OCEANUM
OCEAN NUMERICAL

An introduction to Docker and how it might be used by the WAVEWATCH III community

WAVEWATCH III Developers Meeting
22/01/2020

Who am I?

- From Melbourne, PhD in global wave modelling at University of Melbourne
- 8 years at the Australian Bureau of Meteorology developing global wave forecast and hindcast systems
- 4 years at the NZ Metocean/Metservice developing global wave forecast and hindcast systems and leading a devops team
- 6 months at Oceanum - wave modelling amongst many other things

Outline

- What is Docker?
- How can it be applied in a scientific compute context
- How could it be used by the WAVEWATCH III group
- Wavespectra python library

What is Docker?

The Challenge

Multiplicity of Stacks

Static website

nginx 1.5 + modsecurity + openssl + bootstrap 2

Background workers

Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs

Web frontend

Ruby + Rails + sass + Unicorn

Queue

Redis + redis-sentinel

DB

Java + Cassandra + Spark

API endpoint

Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client

Do services and apps interact appropriately?

Multiplicity of hardware environments

Development VM

QA server

Customer Data Center

Public Cloud

Disaster recovery

Production Servers

Production Cluster

Contributor's laptop

Can I migrate smoothly and quickly?

Dependency madness

	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
DB	?	?	?	?	?	?	?
Analytics	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?

Pre 1960's transport

Multiplicity of Goods







Do I worry about how goods interact (e.g. coffee beans next to spices)

Multipilicity of methods for transporting/storing







Can I transport quickly and smoothly (e.g. from boat to train to truck)

Also a dependency mess

	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
							

Solution: Intermodal Shipping containers

Docker, shipping containers for code

Multiplicity of Stacks

Static website

User DB

Web frontend

Queue

Analytics DB

Do services and apps interact appropriately?

An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…

…that can be manipulated using standard operations and run consistently on virtually any hardware platform

Multiplicity of hardware environments

Development VM

QA server

Customer Data Center

Public Cloud

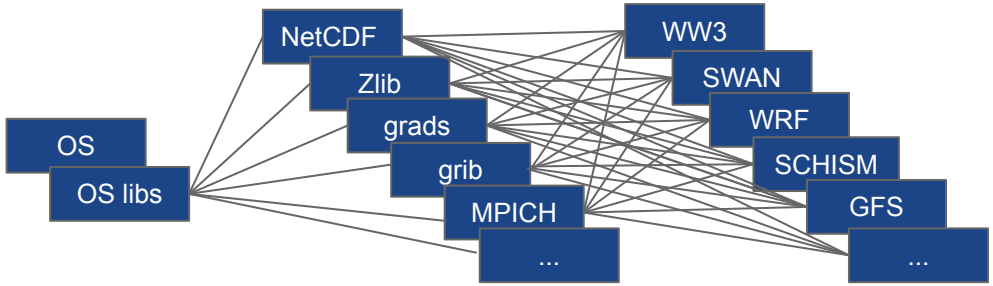
Production Cluster

Contributor's laptop

Can I migrate smoothly and quickly

How can it be applied to a scientific compute context?

Scientific Computing Analogs



Personal Desktop

Development VM

Testing Server

Production Servers

Contributor Laptop

Environment management

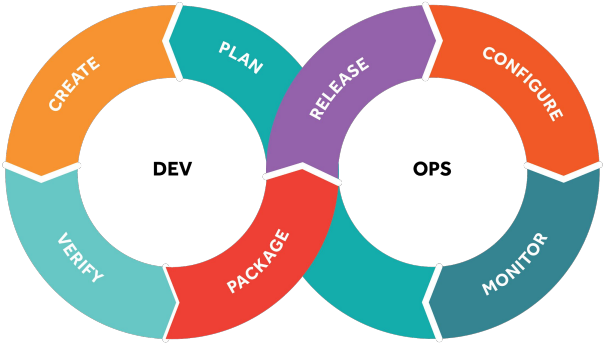
- Have to maintain multiple environments
  - challenging (how to simultaneously develop & test processes with different runtime requirements?)
  - Expensive (setting up multiple environments traditionally laborious and time-consuming)
- Typically multiple staged environments evolve
  - **local** - developer's workstation
  - **dev** (or **sandbox**) - first stage for developers to merge and test
  - **staging** - exact replica of production environment for final verification tests
  - **production**
- The difficulty of progressing through these step mean that changes often bundled together in large batches, making upgrades both slow and high risk
- Introduced tension between scientists who want to deploy things rapidly, and sys admins who (understable) care about stability and security

Advantages of Docker

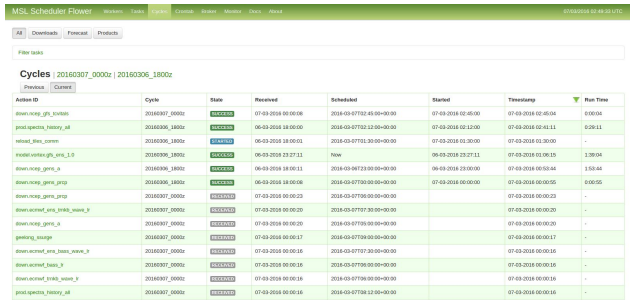
- Build once, deploy anywhere
- Safe, isolated deploys, easy rollbacks
- Consistent results across environments (Dev == Test == Staging == Production)
- Easy disaster recovery

Allows for much more continuous deployment strategies

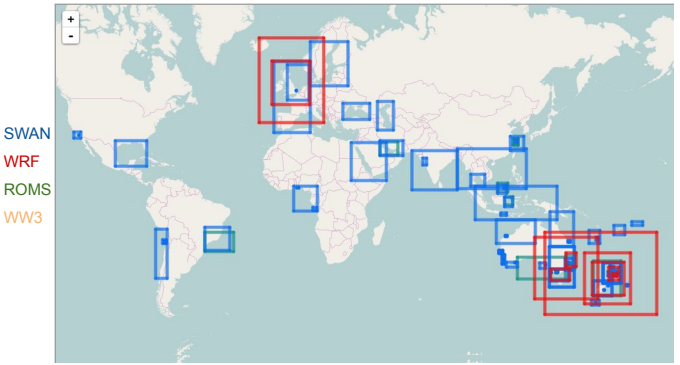
- Move from staged release process that often crosses teams and architectures (and philosophies) to a much more continuous development and release cycle



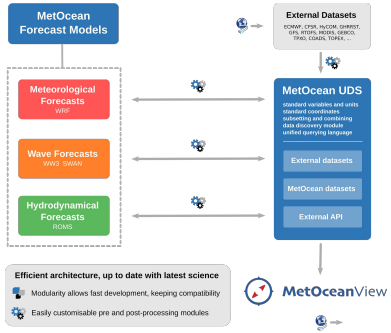
Examples of practical use - Metocean Solutions



In house developed scheduler handles pulling and scheduling models running in dockers on a remote linux cluster



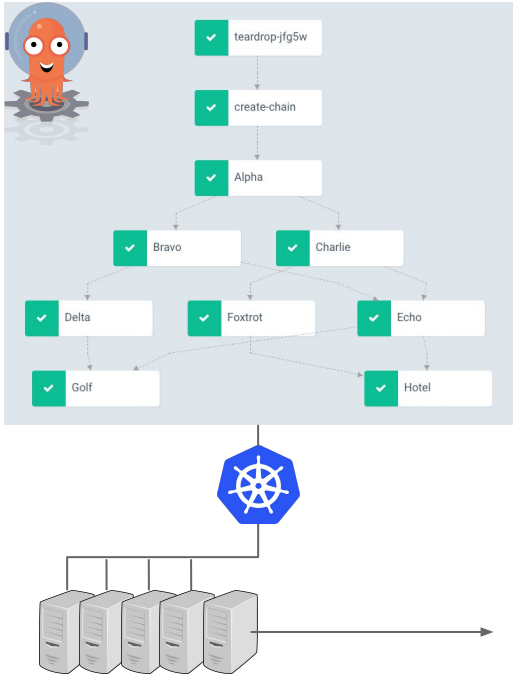
Model domains



High level system architecture

Examples of practical use - Oceanum

- Using kubernetes on top of Docker to abstract the hardware away one step further.
- No longer care about servers at all, hardware is completely ephemeral, and scales on request



Cloud native architecture running models on an autoscaling cluster

How could Docker be used by the WAVEWATCH III group?

Docker for WAVEWATCH III - Possible uses

- Ship built binaries
  - Lower barriers of entry for users who are not necessary comfortable compiling the model (and dependencies)
- Testing
  - Provide a consistent environment for regression testing
- Development
  - Consistent development environment - platform independent
  - Ship development platform with external dependencies already installed (e.g. libraries, coupling frameworks, data assimilation frameworks etc. Keep WAVEWATCH III developers focused on WAVEWATCH III)

Docker - Simple Example

Dockerfile

```
FROM ubuntu:19.04
MAINTAINER Tom Durrant <t.durrant@oceanum.science>

ARG COMP='Gnu'
ARG SWITCH='Ifremer1'
ARG PROGS='ww3_grid ww3_strt ww3_prnc ww3_shel ww3_multi ww3_ounf ww3_ounp'

# Upgrade and install required libs
RUN apt-get -y update &&\
    apt-get -y upgrade &&\
    apt-get -y install make gcc gfortran vim mpich \
                       libnetcdf-dev libnetcdff-dev &&\
    apt-get clean


# Compile and install model
COPY model /source/model
COPY regtests /source/regtests
WORKDIR /source/model/bin


# Set required environment variables
ENV WWATCH3_NETCDF=NC4
ENV NETCDF_CONFIG=/usr/bin/nc-config

# Compile model
RUN ./w3_setup /source/model -q -c ${COMP} -s ${SWITCH}
RUN ./w3_make ${PROGS}
```

On your workstation

```
docker build -t tomdurrant.ww3:v1
docker push tomdurrant.ww3:v1
```

On the server

```
docker pull tomdurrant.ww3:v1
docker run -ti tomdurrant.ww3:v1
```

Docker - More complex Example

Install PGI compiler → Compile Libraries → Compile Model

```
FROM  ubuntu:18.04
LABEL maintainer "Oceanum Developers <developers@oceanum.science>"

ARG pgi_tarball

# Install requirements (gcc & g++)
RUN apt-get update && apt-get install -y --no-install-recommends \
        autoconf \
        automake \
        make \
        gcc \
        g++ \
        wget \
        ca-certificates && \
    rm -rf /var/lib/apt/lists/*

# Install PGI
ENV PGI_VERSION 19.4
ENV PGI_INSTALL_DIR /usr/local/pgi
ENV PGI_HOME    ${PGI_INSTALL_DIR}/linux86-64/${PGI_VERSION}
ENV PGI_BIN_DIR ${PGI_HOME}/bin
ENV PGI_LIB_DIR ${PGI_HOME}/lib
ENV PGI_MAN_DIR ${PGI_HOME}/man

ADD $pgi_tarball /tmp

RUN export PGI_SILENT=true && \
    export PGI_ACCEPT_EULA=accept && \
    export PGI_INSTALL_NVIDIA=false && \
    export PGI_INSTALL_MANAGED=true && \
    export PGI_INSTALL_AMD=false && \
    export PGI_INSTALL_JAVA=false && \
    export PGI_INSTALL_MPI=true && \
    export PGI_MPI_GPU_SUPPORT=false && \
    /tmp/install && \
    rm -rf /tmp/*

RUN echo "${PGI_LIB_DIR}" >> /etc/ld.so.conf.d/pgi.conf

ENV PATH            ${PGI_BIN_DIR}:${PATH}:${PGI_HOME}/mpi/openmpi/bin
ENV LD_LIBRARY_PATH ${PGI_LIB_DIR}:${LD_LIBRARY_PATH}:${PGI_HOME}/mpi/openmpi/lib
ENV MANPATH         ${PGI_MAN_DIR}:${MANPATH}
```

```
FROM registry.gitlab.com/oceanum/docker-pgi/compile-pgi
LABEL maintainer "Tom Durrant <t.durrant@oceanum.com>"

# Install requirements (gcc & g++)
RUN apt-get update && apt-get install -y --no-install-recommends \
        zlib1g-dev curl vim && \
    apt-get clean

# Required build arguments
ARG zlib_version
ARG mpich_version
ARG hdf5_version
ARG netcdf_version
ARG netcdf_fortran_version

# Set required environment variables
ENV MPICH_VERSION=$mpich_version
ENV ZLIB_VERSION=$zlib_version
ENV HDF5_VERSION=$hdf5_version
ENV NETCDF_VERSION=$netcdf_version
ENV NETCDF_FORTRAN_VERSION=$netcdf_fortran_version
ENV PATH=$PATH:/usr/local/bin

# Install model requirements
ADD requires.sh /tmp/
RUN cd /tmp && bash requires.sh
```

```
FROM registry.gitlab.com/oceanum/docker-pgi/pgi-batteries:openmpi
LABEL maintainer Tom Durrant <t.durrant@metocean.co.nz>

ARG PHYSICS
ARG COMP
ARG SWITCH

RUN apt-get -y update &&\
    apt-get -y upgrade &&\
    apt-get -y install man make gcc gfortran &&\
    apt-get clean


# Compile and install model
COPY model /source/model
COPY regtests /source/regtests
WORKDIR /source/model/bin

# Set required environment variables
ENV WWATCH3_NETCDF=NC4
ENV NETCDF_CONFIG=/usr/local/bin/nc-config
RUN ./w3_setup /source/model -q -c ${COMP} -s ${SWITCH} -t /source/model/tmp
RUN ./make_ON
```

Reusable blocks - ‘save as’ at an OS level

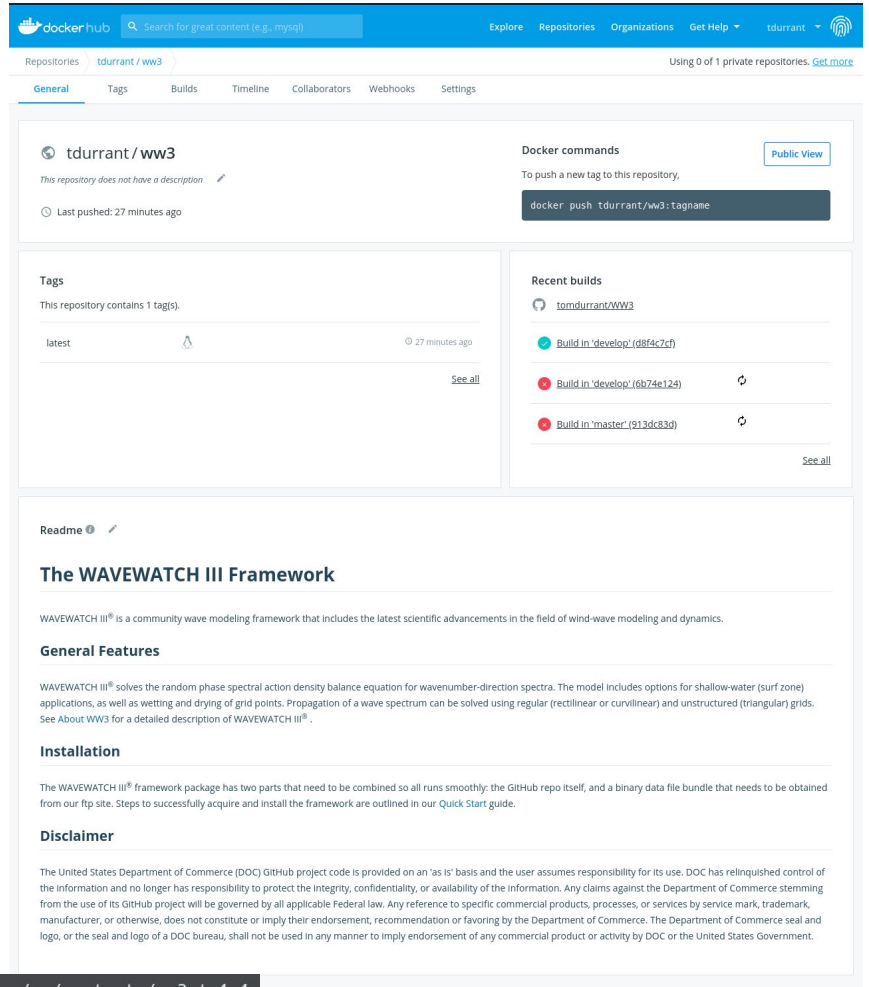
Dockerhub and Autobuilds

- Dockerhub is docker registry and docker building service
- Free for open source projects (such as WW3!)
- Example set up here will rebuild the docker with every commit (possible to refine this)
- https://hub.docker.com/repository/docker/tdurrant/ww3
- Run locally with:

```
docker run -ti tdurrant/ww3:latest
```

- Run regtest:

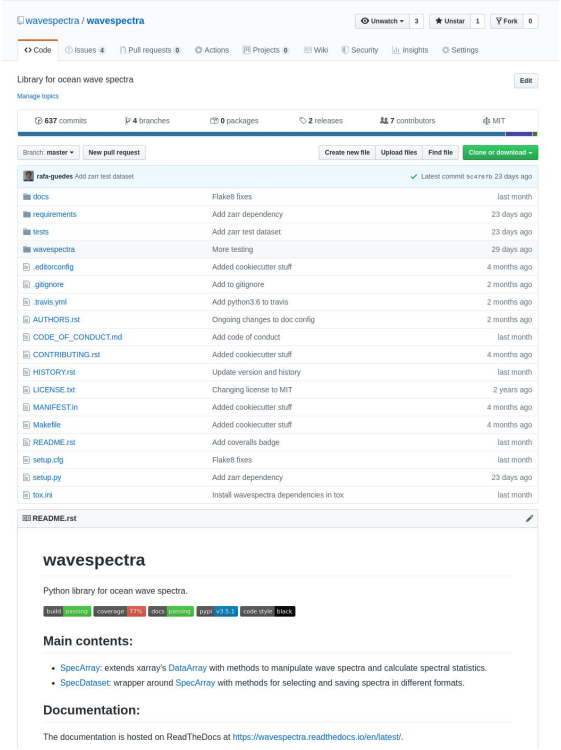
```
docker run ww3 /source/regtests/bin/run_test /source/model ../../regtests/ww3_tp1.1
```



Wavespectra - A python library for processing ocean wave spectral data

Wavespectra python library

- Wavespectra is an open source python library for processing ocean wave spectral data.
- Provides reading and writing of different spectral data formats, calculation of common integrated wave parameters, spectral partitioning and spectral manipulation
- The library is built on top of xarray, providing speed and efficiency for large numbers of spectra
- Code: github.com/wavespectra/wavespectra
- Docs: wavespectra.readthedocs.io



OCEANUM
OCEAN NUMERICAL

Thanks

Tom Durrant
t.durrant@oceanum.science