How to build and test a Docker container for your ICCMA19 solver

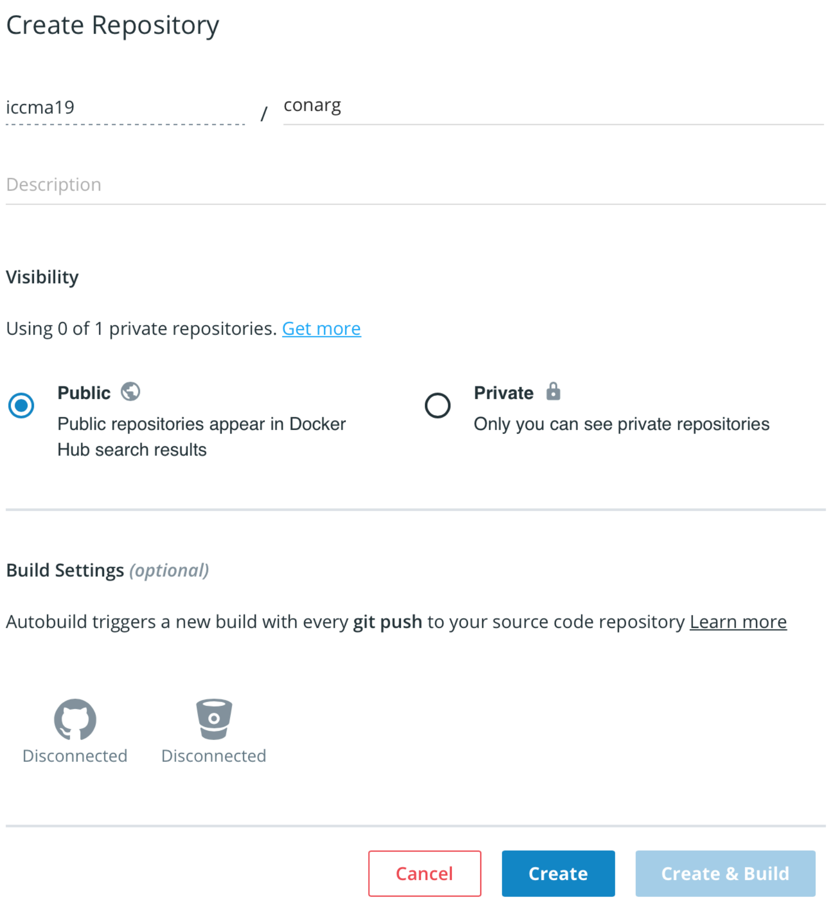
Install and run Docker

This document is a step-to-step guide for packaging your solver to be submitted to International Competition on Computational Models of Argumentation (ICCMA 2019) into a Docker container.

First, create a Docker Cloud account here: https://cloud.docker.com
In the following of this manual we consider as DOCKER_ID the name *iccma19*.

Then, sign in and

1. Click on "Create Repository".
2. Choose a name for the Docker repository of your solver (e.g., "YOUR_SOLVER_REPOSITORY") and a description for your repository, select "public", and then click on "Create". In the following, we suppose the chosen solver name is *conarg*. See Figure 1.



Create Repository

iccma19 / conarg

Description

Visibility

Using 0 of 1 private repositories. Get more

● **Public** Public repositories appear in Docker Hub search results

○ **Private** Only you can see private repositories

Build Settings *(optional)*

Autobuild triggers a new build with every **git push** to your source code repository Learn more

Figure 1: create the repository for your solver.

Your repositories can also be accessed by signing in on Docker Hub: https://hub.docker.com/ (same login name and password). See Figure 2.

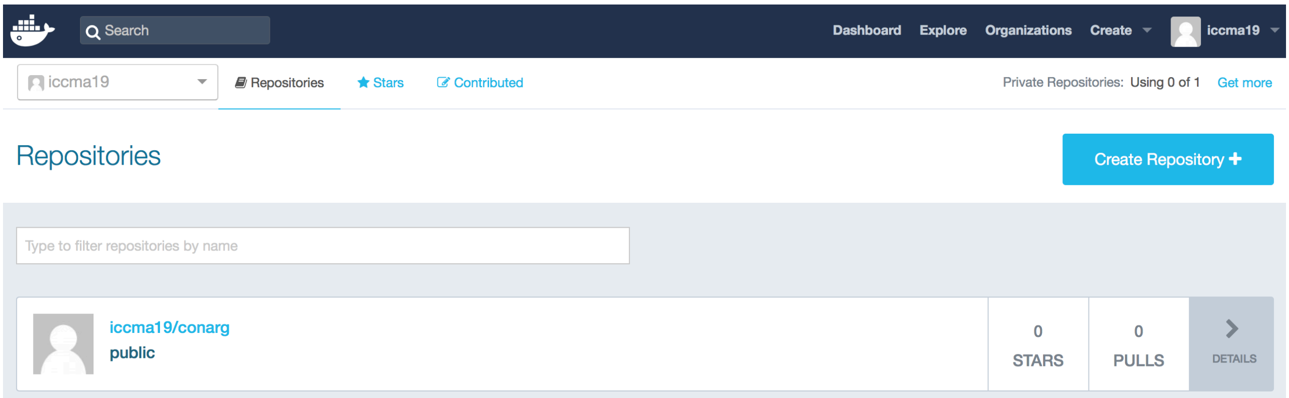


Figure 2: container view from https://hub.docker.com.

Then, install Docker on your machine. Please refer to the official installation Web page: https://docs.docker.com/install/. For instance:
Linux: https://docs.docker.com/install/linux/docker-ce/ubuntu/#install-docker-ce
 or https://linuxize.com/post/how-to-install-and-use-docker-on-ubuntu-18-04/ (for Ubuntu 18.4)
Windows: https://docs.docker.com/docker-for-windows/install/
Mac: https://docs.docker.com/docker-for-mac/install/

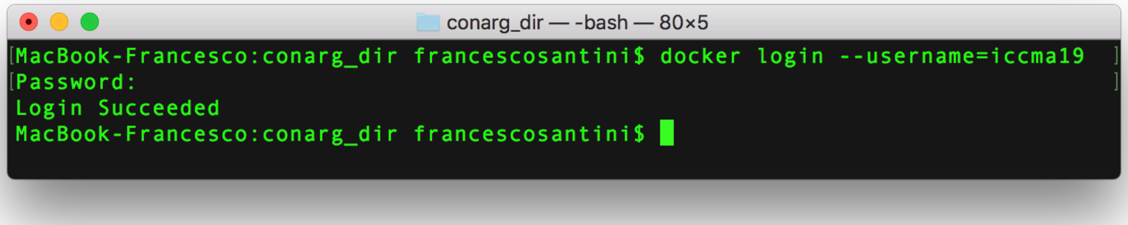
Once accomplished, open a terminal window on your machine and be sure the Docker demon is running. For example, run the *hello-world* container (not that the all the following docker commands may need to be run with *sudo* before them):

docker run hello-world

Then from terminal login to your Docker account by typing:

docker login --username=DOCKER_ID

Where DOCKER_ID is the name of your Docker account (*iccma19* in this running example). You will be also required to type your Docker password. The following screenshot shows this command in the terminal.



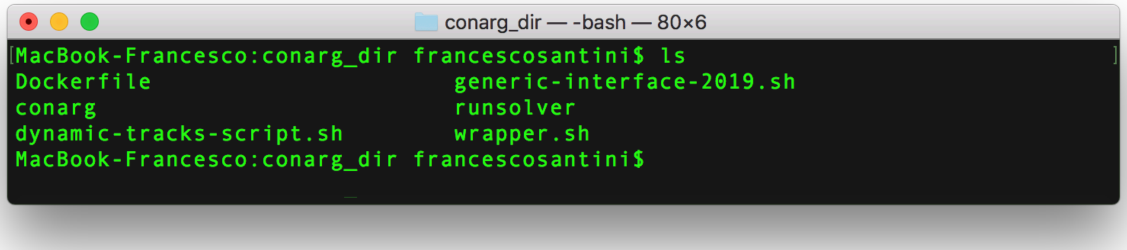
```
[MacBook-Francesco:conarg_dir francescosantini$ docker login --username=iccma19 ]
[Password:                                                                        ]
 Login Succeeded
 MacBook-Francesco:conarg_dir francescosantini$ 
```

Solver Dockerization

In this section we describe how to create a Docker container with your solver. First, create a directory “SOLVER_DIR”, somewhere on your machine. This directory needs to contain at least:

1. All the files needed by your solver; we use “YOUR_SOLVER” as the name of the solver main executable.
2. The *runsolver* tool used to monitor the execution of your solver (see http://www.pragmaticsofsat.org/2011/presentations/slides-or.pdf).
3. A “wrapper.sh” shell script, which will use *runsolver* and call the script at bullet 4.
4. A “generic-interface-2019.sh” shell script, which needs to be adapted in order to implement the required input/output interface (for more details please check the document at http://iccma19.dmi.unipg.it/res/SolverRequirements.pdf).
   If your solver natively implements such an interface, the script at bullet 3 has to directly call your solver and not this script.
5. A file named “Dockerfile” (requiring Alpine Linux for running the solver, and defining *wrapper.sh*, bullet 3, as the entry-point of execution).

When you build an image by using this Dockerfile, the assembled package will contain a minimal distribution of Linux (Alpine Linux: https://alpinelinux.org), and all the files at bullets 1-5. If the solver is composed by several executables/files, add all of them to this directory. Please try to use Alpine Linux: if you use a different Linux distribution, e.g., Ubuntu, the final image size will considerably increase (from ~10 to ~80 GB). The following screenshot shows the minimal content of the “SOLVER_DIR” directory (*conarg_dir* in this example). As running example, we will build a container for *conarg*, which represents an instantiation of the “YOUR_SOLVER” string in this guide.



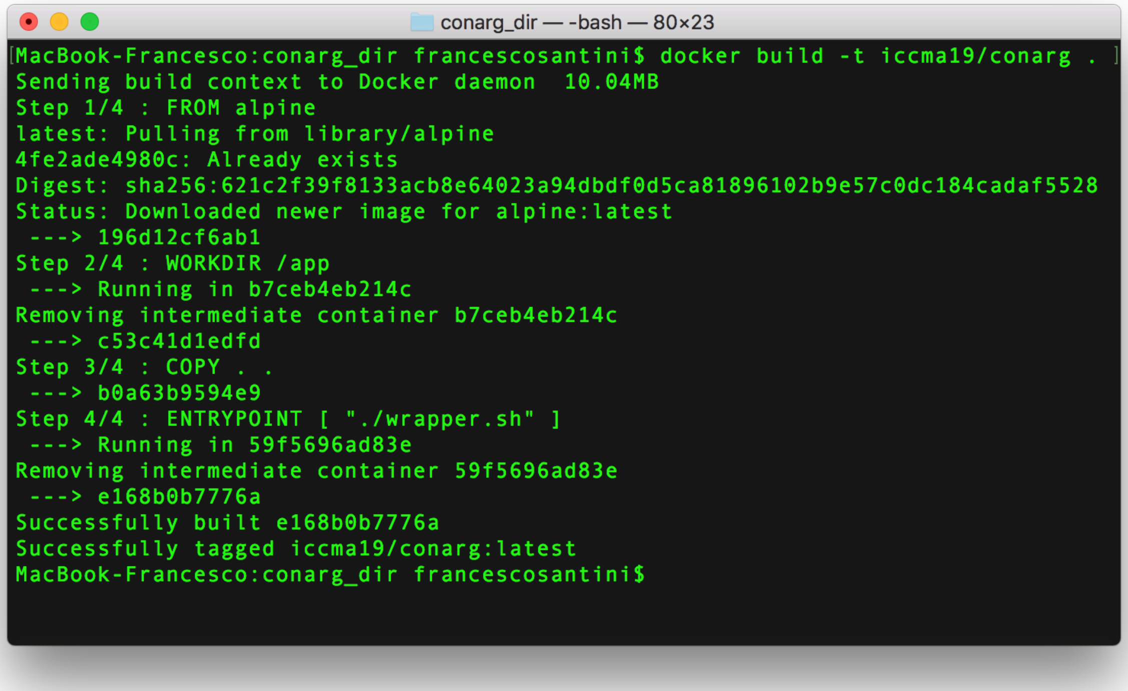
```
MacBook-Francesco:conarg_dir francescosantini$ ls
Dockerfile                      generic-interface-2019.sh
conarg                          runsolver
dynamic-tracks-script.sh        wrapper.sh
MacBook-Francesco:conarg_dir francescosantini$
```

Then, be sure to be inside “SOLVER_DIR”, and type

docker build -t DOCKER_ID/YOUR_SOLVER_REPOSITORY .

where YOUR_SOLVER_REPOSITORY is the name of the repository you have created in this previous section, and “.” (or alternatively “./”) is the current folder that contains all the files.

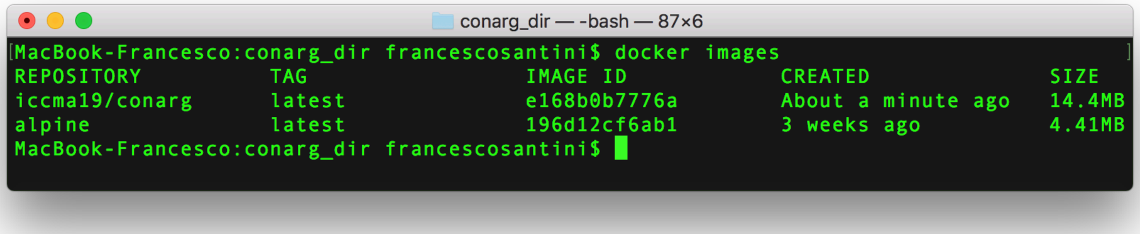
In this example, DOCKER_ID/YOUR_SOLVER_REPOSITORY will then correspond to *iccma19/conarg*. This command builds a Docker image containing everything is inside the current directory. The following screenshot shows what happens when this command is executed to build an image of the ConArg solver.



```
MacBook-Francesco:conarg_dir francescosantini$ docker build -t iccma19/conarg .
Sending build context to Docker daemon  10.04MB
Step 1/4 : FROM alpine
latest: Pulling from library/alpine
4fe2ade4980c: Already exists
Digest: sha256:621c2f39f8133acb8e64023a94dbdf0d5ca81896102b9e57c0dc184cadaf5528
Status: Downloaded newer image for alpine:latest
 ---> 196d12cf6ab1
Step 2/4 : WORKDIR /app
 ---> Running in b7ceb4eb214c
Removing intermediate container b7ceb4eb214c
 ---> c53c41d1edfd
Step 3/4 : COPY . .
 ---> b0a63b9594e9
Step 4/4 : ENTRYPOINT [ "./wrapper.sh" ]
 ---> Running in 59f5696ad83e
Removing intermediate container 59f5696ad83e
 ---> e168b0b7776a
Successfully built e168b0b7776a
Successfully tagged iccma19/conarg:latest
MacBook-Francesco:conarg_dir francescosantini$
```

Afterwards, check if the image "DOCKERID/YOUR_SOLVER_REPOSITORY" has been successfully created (the result for this running example is shown in the following screenshot.

docker images



```
MacBook-Francesco:conarg_dir francescosantini$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED              SIZE
iccma19/conarg      latest              e168b0b7776a        About a minute ago   14.4MB
alpine              latest              196d12cf6ab1        3 weeks ago          4.41MB
MacBook-Francesco:conarg_dir francescosantini$
```

In order to test if your dockerized solver works fine, you first need a second container storing some test-frameworks from ICCMA 2017. The container *iccma19/test_frameworks* stores two frameworks: admbuster_1000.apx and admbuster_1000.tgf.
Please type the following commands one after the other (respectively retrieving from a repository and then running this second container):

docker pull iccma19/test_frameworks

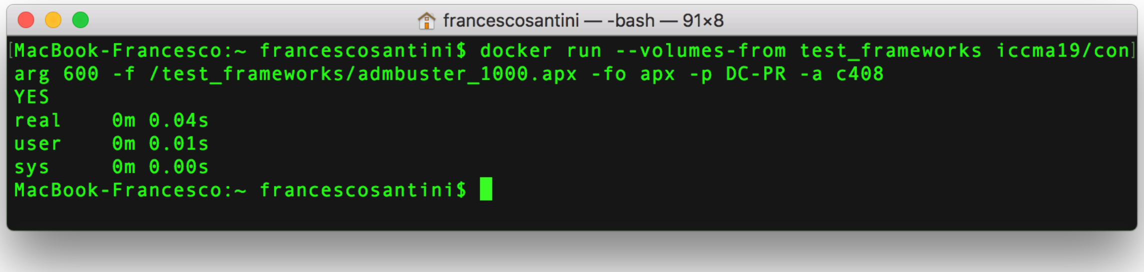
docker run -d --name test_frameworks iccma19/test_frameworks

These commands pull a new container and run it in background (-d option), with name *test_frameworks*.

Now it is possible to launch your dockerized solver on one of the framework instances in *test_frameworks*; use, for instance, the command

docker run --volumes-from test_frameworks DOCKER_ID/YOUR_SOLVER_REPOSITORY 600 -f /test_frameworks/admbuster_1000.apx -fo apx -p DC-PR -a c408

to check the credulous acceptance of argument *c408* with the preferred semantics on file *admbuster_1000.apx*. The first parameter after DOCKER_ID/YOUR_SOLVER_REPOSITORY has always to be the timeout in seconds (600 seconds in this example). The result is shown in the following screenshot.



```
[MacBook-Francesco:~ francescosantini$ docker run --volumes-from test_frameworks iccma19/con]
arg 600 -f /test_frameworks/admbuster_1000.apx -fo apx -p DC-PR -a c408
YES
real	0m 0.04s
user	0m 0.01s
sys	0m 0.00s
MacBook-Francesco:~ francescosantini$ 
```

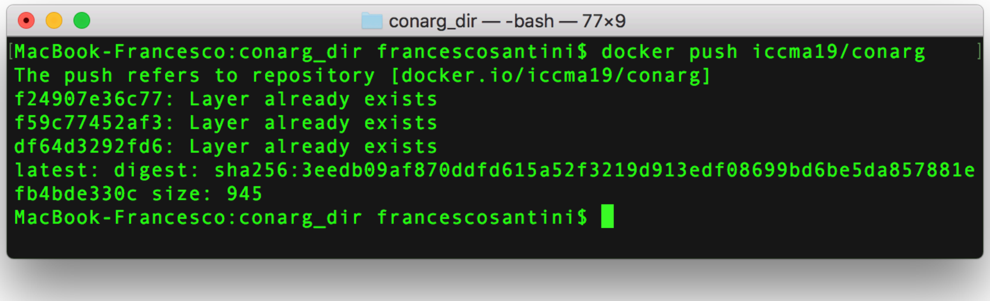
After the timeout in seconds, your solver can be executed by using a superset of the options used in ICCMA 2017 (adding –m is the only change):

- **-f fileinput** (the file storing the framework)
- **-m fileinput** (the file storing the modification on the file passed with –f, used in the dynamic track only)
- **-fo format** (apx or tgf)
- **-p problem** (EE-PR, DS-PR, etc.)
- **-a additional** (e.g., argument to be checked for credulous/skeptical acceptance)

Please refer to http://iccma19.dmi.unipg.it/SolverRequirements.pdf for detailed information on comments. The iccma19/test_frameworks image also contains two modification files (*admbuster_1000.apxm* and *admbuster_1000.tgfm*), in order to test also dynamic solvers.
Finally, you can push the image to your personal repository:

docker push DOCKER_ID/YOUR_SOLVER_REPOSITORY

The result is in the following screenshot:



```
[MacBook-Francesco:conarg_dir francescosantini$ docker push iccma19/conarg     ]
The push refers to repository [docker.io/iccma19/conarg]
f24907e36c77: Layer already exists
f59c77452af3: Layer already exists
df64d3292fd6: Layer already exists
latest: digest: sha256:3eedb09af870ddfd615a52f3219d913edf08699bd6be5da857881e
fb4bde330c size: 945
MacBook-Francesco:conarg_dir francescosantini$ 
```

The repository has been now updated also on Docker Hub https://hub.docker.com/, as Figure 3 shows.

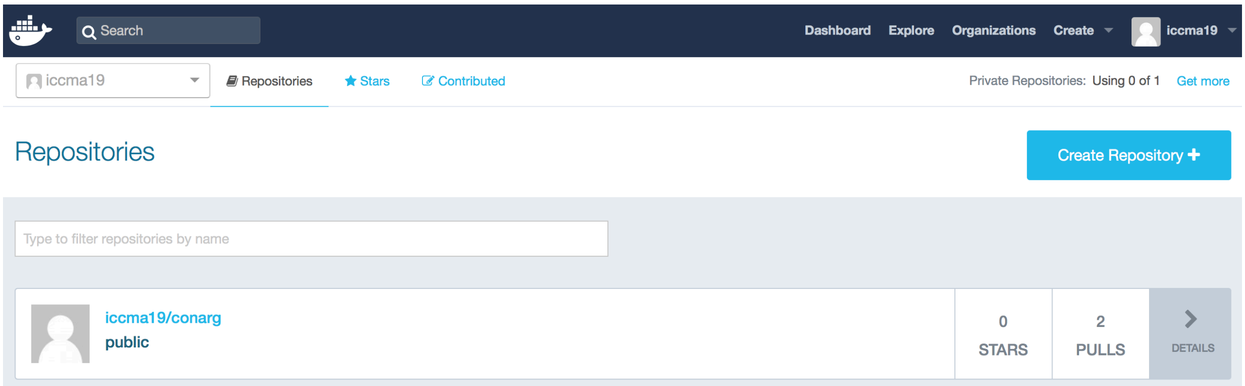
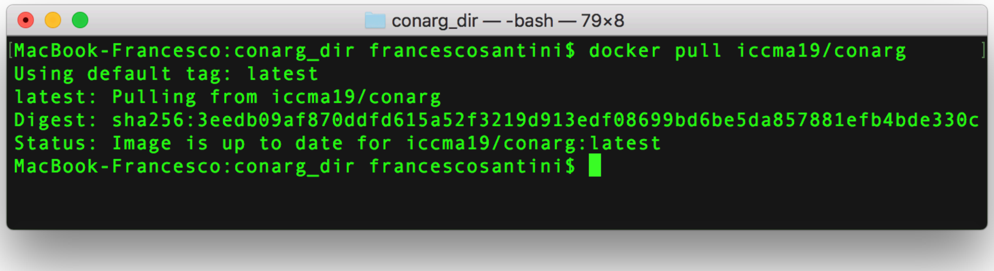


Figure 3: a new pull for this repository.

To pull it again from your repository, first login (e.g., *docker login --username=iccma19*), and then use the command

docker pull DOCKER_ID/YOUR_SOLVER_REPOSITORY



```
MacBook-Francesco:conarg_dir francescosantini$ docker pull iccma19/conarg
Using default tag: latest
latest: Pulling from iccma19/conarg
Digest: sha256:3eedb09af870ddfd615a52f3219d913edf08699bd6be5da857881efb4bde330c
Status: Image is up to date for iccma19/conarg:latest
MacBook-Francesco:conarg_dir francescosantini$
```

A link to a public repository, as *iccma19/conarg* in this example, is what the participants need to clearly state in their solver description (submitted through EasyChair), and represents mandatory information for a solver submission.

All the files used in this guide to dockerize *conarg* (i.e., *conarg_dir*) can be found at the following link:

- http://iccma19.dmi.unipg.it/add/conarg_dir.zip

The sample files used to create *test_frameworks* can be found at:

- http://iccma19.dmi.unipg.it/add/code/test_frameworks.zip

Further commands

We now report a couple of useful additional commands you might use to assemble your container. In case of any problem, please refer to the official documentation:

https://docs.docker.com/engine/reference/commandline/docker/#child-commands

The first one can be used to locally remove a Docker image (*fbff44780fae* is the image ID you can obtain with the *docker images* command, -f is a force flag):

```
docker rmi -f fbff44780fae
```

In order to list all the containers running on your machine, type:

```
docker ps
```

Or *docker ps –a* to get all the containers (also stopped ones). To remove one of such containers, the command is (*3355386d91cb* is the container ID you can obtain with the *docker ps* command):

```
docker rm 3355386d91cb
```

Finally, to stop the execution of the container with ID 3355386d91cb:

```
docker stop 3355386d91cb
```