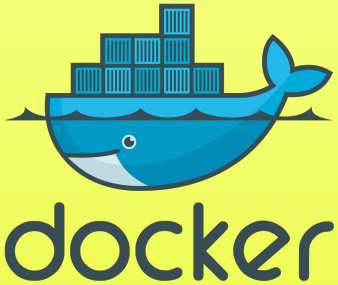
Docker and REANA: a brief intro

Maxim Potekhin
Nuclear and Particle Physics Software Group (BNL)

Phenix School 2021



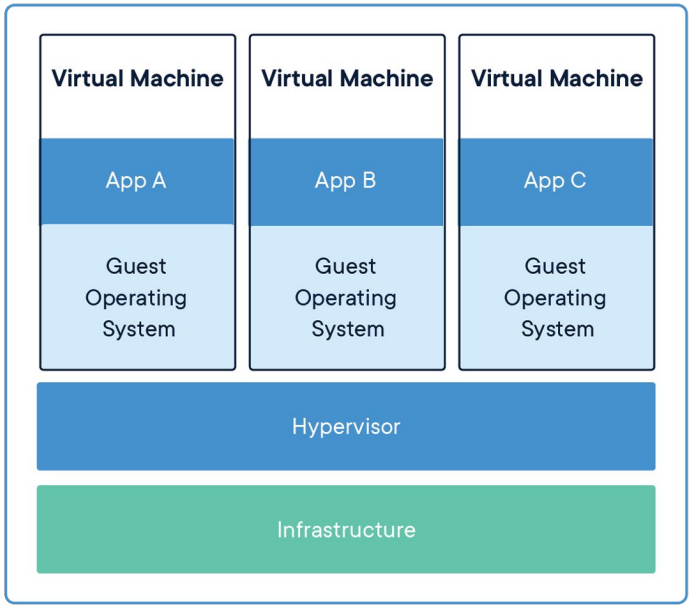
Objectives

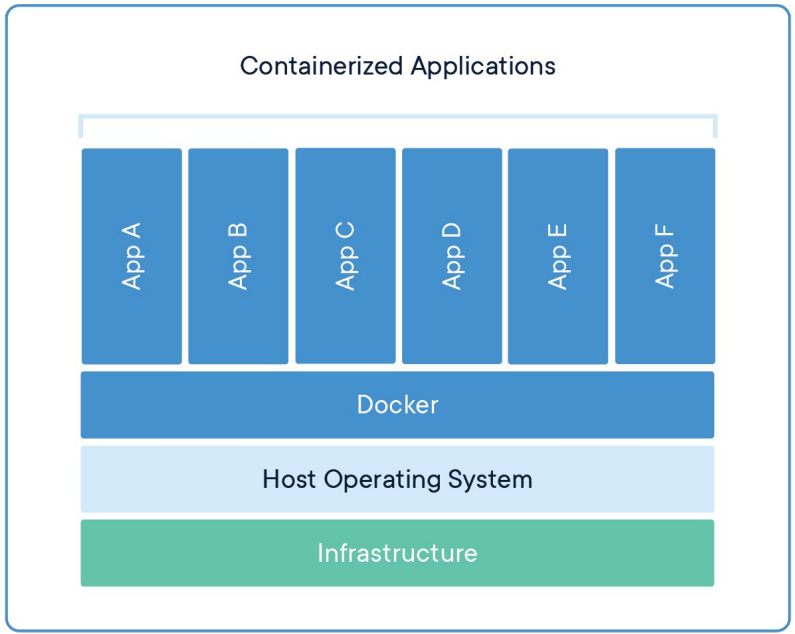
- The main “Data and Analysis Preservation” (DAP) session is scheduled for 6/23/2021 and will involve interactive REANA exercises
  - REANA is the “reproducible analysis” platform, it is based on containerization
- The goal today is to give pointers to those participants who are not very familiar with containerization technology and REANA, so that they have a chance to explore these subjects at their own pace before the REANA session
- Another goal is to make sure participants are able to run the REANA client software and access the REANA Web interface
- Please see our website for more information: https://www.phenix.bnl.gov/analysis/reana.html
- Questions? Contact me at “potekhin at bnl.gov”

Capturing the software environment

- Capturing and preserving the software environment has value both in the short and long-term - to keep analyses organized and self-contained
- Enhances reproducibility
  - cf PHENIX was using containers in production for that purpose
- A common way to achieve this is virtualization
  - A natural way to preserve software in working condition in the long term
  - Two distinct solutions: Virtual Machines and Containers
  - We will focus on containers

Containers vs VM





- Virtual machines require a “hypervisor” which is responsible for **complete emulation** of an OS
- By contrast containers share the same OS kernel resulting in more economical storage and better performance
- Containers are made possible by the Linux resources isolation features (**control groups and namespaces**)

Namespaces and Control Groups

Docker uses a technology called `namespaces` to provide the isolated workspace called the *container*. When you run a container, Docker creates a set of *namespaces* for that container. These namespaces provide a layer of isolation. Each aspect of a container runs in a separate namespace and its access is limited to that namespace.

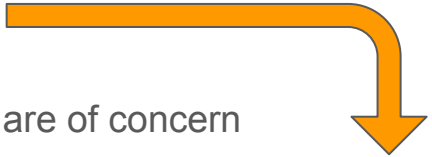
Docker Engine uses namespaces such as the following on Linux:

- **The `pid` namespace:** Process isolation (PID: Process ID).
- **The `net` namespace:** Managing network interfaces (NET: Networking).
- **The `mnt` namespace:** Managing filesystem mount points (MNT: Mount).
- **...and a few others...**

Docker Engine on Linux also relies on a technology called “control groups” (cgroups).
A cgroup limits an application to a specific set of resources. Control groups allow Docker Engine to share available hardware resources to containers and optionally enforce limits and constraints.

Containers, images, repositories

- "Image" is a read-only template residing in storage - essentially a tarball
- It is used to create a running process - the "container"
- "Registry" is a storage and access system for images
- Example: inspect images on a local machine - "docker image ls"
- Docker Hub is a cloud-based registry provided by Docker Inc
- BNL has its own registry which can be uses if privacy and security are of concern



REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
simple_server	latest	3d7b269117a7	20 months ago	158MB
django_import	latest	975596a4f73f	20 months ago	207MB
ubuntu	latest	d131e0fa2585	21 months ago	102MB
alpine	latest	cdf98d1859c1	21 months ago	5.53MB
mediawiki	latest	efd68a02fb8a	21 months ago	691MB
mariadb/server	latest	8fe757be2fd3	23 months ago	368MB
nginx	latest	568c4670fa80	2 years ago	109MB

REANA - “reproducible analysis”

- REANA: captures the workflow, the software, the data
  - Hence it’s capable of true analysis reproducibility
  - If used correctly, helps present a clear description of the computational process
- The software environment is provisioned in the form of Docker containers
- Execution is not interactive - containers are run on a virtual cluster managed with Kubernetes. You don’t need to know details apart from the CLI client.
- The data are staged in (inputs) and staged out (outputs)
- Simple, intuitive description of linear workflows in YAML
  - Can be parametrized, self-documenting
- A more complex syntax is available to describe arbitrary workflow graphs

REANA interfaces

- A basic Web UI provided by the REANA server
  - Can check on the status of jobs running within the system
  - Download files if necessary
- Interaction with the system is mainly via a **CLI client**
  - Richer set of functionality compared to the Web UI, Python-based
  - In our exercise we'll use interactive nodes at BNL SDCC to run the client because it can be set up more easily compared to individual users' machines
  - We will use a SSH tunnel to access the Web UI from the users' machines located outside of the BNL perimeter

SSH tunnel to access the REANA cluster

- You can use utilities like PuTTy, MobaXterm or others capable of ssh tunneling
- Alternatively, on Linux and Linux-like machines you can use straight SSH:

  ssh -L 30443:kubmaster01.sdcc.bnl.gov:30443 USERNAME@ssh.sdcc.bnl.gov
- Once connection is established the REANA Web page will be available at https://localhost:30443/
- Please give it a try
- You should be able to log into the service with the password you chose when you were applying for an account
- If you still don't have an account you can apply for one now

REANA tokens

- The user needs a unique token generated by the service
- There is a simple “sign up process”
- Each account is authorized by the system administrator
- The token can be viewed in the Web UI, and copied to users’ scripts if necessary
- An environment variable must be set for the client to access the service
  - setenv REANA_ACCESS_TOKEN XXXX # user's REANA token

Setting up REANA client

- Log onto an available interactive node (rcas206? machines)
- Use pre-fabricated Python environment to activate the client
  - source /direct/phenix+u/mxmp/.virtualenvs/reana/bin/activate.csh
- Set the environment variables:
  - setenv REANA_ACCESS_TOKEN YOUR_PRIVATE_TOKEN
  - setenv REANA_SERVER_URL https://kubmaster01.sdcc.bnl.gov:30443
- See if the client can run in your account
  - reana-client --help # just to see if it's alive - also a useful summary of available commands, please use this option!
  - reana-client ping # try connecting to the server and see the output

Backup

REANA: a simple example of the YAML input

```
version: 0.0.1
inputs:
  files:
  - ./hello_world_input/hello_world.txt
workflow:
  type: serial
  specification:
    steps:
      - environment: 'ubuntu:18.04'
        commands:
        - echo 'Hello World!' > helloworld1_ouput.txt
outputs:
  files:
    - helloworld1_ouput.txt
```

Staging files to the work area

Pulling image from Docker Hub

Output will be available for download from CLI and GUI