Extractor / Converter Development Story Boards

This story board concentrates on the way a developer would contribute an extractor or a format converter to the Brown Dog service. It includes setup of the development environment, testing, packaging/description and submission.

I am leaning more toward having the developer begin from the template project, i.e. an empty or bare-bones extractor or converter. We can include all the docker files that are necessary for the runtime environment with the template project, making it easy install/start/stop whatever is needed via docker-compose. See the Docker Compose workflow: https://docs.docker.com/compose/gettingstarted/#step-2-create-a-docker-image

Question: Is there any way to make the docker container describe itself in more detail and maybe include some sample files, such that a single upload will capture everything we need for submission of the contribution?

Overview of Steps

1. Developer visits the contributor web page ~~BD-956~~ - Consolidate all contributor instructions on one set of pages DONE
    a. Contributor page explains at a high level how they can develop and contribute a two types of tools to the Brown Dog catalog.
    b. Explains benefits of contributing
    c. Explains intellectual property around contributing
    d. Examples and links to contributions by others
    e. Tools catalogue page has some information on this.
    f. Provide a link to the contributor web page from Tools Catalogue and Brown Dog web pages.
2. Developer visits either extractor or converter how-to page:
    a. Explains where their module fits within system diagram
    b. Explains development instructions step-by-step
        i. how to modify the template
        ii. install path for any dependencies and source code
        iii. input file description
        iv. expected output
        v. how to upload the script to opensource repository.
    c. Provides links to resources they need
    d. ~~BD-957~~ - Create extractor how-to page DONE
    e. ~~BD-958~~ - Create converter how-to page DONE
3. Developer installs prerequisites
    a. VirtualBox or other VM kit
    b. Development VM (if they need a foreign OS)
    c. Docker
    d. Python packages: python, python-pip
4. Developer downloads a project template with all of the basics for an extractor or converter project.
    a. This is an empty project without functionality or dependencies, equivalent to "hello world". (What does exactly Project Template mean? I am assuming that it means a sample for writing extractor/converter- Smruti Padhy)
    b. Template project is documented, with placeholders where contributed code might go.
    c. Might include template files for tools catalog info (A tool metadata file that would be posted upon submission)
    d. ~~BD-959~~ - A template project(s) for converters DONE
        i. Converter Scripts within Polyglot can be found here (https://opensource.ncsa.illinois.edu/bitbucket/projects/POL/repos/polyglot/browse/scripts). Use these as reference to write the template (point (e)).
        ii. May need a template project per target OS?
    e. ~~BD-960~~ - Create standalone extractor template project DONE
        i. Sample-extractors can be found in the pyClowder Library repo (https://opensource.ncsa.illinois.edu/bitbucket/projects/CATS/repos/pyclowder/browse) . Use this as reference to write the template extractor (point (d)).
    f. I think, there are two approaches to this: developer downloads example template such as imagemagick to run example extractor, see its working, study the code and later customizes it. But this will need the dependencies for that extractor which the developer may not need in future with the customization. So, the other way is to give just the skeleton of the extractor and the developer can install dependencies and write code. – Smruti Padhy
        i. I agree, I think the empty template is more useful. We can document it thoroughly and throw in some dummy extracted data ("hello world"), without introducing more dependencies.
5. Developer writes and tests code within template project.
    a. This involves edits to python or customization of a VM image (converters)
    b. Involves edits to the supplied Dockerfile for new dependencies, etc..
        i. For extractors this should contain only the tool, its dependencies, and the extractor
        ii. For converters this should contain the basic Polyglot setup, ideally only the Software Server, the tool, its dependencies, and the converter interface script
            - I have not done a converter, they are packaged as VMs right? How can they be contributed/distributed/packaged? - Gregory Jansen
            - Rob has created two docker images for pecan and imagemagick (https://opensource.ncsa.illinois.edu/bitbucket/projects/BD/repos/dockerfiles/browse/polyglot). Already script is there to set up the environment.
    c. Involves docker-compose up for testing
    d. ~~BD-961~~ - Automated testing based on sample files and their expected output DONE
        i. Developer could edit a template test script for this.
        ii. Tests could be run on catalog submission as a validation step.

6. Developer add sample data files to the sample data folder.
7. Developer edits tool info file (for tools catalog entry)
8. Developer runs Brown Dog contributor script, which submits the key data in the project BD-962 - Publish script for Tools Catalog TO DO
    a. Uses their Brown Dog account
    b. Includes interface script(s), dockerfile, sample input, sample output
    c. If you are submitting a dockerfile file, it should contain the interface scripts.

Notes

- Extractor example project: imagemagick edge detector
- Converter example project: imagemagick jpg to ico converter
- TODO: Create an extractor template written in Java, C++

Extractor How-To Details

1. Create a local project from the template:

```
git clone http://bitbucket.ncsa.illinois.edu/path/to/extractor-template.git myextractor
```

   Note: If the user downloads the docker-extractor bare image, it is not required to git clone the template project. The docker-extractor image should have that extractor template project.
2. Developer creates and activates a python virtual environment for their new project.
    a.
```
virtualenv myextractor
source myextractor/bin/activate
pip install docker-compose
```
3. Developer runs "docker-compose up" to start runtime environment of docker containers (TODO: docker-compose.yml for extractor runtime environment) (TODO: docker-compose.yml for converter runtime environment) (TODO: Create any missing Docker containers)
    a.
```
cd myextractor
docker-compose up
./run-tests.py (or something like that)
```
    b. Runtime containers for extractors: Clowder instance
    c. Runtime containers for converters: Polyglot instance
4. Alternatively, to Step (3-4), Developer looks for docker-extractor or docker-converter to start with and for writing interface script for new tools.
    a. Example docker extractors and converters are there in (https://opensource.ncsa.illinois.edu/bitbucket/projects/BD/repos/dockerfiles/browse)
    b. TODO: Create an repository for a template docker-extractor and a template docker-converter.
    c. While creating basic docker image for extractor, the template project should consists of interface scripts with comments where to modify the code (Follow Step 3-4, within a docker container).
    d. upload the image to dockerhub and create dockerfile and README and upload to the repository.
    e. Include these template docker-images in the docker-compose.yml file for setting up the runtime-environment for developer to enable start writing custom scripts.

TODO: Test the deployment based on Rob's script.