Shibboleth-IdP Standalone Linux Container Release 180401

Introduction

This packaged TIER Shibboleth-IdP release is a standalone Docker container (Linux-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 Linux-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). To do this (substitute below for 'manager' as desired/appropriate):
    - *hostname manager*
    - *echo manager > /etc/hostname*
    - *systemctl restart docker*
- For all node-to-node communications, it is best to use any private IP addresses which may be primary on the VMs. The individual nodes must be able to communicate with each other using the Docker Swarm protocols (see below).

- It is likely easiest to do builds on the/a manager VM, however, builds can be done on either VM or….

    It is possible to use a desktop/laptop as a build machine!

    Docker for Windows

    Docker for Mac

    Others

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 (a script is provided for this). The script's synchronization process requires a regular user account on all swarm nodes to be used for syncing files. The critical part of this is that a single public/private keypair will need to be used for this user on all nodes and the syncFilesToAllNodes.sh script will need access to the private key from this keypair (the script will only run on the manager node). Steps to create this user account are:
            - On Manager node(s):
                - *useradd -m -d /home/swarmuser -s /bin/bash swarmuser*
                - *sudo -u swarmuser ssh-keygen* (leave password blank)
                - *sudo -u swarmuser cp ~swarmuser/.ssh/id_rsa.pub ~swarmuser/.ssh/authorized_keys*
                - *cat ~swarmuser/.ssh/authorized_keys* (then copy/paste to worker nodes)
            - On Worker node(s):
                - *useradd -m -d /home/swarmuser -s /bin/bash swarmuser*
                - *sudo -u swarmuser mkdir -p ~swarmuser/.ssh*
                - *sudo -u swarmuser chmod 700 ~swarmuser/.ssh*
                - *sudo -u swarmuser vi ~swarmuser/.ssh/authorized_keys* (then paste the key from above)
    - **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 config**
    - Use the configBuilder.sh script (in the files from github)
    - After building your IdP config, this script will create a timestamped archive/zipfile with all of your config bits (including sensitive information).
    - If you choose a 'hybrid' build type, then this script also creates a new child directory named 'ConfigNoSecrets' which contains your configuration, but with all secrets moved to a separate directory (including the more dynamic config files listed above).
    - Information needed to run this script:
        - 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)

BUILD the image

- Do this work from the same directory where your 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 configBuilder.sh script
        - *docker build --rm -t my/shibb-idp-tier .*
- **Hybrid**
    - It will be necessary to first transfer your secrets files to your manager VM, if they are not already there.
    - The first command below establishes an alternate location for the config bits (which then gets used in the subsequent 'docker build' command). The default alternate directory referenced is produced by the configBuilder.sh script and has the secrets extracted from the main config. This means that the resulting image will be burned with an incomplete config (some files will be missing). That config will be completed at run-time when the predefined secrets are added by docker.

- *export ALTCFG=ConfigNoSecrets*
- *docker build --rm -t my/shibb-idp-tier --build-arg TOMCFG=${ALTCFG}/config/tomcat *

  *--build-arg TOMLOG=${ALTCFG}/logs/tomcat *

  *--build-arg TOMCERT=${ALTCFG}/credentials/tomcat *

  *--build-arg TOMWWWROOT=${ALTCFG}/wwwroot *

  *--build-arg SHBCFG=${ALTCFG}/config/shib-idp/conf *

  *--build-arg SHBCREDS=${ALTCFG}/credentials/shib-idp *

  *--build-arg SHBVIEWS=${ALTCFG}/config/shib-idp/views *

  *--build-arg SHBEDWAPP=${ALTCFG}/config/shib-idp/edit-webapp *

  *--build-arg SHBMSGS=${ALTCFG}/config/shib-idp/messages *

  *--build-arg SHBMD=${ALTCFG}/config/shib-idp/metadata *

  --build-arg SHBLOG=${ALTCFG}/logs/shib-idp .

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

- Start a new docker swarm with your 2 VMs
  - If needed, set rules in firewalld on all swarm nodes (and disable selinux, if applicable)
    - *firewall-cmd --permanent --zone=public --add-port=2377/tcp*
    - *firewall-cmd --permanent --zone=public --add-port=7946/tcp*
    - *firewall-cmd --permanent --zone=public --add-port=7946/udp*
    - *firewall-cmd --permanent --zone=public --add-port=4789/udp*
    - *firewall-cmd --reload*
  - On the 'manager' VM:
    - *docker swarm init --advertise-addr <ip-addr-of-mgr-eth0>*
  - On the 'worker' VM (command below is the final output of the previous command):
    - *docker swarm join *

      *--token <big-long-string> *

      <managerIP-from-previous-step>:2377

    *you can get the needed cmd above with the actual token on the manager node by running: *docker swarm join-token worker*

- **For a hybrid config type**
  - First define needed secrets (from the original directory where you downloaded the github repo)
    - *docker secret create idp.properties ./config/shib-idp/conf/idp.properties*
    - *docker secret create ldap.properties ./config/shib-idp/conf/ldap.properties*
    - *docker secret create idp-signing.key ./credentials/shib-idp/idp-signing.key*
    - *docker secret create idp-signing.crt ./credentials/shib-idp/idp-signing.crt*
    - *docker secret create idp-encryption.key ./credentials/shib-idp/idp-encryption.key*
    - *docker secret create idp-encryption.crt ./credentials/shib-idp/idp-encryption.crt*
    - *docker secret create keystore.jks ./credentials/tomcat/keystore.jks*
    - *docker secret create sealer.jks ./credentials/shib-idp/sealer.jks*
    - *docker secret create sealer.kver ./credentials/shib-idp/sealer.kver*
    - *docker secret create relying-party.xml ./config/shib-idp/conf/relying-party.xml*
    - *docker secret create attribute-filter.xml ./config/shib-idp/conf/attribute-filter.xml*
    - *docker secret create attribute-resolver.xml ./config/shib-idp/conf/attribute-resolver.xml*
    - *docker secret create metadata-providers.xml ./config/shib-idp/conf/metadata-providers.xml*
- **For a "mounted" config type**
  - First sync config content to all swarm nodes (from Dockerfile directory)
    - *./syncFilesToAllNodes.sh build swarm-idp-config swarmuser swarmuser*

RUN a new service on your swarm

- **Burned config**
  - *docker login <repo.mycampus.edu>:<port>*
  - *docker service create* \

    --replicas 2 \

    --name shib-idp \

    --with-registry-auth \

    --publish 443:443 \

    --health-interval=5m --health-cmd "curl -k --fail https://127.0.0.1/idp/status || exit 1" \

    <repo.mycampus.edu>:<port>/shib-idp-tier:<your userID>

- **Mounted config**
  - Run from the directory where the Dockerfile is located
  - *docker login <repo.mycampus.edu>:<port>*
  - *docker service create* \

    --replicas 2 \

    --name shib-idp \

    --with-registry-auth \

    -p 443:443 \

    --health-interval=5m --health-cmd "curl -k --fail https://127.0.0.1:443/idp/status || exit 1" \

    --mount type=bind,source=${PWD}/config/tomcat,destination=/usr/local/tomcat/conf \

    --mount type=bind,source=${PWD}/wwwroot,destination=/usr/local/tomcat/webapps/ROOT \

    --mount type=bind,source=${PWD}/logs/tomcat,destination=/usr/local/tomcat/logs \

    --mount type=bind,source=${PWD}/credentials/tomcat,destination=/opt/certs \

    --mount type=bind,source=${PWD}/config/shib-idp/conf,destination=/opt/shibboleth-idp/conf \

    --mount type=bind,source=${PWD}/credentials/shib-idp,destination=/opt/shibboleth-idp/credentials \

    --mount type=bind,source=${PWD}/config/shib-idp/views,destination=/opt/shibboleth-idp/views \

    --mount type=bind,source=${PWD}/config/shib-idp/edit-webapp,destination=/opt/shibboleth-idp/edit-webapp \

    --mount type=bind,source=${PWD}/config/shib-idp/messages,destination=/opt/shibboleth-idp/messages \

    --mount type=bind,source=${PWD}/config/shib-idp/metadata,destination=/opt/shibboleth-idp/metadata \

    --mount type=bind,source=${PWD}/logs/shib-idp,destination=/opt/shibboleth-idp/logs \

    <repo.mycampus.edu>:<port>/shib-idp-tier:<your userID>

- **Hybrid config**
  - *docker login <repo.mycampus.edu>:<port>*
  - *docker service create* \

    --replicas 2 \

    --name shib-idp \

    --with-registry-auth \

    --publish 443:443 \

    --secret source=idp.properties,target=/opt/shibboleth-idp/conf/idp.properties \

    --secret source=ldap.properties,target=/opt/shibboleth-idp/conf/ldap.properties \

--secret source=idp-signing.key,target=/opt/shibboleth-idp/credentials/idp-signing.key \

--secret source=idp-signing.crt,target=/opt/shibboleth-idp/credentials/idp-signing.crt \

--secret source=idp-encryption.key,target=/opt/shibboleth-idp/credentials/idp-encryption.key \

--secret source=idp-encryption.crt,target=/opt/shibboleth-idp/credentials/idp-encryption.crt \

--secret source=keystore.jks,target=/opt/certs/keystore.jks \

--secret source=sealer.jks,target=/opt/shibboleth-idp/credentials/sealer.jks \

--secret source=sealer.kver,target=/opt/shibboleth-idp/credentials/sealer.kver \

--secret source=relying-party.xml,target=/opt/shibboleth-idp/conf/relying-party.xml \

--secret source=attribute-filter.xml,target=/opt/shibboleth-idp/conf/attribute-filter.xml \

--secret source=attribute-resolver.xml,target=/opt/shibboleth-idp/conf/attribute-resolver.xml \

--secret source=metadata-providers.xml,target=/opt/shibboleth-idp/conf/metadata-providers.xml \

--health-interval=4m --health-cmd "curl -k --fail https://127.0.0.1:443/idp/status || exit 1" \

<repo.mycampus.edu>:<port>/shib-idp-tier:<your userID>

- **Check your new service**
  - *docker service ls*
  - *docker service inspect shib-idp*
  - *docker service ps shib-idp*
  - *docker ps* (on a node running the new service)
  - To get a shell in one of your containers:
    - *docker exec -it <containerID-from-docker_ps> bash*

Changing the configuration of an existing service

- **Burned**
  - Update the appropriate config file in the directories setup by the configBuilder.sh script (child directories from the Dockerfile directory). Most Shib-IdP config is in 'config/shib-idp/conf'.
  - Re-run the same build command from step 2.b above, with the change below, to build and push a new image.
    - Modify the build command slightly (apply a tag as in 'newBuild' below) to ensure your new build has a unique name
      - *docker build --rm -t my/shibb-idp-tier:newbuild .*
      - *docker tag my/shibb-idp-tier:newbuild <repo.mycampus.edu>:<port>/shib-idp-tier-newbuild:<your userID>*
      - *docker login <repo.mycampus.edu>:<port>*
      - *docker push <repo.mycampus.edu>:<port>/shib-idp-tier-newbuild:<your userID>*
  - Apply a service update to your swarm to roll in the new image
    - *docker service update --update-parallelism 1 --update-delay 60s --image <repo.mycampus.edu>:<port>/shib-idp-tier-newbuild:<your userID> shib-idp*
- **Mounted**
  - Update the appropriate config file in the directories setup by the configBuilder.sh script (child directories from the Dockerfile directory). Most Shib-IdP config is in 'config/shib-idp/conf'.
  - Re-run the *syncFilesToAllSwarmNodes.sh* script (from step 6.b) to push the updated files to the other swarm nodes (this must be done on the manager node).
  - Apply a service update with the 'force' tag to apply a rolling restart of the IdP instances
    - *docker service update --force --update-parallelism 1 --update-delay 60s shib-idp*
- **Hybrid**
  - Secret files
    - On the manager VM, update the relevant secret file either in the child directories created by the configBuilder.sh script -OR- in the 'ConfigNoSecrets/SECRETS' subdirectory
    - Update the relevant secret within docker
      - First, edit the relevant secret file.
      - Then, create a new secret within docker.
      - Then, update the service, removing the old secret and adding the new one in the same command.
        - *docker secret create idp.properties.new ./config/shib-idp/conf/idp.properties*
        - *docker service update* \
          --secret-rm idp.properties \
          --secret-add src=idp.properties.new,target=/opt/shibboleth-idp/conf/idp.properties \
          shib-idp
      - You may want to then delete the old secret

        - *docker secret rm idp.properties*
- Regular config files (non-secret)
    - Use the instructions above for updating a burned config