AMS 2016: UniCloud, Docker at Unidata

LDM, TDS, and RAMADDA on Microsoft Azure VM

Julien Chastang

<2015-12-08 Tue>

Contents

1 Quick Start

In order to understand what you are doing, it is best read the complete contents of this document and follow the instructions herein. And if there are problems you will be able to reason about the errors. However, if you are champing at the bit, you can run the following commands to quickly get you going.

- `git clone https://github.com/Unidata/Unidata-Dockerfiles`
- Download and install[1] `docker-machine`
- Run the `Unidata-Dockerfiles/ams2016/unicloud-1.sh [--azure-host] [--azure-host] [--azure-subscription-id] [--azure-subscription-cert] [--azure-size]` script (this will take few minutes)

At this point you may or may not see an error message pertaining to regenerating the certificates. In this case you will have to:

[1] `https://docs.docker.com/machine/install-machine/`

```
docker-machine regenerate-certs <azure-host>
eval "$(docker-machine env <azure-host>)"
```

Now you are ready to do additional set up on the new Docker host:

```
docker-machine ssh <azure-host> "bash -s" < \
    Unidata-Dockerfiles/ams2016/unicloud-2.sh
```

At this point you are almost done. `ssh` into new Docker host: `docker-machine ssh <azure-host>`.

See the section below about editing the `ldmfile.sh` to correctly handle logging.

Run `~/git/Unidata-Dockerfiles/ams2016/unicloud-3.sh`

2 Preamble

The following instructions describe how to configure a Microsoft Azure VM[2] serving data with the LDM[3], TDS[4], and RAMADDA[5]. This document assumes you have access to Azure resources though these instructions should be fairly similar on other cloud providers (e.g., Amazon). They also assume familiarity with Unix, Docker, and Unidata technology in general. You must have `sudo` priviliges on your Azure host which will hopfully be provided to you be default. You will have to be comfortable entering commands at the Unix command line. We will be using Docker images defined at the Unidata-Dockerfiles repository[6] in addition to a configuration specifically planned for AMS 2016 demonstrations project AMS 2016 demonstrations project[7].

3 Preliminary Setup on Azure

The instructions assume we will create an Azure VM called `unidata-server.cloudapp.net` abbreviated to `unidata-server`. Tailor the VM name for your purposes when following this document. This VM will be our **Docker Host** from where we will run Docker containers for the LDM, TDS, and RAMADDA.

[2] `https://azure.microsoft.com`
[3] `http://www.unidata.ucar.edu/software/ldm/`
[4] `http://www.unidata.ucar.edu/software/thredds/current/tds/`
[5] `http://sourceforge.net/projects/ramadda/`
[6] `https://github.com/Unidata/Unidata-Dockerfiles`
[7] `https://github.com/Unidata/Unidata-Dockerfiles/tree/master/ams2016`

3.1 `docker-machine`

Install[8] `docker-machine` on your local computer. `docker-machine` is a command line tool that gives users the ability to create Docker VMs on your local computer or on a cloud provider such as Azure.

3.2 Create a VM on Azure.

The following `docker-machine` command will create a Docker VM on Azure in which you will run various Docker containers. It will take a few minutes to run (between 5 and 10 minutes). You will have to supply `azure-subscription-id` and `azure-subscription-cert` path. See these Azure `docker-machine` instructions[9], if you have questions about this process. Also the the size of the VM is currently set to `ExtraLarge`. See here[10] to learn more about sizes for virtual machines.

```
# Create Azure VM via docker-machine
docker-machine -D create -d azure \
               --azure-subscription-id=$AZURE_ID \
               --azure-subscription-cert=$AZURE_CERT \
               --azure-size=$AZURE_SIZE $AZURE_HOST
```

3.3 Configure Unix Shell to Interact with New Azure VM.

Execute the following command `eval` in your local computer shell environment to ensure that `docker` commands will be run with the newly created Docker host.

3.4 Restart Azure VM

Mysteriously, when you `ssh` (see next section) into the fresh VM, you are immediately told to restart it so let's preempt that message by doing that now.

3.5 `ssh` into VM with `docker-machine`

```
docker-machine ssh unidata-server
```

[8] `https://docs.docker.com/machine/install-machine/`

[9] `https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-docker-machine/`

[10] `https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-size-specs/`

3.6 Install Package(s) with `apt-get`

At the very least, we will need `unzip` on the Azure Docker host.

```
set -x

# update and install package(s)
sudo apt-get -qq update
sudo apt-get -qq install unzip
```

3.7 Add `ubuntu` User to `docker` Group and Restart Docker

```
# Add ubuntu to docker group
sudo usermod -G docker ubuntu

# Restart docker service
sudo service docker restart
```

3.8 Install `docker-compose` on VM

`docker-compose` is a tool for defining and running multi-container Docker applications. In our case, we will be running the LDM, TDS, TDM (THREDDS Data Manager) and RAMADDA so `docker-compose` is perfect for this scenario. Install `docker-compose` on the Azure Docker host.

"You may have to update version (currently at `1.5.2`).

```
# Get docker-compose
curl -L \
https://github.com/docker/compose/releases/download/1.5.2/docker-compose-‘uname -s‘-‘u
      > docker-compose
sudo mv docker-compose /usr/local/bin/
sudo chmod +x /usr/local/bin/docker-compose
```

4 LDM and TDS Configuration

4.1 Background

At this point, we have done the preliminary legwork to tackle the next step in this process. We will now want to clone two repositories that will allow us to configure and start running the the LDM, TDS, and RAMADDA. In particular, we will be cloning:

- `github.com/Unidata/Unidata-Dockerfiles`[11]
- `github.com/Unidata/TdsConfig`[12]

4.1.1 `Unidata-Dockerfiles`

The `Unidata-Dockerfiles` repository contains a number of Dockerfiles that pertain to various Unidata technologies (e.g., the LDM) and also projects (e.g., ams2016). As a matter of background information, a `Dockerfile` is a text file that contains commands to build a Docker image containing, for example, a working LDM. These Docker images can subsequently be run by `docker` command line tools, or `docker-compose` commands that rely on a `docker-compose.yml` file. A `docker-compose.yml` file is a text file that captures exactly how one or more containers run including directory mappings (from outside to within the container), port mappings (from outside to within the container), and other information.

4.1.2 `TDSConfig`

The `TDSConfig` repository is a project that captures THREDDS and LDM configuration files (e.g., `catalog.xml`, `pqact.conf`) for the TDS at `http://thredds.ucar.edu`. Specifically, these TDS and LDM configurations were meant to work in harmony with one another. We can re-use this configuration with some minor adjustments for running the TDS on the Azure cloud.

4.2 `git clone` Repositories

With that background information out of the way, let's clone those repositories by creating `~/git` directory where our repositories will live and issuing some `git` commands.

```
# Get the git repositories we will want to work with
mkdir -p /home/ubuntu/git
git clone https://github.com/Unidata/Unidata-Dockerfiles \
    /home/ubuntu/git/Unidata-Dockerfiles
git clone https://github.com/Unidata/TdsConfig /home/ubuntu/git/TdsConfig
```

[11] `https://github.com/Unidata/Unidata-Dockerfiles`
[12] `https://github.com/Unidata/TdsConfig`

4.3 Configuring the LDM

4.3.1 LDM Directories on Docker Host

For anyone who has worked with the LDM, you may be familiar with the following directories:

- `etc/`
- `var/data`
- `var/logs`
- `var/queue`

The LDM `etc` directory is where you will find configuration files related to the LDM including `ldmd.conf`, `pqact` files, `registry.xml`, and `scour.conf`. We will need the ability to easily observe and manipulate the files from **outside** the running LDM container. To that end, we need to find a home for `etc` on the Docker host. The same is true for the `var/data` and `var/logs` directories. Later, we will use Docker commands that have been written on your behalf to mount these directories from **outside** to **within** the container. The `var/queues` directory will remain inside the container.

```
# Create LDM directories
mkdir -p ~/var/logs
mkdir -p ~/etc/TDS
```

`var/data` is a bit different in that it needs to be mounted on data volume on the Docker host. We will be handling that step further on.

4.3.2 LDM Configuration Files

There is a generic set of LDM configuration files located here `~/git/Unidata-Dockerfiles/ldm/etc/`. However, we will just grab `netcheck.conf` which will remain unmodified.

```
# Copy various files for the LDM.
cp ~/git/Unidata-Dockerfiles/ldm/etc/netcheck.conf ~/etc
```

The rest of the LDM configuration files will come from our `ams2016` project directory.

Also, remember that these files will be used **inside** the LDM container that we will set up shortly. We will now be working with these files:

- `ldmd.conf`
- `registry.xml`
- `scour.conf`

1. `ldmd.conf`

   ```
   cp ~/git/Unidata-Dockerfiles/ams2016/ldmd.conf ~/etc/
   ```

   This `ldmd.conf` has been setup for the AMS 2016 demonstration serving the following data feeds:

   - 13km Rapid Refresh[13]
   - NESDIS GOES Satellite Data[14]
   - Unidata NEXRAD Composites

   For your information, and for future reference, there is a `~/git/TdConfig/idd/pqacts/README.tx` file that may be helpful in writing a suitable `ldmd.conf` file.

2. `registry.xml`

   ```
   cp ~/git/Unidata-Dockerfiles/ams2016/registry.xml ~/etc/
   ```

   This file has been set up for the AMS 2016 demonstration. Otherwise you would have to edit the `registry.xml` to ensure the `hostname` element is correct. For your own cloud VMs, work with `support-idd@unidata.ucar.edu` to devise a correct `hostname` element so that LDM statistics get properly reported. Here is an example `hostname` element `unidata-server.azure.unidata.ucar.edu`.

3. `scour.conf`

   You need to scour data or else your disk will full up. The crontab entry that runs scour is in the LDM Docker container[15]. Scouring is invoked once per day.

   ```
   cp ~/git/Unidata-Dockerfiles/ams2016/scour.conf ~/etc/
   ```

[13] `http://rapidrefresh.noaa.gov/`
[14] `http://www.nesdis.noaa.gov/imagery_data.html`
[15] `https://github.com/Unidata/Unidata-Dockerfiles/blob/master/ldm/crontab`

4. `pqact.conf` and TDS configuration

   In the `ldmd.conf` file we copied just a moment ago there is a reference to a `pqact` file; `etc/TDS/pqact.forecastModels`. We need to ensure that file exists by doing the following instructions. Specifically, explode `~/git/TdsConfig/idd/config.zip` into `~/tdsconfig` and `cp -r` the `pqacts` directory into `~/etc/TDS`. **Note** do NOT use soft links. Docker does not like them.

   ```
   # Set up LDM and TDS configuration
   mkdir -p ~/tdsconfig/
   cp ~/git/TdsConfig/idd/config.zip ~/tdsconfig/
   unzip ~/tdsconfig/config.zip -d ~/tdsconfig/
   cp -r ~/tdsconfig/pqacts/* ~/etc/TDS
   ```

5. Edit `ldmfile.sh`

   Open the `etc/TDS/util/ldmfile.sh` file the editor of your choice. As the top of this file indicates, you must edit the `logfile` to suit your needs. Change the

   ```
   logfile=logs/ldm-mcidas.log
   ```

   line to

   ```
   logfile=var/logs/ldm-mcidas.log
   ```

   This will ensure `ldmfile.sh` can properly invoked from the `pqact` files.

4.3.3 Upstream Data Feed from Unidata or Elsewhere

The LDM operates on a push data model. You will have to find someone who will agree to push you the data. If you are part of the American academic community please send a support email to `support-idd@unidata.ucar.edu` to discuss your LDM data requirements.

4.4 Configuring the TDS

4.4.1 Edit TDS catalog.xml Files

The `catalog.xml` files for TDS configuration are contained within the `~/tdsconfig` directory. Search for all files terminating in `.xml` in that directory. Edit the

`xml` files for what data you wish to server. See the TDS Documentation[16] for more information on editing these XML files.

Let's see what is available in the `~/tdsconfig` directory.

```
find ~/tdsconfig -type f -name "*.xml"
```

```
/home/ubuntu/tdsconfig/idd/forecastModels.xml
/home/ubuntu/tdsconfig/idd/radars.xml
/home/ubuntu/tdsconfig/idd/obsData.xml
/home/ubuntu/tdsconfig/idd/forecastProdsAndAna.xml
/home/ubuntu/tdsconfig/idd/satellite.xml
/home/ubuntu/tdsconfig/radar/CS039_L2_stations.xml
/home/ubuntu/tdsconfig/radar/CS039_stations.xml
/home/ubuntu/tdsconfig/radar/RadarNexradStations.xml
/home/ubuntu/tdsconfig/radar/RadarTerminalStations.xml
/home/ubuntu/tdsconfig/radar/RadarL2Stations.xml
/home/ubuntu/tdsconfig/radar/radarCollections.xml
/home/ubuntu/tdsconfig/catalog.xml
/home/ubuntu/tdsconfig/threddsConfig.xml
/home/ubuntu/tdsconfig/wmsConfig.xml
```

5 Setting up Data Volumes

As alluded to earlier, we will have to set up data volumes so that the LDM can write data, and the TDS and RAMADDA can have access to that data. The `/mnt` volume on Azure is a good place to store data. Check with Azure about the assurances Azure makes about the reliability of storing your data there for the long term. For the LDM this should not be too much of a problem, but for RAMADDA you may wish to be careful as there is the potential to lose user data.

5.1 Check Free Disk Space

Let's first display the free disk space with the `df` command.

```
df -H
```

[16] `http://www.unidata.ucar.edu/software/thredds/current/tds/catalog/index.html`

Filesystem	Size	Used	Avail	Use%	Mounted	on
/dev/sda1	31G	2.0G	28G	7%	/	
none	4.1k	0	4.1k	0%	/sys/fs/cgroup	
udev	7.4G	8.2k	7.4G	1%	/dev	
tmpfs	1.5G	394k	1.5G	1%	/run	
none	5.3M	0	5.3M	0%	/run/lock	
none	7.4G	0	7.4G	0%	/run/shm	
none	105M	0	105M	0%	/run/user	
none	66k	0	66k	0%	/etc/network/interfaces.dynamic.d	
/dev/sdb1	640G	73M	607G	1%	/mnt	

5.2 Create `/data` Directory

Create a `/data` directory where the LDM can write data soft link to the `/mnt` directory. Also, create a `/repository` directory where RAMADDA data will reside.

```
# Set up data directories
sudo ln -s /mnt /data
sudo mkdir /mnt/ldm/
sudo chown -R ubuntu:docker /data/ldm
sudo mkdir /mnt/repository/
sudo chown -R ubuntu:docker /data/repository
```

These directories will be used by the LDM, TDS, and RAMADDA docker containers when we mount directories from the Docker host into these containers.

6 Opening Ports

Ensure these ports are open on the VM where these containers will run. Ask the cloud administrator for these ports to be open.

Service	External Port
HTTP	80
TDS	8080
RAMADDA	8081
SSL TDM	8443
LDM	388

Note the TDM is an application that works in conjunction with the TDS. It creates indexes for GRIB data in the background, and notifies the TDS via port 8443 when data have been updated or changed. See here[17] to learn more about the TDM.

7 Tomcat Logging for TDS and RAMADDA

It is a good idea to mount Tomcat logging directories outside the container so that they can be managed for both the TDS and RAMADDA.

```
# Create Tomcat logging directories
mkdir -p ~/logs/ramadda-tomcat
mkdir -p ~/logs/tds-tomcat
```

Note there is also a logging directory in `~/tdsconfig/logs`. All these logging directories should be looked at periodically, not the least to ensure that `log` files are not filling up your system.

8 Starting the LDM TDS RAMADDA TDM

8.0.1 RAMADDA Preconfiguration

When you start RAMADDA for the very first time, you must have a `password.properties` file in the RAMADDA home directory which is `/data/repository/`. See RAMADDA documentation[18] for more details on setting up RAMADDA. Here is a `pw.properties` file to get you going. Change password below to something more secure!

```
# Create RAMADDA default password
echo ramadda.install.password=changeme! > /data/repository/pw.properties
```

8.0.2 Final Edit to `docker-compose.yml`

When the TDM communicates to the TDS concerning changes in data it observes with data supplied by the LDM, it will communicate via the `tdm` tomcat user. Edit the `docker-compose.yml` file and change the `TDM_PW` to `MeIndexer`. This is not as insecure as it would seem since the `tdm` user has

[17] `https://www.unidata.ucar.edu/software/thredds/current/tds/reference/collections/TDM.html`

[18] `http://ramadda.org//repository/userguide/toc.html`

few privileges. Optimally, one could change the password hash for the TDM user in the `tomcat-users.xml` file.

8.0.3 Pull Down Images from the DockerHub Registry

At this point you are almost ready to run the whole kit and caboodle. But first pull the relevant docker images to make this easier for the subsequent `docker-compose` command.

```
set -x

# Docker pull all relavant images
docker pull unidata/ldmtds:latest
docker pull unidata/tdm:latest
docker pull unidata/tds:latest
docker pull unidata/ramadda:latest
```

8.0.4 Start the LDM, TDS, TDM, RAMADDA

We are now finally ready to start the LDM, TDS, TDM, RAMADDA with the following `docker-compose` command.

```
# Start up all images
docker-compose -f ~/git/Unidata-Dockerfiles/ams2016/docker-compose.yml up -d
```

9 Check What is Running

At this point, you should have these services running:

- LDM
- TDS
- TDM
- RAMADDA

Next, we will check our work through various means.

9.1 Docker Process Status

From the shell where you started `docker-machine` earlier you can execute the following `docker ps` command to list the containers on your docker host. It should look something like the output below.

```
docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Status}}"
```

CONTAINER	ID	IMAGE	STATUS	
4ed1c4c18814	unidata/ramadda:latest	Up	17	seconds
bdfcf5590bc6	unidata/ldmtds:latest	Up	18	seconds
aee044cf8e66	unidata/tdm:latest	Up	20	seconds
4d0208f85b22	unidata/tds:latest	Up	21	seconds

9.2 TDS and RAMADDA URLs

Verify what you have the TDS and RAMADDA running by navigating to: `http://unidata-server.cloudapp.net/thredds/catalog.html` and `http://unidata-server.cloudapp.net:8081/repository`. If you are going to RAMADDA for the first time, you will have to do some RAMADDA set up[19].

9.3 Viewing Data with the IDV

Another way to verify your work is run the Unidata Integrated Data Viewer[20].

9.3.1 Access TDS with the IDV

In the IDV Dashboard[21], you should be able to enter the catalog XML URL: `http://unidata-server.cloudapp.net/thredds/catalog.xml`.

9.3.2 Access RAMADDAA with the IDV

RAMADDA has good integration with the IDV and the two technologies work well together. You may wish to install the RAMADDA IDV plugin[22] to publish IDV bundles to RAMADDA. RAMADDA also has access to the `/data/ldm` directory so you may want to set up server-side

[19] `http://ramadda.org//repository/userguide/toc.html`

[20] `https://www.unidata.ucar.edu/software/idv/`

[21] `https://www.unidata.ucar.edu/software/idv/docs/userguide/data/choosers/CatalogChooser.html`

[22] `http://www.unidata.ucar.edu/software/idv/docs/workshop/savingstate/`

view of this part of the file system[23]. Finally, you can enter this catalog URL in the IDV dashboard to examine data holdings shared bundles, etc. on RAMADDA `http://unidata-server.cloudapp.net:8081/repository?output=thredds.catalog`.

`Ramadda.html`

[23] `http://ramadda.org//repository/userguide/developer/filesystem.html`