Installation

Download repository

The first thing you need to do is download the adcircreg3simdb GitHub repository. This can be down using the following git command:

```
git clone https://github.com/RENCI/adcircreg3simdb.git
```

After the repository has been downloaded change you directory to the repository:

```
cd adcircreg3simdb
```

The docker file

```
# set the base image
FROM continuumio/anaconda3

# author
MAINTAINER Jim McManus

# extra metadata
LABEL version="1.0"
LABEL description="regin3db image with Dockerfile."

# Set user environment info, USER_ID and USER_GID are 1324, and PG_ID and PG_GID are
# 70 on adcric-db. These are specific to the setup at RENCI, to enable write permission to disk
# storage space defined in the volume. The USER_ID, USER_GID,  and PG_GID need to be
# changed to ones relating to the storage area used in creating the volume.

ENV USER=data GROUP=data USER_ID=1324 USER_GID=1324 PASSWORD=adcircdata
CONDAENV=adcirc PG_USER=postgres PG_GROUP=postgres PG_ID=70 PG_GID=70
PG_PASSWORD=postgres

# update sources list
# install basic apps, one per line for better caching
RUN apt-get clean && apt-get update &&\
    apt-get install -qy nano \
    curl \
    vim \
    lsb-release \
    gnupg2 \
    sudo
```

```
# add user data, specific to adcirc-db and general setup, respectively
RUN groupadd -r -g $USER_GID $GROUP && useradd --no-log-init -r -u $USER_ID -g
$USER_GID $USER

# define working directory
WORKDIR /home/$USER

# Make RUN commands use `bash --login`:
SHELL ["/bin/bash", "--login", "-c"]

# Create the environment, and initialize conda in bash config files.
# Activate the environment, and install 1.44.1 packages:
COPY environment.yml .
RUN conda env create -f environment.yml &&\
 conda init bash &&\
 conda activate $CONDAENV &&\
 conda update -n base -c defaults conda -y --quiet

# Copy bashrc to home directory as .bashrc
ADD bashrc /home/$USER/.bashrc

# restart postgresql, make password for $USER,
# clone adcircreg3simdb repository, and
# change owner and group to data for /home/data
RUN  echo -e "$PASSWORD\n$PASSWORD" | passwd $USER &&\
  mkdir -p /home/$USER/ingestProcessing/csv &&\
  mkdir -p /home/$USER/ingestProcessing/nc &&\
  git clone https://github.com/RENCI/adcircreg3simdb.git &&\
  chown -R $USER:$GROUP /home/$USER

# add user postgres for adcirc-db
# get postgres, postgis and timescaledb repos
RUN groupadd -r -g $PG_GID $PG_GROUP && useradd --no-log-init -r -u $PG_ID -g $PG_GID
$PG_USER &&\
  usermod -a -G $PG_GROUP data &&\
  apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys
B97B0AFCAA1A47F044F244A07FCC7D46ACCC4CF8 &&\
  echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -c -s)-pgdg main" | tee
/etc/apt/sources.list.d/pgdg.list &&\
  wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-key add - &&\
  sh -c "echo 'deb https://packagecloud.io/timescale/timescaledb/debian/ `lsb_release -c -s`
main' > /etc/apt/sources.list.d/timescaledb.list"  &&\
  wget --quiet -O - https://packagecloud.io/timescale/timescaledb/gpgkey | apt-key add -
```

```
# install postgres, postgis and timescaledb
RUN apt-get update && apt-get install -qy postgresql-11-postgis-3 \
   timescaledb-postgresql-11 &&\
   apt-get -qy autoremove

# add user postgres
USER $PG_USER

# add timescaledb to postgresql.conf, and then
# Create a PostgreSQL role named ``postgres`` with ``postgres`` as the password and
# then create a database `postgres` owned by the ``postgres`` role.
# Note: here we use ``&&\`` to run commands one after the other - the ``\``
#       allows the RUN command to span multiple lines.
RUN echo "shared_preload_libraries = 'timescaledb'" >>
/etc/postgresql/11/main/postgresql.conf &&\
   /etc/init.d/postgresql start &&\
   psql --command "ALTER USER "$PG_USER" PASSWORD '"$PG_PASSWORD"';" &&\
   psql --command "CREATE DATABASE reg3sim" &&\
   psql -d reg3sim --command "CREATE EXTENSION postgis;" &&\
   psql -d reg3sim --command "CREATE EXTENSION IF NOT EXISTS timescaledb
CASCADE;" &&\
   psql -d reg3sim --command "CREATE USER data WITH ENCRYPTED PASSWORD
'adcirc';" &&\
   psql -d reg3sim --command "GRANT ALL PRIVILEGES ON DATABASE reg3sim TO data;"
   #psql -d reg3sim --command -f /home/data/adcircreg3simdb/region3-function.sql


USER root

# tune tilescaledb.
# Adjust PostgreSQL configuration so that remote connections to the
# database are possible.
# And add ``listen_addresses`` to ``/etc/postgresql/11/main/postgresql.conf``
RUN timescaledb-tune --quiet --yes &&\
  echo "host all  all    0.0.0.0/0  md5" >> /etc/postgresql/11/main/pg_hba.conf &&\
  echo "listen_addresses='*'" >> /etc/postgresql/11/main/postgresql.conf

# Expose the PostgreSQL port
EXPOSE 5432

# Add VOLUMEs to allow backup of config, logs and databases
VOLUME  ["/etc/postgresql", "/var/log/postgresql", "/var/lib/postgresql", "/home/data"]

# Set the default command to run when starting the container
```

CMD ["/usr/lib/postgresql/11/bin/postgres", "-D", "/var/lib/postgresql/11/main", "-c", "config_file=/etc/postgresql/11/main/postgresql.conf"]

locales to UTF-8
RUN locale-gen C.UTF-8 && /usr/sbin/update-locale LANG=C.UTF-8
ENV LC_ALL C.UTF-8

The Dockerfile uses the continuumio/anaconda3 image. This creates an environment where Anaconda is already installed. Anaconda is a distribution of Python programming language set up for numerical analysis. Using Anaconda simplifies the installation of programs required for the processing of the data, for ingest into the database. The continuumio/anaconda3 image uses Debian GNU/Linux, version 10 (buster) as its operating system.

Build docker images

To build the docker image, first change directory to:

```
cd build
```

The Dockerfile is currently set up for adcirc-db.edc.renci.org at RENCI. In the Dockerfile USER_ID and USER_GID are given the value of 1324, and PG_ID and PG_GID are given the value of 70. This has been done to enable write permission to the storage area defined by the volume, when creating the container. At RENCI the storage area is a shared area, and USER_GID 70 is the group that is used to share the disk. This step may not be necessary if the person who creates the docker image, and the container also owns the disk space defined by volume. If this is the case the user can edit the Dockerfile using their own user id and group id for these values..

ENV USER=data GROUP=data USER_ID=???? USER_GID=???? PASSWORD=adcircdata CONDAENV=adcirc PG_USER=postgres PG_GROUP=postgres PG_ID=?? PG_GID=?? PG_PASSWORD=postgres

After editing the Dockerfile you then run:

```
./buildimage.sh
./createnetwork.sh
```

The command to build the docker image is:

```
docker build -t region3db_image .
```

Where the name of the docker images that is produced is “region3db_image”.

The command to create the network is:

```
docker network create --driver bridge region3db_network
```

Where the name of the network that is created is region3db_network.

Create the docker container

When creating the container you need to define the volume where the postgresql data, and output data will be written too. This disk space for this volume needs to be large enough to write all of the Region 3 simulation data. At RENCI we are using /projects/regionthree/ which is accessible on the dcirc-db.edc.renci.org VM. You can use your own directory path. To create the container you run the following command using you own directory path:

```
./createcontainer.sh /projects/regionthree
```

The command in createcontainer.sh used to create the container is:

```
DIRPATH=$1 # One adcirc-db DIRPATH is /projects/regionthree
PROCESSING=$DIRPATH/ingestProcessing
STORAGE=$DIRPATH/dockerstorage

docker run -ti --name region3db_container --shm-size=4g \
  --network region3db_network \
  -p 5432:5432 \
  --volume $PROCESSING:/home/data/ingestProcessing \
  --volume $STORAGE:/var/local/postgresql \
  -d region3db_image /bin/bash
```

The name of the container created is “region3db_container”. The shared memory size for the container is set to 4 gigabytes using the --shm-size parameter. The network is set, using the --network parameter, to region3db_network, created previously. The port for the database is set to 5432 outside and inside the container using the -p parameter. Two Volumes are set using the --volume parameter. The first volume is set to a location for processing the data before ingesting it into the database. The second volume is set for the location of the data in the database. Finally, the -d parameter is used to define the image used in creating the container. In this case we are using “region3db_image” which we created earlier.

Access the docker container as root

You now should be able to access the container shell using the following command:

```
docker exec -it region3db_container bash
```

Changing postgresql directory

If you are using the container with volumes you will need to issue the following commands as postgres:

```
su postgres
bash
cd /var/local/postgresql
mv /var/lib/postgresql/11 .
vi /etc/postgresql/11/main/postgresql.conf
change data_directory from /var/lib/postgresql to /var/local/postgresql
```

These commands move the data for the Postgres database from /var/lib/postgresql/, which is the standard location, to /var/local/postgresql, which we defined in the container's volume declaration, utilizing the disk space there.

After moving the directory you will need to restart postgresql as user postgres:

```
service postgresql restart
```

Import functions

The next step is to change directory back to the repository, in the container:

```
cd /home/data/adcircreg3simdb
```

and the run a psql command (password is postgres) to load the functions "region3-function.sql" and "region3-function-bufopt.sql" into the database:

```
psql -U postgres --password -d reg3sim -p 5432 -h localhost -f /home/data/adcircreg3simdb/region3-function.sql
```

```
psql -U postgres --password -d reg3sim -p 5432 -h localhost -f /home/data/adcircreg3simdb/region3-function-butopt.sql
```

Now exit from postgres and container root:

```
exit
exit
exit
```

Accessing the container as data

You can logon to the data account with the following command:

```
docker exec -it --user data region3db_container bash
```

Ingest storm data

From there you can run the ingest scripts with the following commands. First change directory to the repository:

```
cd adcircreg3simdb
```

Then activate the conda environment “adcirc”:

```
conda activate adcirc
```

Then run ingest commands for the different ADCIRC variables, using a storm name as input:

```
./ingestZetaFortNcCSV.py storm_name
./ingestSwanAllNcCSV.py storm_name
./ingestVelFortNcCSV.py storm_name
```

Then extract the geometry, and bathymetry from the storm, ingest it into the database:

```
./createGeoNcCSV.py storm_name
./ingestGeoCSV.py
```

An example storm name is BP1_dp1r1b1c1h1l1