VO Applications running in Containers

An example with Topcat and SPLAT

Margarida Castro Neves[1,2]

[1]Astronomisches Rechen-Institut, Zentrum für Astronomie, Universität Heidelberg

[2]German Astrophysical Virtual Observatory

ESCAPE WP4 Technology Forum 1, Strasbourg, 4-6 February 2020

Why Containers

- Current situation: standalone applications (Topcat, Aladin, SPLAT-VO,...) or web-based. Data can be exchanged via SAMP

Why Containers

- Current situation: standalone applications (Topcat, Aladin, SPLAT-VO,...) or web-based. Data can be exchanged via SAMP
- Experiment: each application runs in a docker container

Why Containers

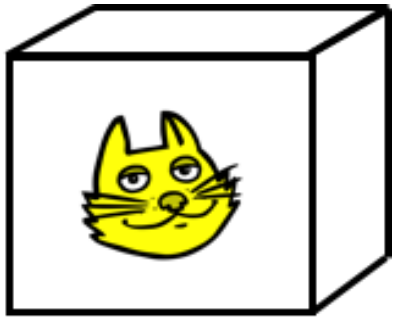
- Current situation: standalone applications (Topcat, Aladin, SPLAT-VO,...) or web-based. Data can be exchanged via SAMP
- Experiment: each application runs in a docker container
- Advantages: Containers use same OS, no need for setup for different OS versions/environments; No need to deliver native code libraries for different architectures (SPLAT);

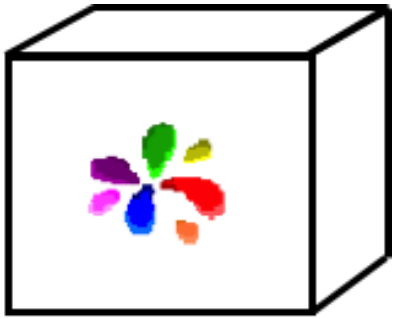
Why Containers

- Current situation: standalone applications (Topcat, Aladin, SPLAT-VO,...) or web-based. Data can be exchanged via SAMP
- Experiment: each application runs in a docker container
- Advantages: Containers use same OS, no need for setup for different OS versions/environments; No need to deliver native code libraries for different architectures (SPLAT);
- Disadvantages: GUI Applications depend on setup of host machine, network communication tricky

Setup Environment and Goal

- Docker installation (Docker for Mac)
- Create containers for some VO Applications (Topcat, SPLAT)
- let them read/write data to the host machine
- let them communicate with each other via SAMP





Setup

Needed to create a container:

- dockerfile: a file containing instructions to create the container
- the application executables (can also be downloaded/built from the dockerfile)
- a startscript to set some environment variables and start the application

Setup: example dockerfile for topcat container

```
# set parent image, working directory
FROM openjdk:8-jre
COPY . /usr/topcat
WORKDIR /usr/topcat
# update and instal needed libraries
 RUN apt-get update; &&\
   apt-get -y install libxrender1 libxtst6 libxi6
# download topcat (topcat-full.jar and topcat startscript
RUN mkdir bin &&\
  wget -O bin/topcat-full.jar http://www.star.bris.ac.uk/~mbt/to
  wget -O bin/topcat http://www.star.bris.ac.uk/~mbt/topcat/topc
# execute startscript
CMD ["/usr/topcat/starttopcat"]
```

Setup: example startscript for topcat container

```
#!/bin/sh

SAMP_HUB=std-lockurl:file://localhost/root/.samp_$HOSTNAME
export SAMP_HUB

bin/topcat -Djsamp.localhost=${hostname} -exthub
```

Running GUI Applications in a Container

Very tricky as the docker container is headless. Many solutions proposed (X-forwarding, ssh, VNC,...), no common solution for all host operating systems, with no extra work for user.
Using X-Forwarding:

- Host machine has to allow container to connect to X11 Server
- Container has to know the ip address of host machine ($DISPLAY)

Running GUI Applications in a Container, X Forwarding

- Host machine has to allow container to connect to X11 Server

- Container has to know the ip address of host machine

Running GUI Applications in a Container, X Forwarding

- Host machine has to allow container to connect to X11 Server
  - Reckless solution that works: 

    ```
    $ xhost +
    ```

- Container has to know the ip address of host machine

Running GUI Applications in a Container, X Forwarding

- Host machine has to allow container to connect to X11 Server
  - Reckless solution that works: 

    ```
    $ xhost +
    ```

  - Better solution: use own ip address. On a MacOs machine:

    ```
    ip=$(ipconfig getifaddr en0)
    xhost + $ip
    ```

- Container has to know the ip address of host machine

Running GUI Applications in a Container, X Forwarding

- Host machine has to allow container to connect to X11 Server
  - Reckless solution that works:

    ```
    $ xhost +
    ```

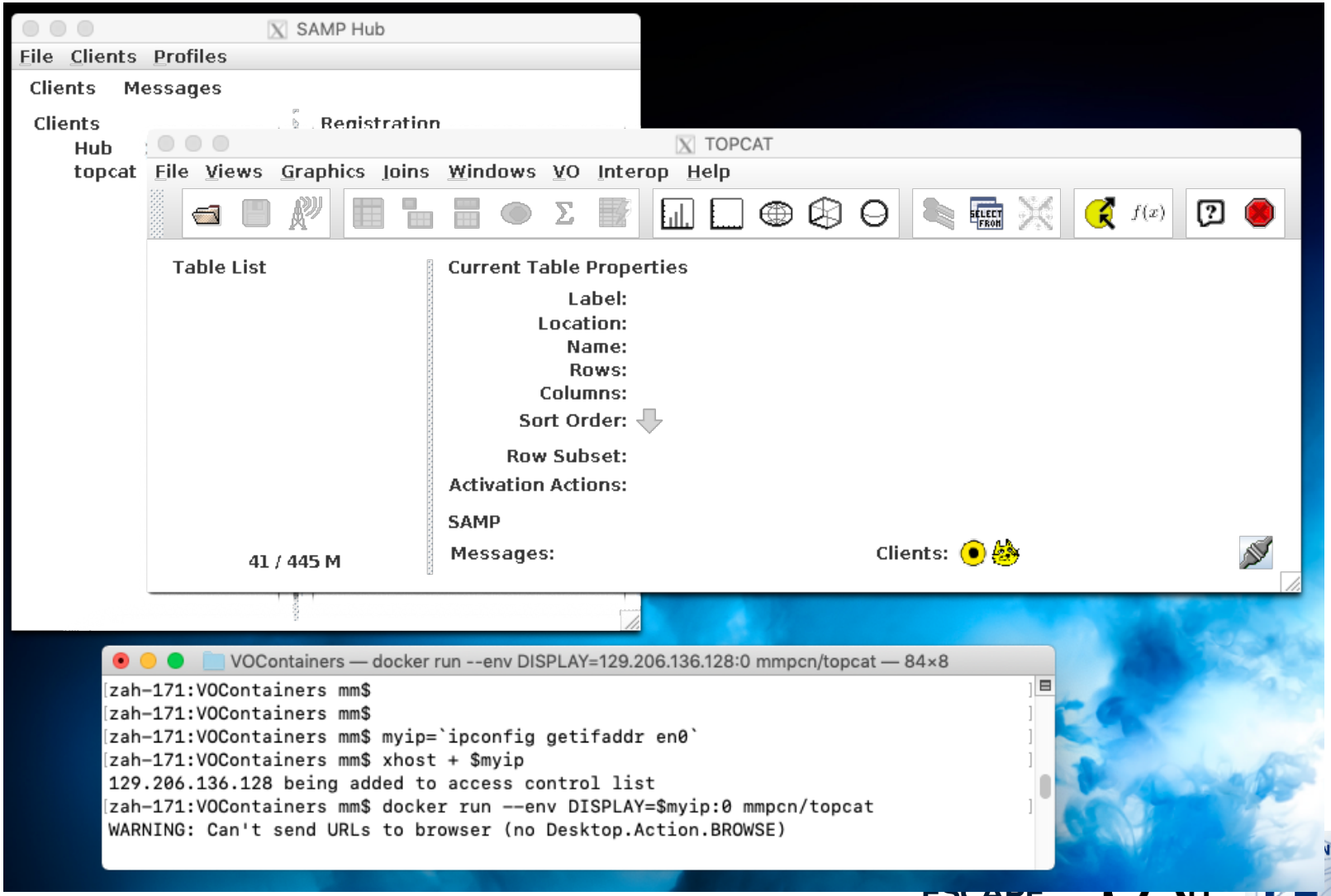
  - Better solution: use own ip address. On a MacOs machine:

    ```
    ip=$(ipconfig getifaddr en0)
    xhost + $ip
    ```

- Container has to know the ip address of host machine
  - passed to container as environment variable DISPLAY

    ```
    docker run --env DISPLAY=$ip:0 -t mmpcn/topcat
    ```

Running GUI Applications in a Container, X Forwarding



Sharing data with the host (and between containers)

Mount a chosen directory to the containers

- containers can save data on the host
- containers can read existing data from the host

```
docker run --env DISPLAY=$ip:0 \
        --volume=~mm/datafiles:/root -t mmpcn/topcat
```

SAMP communication between containerised applications

- Clients in same host, but in different containers (different ip-adresses)
- To make things easy, run containers in same network:

```
docker create samp-network
docker run [options] -net samp-network [container]
```

- Each VO Application container (now SPLAT, TOPCAT) starts a SAMP Hub
- Use the JSAMP bridge functionality:
  `http://www.star.bristol.ac.uk/~mbt/jsamp/bridge.html`
  With JSAMP bridge, clients can send and receive to and from each other clients just as if they were local

SAMP communication: JSAMP bridge

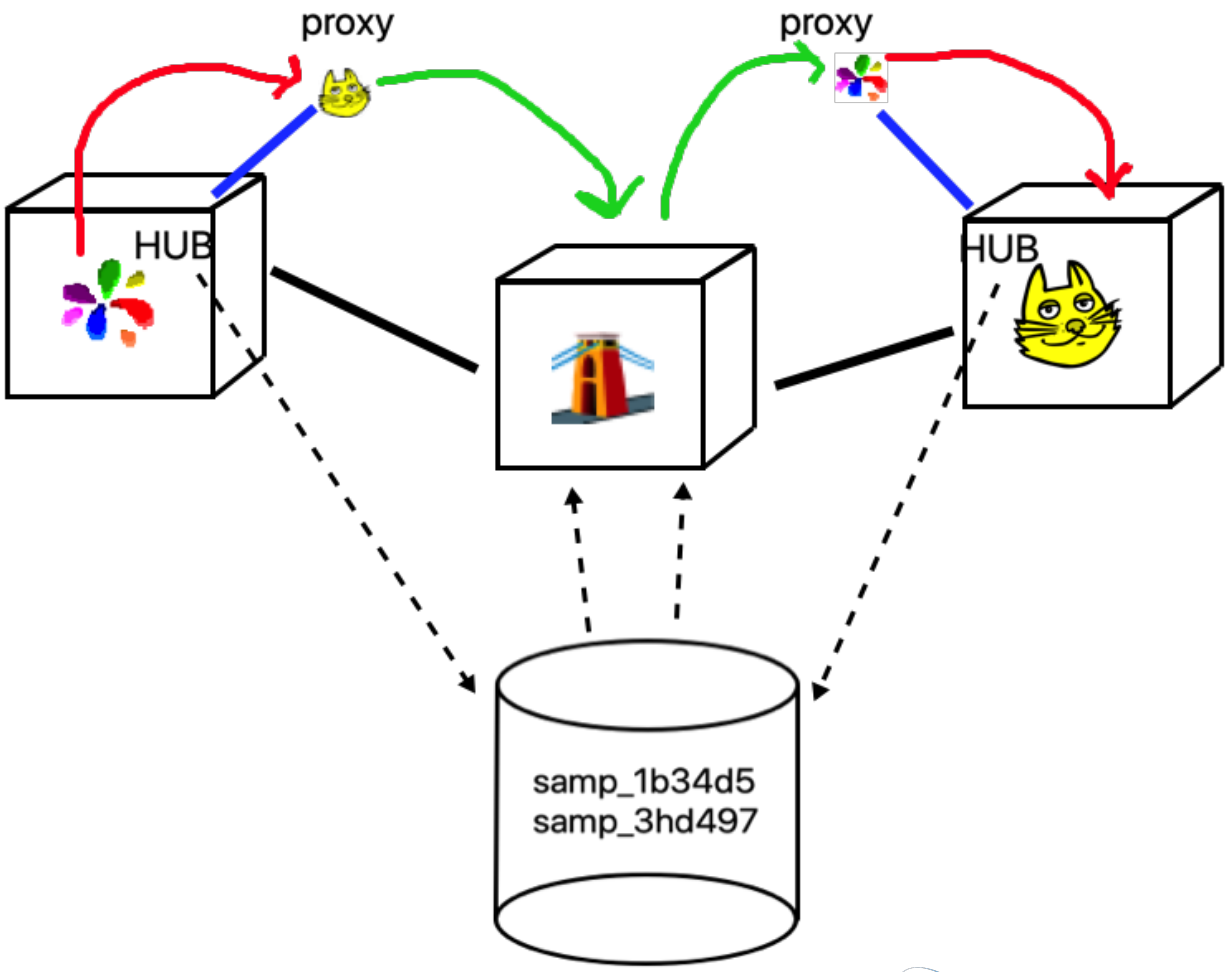
- create a jsamp container, access shared data volume on the host
- SAMP hubs write a lockfile (default: $HOME/.samp)
- But: each container has its own hub, with a different lockfile
- solution: write the lockfiles to a shared directory, and use an unique filename for each one. This is done setting the environment variable SAMP_HUB at the startscript in the container:

```
SAMP_HUB=std-lockurl:file://localhost/root/.samp_$HOSTNAME
export SAMP_HUB
```

SAMP communication: JSAMP bridge

- create a jsamp container in samp-network, and with access to the directory where the lockfiles are
- in the startscript, it looks for existing lockfiles and creates a bridge
- Problem/TO DO: jsamp bridge can't dinamically discover new hubs,so it has to be started after the other containers are running
- Still not tested: how to communicate with local applications or webSAMP

SAMP communication: JSAMP bridge



Containerization of VO Applications: Conclusions

Results:

- this was a short test, to see if it works and if it's worth
- works in certain conditions
- not very comfortable for the user, especially GUIs
- setup may vary according to operating system
- Possibility for running remotely from an application server?
- Kubernetes orchestration, etc. have not yet been considered/tested

Containerization of VO Applications

Please take a look:

- Docker containers: `https://hub.docker.com/u/mmpcn`
- GitHub `https://github.com/mmpcn/VOContainers`

Thank you!