http://wso2.com/

Stratos 2.0 User Guide

Date: 05. April. 2013

Email: support@wso2.com

Contents

- Contents
- Getting Started
  - Stratos2 demo with Openstack as the IaaS
    - What you need
    - Setup the Openstack IaaS
    - Setup Strato2
    - Setup the Demo
    - Running the Demo
    - Tenant Creation
  - Stratos2 setup with EC2 as the IaaS
  - CLI Tool User Guide
    - Usage
    - Console Mode
    - Single Call Mode
  - GIT Repository Structure
  - GUI User Guide
    - Available Cartridges
    - Subscribed Cartridges
  - WSO2 Application Server Express Guide
    - Login to Stratos2.0 using CLI
  - SugarCRM Express Guide
    - Login to Stratos2.0 using CLI
  - Wordpress Guide
    - Login to Stratos2.0 using CLI
    - Upload wordpress
    - Setting up database
    - Create a database
    - Run installer

- PHP cartridge and Identity Server Integration Guide
  - Login to Stratos 2.0 using CLI
  - Upload simplesamlphp
  - Login to IS using tenant info
  - Getting SP Metadata
  - Test SSO
- Deployment Synchronization with Github
  - Login to Stratos2.0 using CLI
  - Custom Domain Mapping
  - Setting Service Hook at Github
- Autoscaling User Guide
  - How can I provide the autoscaling related parameters?
  - What are the parameters and what they mean?
  - Can I set a limit to the number of service instances that are maintained in the system, at any given time?
  - Do I have a control over the number of instances that the autoscaler can start?
  - How can I do a simple autoscaling test?
  - Sample configuration files
    - Properties defined in the defaults section
    - Properties defined within the service element
  - Sample CLI command
- Troubleshooting
  - Stratos2 Core services are not started
  - Started Carbon cartridge instance has not joined ELB
- Known Issues
  - PHP cartridge and Identity Server Integration Guide
- References

Getting Started

To get immediately up with Stratos2 you have two options. First option is you install Openstack in a single node and install the Stratos2 packs on the same node. The other option is you use the EC2 pre-installed demo image. The first option use Openstack as the IaaS. The second option use the Amazon EC2 as the IaaS.

If you are not comfortable in installing IaaS itself(which is by the way very easy) you can go for the EC2 IaaS option.

Stratos2 demo with Openstack as the IaaS

In this demo we show how to use Stratos2 interactive CLI and browser based UI to subscribe to a php cartridge, which require a mysql cartridge for database access. We also show how to subscribe to a WSO2 Application Server(AS) cartridge. Before anything we need to first setup our IaaS, that is Openstack running on a single node.

What you need

- You need to download Stratos2 release pack **wso2s2-1.0.0.zip** and the interactive CLI tool **wso2s2cli-1.0.0.zip**
- Release pack contain

1. All binaries for Stratos2 needed to run Stratos2

2. Installation scripts

3. WSO2 PHP cartridge(php-cartridge-amd64.img). We use this to show you how to subscribe to a PHP cartridge and upload and use your PHP apps using it.

4. WSO2 mysql cartridge(mysql-cartridge-amd64.img). We use this to show how to subscribe to a mysql cartridge.

5. WSO2 carbon cartridge(carbon-cartridge-amd64.img). We use this cartridge to show you how to subscribe and use a WSO2 application server product cartridge. To run this cartridge you need Puppet master run in your node which can be done using a script provided in the Stratos2 pack.

6. Necessary tools to create custom cartridges

- You need at least 4Gb memory and 40Gb hard disk in your node to run Openstack and Stratos2

- Install Ubuntu 12.04 server in the node using normal install procedure. Please refer to Ubuntu installation docs for installing Ubuntu server. You can use any other operating system which support Openstack. However then you need to find some instructions to install Openstack Essex version in that OS.

Setup the Openstack IaaS

You need to install Openstack in your node. For setting up Openstack as the IaaS for Stratos2, please refer to **[1]**. It provide easy to follow steps to get Openstack installed in a short time.

Setup Strato2

Please follow the Stratos2 installation guide to install Stratos2 in the same node as the Openstack installed node. From now on we assume that you have followed the installation steps exactly as in the installation guide to install Openstack and Stratos2 in the same node. We also assume that you have enabled Openstack demo while installing Stratos2.

Setup the Demo

You don't need to do anything new to setup the demo if you have installed Stratos2 with Openstack in the same node as described in the Stratos2 installation guide, with Openstack demo enabled.
To verify certain things log into Openstack dashboard web ui in the Openstack controller node using url

http://<openstack controller node ip>
username: admin
password: openstack

- The security group you have chosen should have the following ports open. Note that this is already done when you install the Stratos2 and Openstack following the Stratos2 installation guide.

icmp, 22, 80, 443, 8080, 3306, 4000, 4025, 4103, 8280, 5001, 5140, 5672, 5673, 7000, 7721, 9000, 9160, 9161, 9443, 9444, 9445, 9446, 9447, 9763, 9764, 9765, 9766, 9767

- Verify that PHP, MYSQL and WSO2 Carbon Cartridges are uploaded to the Glance Server. This was done in the Stratos2 installation process.

- Make sure that you have four java processes running, which mean that all the Stratos2 servers are running. There servers are started at the end of the Stratos2 installation script.

$ ps -A|grep java

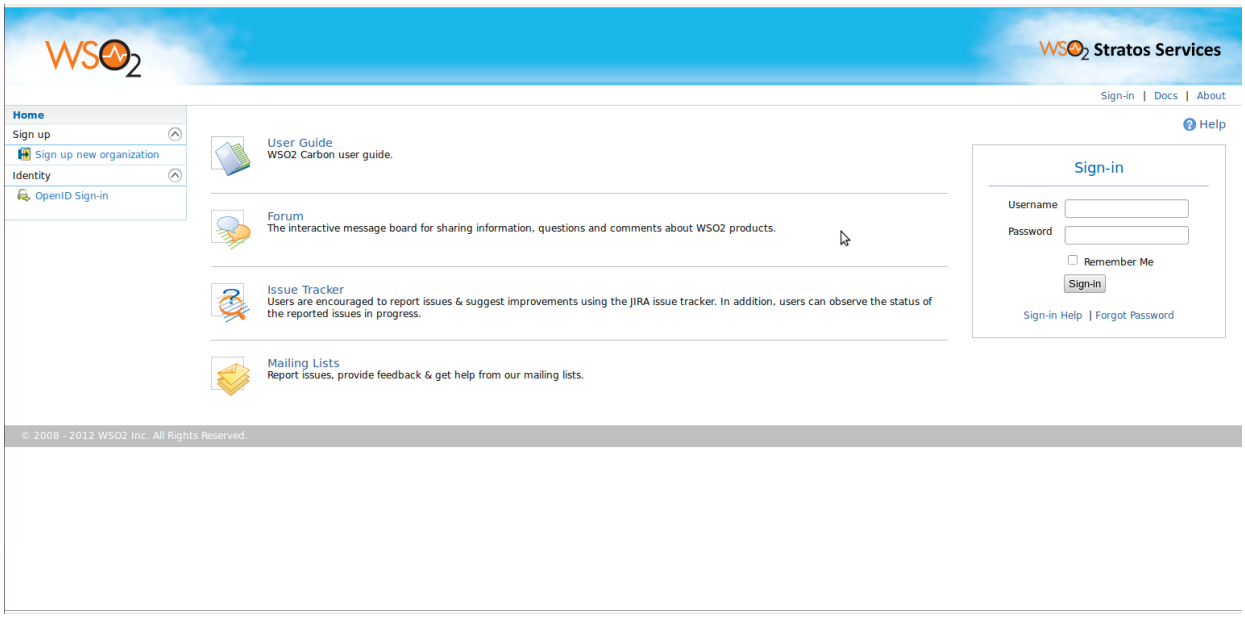
Running the Demo

Tenant Creation

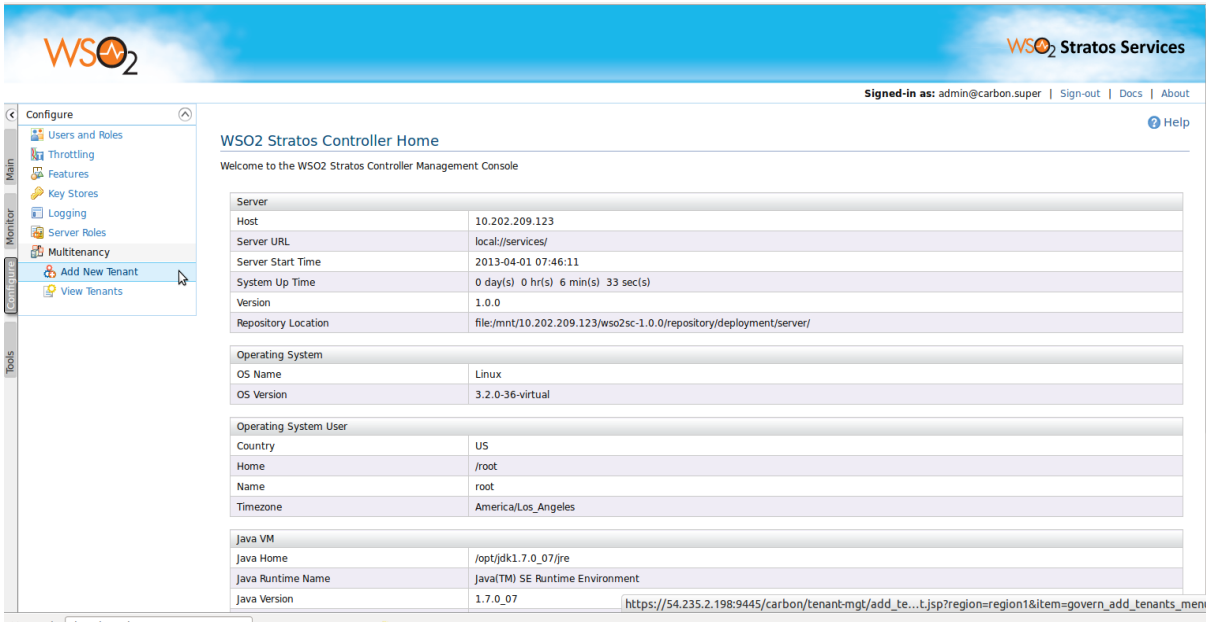
Before moving forward, you need to create a tenant for yourself in Stratos 2 environment. For this you need to,

- Access the URL of the Stratos Controller (https://<**STRATOS_SC_HOST:STRATOS_SC_PORT**/carbon>).

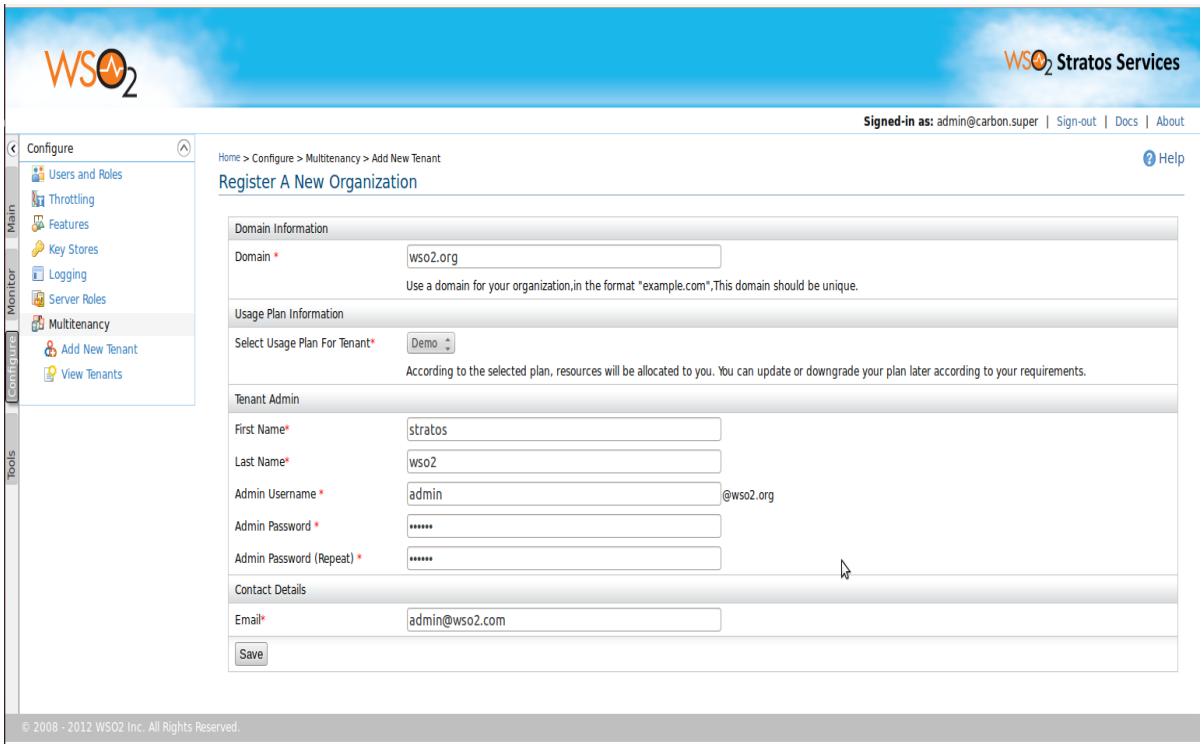
- This will redirect you to a management console.



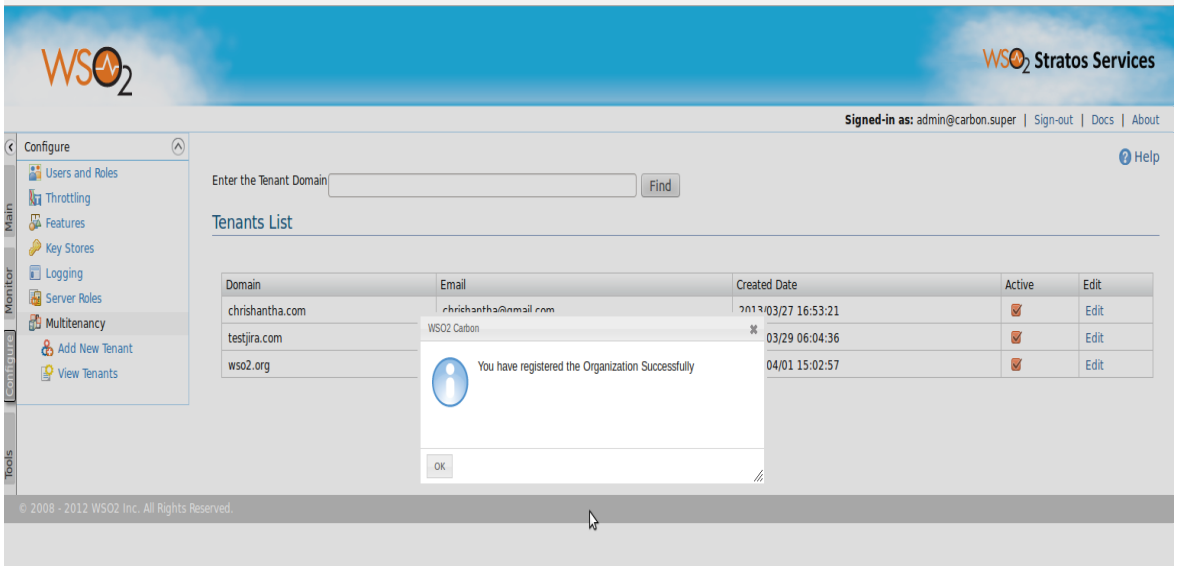
- Login to the management console using “admin” as the username and “admin” as the password.



- Click on the “Add New Tenant” menu item on the left side bar (please see the above image).



- You will be redirected to a page like above image. You can add the tenant details and get registered in Stratos 2, using this page.



- On a successful registration you will see the above message. Now your tenant "wso2.org" and corresponding 'admin' user "admin@wso2.org" is successfully created in Stratos 2 environment and you can proceed with the next steps of the Stratos 2 journey.

Stratos2 setup with EC2 as the IaaS

We have made available following public EC2 images (AMI s), which you can use to setup and run Stratos 2 in EC2, with less time.

1. Stratos 2.0 Components (s2demo.ami / ami-7a3b5f13)
2. PHP cartridge (s2demo_php_cartridge.ami / ami-64b2d70d)
3. MySQL cartridge (s2demo_mysql_cartridge.ami / ami-18bf2571)
4. WSO2 Carbon cartridge (s2demo_carbon_cartridge.ami / ami-7889ec11)

EC2 Configurations

---

1. Key pairs

Create a key pair in EC2, download and save the private key.
set permission: chmod 0600

2. Security groups

Create a security group with following ports enabled.

22 (ssh port)
80 (apache http port)
443 (apache https port)
3306 (mysql)
8280 (elb http proxy port)
8243 (elb https proxy port)
9445 (sc https port)
5001 (sc clustering port)
4000 (elb clustering port)
9443 (default https)
9763 (default http)
4100 (clustering port of carbon instances)
9447 (CartridgeAgentService https port)
8140 (puppet master port)
9764 (carbon cartridge http port)
9444 (carbon cartridge https port)
7714 (BAM port)
9163 (BAM cassandra port)

Stratos 2.0 Components Configuration

Using AMI s2demo.ami, start an instance in EC2 (m1.xlarge is preferred)

Stratos 2 binaries are installed into /opt location.

Most of the configurations are already done, but few more properties need to be set with appropriate values.

In /opt/wso2sc-1.0.0/repository/conf/cartridge-config.properties

cartridge.key=<cartridge_key_path> (Full path to the EC2 private key)

In /opt/wso2cc-1.0.0/repository/conf/cloud-controller.xml

Set following properties accordingly.

ACCESS_KEY - Access Key of EC2 account

SECRET_ACCESS_KEY - Secret key of EC2 account
OWNER-ID - Owner id of EC2 account
AVAILABILITY_ZONE - Availability zone (eg, us-east-1c)
SECURITY_GROUP - Name of the EC2 security group
KEY_PAIR - Name of the key pair

Start the system
---

execute /opt/start_servers.sh to start up the stratos 2 components.

Once the servers are started you can login to Stratos 2 Management Console and register tenants, (described in the section above)

CLI Tool User Guide

You can start playing with the demo setup. For that you need the interactive client CLI tool. You can find the CLI tool in **wso2s2cli-1.0.0.zip** file. Extract this to home directory or a place of your choice and then follow instructions as below.

Usage

You have to export the host and port of SC as environment variables in your bashrc file before using the CLI Tool.
E.g.
export STRATOS_SC_HOST=192.168.92.10
export STRATOS_SC_PORT=9445

Also export the following environment variables by adding them into your bashrc file

export JAVA_HOME=/opt/jdk1.6.0_24/
export PATH=$JAVA_HOME/bin:$PATH

$ cd wso2s2cli-1.0.0

Add execute permission to the script

$ chmod 755 ./stratos.sh

Console Mode

In this mode you can call stratos.sh to log-in to Stratos Controller,

$ stratos.sh [username] [password]

eg.

$ stratos.sh admin@wso2.org admin123

If you want to give the keystore path,

$ stratos.sh [username] [password] -keystore <path_to_keystore>

If you have provided wrong parameters to log-in, log-in help will be shown.

```
Usage for log-in:
    sh stratos.sh [username] [password] -keystore <path_to_keystore>
       -keystore : Default key store for Stratos
```

Now the you are logged into a interactive command line shell interface called stratos.
You can use 'help' command to see the usage,
i.e. **stratos> help**

Help command will be providing usage of all the actions. Here is how it looks like,

stratos>help

Usage: stratos> [action] <mandatory argument>* <option identifier>* <optional argument>*

Action can be one of the following,
list : List the available cartridges with details.
subscribe : Subscribe to a new cartridge.
info: Print a detailed description about a cartridge.
domain-mapping : Give a domain mapping to a subscribed cartridge.
help : Print general help and help for different actions.

Mandatory and optional arguments depend on the action. Following is the list of usages for each action,

Usage for action list:
list

Usage for action subscribe:
subscribe [cartridge type] [cartridge alias] -min <min instances> -max <max instances> -repoURL <GIT Repository URL>
-repoUserName : Username for the given repository
-repoPassword : Password associated with repository username
-min : minimum number of instances. Default is 1
-max : maximum number of instances. Default is 1
-connect : Connects one cartridge to another. This need to use with -alias
-alias : data cartridge alias
eg: subscribe [cartridge type] [cartridge alias] -connect [data cartridge type] -alias [data cartridge alias] -repoURL <GIT Repository URL>
Note: Aliases cannot include uppercase letters or . (dot) character

Usage for action unsubscribe:
unsubscribe [cartridge alias]

Usage for action info:
info [cartridge alias] -v
-v : verbose

Usage for action domain-mapping:
domain-mapping [cartridge alias] [mapped domain]

Usage for action remove-domain-mapping:
remove-domain-mapping [cartridge alias]

Usage for action help:
help [action]

Single Call Mode

You can call the command directly with action. It will not show the “stratos>” prompt and fulfill the request of command immediately for particular action command. This mode is useful for automating your interaction with Stratos2.

In this mode he can call the command as follows
$ stratos.sh [username] [password] [action commands]

If user wants to give the keystore path,

$ stratos.sh [username] [password] [action commands] -keystore <path_to_keystore>

GIT Repository Structure

The GIT repository that configure with a cartridge should have the correct folder structure to add artifacts in order synchronize with the cartridge and add to the correct location of the server to get deployed.

For most of the carbon server cartridges the corresponding GIT repository should have the folder structure of <CARBON_HOME>/repository/deployment/server.

E.g. for an Application Server Cartridge the folder structure should be as follows.

axis2modules, axis2services, aggeryapps, jaxwebapps, jsservices, modulemetafiles, servicemetafiles, webapps.

For the PHP cartridge user should create a folder called **www** inside the GIT repository in order to upload PHP artifacts.

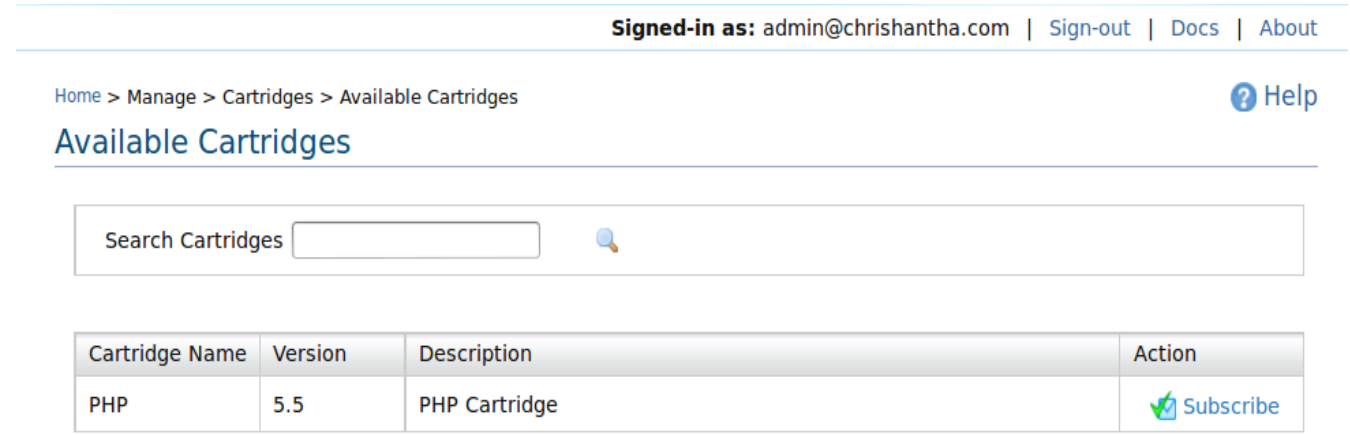
GUI User Guide

GUI can also be used to playing with the demo setup. Url to access the GUI would be (SC),
https://SC_HOST:SC_PORT/
eg.
https://192.168.92.10:9445/

GUI mainly consists of two menus, which will be displayed when a tenant user is logged in.
Eg: Available Cartridges menu can be found in **Main --> Cartridges --> Available Cartridges**

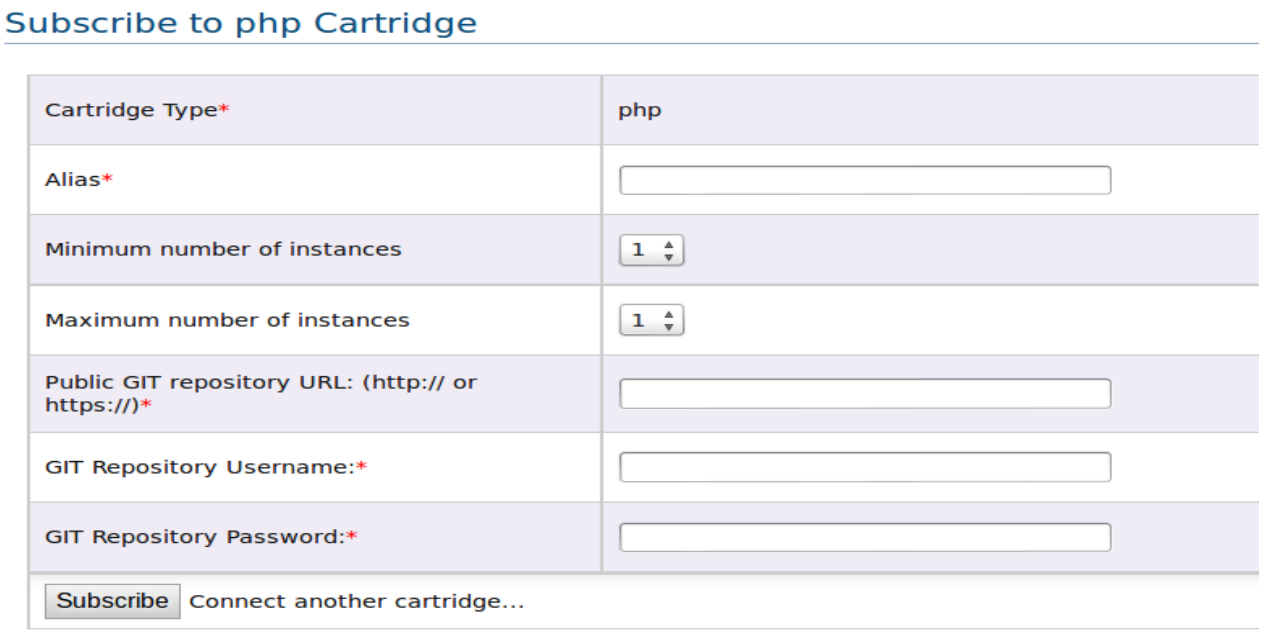
Available Cartridges

This link will display the available cartridges for the user.



User can search for cartridges using Cartridge Name and Alias and subscribe for a selected cartridge using “Subscribe” link.

Subscribe form is displayed when user clicked on the subscribe link.



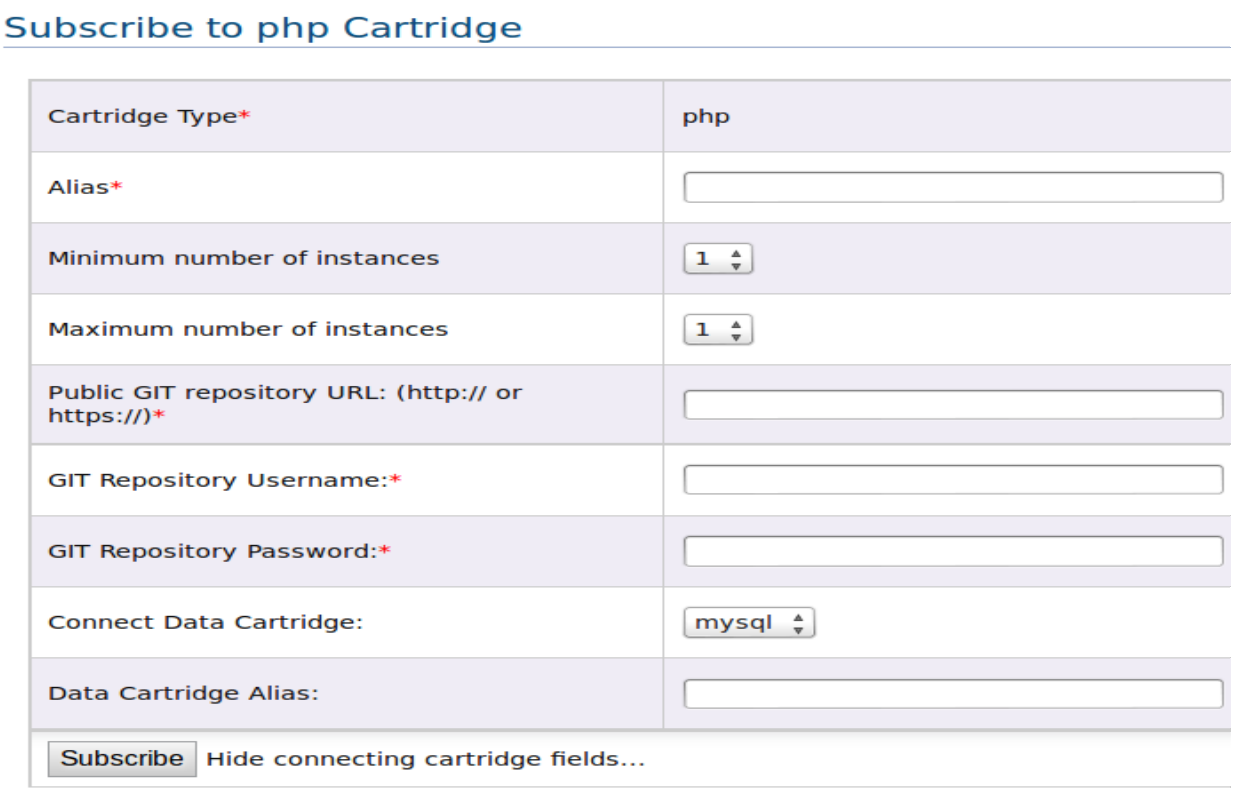
The user has to enter a unique alias for the cartridge , which will be used to identity the cartridge instance.

Please note that the "." (dot) character can be used for alias.

External GIT Repository values (URL, Username & Password) are required. The URL should not contain the username. Username and password should be specified separately as shown above.

Please note that for carbon cartridges, currently only **HTTP git repositories** are supported. You can setup your own GIT server to support GIT repositories with HTTP access, as explained in the section 'Setting up a Glitbit Server and Creating Git Repositories' in Stratos 2.0 Installation guide.

User can also a connect another cartridge when subscribing. If user clicks on "Connect another cartridge" link next to "Subscribe" button, additional two fields will be displayed to selected the cartridge type and alias.

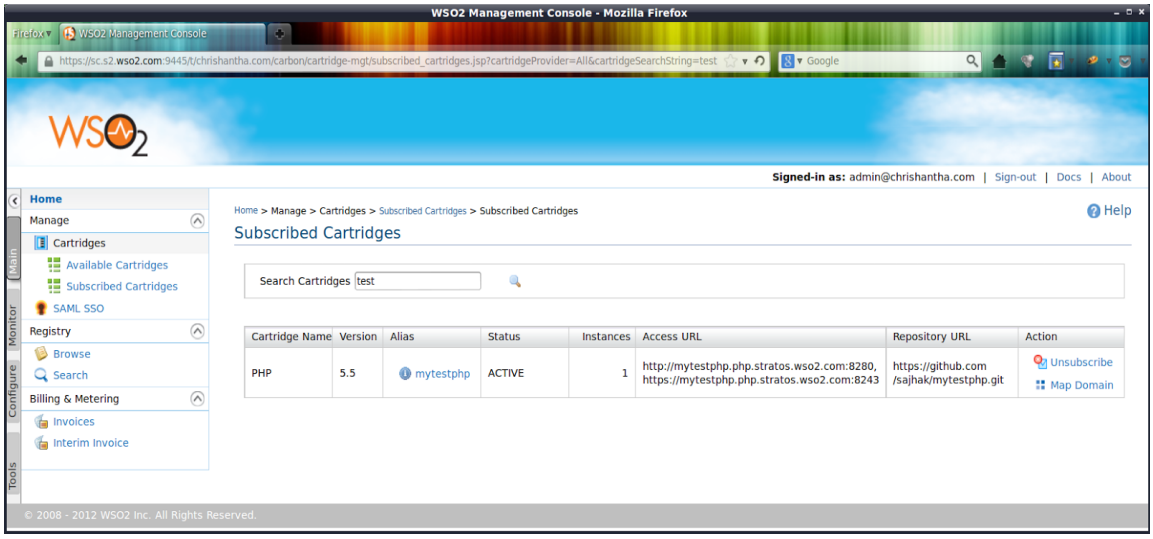


When subscribing to a data cartridge, user need to enter the alias only.

Minimum & Maximum number of instances are not needed for WSO2 Carbon based cartridges.

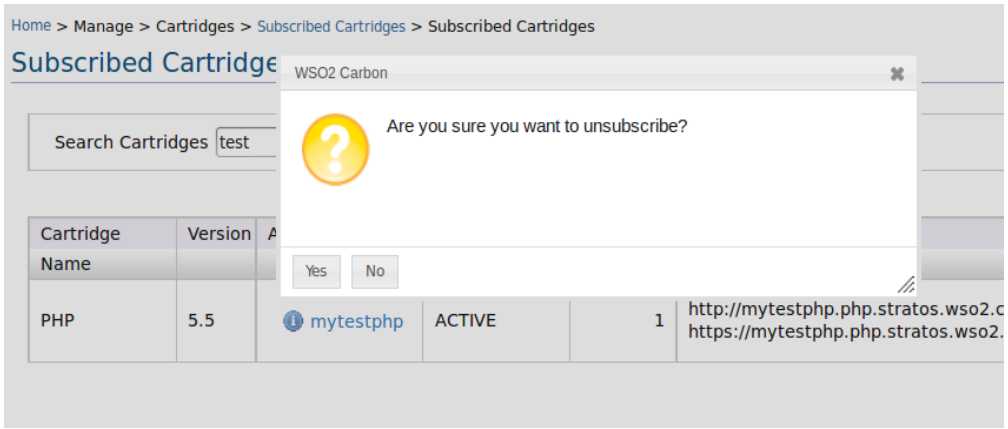
Subscribed Cartridges

Subscribed cartridges page will display all the subscribed cartridges for the tenant user.

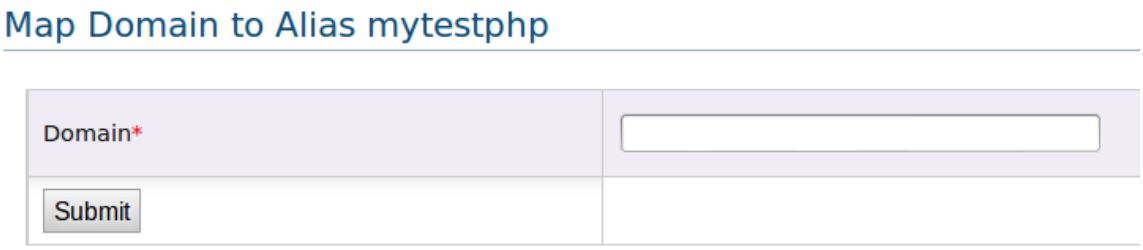


Cartridge Name: Name of the cartridge
Version: Version of the ccartridge
Alias: The unique alias of the cartridge
Status: Current status of the subscribed cartridge which will have the transition as SUBSCRIBED (just after subscribe link is pressed) --> NOT-READY(cartridge instance is starting up) --> ACTIVE (cartridge can be usable in this state)
Access URL: Cartridge can be accessed using this URL. Management console of carbon cartridges can be accessed by this URL.
Repository URL: The URL of the git repository
Action: indicate the action can be performed on a cartridge.

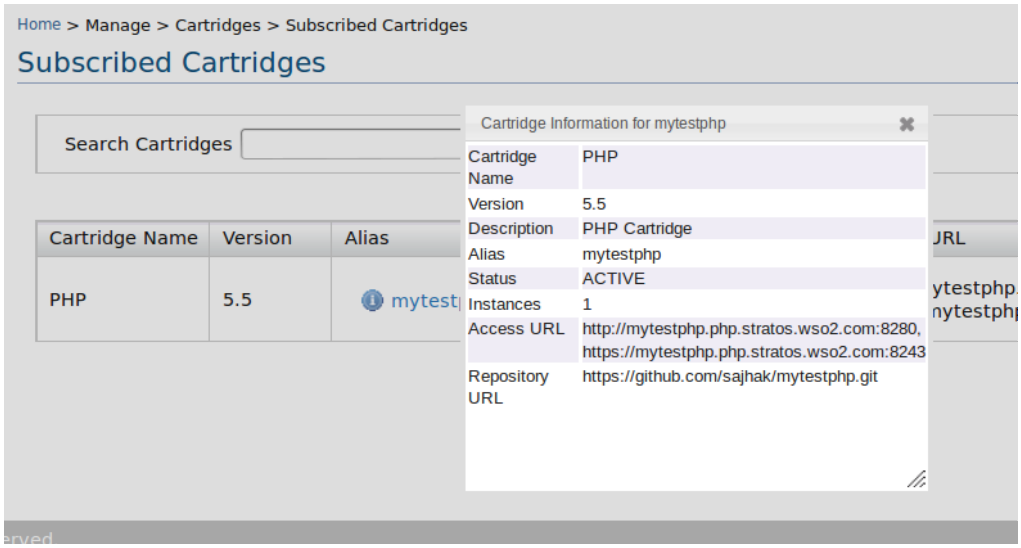
User can click on "Unsubscribe" to unsubscribe from the cartridge. User will be prompted to confirm the unsubscribe action



User can use the “Map Domain” link to add a domain mapping for the subscribed cartridge.



If the user wants to find more information of a subscribed cartridge, then the user can click on the subscribed cartridge alias. Then it will show a pop-up window with information of the subscribed cartridge.



Monitoring Logs of Cartridge instances in EC2

The cartridge instance’s logs can be viewed from Stratos Controller in EC2. Since EC2 is bydefault configured for logging, you can easily browse your cartridge instance’s logs. If you go to Monitor --> System Logs in SC, you can see the logs of SC instance by default.

But in the same page, if you go to “Show advance Search”, you can find the Cartridge Alias List. From that, if you select a cartridge alis of your preferred instance, you can get the logs from that particular instance.

WSO2 Application Server Express Guide

Login to Stratos2.0 using CLI

$ ./stratos.sh admin@telecom.org password

```
Successfully Authenticated!
```

stratos>subscribe as mywso2as

Where **as** is WSO2 Application Server cartridge, **mywso2as** is alias/name for WSO2 Application cartridge.

And it give following output;

```
Subscribing to mywso2as cartridge : as with Alias mywso2as
You have successfully subscribed to as cartridge.
http://git.stratos.com/telecom.org/mywso2as.git repo Created!
Your application is being published here http://as.cartridge-test.wso2.com:8280
```

Note that you have to add an entry in your **/etc/hosts** file for **as.cartridge-test.wso2.com** in order to access the url shown in the output of the command.
In this case it is,
192.168.92.10 as.cartridge-test.wso2.com

SugarCRM Express Guide

Login to Stratos2.0 using CLI

$ ./stratos.sh admin@telecom.org password

```
Successfully Authenticated!
```

stratos>subscribe php mycrm -connect mysql -alias crmdb -repoURL *git-repo-url -repoUserName *username -repoPassword *password

***git-repo-url -** eg like https://github.com/lakwarus/sugarcrm.git

***username -** Username for provided git account

***password -** Password for provided git account

Where **php** is PHP cartridge, **mycrm** is alias/name for PHP application, **mysql** is MYSQL cartridge, **crmdb** is alias for mysql database cartridge and **https://github.com/lakwarus/sugarcrm.git** is ready-to-deploy Stratos2.0 SugarCRM repo.

And it give following output;

```
Subscribing to data cartridge : mysql with Alias crmDB
You have successfully subscribed to mysql cartridge.
Subscribing to mycrm cartridge and connecting with crmDB data cartridge
Success!
Your application is being published here http://mycrm.php.stratos.com:8280
(this might take a minute... depending on repo size)
```

Note that you have to add an entry in your **/etc/hosts** file for **mycrm.php.stratos.com** in order to access the url shown in the output of the command.
In this case it is

192.168.92.10 mycrm.php.stratos.com

- SugarCRM application is now ready to use. Default username and password are admin:admin

It may take sometime to enable the login.

Wordpress Guide

Login to Stratos2.0 using CLI

$ ./stratos.sh admin@telecom.org password

stratos>subscribe php myweb -repoURL *git-repo-url -repoUserName *username -repoPassword *password

Where **php** is PHP cartridge, **myweb** is alias/name for PHP application.

***git-repo-url -** eg like https://github.com/lakwarus/wordpress.git

***username -** Username for provided git account

***password -** Password for provided git account

Please note that Provided PHP cartridge configured to be expect following folder structure for work.

```
simplesamlphp  sql  www
```

Where www folder contain wordpress php files. Please look https://github.com/lakwarus/wordpress.git for sample.

Upload wordpress

Checkout the tenant repo into a folder of your choice

$ git clone http://git.stratos.com/telecom.org/myweb.git

When asked user name and password give the username and password of the tenant.

$ cd myweb

To list the folder

$ ls

```
 simplesamlphp  sql  www
```

copy wordpress/* files into www folder. Wordpress can be downloaded from here [2]

$ git add www/*

$ git commit -a -m "initial commit"

$ git push

If git push take long time with the following output

```
remote: 100   357   0      0  100   357         0      6  0:00:59  0:00:52  0:00:07
```

do a **ctrl+C** to finish it(this is a known issue)

Setting up database

stratos>subscribe mysql mywebdb

Where **mysql** is MYSQL cartridge, **mywebdb** is alias for mysql database cartridge

stratos> info mywebdb

```
Cartridge Info
-----------------
Cartridge: mysql
Alias: mywebdb
Host: 192.168.92.67
Password: cilxskqh
Admin URL: https://mywebdb.mysql.stratos.com
Status: ACTIVE
```

```
Active Instances: 1
```

Goto https://mywebdb.mysql.stratos.com (Login using above info. Note that the username is root)

Note that you have to add an entry in your **/etc/hosts** file for **mywebdb.mysql.stratos.com** in order to access the url shown in the output of the command.
In this case it is

ELB IP Address mywebdb.mysql.stratos.com

Create a database

For that in the phpmyadmin click the **Databases** tab and under **Create new database** box give your database name(eg. mywebdb)

Goto **http://myweb.php.stratos.com:8280**

This request is sent to the WSO2 elastic load balancer. You access your application running in PHP cartridge through the load balancer.
Note that you have to add an entry in your **/etc/hosts** file for **myweb.php.stratos.com** in order to access the url shown in the output of the command.
In this case it is

192.168.92.10 myweb.php.stratos.com

Run installer

Give the details that you get when executing **info mywebdb** command and click submit button.
For this example.

Database Name : mywebdb
User Name : root
Password : cilxskqh
Database Host : 192.168.92.67

Now **http://myweb.php.stratos.com:8280** website is ready for use. You can continue installing the wordpress and after successful install you will be presented with the login page. Under the Known Issues section, see jira issue SPI-21(https://wso2.org/jira/browse/SPI-21). However when you enter **http://myweb.php.stratos.com:8280** again in the browser you can log into the wordpress page.

For issues related look at Known Issues section.

PHP cartridge and Identity Server Integration Guide

Login to Stratos 2.0 using CLI

$ ./stratos.sh admin@telecom.org password
```
Successfully Authenticated!
```

stratos>subscribe php mysaml -repoURL git-repo-url -repoUserName *username -repoPassword *password

Where **php** is PHP cartridge, **myweb** is alias/name for PHP application
***git-repo-url -** eg like https://github.com/lakwarus/wordpress.git
***username -** Username for provided git account
***password -** Password for provided git account

Upload simplesamlphp

From a folder of your choice do
$ git clone http://git.stratos.com/telecom.org/mysaml.git
When asked user name and password, give the username and password of the tenant. User name should be your tenant domain(eg:admin@stratos.com)
$ cd mysaml
$ ls
```
simplesamlphp  sql  www
```

copy **mysaml/*** files into **simplesamlphp** folder. Simplesamlphp can download from here [3]
Set following configuration at simplesamlphp/metadata/saml20-idp-remote.php

```
$metadata['https://idp.stratos.com:8243/samlsso'] = array(
    'name' => array(
        'en' => 'WSO2 IS - guest users',
    ),
    'description'        => '',

 'SingleSignOnService'  => 'https://idp.stratos.com:8243/samlsso',
 'SingleLogoutService'  => 'https://idp.stratos.com:8243/samlsso',
 'certFingerprint'     => '6bf8e136eb36d4a56ea05c7ae4b9a45b63bf975d',
```

```
 /*'certFingerprint'     => '3B:B1:35:E0:B1:35:BB:83:26:F5:00:F6:62:8D:67:B1:DC:98:82:BA'*/
 /*'certificate'       =>'server.pem'*/
);
```

$ git add *
$ git commit -a -m "simplesamlphp commit"
$ git push

If git push take long time with the following output

```
remote: 100   357   0      0  100   357        0      6  0:00:59  0:00:52  0:00:07
```

do a **ctrl+C** to finish it(this is a known issue)

Login to IS using tenant info

Goto https://idp.stratos.com:8243/
Note that you have to add an entry in your **/etc/hosts** file for **idp.stratos.com** in order to access the url shown in the output of the command.
In this case it is

```
192.168.92.10      idp.stratos.com
```

Login details:
user: admin@telecom.org
password:password
Goto SAML SSO
Add new service provider with following details

-- Issuer
-- Assertion Consumer URL
-- ON Enable Assertion Signing
-- ON Enable Single Logout

Below it describe how you can get above details.

Getting SP Metadata

Goto **http://mysaml.php.stratos.com:8280/simplesaml**

Note that you have to add an entry in your **/etc/hosts** file for **mysaml.php.stratos.com** in order to access the url shown in the output of the command.
In this case it is

```
192.168.92.10      mysaml.php.stratos.com
```

Click on Show metadata on Federation tab
It will provide required details for create new service provider
Issuer: http://mysaml.php.stratos.com:8280/simplesaml/module.php/saml/sp/saml2-acs.php/default-sp
Assertion Consumer URL:
http://mysaml.php.stratos.com:8280/simplesaml/module.php/saml/sp/metadata.php/default-sp
Custom Logout URL :
http://mysaml.php.stratos.com:8280/simplesaml/module.php/saml/sp/saml2-logout.php/default-sp

Test SSO

Log out from IS Session
Goto **http://mysaml.php.stratos.com:8280/simplesaml**
Goto **Thentication tab ->Test configured authentication sources -> default-sp** and press Select button

It will prompt IDP login page. Login using admin@telecom.org:password

If you got following issue

Copy highlighted and update simplesamlphp/metadata/saml20-idp-remote.php

```
$metadata['https://idp.stratos.com:8243/samlsso'] = array(
    'name' => array(
        'en' => 'WSO2 IS - guest users',
    ),
    'description'        => '',

 'SingleSignOnService'  => 'https://idp.stratos.com:8243/samlsso',
 'SingleLogoutService'  => 'https://idp.stratos.com:8243/samlsso',
 'certFingerprint'     => '58bf88d54f105a1e7f8d69ac1659033604a92fcd',
 /*'certFingerprint'     => '3B:B1:35:E0:B1:35:BB:83:26:F5:00:F6:62:8D:67:B1:DC:98:82:BA'*/
```

```
 /*'certificate'       =>'server.pem'*/
);
```

Then do

$ git commit -a -m "change cert fingerprint"
$ git push

Do SSO login test again.

See Known Issues section for issues related.

Deployment Synchronization with Github

This section describe how to use existing github repository with Stratos 2.0 Environment.

Login to Stratos2.0 using CLI

$./stratos.sh admin@telecom.org password

stratos>subscribe php myphpapp -repoURL *git-repo-url -repoUserName *username -repoPassword *password

***git-repo-url -** Git repo contain the myphpapp. eg https://github.com/lakwarus/myphpapp.git
***username -** Username for provided git account
***password -** Password for provided git account

myphpapp application is now ready to use. Now you can add your own domain url to access your application. It is described in the following section.

Custom Domain Mapping

With the above example we show how you can use the custom domain mapping feature of Stratos2. Suppose that you need to give your own domain, “myphp.com” to access the application at `http://myphpapp.php.stratos.com:8280`

At the time of subscription you do not have to provide own domain information. It is only after subscription you can add the domain mapping using cli tool.

stratos> domain-mapping as myphp.com

Output will display the Stratos domain you have to CNAME for.

If you don't want to register a CNAME now, you can add following entry to the **/etc/hosts** file for testing purposes.

192.168.92.10 myphp.com

You can access the applications using

`http://myphp.com:8082`

So you have successfully registered your own domain with Stratos2.

Now you can upload your applications to your own github repo at
https://github.com/lakwarus/myphpapp.git

In order to inform Stratos2 about your application, you need to do the following.

Setting Service Hook at Github

Now for github to inform Stratos2 ADC about the applications deployed there, goto github website and select **repository**
Goto **setting -> Service Hooks -> WebHook** URL and add ***ADC Notification Service** and do update settings.

***ADC Notification Service**

Autoscaling User Guide

How can I provide the autoscaling related parameters?

We provide set of parameters to calibrate the elasticity of the system.

- For WSO2 Carbon services, you have to define the autoscaling related parameters before WSO2 ELB starts up (we'll support the dynamic behaviour for Carbon servers soon).
- For non-Carbon services, you can define autoscaling parameters when subscribing to a Cartridge.

What are the parameters and what they mean?

There are few of them and all of the vital ones are configurable using **loadbalancer.conf** file for Carbon services (sample configuration files are provided at the end of this document.) and at the time of subscription for non-Carbon services.

autoscaler_task_interval (t) - time period between two iterations of 'autoscaling decision making' task. When configuring this value, you are advised to consider the time 'that a service instance takes to join ELB'. This is in milliseconds and the default value is 30000ms.

max_requests_per_second (Rps) - number of requests, a service instance can withstand per a second. It is recommended that you calibrate this value for each service instance and may also for different scenarios. Ideal way to estimate this value could be by load testing a similar service instance. Default value is 100.

rounds_to_average (r) - an autoscaling decision will be made only after this much of iterations of 'autoscaling decision making' task. Default value is 10.

alarming_upper_rate (AUR)- without waiting till the service instance reach its maximum request capacity (alarming_upper_rate = 1), we scale the system up when it reaches the request capacity, corresponds to alarming_upper_rate. This value should be 0<AUR<=1 and default is 0.7.

alarming_lower_rate (ALR) - lower bound of the alarming rate, which gives us a hint; that we can think of scaling down the system. This value should be 0<ALR<=1 and default is 0.2.

scale_down_factor (SDF) - this factor is needed in order to make the scaling down process slow. We need to scale down slowly to reduce scaling down due to a false-positive event. This value should be 0<SDF<=1 and default is 0.25.

Can I set a limit to the number of service instances that are maintained in the system, at any given time?

Yes, you can set the **min_app_instances** for any service cluster and the autoscaler will make sure that the system will not scale down below that, even though there is no considerable service requests in-flight.

Do I have a control over the number of instances that the autoscaler can start?

Yes, you can set the **max_app_instances** for any service cluster and the autoscaler will make sure that the system will not scale up above that limit, even though there is a high load of requests in-flight. This is useful, especially when you pay for the instances you start up.

How can I do a simple autoscaling test?

Let's consider a 'PHP Cartridge' case. Load Stratos-2 CLI tool and subscribe to 'PHP' Cartridge as follows.

stratos>subscribe php nirmal -min 1 -max 5

This will result in starting up a php service instance for you along with a GIT repo.

Push a php application to the GIT repository, created just for you. For testing purposes, you can add a php application, which does nothing other than sleeping for 30 seconds. Seconds after committing your app, you should be able to access it.

Now write a small jmeter test script to load your PHP application.

After some time you should see that the nodes are scaling up (given that you loaded the PHP application heavily) and also when the load test is over, the extra nodes should scaling down.

Sample configuration files

Properties defined in the defaults section

```
loadbalancer {
        # minimum number of load balancer instances
        instances           1;
        # whether autoscaling should be enabled or not.
        enable_autoscaler   true;
        #please use this whenever url-mapping is used through LB.
```

```
        #size_of_cache              100;
        # autoscaling decision making task
        autoscaler_task
org.wso2.carbon.mediator.autoscale.lbautoscale.task.ServiceRequestsInFlightAutoscaler;
        # End point reference of the Autoscaler Service
        autoscaler_service_epr <autoscaler_service_epr>;
        # interval between two task executions in milliseconds
        autoscaler_task_interval 30000;
        # after an instance booted up, task will wait maximum till this much of time and let the server started up
        server_startup_delay 60000; #default will be 60000ms
        # session time out
        session_timeout 90000;
        # enable fail over
        fail_over true;
}

# services' details which are fronted by this WSO2 Elastic Load Balancer
services {
        # default parameter values to be used in all services
        defaults {
        # minimum number of service instances required. WSO2 ELB will make sure that this much of instances
        # are maintained in the system all the time, of course only when autoscaling is enabled.
        min_app_instances       1;
        # maximum number of service instances that will be load balanced by this ELB.
        max_app_instances       3;
        max_requests_per_second  5;
        rounds_to_average       2;
        alarming_upper_rate 0.7;
        alarming_lower_rate 0.2;
        scale_down_factor 0.25;
        message_expiry_time     60000;
        }


        appserver {
        hosts   appserver.cloud-test.wso2.com;
        domains  {
                3.appserver.domain {
                tenant_range    *;
```

```
                min_app_instances          0;
                }
        }
        }

}
```

Properties defined within the service element

```
loadbalancer {
        # minimum number of load balancer instances
        instances           1;
        # whether autoscaling should be enabled or not.
        enable_autoscaler   true;
        #please use this whenever url-mapping is used through LB.
        #size_of_cache              100;
        # autoscaling decision making task
        autoscaler_task
org.wso2.carbon.mediator.autoscale.lbautoscale.task.ServiceRequestsInFlightAutoscaler;
        # End point reference of the Autoscaler Service
        autoscaler_service_epr <autoscaler_service_epr>;
        # interval between two task executions in milliseconds
        autoscaler_task_interval 30000;
        # after an instance booted up, task will wait maximum till this much of time and let the server started up
        server_startup_delay 60000; #default will be 60000ms
        # session time out
        session_timeout 90000;
        # enable fail over
        fail_over true;
}

# services' details which are fronted by this WSO2 Elastic Load Balancer
services {
        # default parameter values to be used in all services
        defaults {
                # minimum number of service instances required. WSO2 ELB will make sure that this much of
        instances
                # are maintained in the system all the time, of course only when autoscaling is enabled.
```

```
                min_app_instances           1;
                # maximum number of service instances that will be load balanced by this ELB.
                max_app_instances           3;
                max_requests_per_second   5;
                rounds_to_average           2;
                alarming_upper_rate 0.7;
                alarming_lower_rate 0.2;
                scale_down_factor 0.25;
                message_expiry_time         60000;
        }


        appserver {
                hosts     appserver.cloud-test.wso2.com;
                domains   {
                        3.appserver.domain {
                                tenant_range       *;
                                min_app_instances           0;
                                max_requests_per_second   5;
                                alarming_upper_rate 0.6;
                                alarming_lower_rate 0.1;
                        }
                }
        }

}
```

Sample CLI command

stratos>subscribe <cartridge-type> <alias> -min <min-count> -max <max-count>

stratos>subscribe php nirmalphp -min 1 -max 5

Troubleshooting

Stratos2 Core services are not started

If your Stratos2 setup is running properly there should be four java processes running when the script ends. These are ELB, S2 Agent, SC and CC. If any of these servers are not up, first check whether the pack **wso2s2-openstack-1.0.0-1.0.0.zip** is available in the downloaded location. If not, you have to manually start the servers. In the case of EC2, servers are at /mnt/<folder_with_IP_name>/ , where as in the OpenStack setup they are at /opt/.

Started Carbon cartridge instance has not joined ELB

This could be happened due to the ports of the IaaS security group that should be opened are not opened. You can find the ports that should be opened from user guide and open them.

Known Issues

PHP cartridge and Identity Server Integration Guide

For Known issues in IS
https://wso2.org/jira/browse/IDENTITY-591
For the above issue, there is a patch you have to apply for simplesamlphp library. In ./lib/SAML2/HTTPPost.php file, receive function, comment the following line

$msg = base64_decode($msg);

References

[1]http://damithakumarage.wordpress.com/2013/03/20/easy-installation-of-openstack-essex-on-a-single-node/
[2]http://wordpress.org/latest.tar.gz
[3]http://simplesamlphp.org/download