4/24/2013

C2M2L Final Integration Report

The Dynamic Context Server (DCS) is a semantic web application that provides ontologically-organized environmental modeling artifacts to users. The DCS can be hosted on a local personal computer or via a remotely served location. This report describes the steps needed to integrate the DCS within an Amazon Cloud environment.

Dynamic Context Server

INTEGRATING THE DCS ON THE CLOUD

The development version of the DCS resides on either a Windows or Linux personal computer. To make it accessible to either larger or more geographically dispersed design teams, the plan was to eventually host it on a site such as Vehicle Forge (http://VehicleForge.net). In practical terms, Vehicle Forge duplicates the available cloud-hosted platform such as Amazon Web Services (http://aws.amazon.com/) allowing us to generalize the installation to an all-purpose hosting platform. As we see no loss of capability, for this integration report we describe the steps needed to import and deploy the DCS on a typical AWS node.

The first step is to obtain a web services account, select an EC2 operating system image, and start an instance according to the most recent instructions. Note the IP address selected for you. Interactive tools such as the EC2 Management Console are available to check the health of the instance session (see **Figure 1**)

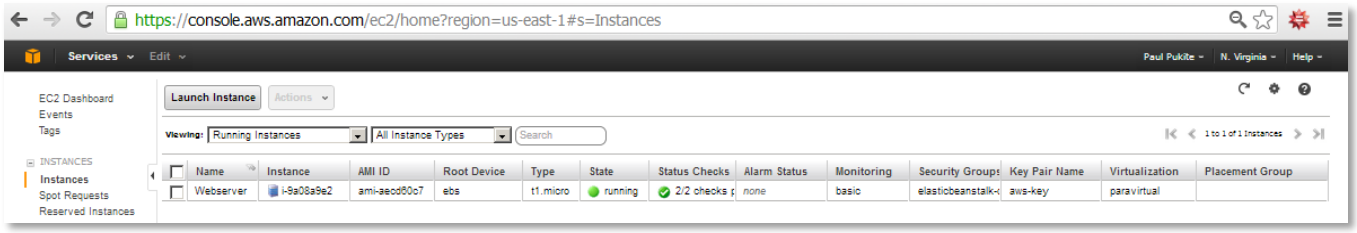

Figure 1: EC2 Management Console showing the named session

Once started, open a terminal session on your local computer, and login to the EC2 instance by using **ssh** or its equivalent:

```
local> ssh user_name@host_address
```

The code for the DCS is available from either the delivered compressed archive, or from a subversion repository located here

https://babelfish.arc.nasa.gov/trac/avm_performers/browser/context/

Preliminary to the installation, unpack the archives or checkout the source from the subversion account.

Prerequisites for the installation are described in Annex 3 (titled "Installation") of Appendix D of the C2M2L final report. This includes a short list of necessary open source supporting packages (swipl, R, graphviz).

These support applications needed by DCS may or may not be available as part of the ECS instance. If something is not available, we use the **yum** package management software to install the requisite apps.

To execute the DCS application, we have a choice to run it as a background daemon or as a no-hang-up (nohup) job, which will remain operational when we log out of the session. Since the latter is very convenient, we have provided an example nohup script called *nohup-run-cloud*. If launched from the main context directory, this will handle all the startup parameters.

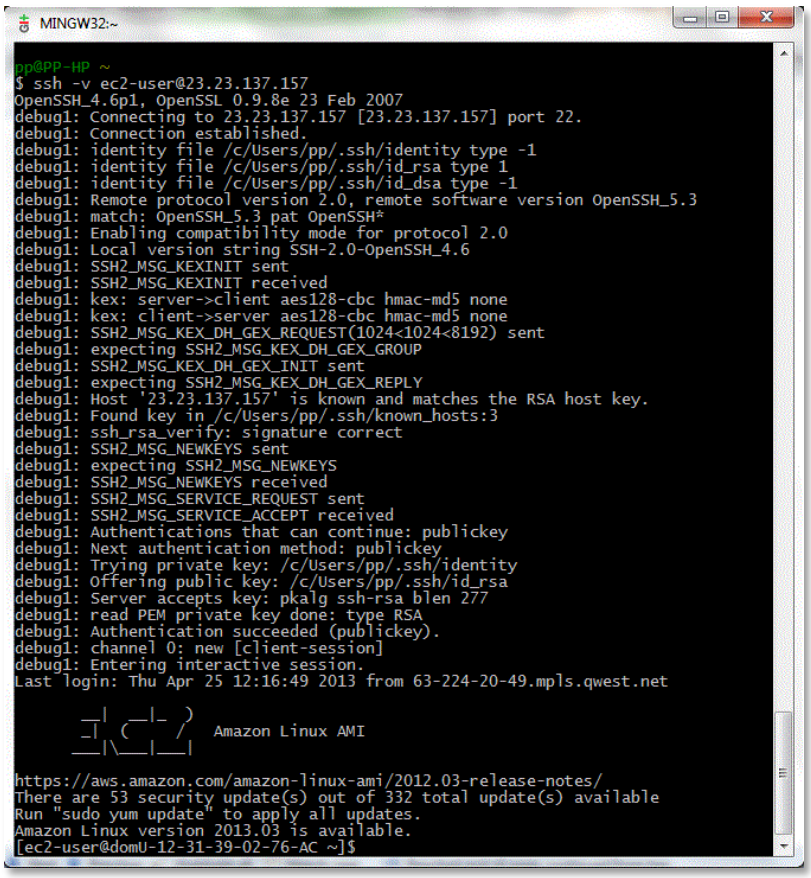

Figure 2: Logging into an AWS session using a terminal emulator and SSH.

```
[ec2-user@domU-12-31-39-02-76-AC ~]$ cd context/
[ec2-user@domU-12-31-39-02-76-AC context]$ pwd
/home/ec2-user/context
[ec2-user@domU-12-31-39-02-76-AC context]$ cat dynamic_context_server/config-en
abled/network.pl
:- module(conf_network, []).
:- use_module(library(settings)).

/** <module> Configure the HTTP server

Change the default port on which the HTTP server listens. If
host-detection does not work or this server is behind a proxy, you may
also need the public_host/public_port settings.

The =prefix= setting rebases all paths on the server to the indicated
path. Note that the prefix has *no* trailing /. E.g. a setting =|/demo|=
changes the root of the server to =|/demo/|=. Rebasing a server is only
possible if internal path dependencies use the HTTP path mechanism to
find paths for internal services.

The setting =workers= sets the number of HTTP worker threads. See the
link below for more info.

@see    localhost.pl
@see    http_location_by_id/2 and http_link_to_id/3 for finding the
        locations of internal services.
@see    http://www.swi-prolog.org/howto/http/HTTPScale.html for more
        info on server scalability.
*/
:- set_setting_default(http:port, 80).
:- set_setting_default(http:public_host, 'entroplet.com').
:- set_setting_default(http:public_port, 80).
% :- set_setting_default(http:port, 8080).
% :- set_setting_default(http:public_host, 'www.example.org').
% :- set_setting_default(http:public_port, 80).
% :- set_setting_default(http:prefix, '/demo').
% :- set_setting_default(http:workers, 16).
[ec2-user@domU-12-31-39-02-76-AC context]$
```

Figure 3 : The file network.pl is modified to use port 80 and a specific domain name.

The script is launched from the ***dynamic_context_server*** subdirectory (see **Figure 4**). As a convenience, we execute as root to get access to port 80. Other methods are available to acquire that port without assuming root privileges, which is recommended to increase security.

```
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$ ps -elf | grep swipl
4 S root      9067  8996  0  80   0 - 36999 poll_s 12:32 pts/1    00:00:00 sudo
nohup env PATH=/bin:/usr/local/bin/:/usr/bin swipl -g cp_server,sleep(100000000)
 -s run.pl -t halt
4 S root      9068  9067 28  80   0 - 236508 hrtime 12:32 pts/1   00:01:12 swipl
 -g cp_server,sleep(100000000) -s run.pl -t halt
0 S ec2-user  9095  8996  0  80   0 - 25861 pipe_w 12:36 pts/1    00:00:00 grep
swipl
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$ sudo kill -9 9068
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$ cat nohup-run-cloud
sudo nohup env PATH=/bin:/usr/local/bin/:/usr/bin swipl -g 'cp_server,sleep(1000
00000)' -s run.pl -t halt &
[1]+  Exit 137                sudo nohup env PATH=/bin:/usr/local/bin/:/usr/bin
swipl -g 'cp_server,sleep(100000000)' -s run.pl -t halt
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$ . ./nohup-run-cloud
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$ nohup: ignoring input
and appending output to `nohup.out'

[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$
```

Figure 4 : To run a DCS application, we first terminate any existing apps identified by "swipl" and then launch the "nohup-run-cloud" script. The script is essentially a one-liner which will run the application in the background and save output to a log file.

To terminate a session, use the kill command on the launched process id (PID). Use the commands **ps** or **top** to identify the PID of the server. There are no mechanisms for restarting a DCS session, other than to terminate and relaunch via the script. Monitoring of the application output is available through tailing the ***nohup.out*** log file (see **Figure 5**).

```
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$ sudo tail -9 nohup.o
ut

% /home/ec2-user/context/dynamic_context_server/run.pl compiled 2.56 sec, 31,375
 clauses
% Started ClioPatria server at port 80
% You may access the server at http://entroplet.com:80/
% Restoring 253 snapshots using 1 concurrent workers
combined_ontology.rdf.................................. 253 of 253 graphs
% Restoring 1 journals using 1 concurrent workers
user.......................................................... 1 of 1 graphs
% Loaded 254 graphs (436,253 triples) in 6.89 sec. (44% CPU = 3.02 sec.)
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$
```

Figure 5 : The output of the log file can be monitored via "tail –f". Otherwise, it is safe to log out from the EC2 session at this point and the app will continue to execute.

To verify a successful integration, navigate to the IP address of the AWS EC2 instance using a Firefox, Chrome, or recent Internet Explorer browser (see **Figure 6**).

Figure 6: Home screen of the Dynamic Context server

As delivered, the default DCS administrative user name is **pp**, and password **pukite.** This is required to load the repository context data, via the menu command shown in **Figure 7.**

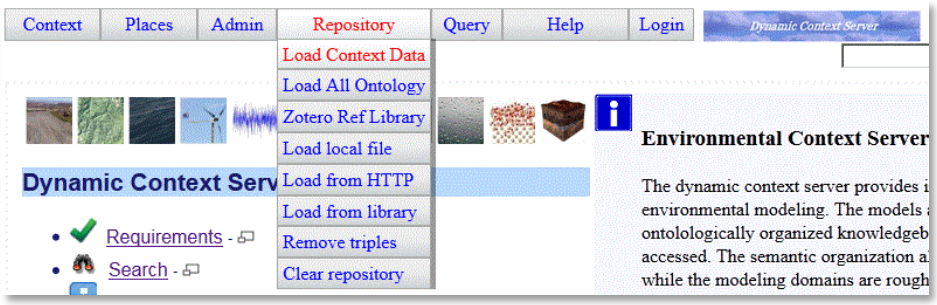

Figure 7 : As a one-time step, the semantic context data needs to be loaded administratively. This will normally take a few seconds at most. Regular users can not access the repository functionality.

The EC2 platform features support such as fail-over whereby a session will automatically restart should it terminate unexpectedly.

That concludes a description of the steps needed to integrate the Dynamic Context Server onto a cloud-based Amazon EC2 platform. Users can then access the DCS by navigating to the chosen IP.