INSTALLAZIONE

Per l'installazione dell'applicazione è necessario l'utilizzo dei servizi web offerti da Amazon (AWS):

1. Creare un'istanza EC2 che assume il compito di service registry
2. Installare su di essa Git per il download del codice sorgente e Go per poter eseguire l'applicazione
3. Inserire in /home/ec2-user/.aws il file "*credentials*" dell'account di AWS utilizzato
4. Copiare il servizio "*registry.service*" in /etc/systemd/system ed avviarlo

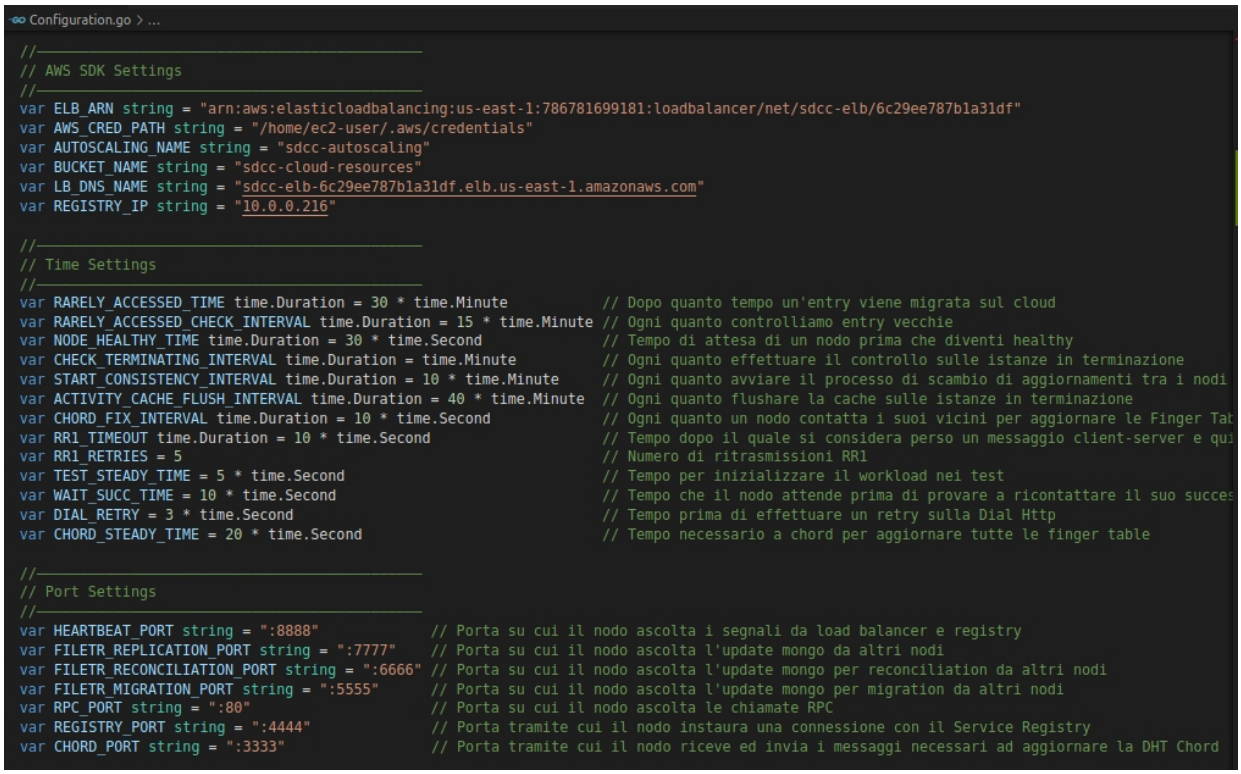
Per la gestione e l'evoluzione del sistema è necessario:

1. Creare un LB per l'instradamento delle richieste verso i nodi
2. Creare un'AMI da un'istanza EC2
3. Installare su di essa Git per il download del codice sorgente, Go per poter eseguire l'applicazione e MongoDB per poter memorizzare e gestire i dati del sistema di storage
4. Inserire in /home/ec2-user/.aws il file "*credentials*" dell'account di AWS utilizzato
5. Copiare il servizio "*node.service*" in /etc/systemd/system e abilitarlo così che venga avviato all'avvio del sistema
6. Creare la configurazione di avvio per l'Autoscaling tramite l'AMI realizzata al punto 2
7. Creare l'Autoscaling e i corrispondenti allarmi per la gestione e realizzazione di *scaleIn/scaleOut*

CONFIGURAZIONE

Per configurare l'applicazione è sufficiente modificare il file "Configuration.go", in cui sono definiti tutti i parametri che vengono utilizzati:

1. Aggiornare **ELB_ARN** con quello creato
2. Aggiornare **AUTOSCALING_NAME** con quello creato
3. Aggiornare **LB_DNS_NAME** con quello creato
4. Aggiornare **REGISTRY_IP** con quello dell'istanza utilizzata



```go
Configuration.go > ...

//------------------------------------------
// AWS SDK Settings
//------------------------------------------
var ELB_ARN string = "arn:aws:elasticloadbalancing:us-east-1:786781699181:loadbalancer/net/sdcc-elb/6c29ee787b1a31df"
var AWS_CRED_PATH string = "/home/ec2-user/.aws/credentials"
var AUTOSCALING_NAME string = "sdcc-autoscaling"
var BUCKET_NAME string = "sdcc-cloud-resources"
var LB_DNS_NAME string = "sdcc-elb-6c29ee787b1a31df.elb.us-east-1.amazonaws.com"
var REGISTRY_IP string = "10.0.0.216"

//------------------------------------------
// Time Settings
//------------------------------------------
var RARELY_ACCESSED_TIME time.Duration = 30 * time.Minute           // Dopo quanto tempo un'entry viene migrata sul cloud
var RARELY_ACCESSED_CHECK_INTERVAL time.Duration = 15 * time.Minute // Ogni quanto controlliamo entry vecchie
var NODE_HEALTHY_TIME time.Duration = 30 * time.Second              // Tempo di attesa di un nodo prima che diventi healthy
var CHECK_TERMINATING_INTERVAL time.Duration = time.Minute          // Ogni quanto effettuare il controllo sulle istanze in terminazione
var START_CONSISTENCY_INTERVAL time.Duration = 10 * time.Minute     // Ogni quanto avviare il processo di scambio di aggiornamenti tra i nodi
var ACTIVITY_CACHE_FLUSH_INTERVAL time.Duration = 40 * time.Minute  // Ogni quanto flushare la cache sulle istanze in terminazione
var CHORD_FIX_INTERVAL time.Duration = 10 * time.Second             // Ogni quanto un nodo contatta i suoi vicini per aggiornare le Finger Tab
var RR1_TIMEOUT time.Duration = 10 * time.Second                    // Tempo dopo il quale si considera perso un messaggio client-server e qui
var RR1_RETRIES = 5                                                 // Numero di ritrasmissioni RR1
var TEST_STEADY_TIME = 5 * time.Second                              // Tempo per inizializzare il workload nei test
var WAIT_SUCC_TIME = 10 * time.Second                               // Tempo che il nodo attende prima di provare a ricontattare il suo succes
var DIAL_RETRY = 3 * time.Second                                    // Tempo prima di effettuare un retry sulla Dial Http
var CHORD_STEADY_TIME = 20 * time.Second                            // Tempo necessario a chord per aggiornare tutte le finger table

//------------------------------------------
// Port Settings
//------------------------------------------
var HEARTBEAT_PORT string = ":8888"            // Porta su cui il nodo ascolta i segnali da load balancer e registry
var FILETR_REPLICATION_PORT string = ":7777"   // Porta su cui il nodo ascolta l'update mongo da altri nodi
var FILETR_RECONCILIATION_PORT string = ":6666" // Porta su cui il nodo ascolta l'update mongo per reconciliation da altri nodi
var FILETR_MIGRATION_PORT string = ":5555"     // Porta su cui il nodo ascolta l'update mongo per migration da altri nodi
var RPC_PORT string = ":80"                    // Porta su cui il nodo ascolta le chiamate RPC
var REGISTRY_PORT string = ":4444"             // Porta tramite cui il nodo instaura una connessione con il Service Registry
var CHORD_PORT string = ":3333"                // Porta tramite cui il nodo riceve ed invia i messaggi necessari ad aggiornare la DHT Chord
```

NOTA: Oltre agli altri parametri di configurazione, è possibile modificare anche le porte utilizzate dall'applicazione, tenere a mente che, per la porta utilizzata dal LB, non basta modificarla sul codice sorgente ma bisogna aggiornarla anche nelle impostazioni dalla console AWS, modificando la porta utilizzata per gli "*healthy check*" dei nodi.

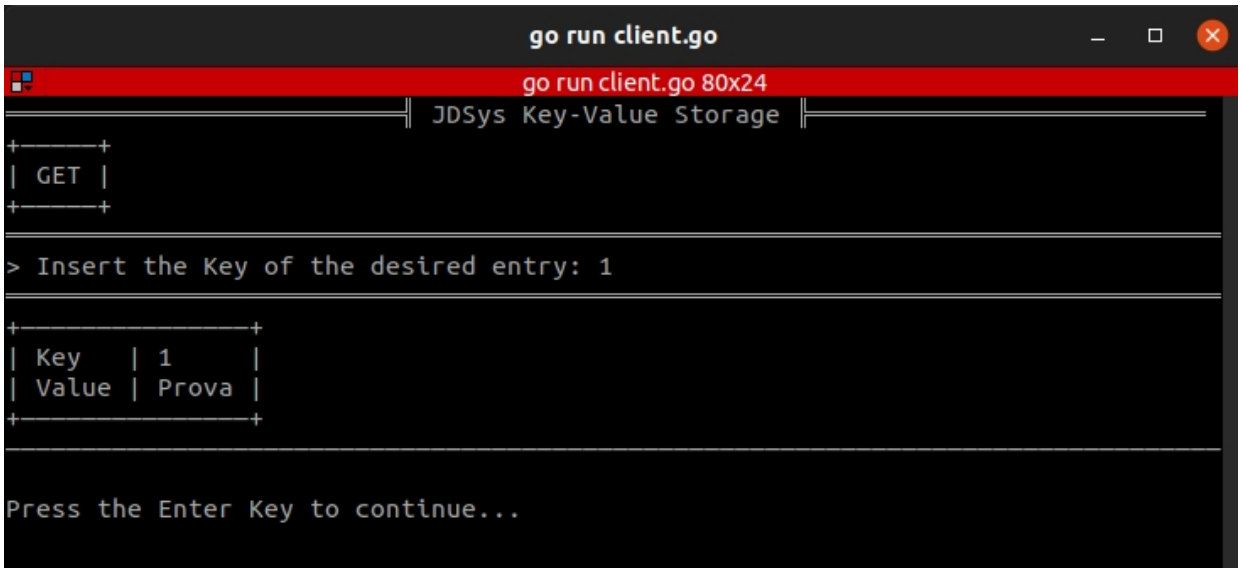
ESECUZIONE

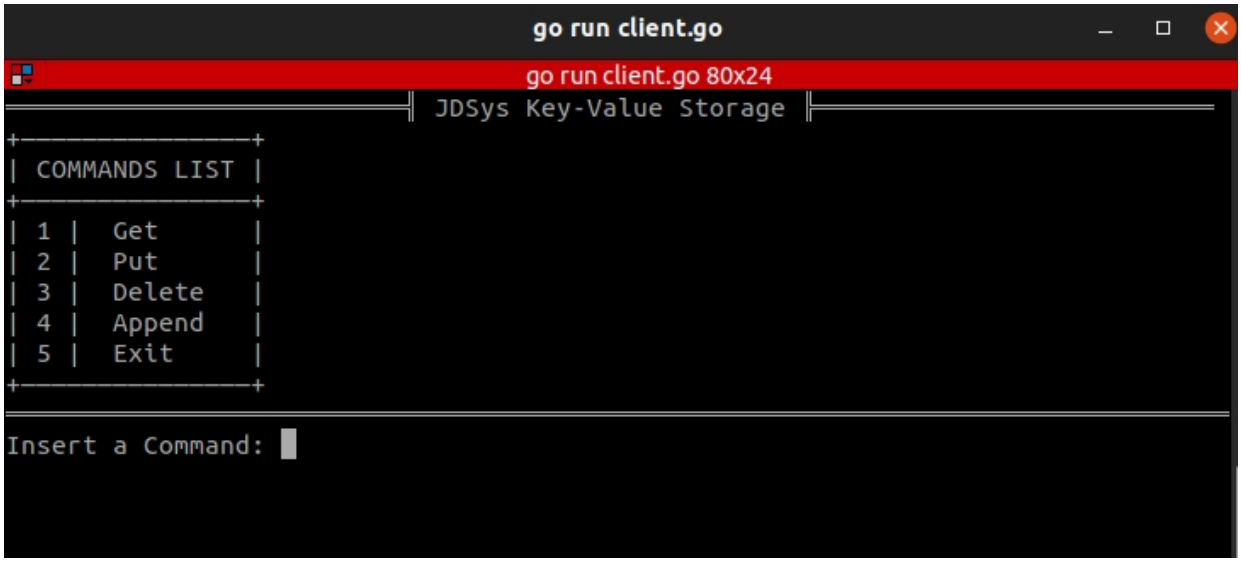
Una volta configurato il back-end del sistema come spiegato in maniera dettagliata nella sezione "**INSTALLAZIONE**", è sufficiente:

1. Scaricare in locale il codice sorgente da Github
2. Eseguire il file "*client.go*" per poter interagire con i nodi e quindi con il sistema di storage distribuito

NOTA: ovviamente, è necessario avere installato Go sulla macchina, altrimenti non sarà possibile eseguire l'applicazione lato client.

Di seguito viene mostrata come appare l'applicazione lato client con la successiva richiesta di "*GET*" verso il sistema di storage per la chiave "*1*":


go run client.go
go run client.go 80x24
JDSys Key-Value Storage
GET
> Insert the Key of the desired entry: 1
Key | 1
Value | Prova
Press the Enter Key to continue...


go run client.go
go run client.go 80x24
JDSys Key-Value Storage
COMMANDS LIST
1 | Get
2 | Put
3 | Delete
4 | Append
5 | Exit
Insert a Command: