Quick Start Guide to Using the Companions Docker Container

Install Docker

If you want to run a Docker container, you need to install Docker. The free versions of Docker are fine for our purposes – Docker Desktop for Windows and Macs (https://www.docker.com/get-started), Docker Community Edition for Linux (https://hub.docker.com/search?q=&type=edition&offering=community&operating_system=linux).

Once you've installed it, you can confirm it's running by opening a command shell and running `docker version`.

Please note that the Companion image includes a large knowledge base, so it is around 4 GB in size. You will want to ensure that the disk on which Docker stores container images has at least that much free space. Different implementations of Docker store container images in different places but they usually give a way to change this location, either through a graphical user-interface or by editing a config file.

Running the Companions Container

You have been provided with scripts named **run-companions.sh** (for the Bash shell) and **run-companions.bat** (for Windows). Normally you will want to map some of the Companions directories to your local file system so that you can more easily view or change those files. So you first need to edit the run-companions script so that you can set the directory locations:

- LOGS_PATH = location on your local computer of agent logs, error logs, reports, and kiosk logs.
- PATCHES_PATH = location of patches you might install for Companions.
- FLAT_FILES_PATH = contains knowledge-base flat-files that you might edit or augment.
- SKETCH_PATH = WebSketch is included with this version of Companions. WebSketch serves sketches from this directory. The distribution includes various sample sketches.
- KIOSK_QUESTIONS_PATH = training question files for the Kiosk are located here.
- KB_PATH = Companions includes the NextKB knowledge base internally. If you put any files in the directory specified by KB_PATH, they will be used as the knowledge base instead. This allows you to swap in alternate KBs. Make sure this location has room for the knowledge base (at least 4 GB).
- PYTHONIAN_CODE = You can add your own Python code to this directory and make use of it with the Pythonian Companions Agent.

If you don't want to set any of these paths, delete the lines that use those paths in the call to **docker run**. Each of these lines begins with **-v**, which is a Docker command argument for mapping a volume.

After you've edited the script, run it to start the Companions container. This will open a web server at port 9100. Point your browser to http://localhost:9100/smgr.html to start using Companions.

If you have a firewall blocking local connections to port 9100, you'll have to unblock that port.

Note that after starting the container, your command shell's stdout and stdin are bound to those of the container. This makes it easier to see the various Companions status messages. If you want to detach from that and return to your normal command prompt but keep Companions running, press **ctrl-P** then **crtl-Q**. If you want to re-attach to the Companions container, call `docker attach companions`.

When you end your Companions session, the container is terminated. To start a new session, you'll have to call the run-companions script again.

Pulling a New Version of the Companions Container Image

The **run-companions** script discussed in the previous section pulls the Companions container image from our repository if it doesn't already exist on your computer. This does not, however, check to see if a new version has been released in the repository. So you'll have to pull the image yourself if you want the new version. The **pull-companions.{sh|bat}** script does that. Just run the script and it will pull the new image to your local computer.

Using Pythonian with the Companions Container

- The Companions Docker container includes the Python-based Pythonian Companions Agent. To use this agent, you'll use the same **run-companions.{sh|bat}** script discussed in the previous section to start Companions. Be sure to specify a path for **PYTHONIAN_CODE**. You will put your python code in this directory on your local computer and it will be mounted in the container as the **/code** directory.
- Run the **run-companions.{bat|sh}** script.
- Open http://localhost:9100/smgr.html in your web browser and start a session.
- The run-companions.{bat|sh} script will leave your command/shell attached to the stdin and stdout streams of the Docker container. Either use **ctrl-P**, **ctrl-Q** to detach from that session and return to local command prompt or start a new command/shell window. Then call `docker exec -it companions python3 /app/companions/pythonian/<yourAgent.py>` to run your python agent
  OR
  `docker exec -it companions python3` to get a python REPL
  OR
  `docker exec -it companions bash` to run the bash shell. From there you can change to the /app/companions/pythonian directory and run your python code. Note that the container user has the following bash aliases defined:
    - python = python3
    - pip = pip3
- If you ever want to re-attach to the stdin and stdout of a running Companions container, use `docker attach companions`.

Configuration Parameters

The Companions container can also be configured using the following environment variables.

- COMPANIONS_HTTP_PORT <integer> = The port on which the HTTP server will listen. This is the port from which the WebUI and RBrowse pages are served. [default = 9100]
- COMPANIONS_FACILITATOR_PORT <integer> = The port on which the facilitator will listen. [default = 9000]
- COMPANIONS_SESSION_DOMAIN <case-sensitive string> = When specified, Companions will start a session automatically upon launch with the specified domain (e.g. *Interaction-Manager*).
- COMPANIONS_SESSION_USER <case-sensitive string> = The username that will be used for the initial session if a session domain was specified.
- COMPANIONS_EXIT_ON_SESSION_END <TRUE|FALSE> = This controls whether or not the Companions executable will exit after a session has been ended. (default = TRUE)

If you are starting the container using *docker run*, you can use -e to specify environment variable values. For example:

```
docker run -it -p 9100:9100 -e COMPANIONS_SESSION_DOMAIN=Interaction-Manager
companions
```

Container environment values can also be specified in Kubernetes and Docker Compose YAML files.