Introduction to Docker containers

containerized UM-Bridge models/benchmarks

Anne Reinarz

Agenda

Goal: Introduce Docker containers and how they are used in UM-Bridge to provide models/benchmarks.

This Talk covers:

- Introduction to Docker Containers
  - How to set them up
  - How to run them
- UM-Bridge models/benchmarks
  - How docker containers are used in UM-Bridge

Agenda

Goal: Introduce Docker containers and how they are used in UM-Bridge to provide models/benchmarks.

This Talk covers:

- Introduction to Docker Containers
  - How to set them up
  - How to run them
- UM-Bridge models/benchmarks
  - How docker containers are used in UM-Bridge

Docker Containers

UM-Bridge

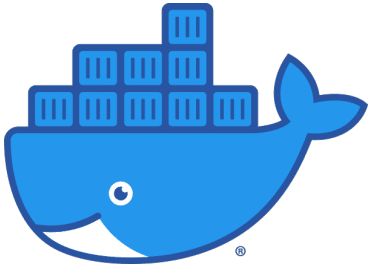
- Models and benchmark implementations support the UM-Bridge interface and are provided as containers.
- UM-Bridge is an abstract interface between UQ methods and models. It mimicks the mathematical "model interface" required by many UQ methods, essentially treating a model as a function mapping vectors onto vectors.

Why Docker?

- **Easy installation:** The installation process for any model or benchmark comes down to a single docker command.
- **Easy access to HPC resources:** UM-Bridge allows for easy access to HPC-scale performance in cloud computing environments. Model containers can be run on any kubernetes cluster without modification.

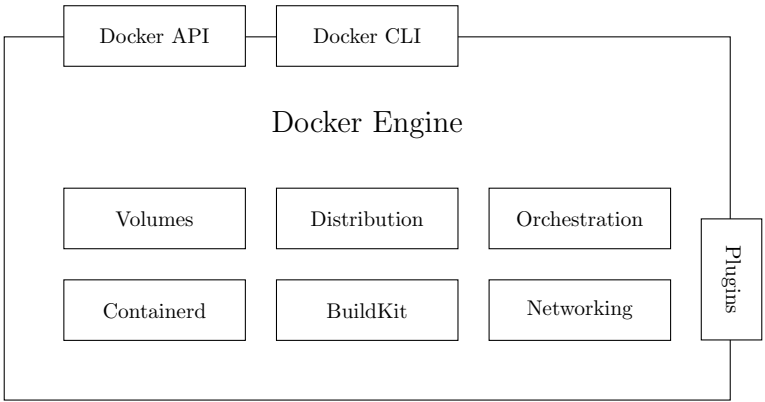
  UM-Bridge provides a reference kubernetes configuration which takes care of load balancing on large-scale clusters. The UQ software can then make use of parallel model instances simply by making concurrent model calls.

What is Docker?



- A Docker container image is a lightweight, standalone, executable package of software.
- The container packages code, runtime, system tools, system libraries and settings.

Docker Engine



Useful Links

- For installation guidelines for most operating system look at: `https://docs.docker.com/engine/install/ubuntu/`
- Docker has an extensive documentation at: `https://docs.docker.com/get-started/`
- The tutorials at `https://diveintodocker.com/` may also be interesting

Hello World

Let's try it for a simple container

```
docker run hello-world
```

- If the hello-world image does not exist locally it will be pulled from dockerhub
- Allows you to test your docker installation

A simple container

Let's try it for another simple example

```
docker run -it ubuntu
```

- This is running new container, we are simply pulling the latest ubuntu image from dockerhub
- `-it` is shorthand for `-i -t`
- `-i` or `--interactive=` starts an interactive docker container. We connect to the container's stdin.
- `-t` gives us access to the terminal in the docker container

A simple container

Inside the container we can run ubuntu commands:

```
apt update && apt install -y figlet
```

```
figlet hello world
```

A simple container

The corresponding Dockerfile would look like this:

```
FROM ubuntu

RUN apt update && apt install -y figlet
CMD figlet hello world
```

A simple container

To build an image from a Dockerfile running

```
docker build -t image-name .
```

- `-t` means tag and gives the image a name rather than meaning terminal as in the run command

Other basic Docker CLI commands

To see current images

```
docker image ls
```

To remove specific images

```
docker image rm name
```

A simple container

Some commonly used docker commands:

- `FROM` sets the base image
- `RUN` executes linux commands while building
- `CMD` executes linux commands while running
- `WORKDIR` sets the working directory
- `ENV` sets environment variables
- `COPY` copies data into the image

Docker Workflow

- Create container image with our application
- Create an UM-Bridge server to communicate with that docker container
- Run the container with that image
- Send requests for model/benchmark evaluations via UM-Bridge
- If we need to tweak the model, update the Dockerfile and push

Simple Example with UM-Bridge

- Grab the minimal server and copy it into the Dockerfile
- Run the UM-Bridge server instead of our figlet command
- When running we need to specify the ports with

  `-p 4242:4242`

Pushing to dockerhub

- Makes your image accessible from anywhere
- Requires an account on dockerhub: `https://hub.docker.com/`
- Possible via CLI:

```
docker login
docker push name/image-name
```

UM-Bridge Models/Benchmarks

UM-Bridge Benchmarks

One of the purposes of UM-Bridge was to create an easily accessible set of benchmarks for the UQ community.

These benchmarks should be:

- Representative
- Make it easy to compare an algorithm to current state of the art algorithms
- Maintainable

UM-Bridge Benchmarks

UM-Bridge benchmarks provide:

- Automated testing and building via Github actions.
- Partially-automated documentation.
- Use of containers keeps maintenance effort for models minimal.

UM-Bridge Benchmarks

Testing a new algorithm on a problem from the benchmark collection offers the following advantages:

- **Ease of use:** Each model or benchmark can be accessed from any supported language through a simple function call
- **Separation of concerns:** The structure of UM-Bridge enables UQ and model experts to work together effectively, since each side only needs to support UM-Bridge.

UM-Bridge Benchmarks

The benchmarks can be found on Github
`https://github.com/UM-Bridge/benchmarks`

Their documentation is at
`https://um-bridge-benchmarks.readthedocs.io/en/docs/`

UM-Bridge Benchmarks

The benchmarks repository consists of two components:

1. Models: Mathematically, an UM-Bridge model is a function mapping parameter vectors onto model output vectors, supporting some of the following:
   - Simple evaluation,
   - Gradient evaluation,
   - Jacobian action,
   - Hessian action.
2. Benchmarks: Define a full UQ problem rather than just a forward model.

A complex UQ problem will thus consist of two seperate directories; one for the forward model and one for the UQ benchmark.

UM-Bridge Benchmarks

The benchmarks repository consists of two components:

1. Models: Mathematically, an UM-Bridge model is a function mapping parameter vectors onto model output vectors, supporting some of the following:
   - Simple evaluation,
   - Gradient evaluation,
   - Jacobian action,
   - Hessian action.
2. Benchmarks: Define a full UQ problem rather than just a forward model.

A complex UQ problem will thus consist of two seperate directories; one for the forward model and one for the UQ benchmark.

UM-Bridge Models

Current models:

- Tsunami
- Composite material with random wrinkle
- L2-Sea
- Tritium Diffusion
- Euler-Bernoulli Beam

UM-Bridge Benchmarks

Current benchmarks:

- Propagation Benchmarks
  - Uncertainty propagation of material properties of a cantilevered beam
- Inference Benchmarks
  - Inferring coefficient field in two-dimensional Poisson PDE
  - Analytic Funnel, Analytic Donut, Analytic Gaussian Mixture, Analytic Banana
  - Tsunami source inference
  - Agent based disease transmission model
  - Boundary condition inversion in a three-dimensional p-Poisson nonlinear PDE
  - Inferring material properties of a cantilevered beam
  - Tritium Diffusion Posterior

UM-Bridge Benchmarks

Current benchmarks:

- Propagation Benchmarks
  - Uncertainty propagation of material properties of a cantilevered beam
- Inference Benchmarks
  - Inferring coefficient field in two-dimensional Poisson PDE
  - Analytic Funnel, Analytic Donut, Analytic Gaussian Mixture, Analytic Banana
  - Tsunami source inference
  - Agent based disease transmission model
  - Boundary condition inversion in a three-dimensional p-Poisson nonlinear PDE
  - Inferring material properties of a cantilevered beam
  - Tritium Diffusion Posterior

UM-Bridge Models/Benchmarks

Models and Benchmarks consist of at least:

- A Dockerfile
- A README following a pre-defined structure
- An UM-Bridge server

They may also have

- Additional required files, e.g. code, input data
- An UM-Bridge client

UM-Bridge Models/Benchmarks

Models and Benchmarks consist of at least:

- A Dockerfile
- A README following a pre-defined structure
- An UM-Bridge server

They may also have

- Additional required files, e.g. code, input data
- An UM-Bridge client

This afternoon

Try this yourself by working throught the tutorial at

`https://um-bridge-benchmarks.readthedocs.io/en/docs/tutorial.html`