**Accounts**

Created a Batch account and a Storage account on the Azure portal with NOAA. Stored the credentials for these in the credentials file

**Controller node**

First we load on the controller node the script that Hem shared, and we interpret that. Somewhere in there we also need to load the controller and credentials files.

**Docker**

It seems like the cluster configuration file allows you to point to a docker image that has all the necessary packages and software to perform the task (Atlantis, R, all dependencies, etc).

It seems like the docker image that is referenced in the cluster configuration file needs to be on a container registry, either docker hub or Azure Container Registry for instance. Now the issue is, the image would contain the Atlantis code (because the image needs to contain the code and executables), so I guess we need a private storage. Details here: <https://github.com/Azure/doAzureParallel/blob/master/docs/33-building-containers.md>

What goes at the end of the Dockerfile? R or RunAtlantis? And do we need to mount the model folder into the Container from the controller? Hem’s code is not running any docker command – maybe it happens under the hood in the doAzureParallel loop?

Let’s think of how we set up the dockerfile and docker image.

The container must have on it:

1. Atlantis and dependencies
2. R packages

Here is the result of reverse-engineering Hem’s Docker image to create a Dockerfile:

# The base image seems to be some Debian-based image, but the exact base is not mentioned.

FROM debian:latest

LABEL org.label-schema.license=GPL-2.0 \

org.label-schema.vcs-url=https://github.com/rocker-org/rocker-versioned \

org.label-schema.vendor=Rocker Project \

maintainer="Carl Boettiger <cboettig@ropensci.org>"

ARG R\_VERSION=3.6.0

ARG BUILD\_DATE

ENV R\_VERSION=${R\_VERSION} \

LC\_ALL=en\_US.UTF-8 \

LANG=en\_US.UTF-8 \

TERM=xterm

# Apt-get updates and installations

RUN apt-get update && apt-get install -y --no-install-recommends [ ... list of packages ... ]

# Setting up R and some R packages

RUN [ ... R installation commands ... ]

# Other installations and environment setups

RUN [ ... various commands ... ]

COPY [source] [destination]

COPY [source] [destination]

COPY [source] [destination]

COPY [source] [destination]

EXPOSE 8787

CMD ["/init"]

MAINTAINER hmorzaria@hotmail.com

# More apt-get installations

RUN apt-get update -qq && apt-get -y --no-install-recommends install [ ... list of packages ... ]

# R package installations using devtools

RUN Rscript -e "install.packages( [ ... list of packages ... ], dependencies = TRUE)"

RUN Rscript -e "devtools::install\_github('jporobicg/shinyrAtlantis')"

RUN Rscript -e "devtools::install\_github('Atlantis-Ecosystem-Model/ReactiveAtlantis')"

RUN Rscript -e "devtools::install\_github('Azure/rAzureBatch')"

RUN Rscript -e "devtools::install\_github('Azure/doAzureParallel')"

ENV TZ=America/Los\_Angeles

RUN ln -snf /usr/share/timezone/$TZ /etc/localtime && echo $TZ > /etc/timezone

# ... any other relevant commands ...

Which by itself is not much help but with the history snippet in this folder gives some indication of what goes on in there. It seems to be setting up mostly the environment to run Atlantis and dependencies and R packages (including Reactive etc.), but it does not seem to build Atlantis (possibly because it could not be stored on Docker Hub?). I’d imagine that the image should need to have Atlantis built within it (we need to run Atlantis there).

So, a merge of Hem’s and Andy’s dockerfiles should do the trick. Build a dockerfile with that, mounting volumes from the local host (Controller) with the code onto the container. Still unsure what the final command should be, but likely “R”, so that all commands in the foreach loop are fed to an R session. I think.

**The loop**

The %do% loop in doAzureParallel distributes tasks across clusters of each node depending on the maxTasksPerNode argument in the cluster configuration file. Meaning that if you have 2-core VM’s keep that parameter to 2 and it should allocate 2 tasks to the node.

The loop runs the command in a container on each node.

Be careful with the configuration of "dedicatedNodes": {

"min": 48,

"max": 48

}

Because if the pool is not deallocated we will always pay for 48 VMs running. However, the command doAzureParallel::stopCluster(cluster) will deallocate the pool and so we’ll stop paying for that.

**Installing R packages**

Seems like we have a few options here.

1. Hem’s script does it in the loop, so for each task
2. We could include them in the docker image and so the container that gets spun up should have the required packages
3. We can do it in the JSON configuration, but it sounds like this install the R packages on the VM that is created but then these packages are not necessarily accessible to the Docker container and the task (Atlantis) that is ran in it.

So it seems like installing the packages in the docker image makes the most sense. It could be as easy as adding a line like: # Install R packages

RUN R -e "install.packages(c('package1', 'package2'), repos='https://cran.rstudio.com/')"

We will need to also set up some kind of pipeline, i.e. what happens after Atlantis is ran? We can (should?) save all output on the Blob storage, perhaps, but then we’ll need to find a way to process that. One option could be adding R code to the loop.

A related question then, if we have an R script that we want to run, is where that exists. Having it on the controller won’t help. If Atlantis runs in the container, then the only option will be having the script itself on the container somehow. A clunkier option would be having the content of the script in the foreach loop itself.

One option may be using the Blob storage (which BTW, will we need to create Containers in the storage account?). Adding this to the loop:

opts <- list(cloudCombine=TRUE, packages="doAzureParallel")

foreach(i=1:10, .options.azure=opts, .export="./path\_to\_local\_file/myscript.R") %dopar% {

source("myscript.R")

# ... rest of your code ...

}

The .export parameter in the foreach function specifies which local files should be uploaded to Azure Blob Storage and then made available to the Azure VMs. To have more than one file / path go with this:

.export=c("./path\_to\_first\_folder/", "./path\_to\_second\_folder/", "./path\_to\_file.R")

These paths can be relative or absolute.

Need to find a way to map each data file to the correct iteration though. Perhaps simple indexing? That is, upload folder with all the run and prm files to the Blob via export, then it gets read in on the VM, mounted to the container (seems to get handled automatically by doAzureParallel), and then we get to use indexing in the foreach loop to pull the relevant files for this run, and feed them to the sh line.