Nouvelle conversation

Mes groupes
Groupes récents
Groupes favoris

Conversations suivies

xnat_discussion

Conversations 99+
Libellés
À propos de :
Mes paramètres d'adhésion

à xnat_di...@googlegroups.com

Yes, it is not just theoretically possible but practically quite possible to do this. It requires a decent familiarity with the REST API, but for the most part doing this through the UI is just using the UI as a proxy to the REST API. In fact, using scripts calling the REST API to launch containers is a very common operation.

Exactly how you'd do this depends a lot on the type and structure of your data and your use case(s), so it's hard to describe a general process for this, but if you have a container that runs at, e.g., the experiment level, you can take the API call(s) for launching the container, parameterize so that you can pass experiment-specific values for each call, then iterate all of the experiments in your project, calling your REST API with the appropriate values.

Containers do not have direct access to the database by default. You *could* do this by passing credentials to the container (or using some sort of secret store like Vault or Keycloak), but generally it's probably not a great idea: outside of the security concerns there's also a non-trivial risk of doing something to the database that confuses XNAT and/or breaks the database altogether. If there are some operations that you really require, I'd probably recommend writing a plugin with its own API endpoints for querying and updating XNAT through the internal API and have your scripts/processing in the container use those instead (not to mention that, depending on where the containers are running, they may not even be able to access the database, e.g. security restrictions on PostgreSQL usually whitelist IP addresses that can access the database remotely, which would completely break containers running under Docker Swarm or Kubernetes).

This approach would cause no mounting or storage space issues outside of the standard mounting or storage space issues that you can run into with the container service, simply because this is how the container service always works (i.e. through REST calls). The input and output mounts are usually parameters for the container launch, so if it works running from the UI it'll work running from your script. If your container generates a great deal of data (e.g. Freesurfer can produce many GBs of output from a few MBs of input data) you'll need a corresponding large amount of storage on which to store that data.

One thing to be *very* aware of is storing session state with whatever tool you're using for scripting. XNAT's underlying application server is Tomcat, which has a default session limit of 1K. That means that there can only be 1K worth of active sessions on the server at one time, after which new session requests are denied until an existing session is terminated or times out. A common mistake people make when scripting is doing something like this:

```
curl -u admin:admin http://server/data/projects?format=xml
curl -u admin:admin http://server/data/projects/Project_A/subjects?format=xml
curl -u admin:admin http://server/data/projects/Project_A/subjects/Subject_1/experiments?format=xml
```

Each of those calls creates a new session, each of which will persist for however long the session timeout is configured on the system. A script that's going through 1,000 experiments without storing session state will make the server inaccessible to other users. How you store session state depends on the tool, e.g. for curl you can do:

```
curl --user admin:admin --cookie-jar cookies.txt --cookie cookies.txt http://server/data/projects?format=xml
```

Other tools like xnatpy or httpie have their own means of persisting session state, so make sure you know how to use that and use it!