

Orquestrador de `jobs`

Computação em Nuvem

Igor Montagner

O projeto deste ano da disciplina de Cloud Computing será criar um orquestrador que gerencie a execução de `jobs`. Cada `job` é um script que será rodado no serviço *Amazon Lambda*, recebendo uma entrada passada pelo usuário e devolvendo como saída um valor de retorno e uma string contendo a saída desse script no terminal.

Seu orquestrador irá receber requisições de execução de scripts enviados por vários usuários (identificados por um `uid>=0`), executá-los e receber requisições de status de cada `job` e também listar todos os `jobs` enviados por um usuário. Isso será feito via interface *REST*, com o desenvolvimento opcional de um cliente de linha de comando.

Especificações do serviço REST

Seu serviço REST deverá seguir as seguintes especificações.

- `/(jobs)?` (GET) - devolve um JSON contendo uma lista de jobs seguindo o padrão abaixo. Um `job` deverá retornar `status = ERROR` quando a sua execução tiver valor de retorno diferente de 0. Um `job` está em `status = WAITING` quando ele já foi recebido por seu serviço mas a infraestrutura necessária para rodá-lo ainda não está pronta.
[{ "job_id": int, "uid": int, "status": ["DONE" | "ERROR" | "WAITING" | "RUNNING"], "result": string }]
- `/jobs/` (POST) - envia um código para ser rodado no nosso serviço. O texto passado em *input* deverá ser redirecionado para a entrada padrão do script passado em *code*. O *timeout* de cada função deverá ser de 10 segundos.

Entrada:

```
{"uid": int, "code": string, "input": string}
```

Saída:

```
{"job_id": int}
```

- `/jobs/<job_id>` (GET) - retorna o status do `job_id` passado.
[{ "job_id": int, "uid": int, "status": ["DONE" | "ERROR" | "WAITING" | "RUNNING"], "result": string }]
- `/users/<uid>` - lista todos os `jobs` de um certo usuário.
[{ "job_id": int, "uid": int, "status": ["DONE" | "ERROR" | "WAITING" | "RUNNING"], "result": string }]

Estes são os requisitos básicos para um projeto **C**. Um projeto que não implemente corretamente os três itens ficará com conceito **D**.

Para avançar na nota do projeto você deverá implementar as seguintes funcionalidades.

- Crie uma ferramenta de linha de comando (similar ao `openstack-client` e `kubect1`) que interage com o serviço acima.
- Crie um script de deploy para sua aplicação (pode ser Juju, Docker ou shell).
- `/jobs/<job_id>` (DELETE) - cancela a execução do script identificado por `job_id`.

- Implemente um sistema de autenticação para este serviço. Desta maneira, para que um usuário submeta um serviço será necessário entrar com seu `uid` e sua senha.
- Proponha uma nova funcionalidade para este serviço.

Entrega

A entrega será feita via Blackboard e os testes serão todos feitos externamente via requisições REST. Qualquer linguagem poderá ser usada para programar o serviço, mas recomendamos usar Python e a biblioteca `boto3`. A data final de entrega é **12/06**.

Roteiro de estudo

Sugerimos que você estude como realizar as seguintes tarefas antes de iniciar o projeto:

1. lançar uma função para execução via AWS Lambda usando `boto3`;
2. recuperar o status de um script em execução;
3. enviar dados para esse script e receber seu resultado (tanto de retorno quanto da saída no terminal)