Cloud Bursting Example Application – Cheatsheet

Introduction

This cheatsheet captures the demo of DB replication over WAN between EC2 and RackSpace and cloud bursting.

Architecture

Topology: 2 sites, single mysql server on each site with WAN replication (space+gateway). Tomcat servers running Petclinic against the mysql on each site.

1. IaaS providers: EC2 and RackSpace
2. DB: MySQL
3. WAN replication channel: the following XAP PUs: G/W, feeder, processor.
    - Listening on DB queries log file to intercept data mutation.
4. Load Balancer: RackSpace LB
5. Demo application: PetClinic running in Tomcat, packaged as a Cloudify application.

Order of system bootstrap is as per above.

Notes for current version:

- DB, PUs and application (#2, #3, #5 above) should be co-located on the same host.
- The only part that is currently managed by Cloudify is item #4. The rest is non-managed and is handled manually.

Cloud setup

- Configure the firewall
    - E.g. brute force: /etc/init.d/iptables stop
- Security group to allow access
- Key pair if needed

DB setup

- Install MySQL server
- Update the MySQL config file to write a queries log (make sure MySQL reloads the conf).
    - /etc/my.cnf
    - See example my.cnf
    - Point to the queries log file: /var/log/queries.txt
    - Touch the queries.txt and chmod to allow access by MySQL user
- Execute schema script
- Execute data script
- Validation: perform sql query and see it written in the queries log file.

WAN Replication setup

There are 3 PUs:

1. db-replication-processor
    - containing embedded space
    - reads SQL commands and executes them to the local DB
2. db-replication-feeder
    - parses the DB queries log file for new SQL commands and writes them to the local space
        - In the Spring application context pu.xml, configure the property queriesLogFilePath of the bean dataFeeder to point to the MySQL query log file (same as in the DB setup above).
3. wan-gateway-XXX
    - XXX being RS or EC2
    - Update G/W pu.xml to point to the remote G/W address
    - Local G/W is assumed to be collocated with the space on same host
    - On the RS gateway machine, you need to have use NAT translation
        - Update network_mapping.config to map the private->public EC2 IPs. See example network_mapping.config
    - Set NIC_ADDR to the right IP
        - NIC_ADDR="#eth0:ip#"

Set the above configurations and then deploy the 3 PUs in the above order.

Validation: make update on one table on local DB and see the change propagated to the remote DB. Repeat validation on both directions.

Load Balancer setup

- Allocate a LB from RS, and configure with address of the EC2 machine and RS machine.
    - Add a third dummy LB node (disabled) just since LB must have at least one node defined.

Demo application setup

Demo application is PetClinic running on Tomcat, managed by Cloudify

- Packaged as an application with a single tomcat service.
- The application is under petclinic-mysql-recipes\petclinic-mysql-XXX
    - XXX is RS or EC2
- Update tomcat.properties with URLs of Tomcat distro, petclinic .war file etc.
- Update RSloadbalancer.properties with LB details
    - During post-start/pre-stop stages the tomcat is registered/unregistered in the LB, using designated scripts.
- The petclinic .war file should be updated:

- Update jdbc.properties with the address of the DB
  - jdbc.url=jdbc:mysql://<IP-ADDRESS>:3306/petclinic