Deployment using Docker Container

This document outlines an approach to fully automate configuration of a new EC2 Instance as an example of how to put an entire WebObjects/Wonder configuration into a container and mount the necessary files and folders to completely and independently run your WO instance on any platform.

This example is based on the TruAnon docker, found here https://hub.docker.com/r/truanon/server

The process starts with a stub script where EC2 instances can update and install the minimal requirements for bootstrapping your container installation. There are also two scripts linked from your own app deployment, one configures the rest of your instance and the other simply runs the application. You must login to your newly Stubbed out EC2 Instance where your deployment has already been checked out. Your deployment WebserverResources can contain a simple sbin where you can keep scripts like these examples.

```
.../Contents/Resources/sbin/configure.sh
```

and

```
.../Contents/Resources/sbin/run.sh
```

Example Dockerfile Template

```
FROM alextu/wofull:latest
MAINTAINER TruAnon
RUN apt-get update
RUN apt-get install python3 -y
RUN apt-get install libdigest-md5-file-perl -y
RUN apt-get install python3-pip -y
RUN apt-get install ImageMagick -y
RUN pip3 install tweepy && \
    apt-get autoremove && \
    mkdir /awsbin && \
    cd /awsbin && \
    curl https://raw.githubusercontent.com/timkay/aws/master/aws -o aws && \
    chmod +x aws && \
    ./aws --link
RUN echo "Include /mywoapps/TruAnon.woa/Contents/WebServerResources/apprules.conf" >> /usr/local/apache2/conf
/httpd.conf
```

NOTE: You can see in the above example Docker template file that the echo statement is used to push a request to load your own "apprules.conf" apache configuration file. This is useful as a way to separately copy in your own rules along with your own deployment and your container will always include those rules when apache runs. The configurations you need can likely all be installed using RUN commands in much the same way. Be aware if you update the apprules.conf file after your container is built, you'd need to exec into your container and gracefully restart apache from inside.

Explanation of Docker Mount Points

By linking to your deployment folder, you can mount scripts and configurations all so they are always pointing to whatever you have in deployment and the container will use those files. This includes handy items like your httpd.conf and JavaMonitor settings which you might update over time or configure specifically for your deployment.

You can add your own dependencies in much the same way.

Mounts	On	In Order To
$HOME/.awssecret	/root/.awssecret	Push your AWS keys into the container
$HOME/wodocker/git/XXX_Deploy	/mywoapps	Mount your app for launching
$HOME/wodocker/git/XXX_Deploy/Split /WebObjects	/usr/local/apache2/htdocs/WebObjects	Try to make web server files visible to the container
$HOME/wodocker/logs	/var/log/WebObjects	Your app output Logs
$HOME/wodocker/conf	/opt/Local/Library/WebObjects /Configuration	Separate your JAVAMonitor Settings

Example Stub Initializes a new EC2 Instance

This stub can be pasted into a text area on the EC2 Launch Instance Wizard or you can link to a copy on your local computer. I find it easier to simply paste this into the text before launching. You can see this updates the server instance, installs GIT and Docker. It then starts up Docker and stuff some credentials and keys into the .netrc for the ec2-user. A short script to clone the GIT repo and make some links is generated and then run on the fly as the ec2-user and this finalizes the one-time setup/configuraiton.

This stub writes a small init.sh file into the ec2-user's home folder there which you must login to your instance after launch and once you see that script written, you must run that by hand to complete configuration and see your app come up live and running.

```
#!/bin/sh
# this stub will run as root
# you must set intended hostname
#

yum update -y
yum install git -y
yum install docker -y
service docker start
usermod -a -G docker ec2-user

mkdir -p ~ec2-user/wodocker/git
chown -R ec2-user:ec2-user ~ec2-user/wodocker

# stuff git credentials .netrc
touch ~ec2-user/.netrc
cat >~ec2-user/.netrc <<EOF
machine github.com
       login XXX
       password XXXXX
EOF

chown ec2-user:ec2-user ~ec2-user/.netrc
chmod go-rwx ~ec2-user/.netrc

# stuff awssecret
touch ~ec2-user/.awssecret
cat >~ec2-user/.awssecret <<EOF
XXX
XXXXX
EOF

chown ec2-user:ec2-user ~ec2-user/.awssecret
chmod go-rwx ~ec2-user/.awssecret

chown -R ec2-user:ec2-user ~ec2-user/wodocker/

# git clone deployment
# link scripts to ~ec2-user
# generate init.sh to later be run as ec2-user
#
touch ~ec2-user/init.sh
cat >~ec2-user/init.sh <<EOF
#!/bin/sh
cd ~/wodocker/git ; git clone https://github.com/XXXXX.git
ln -s ~/wodocker/git/XXXXX/XXXXX/Contents/Resources/sbin/run.sh ~/run.sh
ln -s ~/wodocker/git/XXXXX/XXXXX/Contents/Resources/sbin/configure.sh ~/configure.sh
sh ./run.sh XXXXX.com ; sh ./configure.sh
EOF

chown ec2-user:ec2-user ~ec2-user/init.sh
chmod go-rwx ~ec2-user/init.sh

# done
```

Example configure.sh sets up JavaMonitor and preps the installation once

This script sets up JavaMonitor however you want, add instances and setup the host names and apps and password all using CURL.

```
#!/bin/sh
# new EC2 Instance configure script
# /home/ec2-user/wodocker/git/XXX_Deploy/XXX.woa/Contents/Resources/sbin/configure.sh
#

echo "Add 127.0.0.1 to JavaMonitor"
curl -X POST -d "{id: '127.0.0.1',type: 'MHost', osType: 'UNIX',address: '127.0.0.1', name: '127.0.0.1'}" http://1
27.0.0.1:56789/cgi-bin/WebObjects/JavaMonitor.woa/ra/mHosts.json

echo "Configure WOAdaptorURL"
curl -X PUT -d “{woAdaptor:'http://XXX.com/apps/WebObjects'}" http://127.0.0.1:56789/cgi-bin/WebObjects
/JavaMonitor.woa/ra/mSiteConfig.json

echo "Add XXX”
curl -X POST -d "{id: ‘XXX',type: 'MApplication', name: ‘XXX', additionalArgs: '-Xms64m -Xmx300m -
DWOAllowsConcurrentRequestHandling=true -WOStatisticsPassword wonderpass',unixOutputPath: '/var/log/WebObjects',
unixPath: ‘/mywoapps/XXX.woa/XXX’}" http://127.0.0.1:56789/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications.
json
curl -X GET http://127.0.0.1:56789/cgi-bin/WebObjects/JavaMonitor.woa/ra/mApplications/XXX/addInstance

echo "Configure JavaMonitor Password"
curl -X PUT -d "{password:'XXXXXXX'}" http://127.0.0.1:56789/cgi-bin/WebObjects/JavaMonitor.woa/ra/mSiteConfig.
json

echo "Initialization Complete"
```

Example run.sh simply starts up the app because the mount points and launch args are a bit cumbersome

This run.sh script is meant to simply contain the launch stuff because the container and mount points are complicated. There are ways to reduce that clutter, but this is where I left off.

```
#!/bin/sh
# new EC2 Instance preparation script
# sh ./run.sh hostname.com
#

display_usage() {
  echo "Include a host name"
  echo -e "\nUsage:\n sh ./run.sh [machine.domain.com]\n\n"
}

if [ $# == 0 ]; then
  display_usage
  exit 1
fi

echo "Spark XXX at host name " $1
# visibly mark this EC2 with the machine/host name
touch ~/$1

# start docker by passing in that host name

docker run -d -ti --rm -p 80:80 -p 56789:56789 --hostname $1 -v $HOME/.awssecret:/root/.awssecret -v $HOME
/wodocker/git/XXXX_Deploy:/mywoapps -v $HOME/wodocker/conf:/opt/Local/Library/WebObjects/Configuration -v $HOME
/wodocker/git/XXXX_Deploy/Split/WebObjects:/usr/local/apache2/htdocs/WebObjects -v $HOME/wodocker/logs:/var/log
/WebObjects truanon/server
```

The End Result

The result of all this setup and configuration is you can spark up a new EC2 instance using the stub. This starts and kicks off a configuration that sets up the entire server and application from end-to-end. You do need to log into your EC2 instance after the stub completes and run the init.sh script yourself, in order to make the new instance fully configured and operational. That init checks out deployment files, links and runs your run.sh and configure.sh from your deployment.

Any configuration can be controlled in a similar way and your deployment is reliably portable.