Clowder, extractors and docker

I was talking with Luigi a bit and want to redo how we are using the entry point and configuration aspects of both clowder and extractors. This stems from the docker-compose file we have in the clowder repository and the fact that eventually we would like to run everything as docker.

I would like to move us closer to how the new docker-compose format (v3) as well as docker and kubernetes are doing things. Both try to use a lot more network isolation, where we can create different networks for each stack. For example the clowder stack would use the clowder network, and the pecan stack would use the pecan network. The nice thing is that we can now have a machine called rabbitmq in each network, but there is no confusion which machine you connect to (the one in same network as you).

Another thing we should really start to leverage is making our entrypoints simpler. Right now the entrypoint does a bunch of magic to make sure the right service is up and running (again everybody waits for rabbitmq to be up and running). Simpler would be for the container to start, try to connect and if this fails it will crash and restart. This can be done by adding the "restart: unless-stopped" to the compose file.

Finally I would like for the compose file to be ready for either docker swarm or kubernetes. With the newer versions of docker you can now start a full stack in swarm or kubernetes using docker stack deploy. This will deploy the full stack just like docker compose. Nice thing is that this adds an extra set of instructions to the docker compose file as part of deploy section. Here we can control where the containers run, restart policy as well as how many we need to run. This allows us to easily spin up a clowder instance with multiple image extractors deployed.

As for the configuration aspect I would like to move away of overwriting the configuration files like we do now in the clowder entrypoint. I think we should deploy a standard custom.conf and play.plugins as part of the docker build process. You can overwrite these by mounting in the right configuration file. This is what a lot of other programs do as well. So we would have clowder docker be preconfigured to connect to amqp://guest:guest@rabbitmq:5672/%2F, no need to specify this. If you want clowder to connect to a different rabbitmq server you will need to create a conf file and mount that into the clowder container. Same goes for the plugins (or how this works in clowder 2.0).

Clowder

Clowder will now be preconfigured with a custom.conf and play.plugins that will be configured with all the options needed to run in docker-compose (i.e. elasticsearch, mongo and rabbitmq). It will also assume that the machines that run these external services are named exactly the same. Clowder will just start, and if it can not connect to mongo it will stop (it continue to try and connect to elasticsearch and rabbitmq). To disable/enable any configuration options you can mount a folder in as /home/clowder/custom which will contain the custom.conf, play.plugins and any other customizations.

Extractors

Extractors will have pre-configured the RABBITMQ_URI to be "amqp://guest:guest@rabbitmq:5672/%2". You can override this by setting the environment variable or calling the extractor with the right command line arguments. The entrypoint will not check to see if rabbitmq is up and running, however will simply try to connect and fail if it can't. It is up to the user to either launch the extractor after rabbitmq finishes starting up, or use the right restart policy to start the extractor. The dockerfile can now use CMD ["python", "$MAIN_SCRIPT"]. When starting the extractor you will need to make sure it is started in the same network as rabbitmq (using the --network flag for docker run)