Deploying Grouper Standalone Container - AWS and Elastic Container Service (ECS)

This is an example of a script that will build a Grouper Docker container based on the TIER container. If the build is successful, it will push the new image to a private AWS Elastic Container Registry (ECR). In this case, the build host runs with an IAM role that with an IAM policy attached to it that allows it access to push to this repo. This script could be modified to run as part of a Jenkins pipeline.

build.sh

```
#!/bin/bash

VERSION="grouper-v0.1.5"

# IMPORTANT !!!!!!!!!!!!!!!!!!!!!!!!
# uncomment the line below for the environment you're building for and the proper tag will be applied
# defaults to dev

ENVIRONMENT="comanage-dev-host-configs"
#ENVIRONMENT="comanage-staging-host-configs"
#ENVIRONMENT="comanage-prod-host-configs"

CONFIG="s3url=s3://$ENVIRONMENT"

SHORTENV=$(echo $ENVIRONMENT | cut -f 2 -d '-')
DOCKERENV="SHORTENV=$SHORTENV"


docker build --no-cache --pull --build-arg $CONFIG --build-arg $DOCKERENV -t grouper.at.internet2.edu:${VERSION}
-$SHORTENV .
if [ "$?" -ne "0" ]
then
  echo "# Build failed"
  exit
fi

docker tag grouper.at.internet2.edu:${VERSION}-$SHORTENV 992649280435.dkr.ecr.us-east-2.amazonaws.com/grouper.
at.internet2.edu:${VERSION}-$SHORTENV
if [ "$?" -ne "0" ]
then
  echo "# Tag failed"
  exit
fi

`aws ecr get-login --no-include-email --region us-east-2`
if [ "$?" -eq "0" ]
then
  docker push 992649280435.dkr.ecr.us-east-2.amazonaws.com/grouper.at.internet2.edu:${VERSION}-$SHORTENV
else
  echo "# Login to AWS Docker repo failed. If running on an EC2 instance with an IAM role, ensure it has"
  echo "# the right policy attached to it. "
  echo "# To see the IAM Role, issue the following command:"
  echo "curl http://169.254.169.254/latest/meta-data/iam/info"
  echo "# If not using an IAM instance role, make sure the ~/.aws/credential is file configured correctly."
fi
```

The following is a Dockerfile that is used by the above build script that layers in the necessary local configurations for running Grouper needed by the Internet2 Collaboration Platform that houses spaces.internet2.edu. The files that contain secrets are located in an encrypted S3 bucket and the build host runs with an IAM role that allows it to access this bucket. This example is for a Grouper UI container. The AWS RDS (Relational Database Service) is used by this service.

Dockerfile

```
FROM tier/grouper:2.4.0-a27-u11-w2-p2-20190211

ARG s3url
ARG SHORTENV

ENV ENV=$SHORTENV
ENV CONFIG_BUCKET=$s3url/grouper.at.internet2.edu

ENV CATALINA_OPTS="-XX:+UseG1GC -Xmx3000m"

RUN yum update -y && \
    yum -y install epel-release && \
    pip install awscli && pip install --upgrade pip

RUN aws s3 cp $CONFIG_BUCKET/shib/sp-cert.pem /etc/shibboleth && \
    aws s3 cp $CONFIG_BUCKET/shib/sp-key.pem /etc/shibboleth  && \
    aws s3 cp $CONFIG_BUCKET/shib/attribute-map.xml /etc/shibboleth  && \
    aws s3 cp s3://comanage-dev-host-configs/general_metadata/login.at.internet2.edu-metadata.xml /etc
/shibboleth && \
    cp /dev/null /etc/httpd/conf.d/ssl-enabled.conf && \
    mkdir /opt/grouper/grouper.apiBinary/ddlScripts && \
    chown tomcat:tomcat /opt/grouper/grouper.apiBinary/ddlScripts

COPY container_files/shibboleth/shibboleth2.xml /etc/shibboleth/
COPY container_files/httpd/grouper.conf /etc/httpd/conf.d
# uncomment next line to turn on shibd debug
COPY container_files/shibboleth/shibd.logger /etc/shibboleth/shibd.logger
ENV LD_LIBRARY_PATH=/opt/shibboleth/lib64

ADD https://s3.us-east-2.amazonaws.com/comanage-metadata-public/certs/icmp_signing.crt /etc/pki/tls/certs/

COPY container_files/tomcat/server.xml /opt/tomcat/conf/

# Configuration files for Grouper

RUN aws s3 cp $CONFIG_BUCKET/grouper/grouper.hibernate.properties /opt/grouper/conf/ && \
    aws s3 cp $CONFIG_BUCKET/grouper/ldap.properties /opt/grouper/conf/ && \
    aws s3 cp $CONFIG_BUCKET/grouper/grouper.client.properties /opt/grouper/conf/ && \
    aws s3 cp $CONFIG_BUCKET/grouper/grouper.properties /opt/grouper/conf/ && \
    aws s3 cp $CONFIG_BUCKET/grouper/subject.properties /opt/grouper/conf/ && \
    aws s3 cp $CONFIG_BUCKET/grouper/log4j.properties /opt/grouper/conf/ && \
    aws s3 cp $CONFIG_BUCKET/grouper/grouper-loader.properties /opt/grouper/conf/


# update local text configurations
RUN aws s3 cp $CONFIG_BUCKET/grouper/grouper.text.en.us.properties /opt/grouper/grouper.ui/WEB-INF/classes
/grouperText/
RUN aws s3 cp $CONFIG_BUCKET/grouper/grouper-ui.properties /opt/grouper/grouper.ui/WEB-INF/classes/grouper-ui.
properties
```

The following is an example Elastic Container Service (ECS) task definition for running this container. Configuration such as log groups for Cloudwatch, the port to direct traffic from the Elastic Load Balancer (ELB), the memory requirements for the container, as well as the ECR location to pull the container from. Note the "command" value, as this instructs the container as what component to act as. In this case, it is going to run as a Grouper UI. Alternately, the command could be "loader" (to run as a Grouper Loader), or "ws" (to run as a Grouper Web Services container). This task definition is part of an overall Cloudformation template that builds out the environment that houses other TIER containers used by the Internet2 Collaboration Platform.

Grouper Task Definition for ECS

```
{
  "ipcMode": null,
  "executionRoleArn": null,
  "containerDefinitions": [
    {
```

```
    "dnsSearchDomains": null,
    "logConfiguration": {
      "logDriver": "awslogs",
      "options": {
        "awslogs-group": "comanage-dev",
        "awslogs-region": "us-east-2",
        "awslogs-stream-prefix": "grouper"
      }
    },
    "entryPoint": null,
    "portMappings": [
      {
        "hostPort": 0,
        "protocol": "tcp",
        "containerPort": 80
      }
    ],
    "command": [

      "ui"
    ],
    "linuxParameters": null,
    "cpu": 0,
    "environment": [],
    "resourceRequirements": null,
    "ulimits": null,
    "dnsServers": null,
    "mountPoints": [],
    "workingDirectory": null,
    "secrets": null,
    "dockerSecurityOptions": null,
    "memory": null,
    "memoryReservation": 2048,
    "volumesFrom": [],
    "image": "992649280435.dkr.ecr.us-east-2.amazonaws.com/grouper.at.internet2.edu:grouper-v0.1.5-dev",
    "disableNetworking": null,
    "interactive": null,
    "healthCheck": null,
    "essential": true,
    "links": null,
    "hostname": null,
    "extraHosts": null,
    "pseudoTerminal": null,
    "user": null,
    "readonlyRootFilesystem": null,
    "dockerLabels": null,
    "systemControls": null,
    "privileged": null,
    "name": "grouper-at"
  }
],
"placementConstraints": [],
"memory": null,
"taskRoleArn": null,
"compatibilities": [
  "EC2"
],
"taskDefinitionArn": "arn:aws:ecs:us-east-2:992649280435:task-definition/grouper-at:15",
"family": "grouper-at",
"requiresAttributes": [
  {
    "targetId": null,
    "targetType": null,
    "value": null,
    "name": "com.amazonaws.ecs.capability.ecr-auth"
  },
  {
    "targetId": null,
    "targetType": null,
    "value": null,
    "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
```

```
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.docker-remote-api.1.21"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
    }
  ],
  "pidMode": null,
  "requiresCompatibilities": [],
  "networkMode": null,
  "cpu": null,
  "revision": 15,
  "status": "ACTIVE",
  "volumes": []
}
```

The following is an apache configuration for running the Grouper container in ECS. Since containers have to sit behind load balancers, the remote IP address is that of the load balancer. However, the load balancer inserts a header (X-Forwarded-For) that contains the IP address of the actual client, so that is logged. Rather than having the httpd listen on 2 ports, if the request comes in as http (as seen in the X-Forwarded-Proto), it will redirect to https.

grouper.conf

```
ServerName https://grouper.at.internet2.edu:443
UseCanonicalName On

RemoteIPHeader X-Forwarded-For

RewriteEngine On
RewriteCond %{X-Forwarded-Proto}i http
RewriteRule ^ https://%{HTTP_HOST}:443%{REQUEST_URI} [R=302,L,QSA]

RewriteEngine on
RewriteRule   "^/$"  "/grouper/"  [R]

# log the X-Forwarded-For (the real client IP) header if present
LogFormat "httpd;access_log;%{ENV}e;%{USERTOKEN}e;%{X-Forwarded-For}i %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%
{User-Agent}i\"" proxy
SetEnvIf X-Forwarded-For "^.*\..*\..*\..*" forwarded
CustomLog "/tmp/logpipe" combined env=!forwarded
CustomLog "/tmp/logpipe" proxy env=forwarded

ErrorLogFormat "httpd;error_log;%{ENV}e;%{USERTOKEN}e;[%{u}t] [%-m:%l] [pid %P:tid %T] %7F: %E: [client\ %a] %
M% ,\ referer\ %{Referer}i"
```

Note: Since TLS termination typically happens at the ELB, inform Tomcat that it should create URLs of the form https by specifying

secure="true" in the Connector declaration.