid**

SHIP the image to a repository

- Tag your new image for shipping to a private repository (for info on the free Docker repo, see Docker Private Repo)
    - *docker tag my/shibb-idp-tier <repo.mycampus.edu>:<port>/shib-idp-tier:<your userID>*
- Ship the image to the private repo
    - *docker login <repo.mycampus.edu>:<port>*
        - Username: <your repo username>
        - Password: <your repo password>
    - *docker push <repo.mycampus.edu>:<port>/shib-idp-tier:<your userID>*

Prepare to run a new swarm service

RUN a new service on your swarm

Changing the configuration of an existing service