Polyglot Refactoring for Dockerizing SSes

For BD-1007: Polyglot refactoring for dockerizing SSes, mainly in the POL-SS communication part.

Current Polyglot-SS communication architecture:

In the existing architecure, both Polyglot and SS are servers. Polyglot-SS communication mainly occurs at SS registration, SS heartbeat, SS checking in converted files, and Polyglot redirecting file-getting requests to a SS.

1. SS Registration and heartbeat.
   Refer to the development notes at the "Polyglot / SoftwareServer Documentation" page, POL goes to RabbitMQ, gets the consumer IPs as the SSes, connects to the URL "<ip>:8182/applications". If the URL is accessible and contains valid content, Polyglot adds the IP to its server list.

   SS supports many REST endpoints. PolyglotStewardAMQ uses "/alive" and "/applications" for SS registration and "/alive" for heartbeat:
   - /alive
     This returns a timestamp (a long integer) to indicate the start time of the SS. (long int, plain text)
     Used for adding a SS in discoveryAMQ() if POL gets a valid long int, and removing a SS in heartbeat() if POL does not get a valid long int from the ep.
   - /applications
     This returns a JSON array, such as "[{"alias":"daffodil","conversions":[{"inputs":["csv"],"outputs":["xml"]},{"inputs":["pgm"]," outputs":["xml"]}]},{"alias":"flac","conversions":[{"inputs":["aif","aiff","fla","flac","wav"],"outputs":["aif","aiff","fla","flac","wav"]}]}]".
     Used for adding a SS and updating the IO graph in discoveryAMQ().
2. SS checkin.
   a. SS check in the result to Polyglot (<url1>).
   b. Polyglot updates Mongo, if all steps done, creates a file "<filename1>.url" with the 2nd line as "URL=<url1>", such as "URL=http://141.142.210.2:8182/file/81_browndog.jpg".
   c. When a user accesses http://pol1:8184/file/filename1, POL redirects the request to <url1>.

Proposed design:

1. SS registration with polyglot.
   - Add 1 exchange/queue for SS registration with polyglot.
     Set its TTL to say 5 secs, so msgs are discarded in such time.
   - SS publishes msgs to this queue, each msg contains both start_time and conversion capabilities.
   - Polyglot listens to the queue, adds/updates its SS list and updates IO-graph accordingly. Adds a "latest heartbeat time" for each SS.
2. In "checkin", SS uploads the converted file – instead of sending only the filename. Polyglot saves it, instead of creating a "<file>.url" file.
3. SS heartbeat with polyglot.
   - Periodically Polyglot removes SSs that have expired "latest heartbeat time" from its SS list, and updates IO-graph accordingly. No need to process RabbitMQ msg in this thread.

What need to be added or changed:

1. POL now adds SSes in discoveryAMQ(), removes (non-responsive) SSes in heartbeat(). Change to: POL listens to a new queue, adds SSes and updates timestamps in discovery thread, and removes SSes with expired timestamps in heartbeat thread. NO querying of SSes.
2. In SS: add code to publish conversion capability msgs to a new queue in RabbitMQ for POLs to listen to for SS registration and heartbeat.
3. In SS: add code to upload converted files, in polyglot: add code to save them;
4. In PolyglotStewardAMQ: remove the code that creates "<file1.url>".

With this design, a SS can keep its REST endpoints, but POL does not query them. So it does not matter whether multiple SS containers run on the same VM or different VMs.

Notes:

1. use Java ConcurrentSkipListMap for the server_map in PolyglotStewardAMQ discoveryAMQ() for multi-threaded access.
2. The following Polyglot REST endpoints in PolyglotRestlet.java accesses/redirects to SS.

- /file/<file1>
- servers/<server_ip1>[/...]
- software/<sw1>[/...]