Running GUI apps with Docker

12 Nov 2016, 10:24
Docker / X11

Already using Docker containers? How about run your GUI apps with it? That's right, i'm talking about running your browser, text editor, games, etc.

Why? X11 running at Docker seems complicated

That's quite easy (at least easier than most people think) and it gives you an astounding control over the application… Dont want the app to access (inter)net for some reason? Just use `--net null`. Maybe a isolated network? `--net networkname` (in this case you need to create the Docker Network before). Want to limit how much RAM it will use? `--memory mem` (eg: `--memory 1gb`). Want to choose which cpu cores will run the app? `--cpuset-cpus cpus` (eg: `--cpuset-cpus 0,3` or `--cpuset-cpus 0-3`). Anyway… Feel free to use your imagination. :) (yeah, i know that's possible to do that kind of stuff at linux… But it's not as easy as setting a flag at a command)

Another advantage is that program installs are just "running the app", when you run "`docker run [...] imagename`" Docker Engine verify if this image is available at local repository, and if not, download it and run.

And the application already comes with its dependencies at the container, and that means no "garbage" staying behind as orfan packages (and config files) when the app is removed for some reason. ;)

This experience beyond being fun, takes you to a total immersion with Docker operation, you simply can't use your tools in a daily basis without needing use various Docker resources.

Oh yeah, like i believe that everything is easy, simple and life turns into rainbows and unicorns… Tell me the truth

Ok, ok… Not everything is chocolate and cakes, mainly when you get to the point there's no image made to the app you need…

The dependencies listed by the developers aren't always the only needed… Which is acceptable if you really think about it, because they are usually already installed at your computer, but in containers case it doesn't turn out that way. This may end pretty frustrating some times, there's another moments that may frustrate you a little, but that's the most common to me (but when you find out what was the problem, the sensation is indescripable)

Ok, then if i only run apps that already have a image, i won't have problems at all?

Well, like i said before, the problems are **mainly** when you make a image, but not the only ones… Some integrations with another softwares are pretty hard to get to work (actually until now i didn't got any to work, but as "nothing is impossible"…), let's have an example: some softwares allow you to just "click a link" and open the page at your default browser, but when you have a container of, let's say "Telegram-desktop", there's no browser installed at the "machine" (most times not even the software that checks your default browser and runs it).

Nothing is 100% bullet-proof and it's almost certain that some problems will occur… And sometimes you'll think that the problem is exactly because you're running the software at a container (in my case it never was, but…).

I really don't like to jump at new stuff…

Well, that depends of what you think as new :p
Docker itself isn't exactly old (actually is pretty new… Just 3 years old), but many people already does use it at production (and says it have some challenges but it's worth the effort) and… (roll the drums)
…what i'm talking about started practically together with Docker first public release :)

Really?

Yes! here's the timeline i could get with my searches:

- March 2013: Docker 0.1 public release
- Abril 2013: First known X11 implementation at Docker (using vnc)
- July 2013: X11 implementation at Docker using ssh with X11-Forward
- September 2014: implementation using host's X11 socket
- Fevereiro 2015: Jessica Frazelle posts about running gui apps with Docker
- Abril 2015: Jessica Frazelle makes a talk at Container Camp 2015 showing that she runs (almost) everything at her computer with containers

Why the last ones were included? Well, basically because everytime someone speaks about GUI at Docker, or (s)he will talk about Jess our someone remembers her (there's other videos of her talking about it, but i thought it was better to include only the first one)
As you could see, only a month after Docker public release there were already people wanting to run GUI apps at Docker :)

Alright, you convinced me… Tell me more, how it works? How simple (or complicated) it will be to me?

Works basically like this, you mount a volume of the X11 socket file of the host (/tmp/.X11-unix) at the same path at the container and define the environment variable `DISPLAY` of the container as the display of the X server of the host, so when the app inside the container send the rendering instructions, it will be sending them to the host X server.

Erm… It kind of looks complicated to me, huh?

Relax… I was just explaining the theory behind the stuff, what we're gonna do is just use two flags: `-e DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix` and the "magic" is done. :)

Ok, let's get down to the commands?

Before everything… 99% of the cases you need to give access to X: `xhost local` (this access will be granted until you shutdown/reboot your host).

Simplest command, mounts X11 socket and defines the display (please note that we will "evolving" the command bit by bit, but you can use just the needed flags)

```
docker run [--rm [-it]|-d] \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-e DISPLAY \
imagem [cmd]
```

At some cases the `DISPLAY` variable have to be `DISPLAY=unix$DISPLAY` (to be honest i don't really know why, that's what was pointed out by who made the image and it didn't worked other way) *Obs: "`[--rm [-it]|-d]`" means "Optionally you can use `--rm` which may or not be with `-it` OR you can (or not :p) use `-d`*

3D hardware acceleration support:

```
docker run [--rm [-it]|-d] \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-e DISPLAY \
--device /dev/dri \
imagem [cmd]
```

Audio:

```
docker run [--rm [-it]|-d] \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-e DISPLAY \
--device /dev/dri \
--device /dev/snd \
imagem [cmd]
```

Webcam:

```
docker run [--rm [-it]|-d] \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-e DISPLAY \
--device /dev/dri \
--device /dev/snd \
--device /dev/video0 \
imagem [cmd]
```

Using same date/time as host:

```
docker run [--rm [-it]|-d] \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-e DISPLAY \
--device /dev/dri \
--device /dev/snd \
--device /dev/video0 \
-v /etc/localtime:/etc/localtime:ro \
imagem [cmd]
```

Please note: some distros don't use `/etc/localtime` to set the timezone, in those cases you'll need to check how it does it and "replicate" at the container.

Keeping app configs:

```
docker run [--rm [-it]|-d] \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-e DISPLAY \
--device /dev/dri \
--device /dev/snd \
--device /dev/video0 \
-v /etc/localtime:/etc/localtime:ro \
-v $HOME/.config/app:/root/.config/app \
imagem [cmd]
```

Obs: the path is just an example :p

Bonus: Video-game controller :) (actually, any input device)

```
docker run [--rm [-it]|-d] \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-e DISPLAY \
--device /dev/dri \
--device /dev/snd \
--device /dev/video0 \
```

```
-v /etc/localtime:/etc/localtime:ro \
-v $HOME/.config/app:/root/.config/app \
--device /dev/input \
imagem [cmd]
```

Right, i think i can start with that… Any tips?

- Sort the flags alphabetically (makes troubleshootings easier) - *yeah, i know that at the examples i wasn't following my own tip, but that was to make easier to recognize the flag added*
- Make a script with the commands that will run the containers as functions, that way you can "include" the script to your bash profile and run the it as a "normal" application command (and make "use launchers" easier) - *i'll link the script that i use below*
- Use detach (`--detach` or `-d`) and a function to remove the container that is stopped when running the function again - *that will be at my script too ;)*

Nice! Anything else?

Here's a "bonus": https://github.com/somatorio/dockercompose-sample-workbench
If we can run apps at containers, how about run it through Docker Compose?
That idea is from @gomex, not mine… He talked about it at a conversation about GUI at Docker at a meetup (if you look at the meetup date you'll see that was a some time ago… my bad :p) and i just "executed" (read that as: "tested and wrote about") it.

And here's my humble script - highly based on Jess dockerfunc script (nope, definitely not just a coincidence)

OH, some friends of mine use Mac and Windows, can they do that too?

Sure, they'll just need some hacks (not that what we did until now wasn't hacks :p)

- Mac OS X

Install Docker for Mac

```
brew install socat
brew cask install xquartz
open -a XQuartz

socat TCP-LISTEN:6000,reuseaddr,fork UNIX-CLIENT:\"$DISPLAY\"
docker run -e DISPLAY=hostip:0 [...] image OR DISPLAY=hostip:0 docker-compose u
```

- Windows

Install xming
Install Docker for Windows

```
xming :0 -ac -clipboard -multiwindow
docker run -e DISPLAY=hostip:0 [...] image OR DISPLAY=hostip:0 docker-compose u
```

Everything works like a charm?

Unfortunately not everything (at least until now…), sound doesn't work for exemple and as Docker for Mac/Windows runs with VM, you may expect a little performance loss…

So, it's not a good idea to tell them about it?

Hey, i was just telling you what they can expect, not to avoid it :) As far as i tested it, it just affects games and software that heavily uses hardware.
They can even find new ways to solve the issues i told you about… neat, huh?

Para editar este post basta clicar aqui.