How To

Luca Mastrobattista, 0292461

Indice

1 Consumer **2**
1.1 creator.sh 2
1.2 destroyer.sh 2

2 Producer **2**
2.1 Configurazione 2
2.2 Modalità di avvio 2
2.3 Comandi CLI 3
2.3.1 Sempre, indiffirentemente dall'accesso 3
2.3.2 Utenti che non hanno effettuato l'accesso 3
2.3.3 Utenti che hanno effettuato l'accesso 3
2.3.4 Durante la lettura 3

1 Consumer

Nella cartella `src/consumer` sono presenti due script *bash* che hanno lo scopo di inizializzare e distruggere l'infrastruttura cloud.

1.1 creator.sh

Permette la creazione dell'infrastruttura tramite il file *Terraform* `src/consumer/init_infrastructure/infrastructure.tf`. Questo script può ricevere due parametri di input:

- `-env_var`: questo parametro specifica di non usare il file `~/.aws/credentials` per leggere le credenziali di accesso, che vengono, invece, richieste a riga comando per essere settate come variabili di ambiente;
- `-no_db`: questo parametro specifica di non re-inizializzare il database. In realtà, è maggiormente utile in fase di sviluppo, quando si effettua un aggiornamento sul codice e si vuole mantenere il database intatto.

Lanciare lo script senza parametri porta alla creazione di un ambiente cloud *pulito*, prendendo le credenziali dal file `~/.aws/credentials`.

1.2 destroyer.sh

Distrugge l'infrastruttura creata precedentemente. Anche questo script, può ricevere in input il parametro opzionale `-env_var`, che ha lo stesso scopo e produce lo stesso comportamento del caso precedente.

2 Producer

2.1 Configurazione

Anche qui, è necessario specificare dove debbano essere recuperate le credenziali di accesso per interagire con i servizi AWS.
La configurazione viene effettuata tramite un `Makefile`: lanciando `make`, verrà predisposto l'utilizzo del file `~/.aws/credentials`, che deve essere correttamente configurato al di fuori dell'applicazione; se si sceglie di utilizzare le variabili d'ambiente, invece, si può lanciare `make env-var`, che lancerà lo script bash `configure.sh`. Questo script chiederà interattivamente le credenziali e le memorizzerà su un file nascosto chiamato `.env`. Questo file verrà usato dall'applicazione per impostare le variabili d'ambiente grazie alla libreria `decouple`.

Nota: non è previsto un meccanismo di eliminazione automatica del file `.env`, ma il file può essere *svuotato* invocando `make` senza parametri.

2.2 Modalità di avvio

Il client può essere lanciato in tre modalità differenti:

- `$ python main.py`: il client viene lanciato con un'interfaccia `CLI`;

- `$ python main.py -g`: il client viene lanciato con il supporto `GUI`;
- `$ python main.py -h`: viene mostrato un messaggio di aiuto che riassume quanto detto in *Comandi Cli*.

2.3 Comandi CLI

Una volta avviata l'applicazione in modalità `CLI`, si possono eseguire comandi discriminando se l'utente ha effettuato l'accesso oppure no.

2.3.1 Sempre, indiffirentemente dall'accesso

Questi comandi possono essere sempre invocati, tranne durante la lettura dei messaggi in cui sono ammessi solo i comandi listati in *Durante la lettura*:

- `clear`: pulisce la *shell*;
- `help`: mostra il messaggio di aiuto;
- `exit`: termina l'applicazione.

2.3.2 Utenti che non hanno effettuato l'accesso

- `reg -u <username>`: permette di registrarsi al servizio con l'username passato come parametro. La password viene richiesta interattivamente.
- `log -u <username>`: permette di effettuare l'accesso al servizio con l'username passato come parametro. La password viene richiesta interattivamente.

2.3.3 Utenti che hanno effettuato l'accesso

- `usr_list`: permette di recuperare la lista degli utenti registrati;
- `read [-n]`: permette di leggere tutti i messaggi ricevuti. Se invocato col parametro, invece, restituisce solo i nuovi messaggi;
- `send -t <dest_1>, <dest_2>, ..., <dest_n> -o <object>`: permette di inviare un messaggio a uno o più destinatari. Il testo del messaggio verrà richiesto interattivamente. Gli utenti nella lista possono anche essere separati dal solo spazio.

2.3.4 Durante la lettura

Per ogni messaggio che si sta leggendo, è possibile effettuare diverse operazioni:

- `c`: *continue*. Visualizza il messaggio successivo;
- `b`: *break*. Interrompe la visualizzazione;
- `d`: *delete*. Elimina il messaggio visualizzato;
- `j <n>`: *jump to* n^{th}. Visualizza l'*n-esimo* messaggio;
- `r [-a]`: *reply [all]*. Permette di rispondere al mittente. Se invocato con il parametro, invia il messaggio al mittente e a tutti gli utenti presenti nel campo *To*, escludendo se stesso.