Grouper container running on OpenShift

The major obstacle to getting the Grouper docker container to run in the OpenShift environment is that the process user runs as a userid with few permissions, and which isn't known at build time. This means that the Grouper container processes that normally run as either tomcat or root must be runnable under this low-permission context. These issues can be solved by building a derived container that includes a combination of file permission changes plus minor changes to the library scripts in /usr/local/bin. Since version 2.5.40, file permissions in the Grouper container have been adjusted to better support OpenShift out of the box. In addition, this version adds a new environment variable GROUPER_OPENSHIFT, which triggers the setting of other variables needed in OpenShift – GROUPER_CHOWN_DIRS=false, GROUPER_USE_PIPES=false, GROUPER_GSH_CHECK_USER=false, and GROUPER_RUN_PROCESSES_AS_USERS=false.

One trait of the OpenShift container (and Docker containers in general) is that if a user does not have a primary group, it will run with the `root` group. This can help in the early stages of development. By building an image from a Dockerfile including a `USER <random id>` statement, you can start it up locally as a simple Docker image and look for errors, without needing to do frequent OpenShift deployments.

In OpenShift, the Grouper UI or WS only needs to run Tomcat as its only process, on port 8080. Shibboleth can be run as a separate sidecar container, proxying both the login and SSL.

Configuration (since 2.5.40)

For configuration, it works well to set (a) properties that vary between environments in database configuration (Miscellaneous > Configuration in the UI); (b) properties the same in all environments as either property files in the subimage, or in the database; (c) the minimal set of properties needed for startup as environment secrets; and (d) properties specific to OpenShift as normal deployment env variables. Although secrets could be bind mounted in /run/secrets /grouper_* files, the ease of secret setup in OpenShift makes the setup in (c) preferred. Minimally, the morphString key and the database password are the only two properties that need to be set up this way. However, it may simplify maintenance if the database url and username are also included, since they are likely to be maintained at the same time.

opaque secret: grouper_env_prod
GROUPER_MORPHSTRING_ENCRYPT_KEY: ***** GROUPER_DATABASE_URL: ***** GROUPER_DATABASE_USERNAME: ***** GROUPER_DATABASE_PASSWORD: *****

Create an opaque secret with these values in it, and then add it to the deployment configuration as Environment from Secret.

Other environment variables necessary or optional for OpenShift are:

- TZ: America/New_York (set to local time zone, otherwise the container may default to UTC)
- GROUPER_OPENSHIFT: true (if you only want to run your image in OpenShift you can add to the Dockerfile, but if not add, it can be inserted here)
- GROUPER_RUN_TOMCAT_NOT_SUPERVISOR: true (this should be added as part of the GROUPER_OPENSHIFT switches in a future version)
- GC_MAX_METASPACE_SIZE: 256 (helps with startup memory)
- GROUPER_MAX_MEMORY: 1792m (the default is 1500m which actually may be enough; the critical factor in JVM heap sizing is that the container's memory seems to need at least 700M to avoid out of memory crashes at the OS level; in our case the container limit was 2560Mi, and a value of 1792m gave sufficient spare memory to the OS to prevent crashes)
- ~~JVM_OPTS: -XX:+UnlockExperimentalVMOptions -XX:+UseCGroupMemoryLimitForHeap -XX:MaxRAMFraction=1 -Xms64M (the extra options are supposed to help the JVM have awareness of the actual VM memory it can use)~~

The startup command in theory should just be "ui" or "ws". But OpenShift doesn't pick up the entrypoint value of "entrypoint.sh" in the Grouper standard image for some reason, so it needs to be included. Thus, the startup command would be "/usr/local/bin/entrypoint.sh ui". If your OpenShift UI doesn't have a place to configure the start command, it can be added by editing the deployment config yaml:

```
spec:
  template:
    spec:
      containers:
        - args:
            - ui
          command:
            - /usr/local/bin/entrypoint.sh
```

The readiness check can utilize the diagnostic status page. This can be useful not only to detect pod availability, but also helps in WS to force a preload of object data to speed up the first external call. I could not get the "HTTP GET" health check to work. But a curl command also works:

```
spec:
  template:
    spec:
      containers:
      - ...
          readinessProbe:
            exec:
              command:
                - /bin/bash
                - '-c'
                - curl --fail http://$(cat /etc/hostname):8080/grouper/status?diagnosticType=sources
```

Sample Dockerfile:

```
FROM i2incommon/grouper:2.5.42

# 5005 is for JVM debugger
EXPOSE 8080 5005

# Copy in customizations as needed
COPY files/etc/ /etc/
COPY files/grouperWebapp/ /opt/grouper/grouperWebapp/

# TEMPFIX: Grouper issue
#grouperContainer; INFO: (librarySetupFilesTomcat.sh-setupFilesTomcat_loggingSlf4j) rm -f /opt/grouper
/grouperWebapp/WEB-INF/lib/slf4j-jdk*.jar , result: 1
RUN cd /opt/grouper/grouperWebapp/WEB-INF/ && chgrp 0 lib lib/slf4j-jdk14-1.7.21.jar && chmod g+rws ./lib &&
chmod g+rw ./lib/slf4j-jdk14-1.7.21.jar

# TEMPFIX: Grouper UI startup issue
#java.lang.RuntimeException: Cant make dir: /opt/grouper/grouperWebapp/WEB-INF/ddlScripts
RUN cd /opt/grouper/grouperWebapp/WEB-INF && mkdir ddlScripts && chgrp 0 ddlScripts && chmod g+rws ddlScripts

# TEMPFIX: Grouper gsh issue with unwritable WEB-INF/.groovy (defaults to current directory if user has no home
directory?)
RUN chgrp 0 /opt/grouper/grouperWebapp/WEB-INF && chmod g+ws /opt/grouper/grouperWebapp/WEB-INF
```

Note that when building an image from a Dockerfile, it will inspect the code, see that its base image is i2incommon/grouper, and create an image in the registry for the specified version. Once the buildConfig has been created with the base image, subsequent builds will ignore the FROM version in the Dockerfile even if it changes, preferring the one set up in the buildConfig. Thus, when upgrading versions of Grouper, both the FROM base image in the Dockerfile, and the base image defined in the buildConfig need to be modified.

Schema of

Changes to startup scripts (prior to 2.5.40)

The following changes to the Grouper library scripts in /usr/local/bin fix some issues with temporary files, running `rm` as a non-owner, and the non-existent userid. Until the changes are made to the base Grouper container, they should be copied from the Grouper container Git project, patched locally, and then included in the new Docker build.

container_files/usr-local-bin/gsh

```
@@ -7,13 +7,10 @@ runCommand_unsetAll

 export GSH_JVMARGS="$GSH_JVMARGS -DENV=$ENV -DUSERTOKEN=$USERTOKEN"

-username=$(whoami)

-if [ "$GROUPER_GSH_CHECK_USER" = "true"  ] && [ "$GROUPER_GSH_USER" != "$username" ]
+if [ "$GROUPER_GSH_CHECK_USER" = "true" ] && [ "$GROUPER_GSH_USER" != "$(whoami)" ]
   then
     sudo -u tomcat bin/gsh.sh "$@" | tee /tmp/loggrouper
   else
     exec bin/gsh.sh "$@" | tee /tmp/loggrouper
```

container_files/usr-local-bin/library.sh

```
@@ -1,10 +1,13 @@
 #!/bin/bash

+# Run dos2unix; use temp files in case running as unprivileged user
 echo "grouperContainer; INFO: (library.sh) Start loading library.sh"
-dos2unix /usr/local/bin/library*.sh
-echo "grouperContainer; INFO: (library.sh) dos2unix /usr/local/bin/library*.sh , result=$?"
-dos2unix /usr/local/bin/grouper*.sh
-echo "grouperContainer; INFO: (library.sh) dos2unix /usr/local/bin/grouper*.sh , result=$?"
+for f in /usr/local/bin/library*.sh /usr/local/bin/grouper*.sh; do
+  TFILE=$(mktemp) && dos2unix -q -n $f $TFILE && cat $TFILE > $f
+  echo "grouperContainer; INFO: (library.sh) dos2unix $f, result=$?"
+  rm $TFILE
+done
+

 . /usr/local/bin/libraryPrep.sh
 . /usr/local/bin/libraryPrepOnly.sh
```

container_files/usr-local-bin/librarySetupFiles.sh

```
@@ -64,7 +64,11 @@ setupFiles_storeEnvVars() {
   # go through env vars, should start with GROUPER and have an equals sign in there
   env | grep "^GROUPER" | grep "=" | sort >> /opt/grouper/grouperEnv.sh

-  sed -i "s|^GROUPER|export GROUPER|g" /opt/grouper/grouperEnv.sh
+  # use sed with an intermediate temp file, in case running as n unprivileged user
+  TFILE=$(mktemp) \
+  && sed "s|^GROUPER|export GROUPER|g" /opt/grouper/grouperEnv.sh > $TFILE \
+  && cat $TFILE > /opt/grouper/grouperEnv.sh
+  rm $TFILE

   if [ ! -f /home/tomcat/.bashrc ]
     then
```

container_files/usr-local-bin/librarySetupFilesForProcess.sh

```
@@ -14,8 +14,8 @@ setupFilesForProcess_hsqldb() {
 setupFilesForProcess_hsqldbVersions() {

     # tomee hsql must match the grouper one, and the version cannot be 2.3.2 since it is query bugs (unit
tests fail)
-    rm /opt/tomee/lib/hsqldb-*.jar
-    echo "grouperContainer; INFO: (librarySetupFilesForProcess.sh-setupFilesForProcess_hsqldbVersions) rm /opt
/tomee/lib/hsqldb-*.jar , result: $?"
+    rm -f /opt/tomee/lib/hsqldb-*.jar
+    echo "grouperContainer; INFO: (librarySetupFilesForProcess.sh-setupFilesForProcess_hsqldbVersions) rm -f
/opt/tomee/lib/hsqldb-*.jar , result: $?"
     cp /opt/grouper/grouperWebapp/WEB-INF/lib/hsqldb-*.jar /opt/tomee/lib/
     echo "grouperContainer; INFO: (librarySetupFilesForProcess.sh-setupFilesForProcess_hsqldbVersions) cp /opt
/grouper/grouperWebapp/WEB-INF/lib/hsqldb-*.jar /opt/tomee/lib/ , result: $?"
```

container_files/usr-local-bin/librarySetupFilesTomcat.sh

```
@@ -142,18 +142,18 @@ setupFilesTomcat_authn() {

 setupFilesTomcat_loggingSlf4j() {

-  rm /opt/tomee/lib/slf4j-api*.jar
-  echo "grouperContainer; INFO: (librarySetupFilesTomcat.sh-setupFilesTomcat_loggingSlf4j) rm /opt/tomee/lib
/slf4j-api*.jar , result: $?"
-  rm /opt/tomee/lib/slf4j-jdk*.jar
-  echo "grouperContainer; INFO: (librarySetupFilesTomcat.sh-setupFilesTomcat_loggingSlf4j) rm /opt/tomee/lib
/slf4j-jdk*.jar , result: $?"
+  rm -f /opt/tomee/lib/slf4j-api*.jar
+  echo "grouperContainer; INFO: (librarySetupFilesTomcat.sh-setupFilesTomcat_loggingSlf4j) rm -f /opt/tomee/lib
/slf4j-api*.jar , result: $?"
+  rm -f /opt/tomee/lib/slf4j-jdk*.jar
+  echo "grouperContainer; INFO: (librarySetupFilesTomcat.sh-setupFilesTomcat_loggingSlf4j) rm -f /opt/tomee/lib
/slf4j-jdk*.jar , result: $?"
   cp /opt/grouper/grouperWebapp/WEB-INF/lib/slf4j-api-*.jar /opt/tomee/lib
   echo "grouperContainer; INFO: (librarySetupFilesTomcat.sh-setupFilesTomcat_loggingSlf4j) cp /opt/grouper
/grouperWebapp/WEB-INF/lib/slf4j-api-*.jar /opt/tomee/lib , result: $?"
   # tomee uses the jdk one
   cp /opt/grouper/grouperWebapp/WEB-INF/lib/slf4j-jdk*.jar /opt/tomee/lib
   echo "grouperContainer; INFO: (librarySetupFilesTomcat.sh-setupFilesTomcat_loggingSlf4j) cp /opt/grouper
/grouperWebapp/WEB-INF/lib/slf4j-jdk*.jar /opt/tomee/lib , result: $?"
   # grouper uses the log4j one
-  rm /opt/grouper/grouperWebapp/WEB-INF/lib/slf4j-jdk*.jar
-  echo "grouperContainer; INFO: (librarySetupFilesTomcat.sh-setupFilesTomcat_loggingSlf4j) rm /opt/grouper
/grouperWebapp/WEB-INF/lib/slf4j-jdk*.jar , result: $?"
+  rm -f /opt/grouper/grouperWebapp/WEB-INF/lib/slf4j-jdk*.jar
+  echo "grouperContainer; INFO: (librarySetupFilesTomcat.sh-setupFilesTomcat_loggingSlf4j) rm -f /opt/grouper
/grouperWebapp/WEB-INF/lib/slf4j-jdk*.jar , result: $?"

 }
```

Dockerfile (prior to 2.5.40)

The Docker subimage includes some extra environment variables, imports the modified library scripts, and sets the extra file permissions needed to do file preparation and run the TomEE process. Note that this also includes a USER directive to set the user to an invalid userid. This is for quick testing in the local environment, and should be removed when building the image in OpenShift.

```
FROM i2incommon/grouper:2.5.39

ENV GROUPER_CHOWN_DIRS=false

ENV GROUPER_SHIB_LOG_USE_PIPE=false

ENV GROUPER_GSH_CHECK_USER=false

EXPOSE 8080

COPY files/container_files/usr-local-bin/ /usr/local/bin/

RUN touch /opt/grouper/grouperEnv.sh && chgrp root /opt/grouper/grouperEnv.sh  \
    && chmod g+rwx /opt/grouper/grouperEnv.sh /etc/httpd/conf/httpd.conf && chgrp root /opt/grouper/grouperEnv.
sh /etc/httpd/conf/httpd.conf \
    && chmod g+rx /home/tomcat/ && chgrp root /home/tomcat/ \
    && chmod g+rw /home/tomcat/.bashrc /opt/tomee/conf/server.xml && chgrp root /home/tomcat/.bashrc /opt/tomee
/conf/server.xml \
    && for d in /usr/local/bin \
                  /etc/httpd/conf/ \
                  /etc/httpd/conf.d/ \
                  /opt/hsqldb/ \
                  /opt/tomee/lib/ \
                  /opt/tomee/conf/ \
                  /opt/grouper/grouperWebapp/WEB-INF/lib/ \
                  /opt/tomee/conf/Catalina/localhost/ \
                  /opt/tier-support/ \
                  /opt/grouper/grouperWebapp/WEB-INF/ \
                  /opt/grouper/grouperWebapp/WEB-INF/classes/; do chmod g+rwx $d; chgrp root $d; done \
    && for d in /opt/tomee/webapps/ /opt/tomee/temp/ /opt/tomee/work/Catalina/localhost/; do mkdir -p $d &&
chgrp root $d && chmod g+rws $d; done

# To simulate what would happen when the OpenShift container starts, uncomment this, build, then run locally
#USER 123456
```

Running a container

Besides the environment variables baked into the image, there are also some additional variables that are better left to the Docker run command, since they differ depending on which Grouper component is run. The `ui`, `ws`, etc. commands that are normally used to start a container can't be used, because they add additional environment variables that force Apache and Shibboleth to be run as services, which we don't need in OpenShift.

UI (leave out `ui` in container run command):

- GROUPER_UI=true
- GROUPER_RUN_TOMEE=true
- GROUPER_RUN_PROCESSES_AS_USERS=false

WS (leave out `ws` for container):

- GROUPER_WS=true
- GROUPER_RUN_TOMEE=true
- GROUPER_RUN_PROCESSES_AS_USERS=false
- GROUPER_TOMCAT_CONTEXT=grouper-ws

Daemon (use `daemon` to invoke)

- GROUPER_DAEMON=true
- GROUPER_RUN_TOMEE=true
- GROUPER_RUN_PROCESSES_AS_USERS=false

gsh (use `gsh` to invoke)

- GROUPER_RUN_PROCESSES_AS_USERS=false

UI + WS: Because this is running only TomEE without Apache, there is only one context ("grouper", or defined by GROUPER_TOMCAT_CONTEXT) running. Thus, both the UI and WS will be accessed under the single context. If you need the UI running under /groupre and WS running under grouper-ws, further modifications (as a future exercise) would be needed.