Project: Youtube Summarizer Server Microservice Deployment

Arnold Doray - 6 Jan 2024

Discussion

- Many microservices = headache to startup.
- How to easily startup microservices.

Starting Microservices

- Microservices often need to be started up in the right order, because they depend on each other.
- We can't easily use Bash because often times the libraries need to be downloaded. It is hard to do this in Bash.
- Also, Bash isn't good for distributed deployments.
- The task of getting your microservices started is called "deployment".
- arnold/shell has a few powerful words to help you deploy your microservices.
- Our goal is to keep the settings and startup conditions of your application in a single file.
- This makes deployment easy— just one command to run, (for now, if it's on a single machine, but we are working to expand this to multi-machine).
- It also means the deployment program is a one-stop-shop where all knowledge of the deployment is stored.

*/shell words

- EXEC — runs a bash command
- RUN — runs a smojo microservice, taking care to wait until it is fully compiled and running.
- ECHO — outputs text to the console
- All these words are substitution-enabled.
- Eg you can define any substitution word (— “s”) and call it in the command using #{mysub}
- Lastly, USER sets the username, and can be accessed through #{user} substitution.
- Insert these commands in words, as you see fit.

Example 1: Starting 2 services

- USER arnold
- ECHO ==== Starting Microservices ====
- EXEC pkill -f #{user}/ytsum
- EXEC mkdir -p ./logs/
- ECHO [Starting User Model]
- RUN #{user}/ytsum/users > ./logs/users.log
- ECHO [Starting Main Server]
- RUN #{user}/ytsum > ./logs/server.log

Example 2: Using substitution

```
0 variable TIME
: timer ( — ) now TIME ! ;
: elapsed ( — “s” )
        now TIME @ - 1000 / “ sec” concat ;

main
    timer
    ECHO This many seconds elapsed: #{elapsed}
;
```

Some Tips…

- Use hierarchical names (eg arnold/ytsum/user) so they can be easily pkilled using the common prefix.

- Always log output from your microservices to aid debugging. This is can really save your day!
- Put all your log files in the same place, eg ./logs/

- Use >> to append to a log file. Use this by default.
- Use > to overwrite a log file on restart. Only use this if you are absolutely sure you won't keep the logged output.

- Turn commonly used paths into substitutions. Eg #{logs}/queue.log rather than ./logs/queue.log
- Of course, you need to define : logs (— "s") …. ;

Homework

- Run ./smojo.sh -r arnold/queue/test to see a simple deployment script in action.
- In the listing below, create the timer, elapsed, logs and queue words
- Once this is tested out and works, refactor your main, so that it calls other words that build up your microservices.
- Eg, write an init-paths (—) word that runs the EXEC mkdir -p … commands, and use in your main.
- Write a complete deployment script for ytsum.

```
: main
    timer
    USER arnold
    ECHO ===== STARTING SERVER MICROSERVICES =====
    \ Stop all SDF processes.
    EXEC pkill -f #{user}/sdf/
    \ Make the "logs" folder if it does not already exist.
    EXEC mkdir -p #{logs}
    EXEC mkdir -p #{queue}
    \ Start the main Message Queue
    ECHO [Starting MQ Server]
    RUN #{user}/sdf/queue 4041 #{queue} > #{logs}/queue.log
    \ Start the test server.
    ECHO [Starting Test Server]
    RUN #{user}/sdf/queue/test 8080 4041 > #{logs}/test.log
    \ Start 2 test workers
    ECHO [Starting Test Worker #1]
    RUN #{user}/sdf/worker/poll/test "localhost" 4041 > #{logs}/worker-1.log
    ECHO [Starting Test Worker #2]
    RUN #{user}/sdf/worker/poll/test "localhost" 4041 > #{logs}/worker-2.log
    ECHO ===== DONE (#{elapsed}) =====
    \ Display the running processes.
    EXEC ps -x
;
```