BriCS

University of Bristol Cloud Service Simulation Runner

User & Developer Guide

1 October 2013

John Cartlidge & M. Amir Chohan

BriCS Architecture

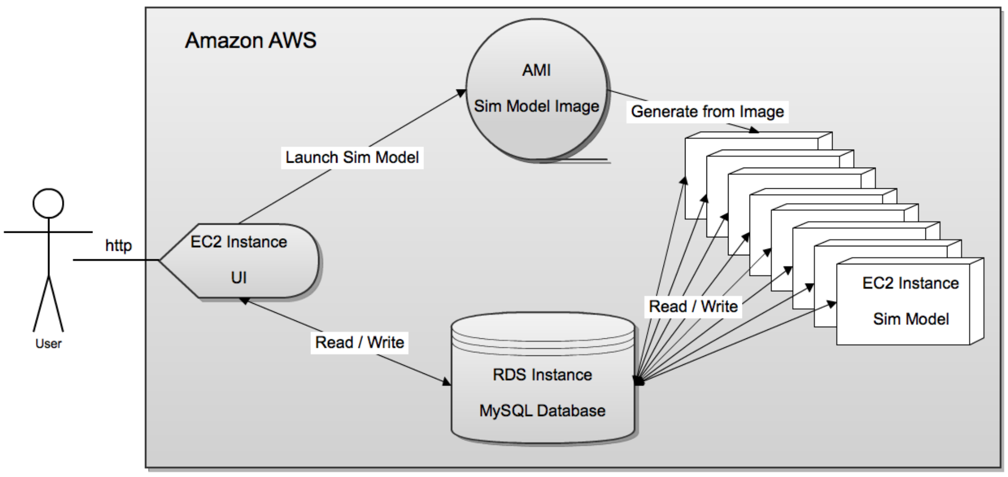


Fig. 1: Architecture design for the BriCS simulation model running on Amazon Web Services (AWS). An EC2 BitNami instance assigned an elastic IP (EIP) address hosts a UI web service, enabling users to configure simulation parameters, launch the simulation and observe simulation progress and results. Simulation data (configuration settings and results) are stored in an RDS instance running a MySQL DB. The UI has read/write access to the database, through which user-inputted configuration settings are written and simulation results are read. An Amazon Machine Image (AMI) stores an image of the simulation model. When a launch signal is received from the UI, the AMI generates multiple parallel EC2 simulation instances (SIs). SIs have read/write access to the DB, reading in configuration parameters and writing progress updates and results. Upon completion, the ephemeral EC2 simulation instances terminate.

Simulation models are run in parallel “in the cloud” using Amazon Web Services (AWS). An architecture diagram is shown in Fig. 1, *above*, containing the following components:

1. The User Interface (UI): Users interact with the simulation model through a web browser connected to the user interface of the model – a graphical web service running on a BitNami EC2 instance. Users are able to configure model settings, view results and stop/start a simulation run.
2. The Database (DB): All simulation data, including configuration settings and simulation results, is stored in an RDS instance configured to run a MySQL database. The database enables read/write access from the User Interface (to write configuration settings and to read simulation results) and the Simulation Instances (to read configuration settings and write results).
3. The Model Image (MI): The simulation code is stored as an image on an Amazon Machine Image (AMI) component. When the AMI receives a “launch” command from the User Interface, it generates a set of EC2 instances, each executing a copy of the simulation code.
4. The Simulation Instances (SIs): Simulation code is executed on a set of EC2 instances running in parallel. Simulations read configuration data from the database and write results to the database. On completion, instances self-terminate.

Component Design:

1. The User Interface (UI)

The UI web service will be written in Python using the Django framework using the BitNami AMI. An elastic IP (EIP) address will be attached to the EC2 instance hosting the UI, enabling a fixed IP / domain address. The UI communicates with the DB using Python's MySQLdb library. Configuration data input by the user is written to the database; results data is read from the database and presented to the user graphically. When a user starts a simulation, the UI communicates the command with the MI to launch simulation instances. Communication with the MI is performed using Python's AWS Boto interface.

2. The Database (DB)

The DB uses MySQL hosted on an RDS instance. Any MySQL API can be used to communicate with the database, e.g., MySQLdb for Python, or JDBC for Java. The database stores configuration data and results data for all simulation runs.

3. The Model Image (MI)

The MI contains an image of the executable simulation code, stored on an AMI. The AMI is used to launch SIs, when necessary. This can be performed manually via the AWS console, or automatically using Python's Boto framework.

4. The Simulation Instances (SIs)

Each SI runs executable simulation code. This code can be written in any language, but Python is preferred for architecture homogeneity. The simulation is launched with an id that is used to query the relevant DB data. Configuration parameters are read from the database and results data are written to the database.

Developer's Guide: Setting up AWS components

1. Database EC2 Instance:

From AWS EC2 console, under NETWORKS & SECURITY, go to Key Pairs and create a new Key Pair BoEkey. Save the download.

Go to instances and launch a Linux instance using BoEkey. Set instance to "terminate on close".

Note the Security group being used by this instance. Go to Security Groups under NETWORK & SECURITY, select the Security group, select Inbound and make sure there is a to allow SSH and MYSQL connection.

Under NETWORK & SECURITY, go to Elastic IPs. Allocate a New address and then associate it with this instance. This Elastic IP is referred to as the database instance address in this guide.

Create a folder `BoE` in your home directory and copy all the project files in there.

#secure the BoEkey, copy the setup script and connect to the database instance:
```
~/BoE/connect_db.sh <BoEkey> <database instance address>
```

#from the database instance, run set up script [select your own rootpassword for mysql root user]
```
./setup_db.sh rootpassword
```

This script installs everything necessary to set up the database. It will take some time to complete.

Edit the script if you want to change the default passwords, usernames and database names. By default, the script creates a database with the following user privileges:
```
mysql -uroot -p<rootpassword>
CREATE USER 'boeuser'@'localhost' IDENTIFIED BY 'boepassword';
GRANT ALL PRIVILEGES ON *.* TO '<boeuser>'@'localhost' WITH GRANT
OPTION;
CREATE USER '<boeuser>'@'%' IDENTIFIED BY '<boepassword>';
GRANT ALL PRIVILEGES ON *.* TO '<boeuser>'@'%' WITH GRANT OPTION;
exit
mysql -u<boeuser> -p<boepassword>
CREATE DATABASE boedatabase;
exit
```

Answer the MySQL configuration questions when prompted. The script will request you to reenter root password. Select:
```
Change root password: "n"
Remove anonymous users: "Y"
Disallow root login remotely: "Y"
Remove test database and access: "Y"
Reload privileges tables now? "Y"
```

When the script completes, the Database Server is complete.

Using the AWS EC2 console, save the instance as an image so that it can be restored at a later date, without having to reinstall. From the EC2 console, right click on instance, then: Create Image (EBS AMI). Give the image an easily identifiable name, such as: "BriCS DB Server".

Troubleshooting: It is possible for the script to fail if, for instance, it is stopped before completing and then re-started. The most likely reason for it to fail is if mysqld is already running. If this is the case, you will see an error of the kind:
```
ERROR 2002: Can't connect to local mysql server through socket
```

`'/var/lib/mysql/mysql.sock'`.

To correct this, you need to stop the running mysql server:
```
Sudo /etc/init.d/mysqld stop
```

Then, re-run the set-up script. It should now succeed:
```
./setup_db.sh rootpassword
```

2. Bitnami Django EC2 Instance:

For the web server, we use a third party instance from BitNami that comes pre-configured with Django. Go to
`http://bitnami.org/stack/djangostack#cloudImage`
Select "Cloud Server: Django Stack on Amazon", then select the link to "Amazon EC2".
Set the region to Europe (Ireland). Then, from the table choose a **free** ubuntu AMI from EBS. This guide is known to work on DjangoStack 1.4.8-0 (64 bit) with python version 2.7.
Note that the **username = bitnami.** If this changes, then the scripts will need to be edited accordingly.
You will now be re-directed back to AWS instance wizard. Launch an instance using the BoEkey. This instance will be used as the webserver. Give it a name such as: "BriCS WebServer". You will now see your instance launching in the AWS EC2 dashboard.

Note the Public DNS (referred to as the bitnami instance address in this guide).
Recommended: Associate an Elastic IP address, this will then be the bitnami instance address

Note the Security group being used by this instance. Go to Security Groups under NETWORK & SECURITY, select the Security group, select Inbound and make sure there is a rule to allow SSH connection and HTTP connection.

Then go to Security Credentials [accessed via your username drop down menu on top navigation bar]
Note the following details:
+ Access keys: access key id and secret access key
+ Account identifiers: AWS account ID
WARNING: The account credentials provide unlimited access to your AWS resources. Keep them secret and safe.

From your local BoE folder open `boto.cfg` and update the keys:
aws_access_key_id = access key id
aws_secret_access_ key = secret access key

Then open `webserver_settings.py` and update your owner_id variable.
owner_id = AWS account ID
Note: your account ID should contain digits only, do not include hyphens, i.e, enter 123412341234; not 1234-1234-1234

#From your local machine run 'create_project.sh' script to create a new project on the bitnami instance:
```
~/BoE/create_project.sh <BoEkey> <bitnami instance address>
```

Press 'Y' when prompted to remove unwanted packages. The script will leave you remote logged in to the bitnami instance.

#From the remote bitnami instance, run the web server:
```
sudo python apps/django/django_projects/BoE/manage.py runserver
0.0.0.0:8000
```

#open a new terminal, connect to the instance and set BoE settings:

```
ssh -i <BoEkey> bitnami@<bitnami instance address>
sudo nano apps/django/django_projects/BoE/BoE/settings.py
```

In the tuple DATABASES change the settings to:

ENGINE:	django.db.backends.mysql
NAME:	`<boedatabase>`
USER:	`<boeuser>`
PASSWORD:	`<boepassword>`
HOST:	`<database instance address>`

TIMEZONE = 'Europe/London'
LANGUAGE_CODE = 'en-GB'

Save file.

#run script to install the simulations application webserver:
```
./install_app.sh
```

Update the settings to tell the system where to find files and activate the app and the admin site:

#open `settings.py`:
```
cd apps/django/django_projects/BoE/
sudo nano BoE/settings.py
```

#set MEDIA_ROOT to `'/opt/bitnami/apps/django/django_projects/BoE/media'`
#set MEDIA_URL to `'/media/'`
#set STATIC_ROOT to `'<django source files directory>/contrib/admin/static'`
where `<django source files directory>` is `/opt/bitnami/apps/django/lib/python2.7/site-packages/django`
#under STATICFILES_DIRS tuple, add `'/opt/bitnami/apps/django/django_projects/BoE/static',`
#under TEMPLATE_DIRS add,`'/opt/bitnami/apps/django/django_projects/BoE/templates',`
#under INSTALLED_APPS tuple, uncomment `'django.contrib.admin'` and add `'simulations',`

Save the file. Finally, we need to deploy files and settings. From the same directory, do:

#add the tables in the database. You will be requested for a username and password.
#Remember these credentials; they are later required to login to the web server.
```
sudo python manage.py syncdb
```

#get django to collect all the static files. Select 'yes' to overwrite.
```
sudo python manage.py collectstatic
```

From the AWS EC2 console reboot the instance (select instance, then actions: reboot)

The Web Server is now complete.

Save the instance as an image so that it can be restored at a later date, without having to reinstall. From the EC2 console, right click on instance: Action: Create Image (EBS AMI). Give the image an easily identifiable name, such as: "BriCS Web Server".

3. SMI (Simulation Model Image) Amazon Machine Image:

As explained below, we first create an Amazon Linux EC2 instance with the simulation model installed. We then create an Amazon Machine Image (AMI) for this simulation model. The resulting AMI will be the Simulation Model Image (SMI), which will be used to launch instances (Simulation Instances) from the web server when required.

From AWS EC2 console, go to instances and launch an Amazon Linux instance using BoEkey.

Note the Security group being used by this instance. Go to Security Groups under NETWORK & SECURITY, select the Security group, select Inbound and make sure there is a to allow SSH connection.

Note the Public DNS (referred to as the smi instance address in this guide).

From your local BoE folder, open `smi_settings.py` and update all the variables.
Note: The Sim Model will not work if these settings are not correct. In particular, edit the:
db_endpoint = "`<database instance address>`"
webserver_public_dns = "`<bitnami instance address>`"

#run script to secure the BoEkey, copy sim model files and connect to the simulation model instance:

```
~/BoE/connect_sm.sh <BoEkey> <smi instance address>
```

#from the remote simulation model instance, run the setup script to install and configure.

```
./setup_sm.sh
```

Select 'y' when prompted by the script.

Note: the above install script will delete itself after it has run. Your simulation model instance is now complete.

Exit the instance from the command line.

Save the instance as an image:
Go to EC2 Amazon console, right-click on this instance and select "Create Image (EBS AMI)".
Give the image an easily identifiable name, such as: "BriCS Sim Model".

The simulation model is now complete. BriCS is now ready to use. See User Guide for operating BriCS.

About the Simulation Model:

The SMI contains a placeholder simulation model "boesimmodel.py" that multiplies two numbers together. It uses a sleep loop to simulate a long-running process.
The simulation model reads in a parameters file containing two numbers in the form:
a = 5 b = 6
The simulation then multiplies these two numbers together. The result is written to a results file. The model iterates through a looped sleep cycle and reports an update status (% complete) to the webserver.

When the SMI is launched, the simulation model is automatically started when the instance boots. On boot, the file \etc\rc.local automatically runs the script "launchboe.py". This launcher queries the Database Server to see if there are any simulations to run. If not, the instance will shutdown. Otherwise, while there are still simulations to run, the launcher will continue to get the next parameters file and pass it to "boesimmodel". The simulation model then runs.

The simulation model in file "boesimmodel.py" is simply a placeholder. You can replace this file with another simulation model of your choice. To change the model, it will be necessary to launch a new instance, configure the SMI and then create an image.

How to:

Update webserver instance address:

If the webserver instance address has been changed then this would need to be updated in the simulation model image.
To do this, launch an instance of the simulation model image, ssh into it and edit the `smi_settings.py` file by setting the `webserver_public_dns` variable to the new address. Create an Image of this and use the newly created AMI to launch simulation model instances in future.

Update database instance address:

If the database ip address has been changed then this would need to be updated in the simulation model image and the bitnami django webserver.

To update the address in the simulation model image, follow the 'update webserver instance address' and change the `rds_endpoint` variable instead of `webserver_public_dns` variable.
Then to update the address in the bitnami django webserver, ssh into it and edit the settings file:

```
sudo apps/django/django_projects/BoE/BoE/settings.py
```

Under the `DATABASES` tuple, update the `HOST` address.

Use a different AMI to launch database instance:

SSH into the bitnami django webserver and edit the db_ami.id file:

```
sudo apps/django/django_projects/BoE/db_ami.id
```

Change the max number of instances:

There are two places where the restriction has been applied. First change the `max_instances` variable in `launch_instances.js`. To edit this file use the following commads after connecting to the bitnami webserver:

```
sudo nano
apps/django/django_projects/BoE/static/js/launch_instances.js
sudo python apps/django/django_projects/BoE/manage.py collectstatic
```

Then edit `webserver_settings.py` and change `max_instances` there as well:

```
sudo nano
apps/django/django_projects/BoE/simulations/webserver_settings.py
```

User Guide

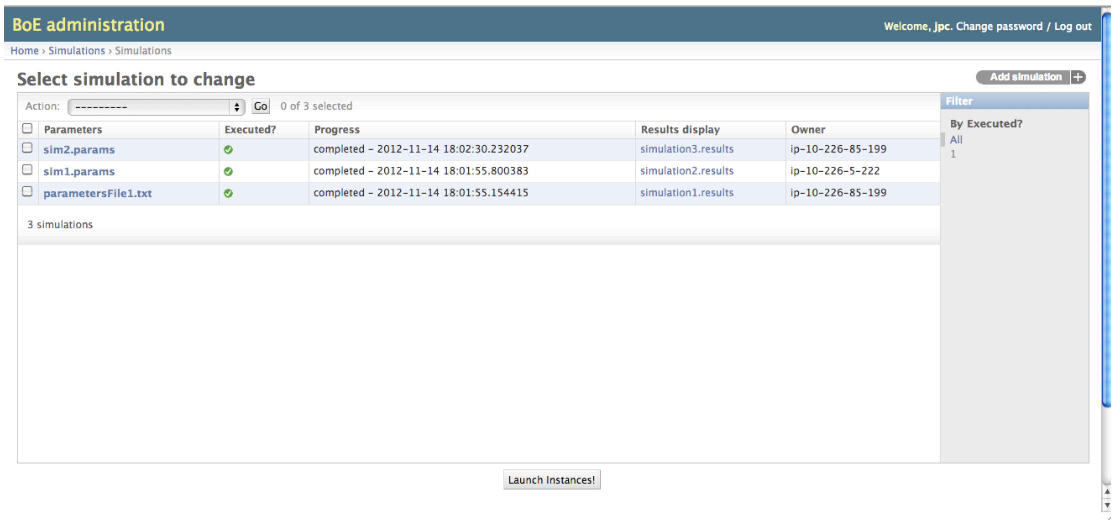
Getting started

Logging in:

- Access the website using URL `http://<Elastic IP>/BoE/admin/` (see developer's guide).
- Login credentials will be the ones that were created at the first database syncronisation i.e. when the command `sudo python manage.py syncdb` was executed for the first time.
- After successful login the admin page will open.

Simulation page

- To access this page go to `http://<Elastic IP>/BoE/simulations/`
- This page will display all the available simulations models.
- To download a the results or the parameters, simply click on it.
- To modify a simulation or launch instances, go to simulation admin page.



BoE administration

Welcome, jpc. Change password / Log out

Home › Simulations › Simulations

Select simulation to change

Action: --------- Go 0 of 3 selected

Parameters	Executed?	Progress	Results display	Owner
sim2.params	✓	completed – 2012-11-14 18:02:30.232037	simulation3.results	ip-10-226-85-199
sim1.params	✓	completed – 2012-11-14 18:01:55.800383	simulation2.results	ip-10-226-5-222
parametersFile1.txt	✓	completed – 2012-11-14 18:01:55.154415	simulation1.results	ip-10-226-85-199

3 simulations

Figure: Simulation Page

Simulation admin page

- Once logged in, simulations page can be directly accessed by going to /BoE/admin/simulations/simulation.
- Alternatively from admin page under Simulation tab, click on simulation.

Actions:

- These are located in a drop down list near the top left corner of the page. There are 4 actions. To carry out an action first mark the simulations; from the drop down list select the action and then hit `'Go'` button next to the drop down list.
- Delete action will simply delete the selected simulations from the database.
- To not execute some simulations when the instances are launched select `Mark selected as executed.` This will change the `status` field of the selected simulations to `'0'` in the database.
- In contrast, the "`Mark selected as not executed`" action will change the `status` of selected simulations to `'0'` in the database, so they can be executed again. This will

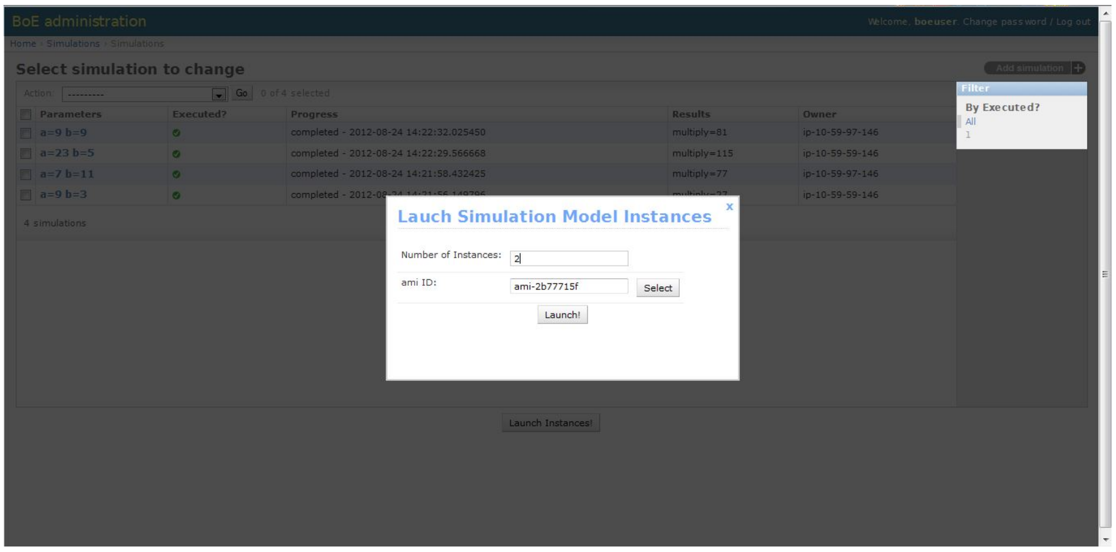
also delete the `owner`, `results` and `progress` parameters of the selected simulations.

Add a simulation:

- Click the grey `'Add Simulation'` button near the top right corner.
- Upload the parameters file. [For an example parameters file, use: **example_sim_params.txt**]
- Hit save. This will add a new simulation in the database.

Launch instances:

- Click the `'Launch instances'` button at the bottom of the page.
- Enter the number of instances. Notice you won't be able to add more than 3 digits. To get rid of this, see developer's guide. Also the number of instances entered must be less than the maximum number of instances (see developer's guide).
- Optionally, enter the Amazon Machine Image Id to launch instances from. If no Id is provided then the instances will be launched from the first available AMI.
- Hit launch.
- Once all the instances have their status as `'running'` an alert dialog will be display notifying of the success.
- The log file can be found in the log_files directory of the BoE project.
- **You must manually refresh the page to see the status updates of running simulations**



Select simulation to change

Parameters	Executed?	Progress	Results	Owner
a=9 b=9	✓	completed - 2012-08-24 14:22:32.025450	multiply=81	ip-10-59-97-146
a=23 b=5	✓	completed - 2012-08-24 14:22:29.566668	multiply=115	ip-10-59-59-146
a=7 b=11	✓	completed - 2012-08-24 14:21:58.432425	multiply=77	ip-10-59-97-146
a=9 b=3	✓	completed - 2012-08-24 14:21:56.149796	multiply=27	ip-10-59-59-146

4 simulations

Lauch Simulation Model Instances

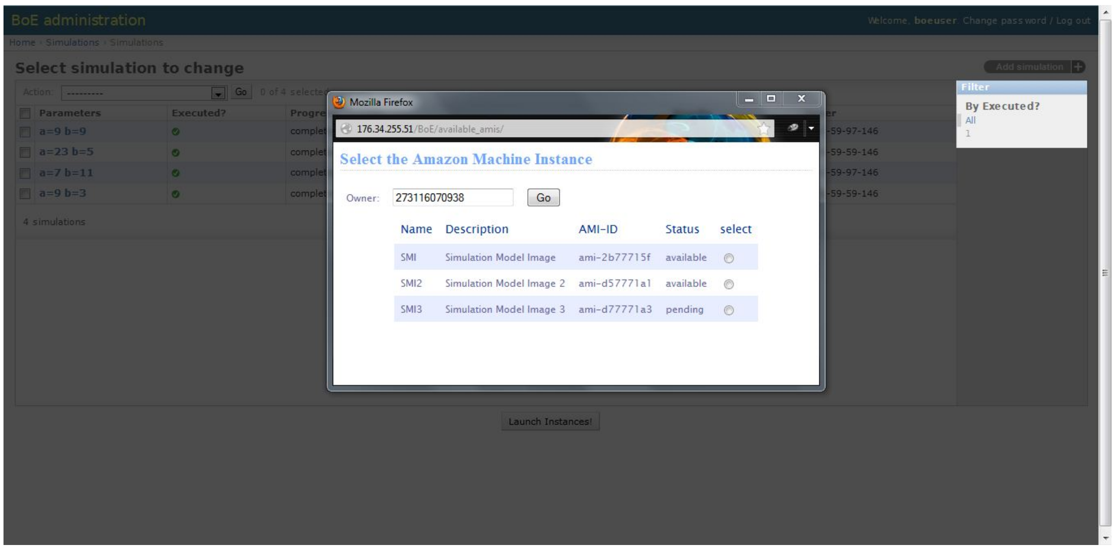
Number of Instances: 2

ami ID: ami-2b77715f

Figure: Launch instances pop-up

Select Amazon Machine Image:

- When launch instances form is open, click `'Select'` in front of ami-id field.
- In the new popup window, select the ami you wish to launch instances from.
- Alternatively, near the top left, enter the owner id of the ami. Hit Go and wait.
- If you wish to change the default owner id, change the `owner_id` variable in `webserver_settings.py`.


BoE administration
Welcome, boeuser. Change password / Log out
Home › Simulations › Simulations
Select simulation to change
Add simulation
Action: --------- Go 0 of 4 selected
Parameters
Executed?
Progre
a=9 b=9
a=23 b=5
a=7 b=11
a=9 b=3
4 simulations
Filter
By Executed?
All
1
Mozilla Firefox
176.34.255.51/BoE/available_amis/
Launch Instances!

Select the Amazon Machine Instance

Owner: 273116070938 Go

Name	Description	AMI-ID	Status	select
SMI	Simulation Model Image	ami-2b77715f	available	○
SMI2	Simulation Model Image 2	ami-d57771a1	available	○
SMI3	Simulation Model Image 3	ami-d77771a3	pending	○

Figure: Select AMI pop-up

Database

Starting database:

- To start the database go to `http://<Elastic IP>/BoE/start_database`.
- This will check whether the current database is live or not, if not it'll launch an instance of a previously saved database ami from db_ami.id in the BoE project folder.
- The log file can be found in the log_files directory of the BoE project by the name of db_start.log

Shutting database:

- To shutdown the database go to `http://<Elastic IP>/BoE/admin`.
- At the bottom of the page there is a 'Shutdown Database' button which redirects to `http://<Elastic IP>/BoE/shut_database`. Only an admin can go this page.
- This will create an AMI of the existing database instance, deregister the previous database AMI (if any) and shutdown this existing database instance.
- The log file can be found in the log_files directory of the BoE project by the name of db_shut.log