http://wso2.com/

Stratos 2.0 Installation Guide

Date: 05 April 2013

Email: support@wso2.com

Stratos 2.0 Installation Guide

Introduction

The aim of this document is to capture the Stratos 2.0 deployment methodologies in OpenStack and EC2. The deployment is described for a setup delivered as a **wso2s2-openstack-1.0.0.zip** which contains a shell script based deployment automation for OpenStack and a puppet based automation for deployment as **wso2s2-ec2-1.0.0.zip** in Amazon EC2. Apart from these two deployment methodologies we also provide a **Public Stratos 2.0 Setup on Amazon EC2** to play around with Stratos 2.0 straight away.

All these deployment artifacts contains the functional components of WSO2 Stratos 2.0 which are WSO2 Stratos Controller(SC), WSO2 Elastic Load Balancer(ELB), WSO2 Cloud Controller(CC), WSO2 Stratos Agent (S2Agent) and Interactive Command Line Interface (CLI).

Conventions used in this document

- Configuration file entries are written as in the following example.

```
This is a configuration file entry
```

- Commands typed into linux shell and linux bash scripts are shown as in the following example.

$ example_command parameters

-**Important:** This is very important fact

When something need to be highlighted within a sentence the text is written in **bold**

What you need

In this installation guide for OpenStack we will install Stratos 2.0 specific stuff in one physical node. In a real deployment you can follow the same guidelines to install these servers in different nodes as deemed suitable. The node should have at least 4G memory, 40G hard disk space and 1 cpu of 2.8GHz. We need 64bit Ubuntu 12.04 server installed in the node.

Deployment Options

OpenStack

- For Openstack we use a single node, where Openstack controller and Openstack compute is installed in the same node. You can install WSO2 Stratos 2.0 servers in to the same single node or on a separate node. In this guide we will install them in the same node as the Openstack installation.

- Please refer the OpenStack installation documentation at [1] in order to install OpenStack Iaas. OpenStack installation and deployment of Stratos 2.0 is thoroughly tested on Ubuntu 12.04.

- Download wso2s2-openstack-1.0.0.zip and extract it on your OpenStack installed server and Follow README.txt

- Install Java in your server by copying the JDK to any prefered location and by exporting environment variable JAVA_HOME to point to your java runtime. We recommend to use Java 6.0.

  export JAVA_HOME="/opt/jdk1.6.0_24"

- Download mysql-connector-java-5.1.17-bin.jar and copy the jar inside wso2s2-openstack-1.0.0. If you intend to use a different mysql connector version then make

sure you update the

```
export mysql_connector_jar="mysql-connector-java-5.1.17-bin.jar"
```

line in wso2s2-openstack-1.0.0/conf/setup.conf file.

- Next the user has to create a Public Key to use with OpenStack, This can be done by using the following command.

$ ssh-keygen -t rsa

For the ease of use make sure that you do not enter any passwords while creating the public key, just press ENTER.

- User can check the created public key as follows.

$ cat ~/.ssh/id_rsa.pub

- Edit wso2s2-openstack-1.0.0/conf/setup.conf according to your environment. There you need to give information about your OpenStack environment.

Make sure that you give the following information in the wso2s2-openstack-1.0.0/conf/setup.conf .

`openstack_public_key="/.ssh/id_ras.pub"` (file path of the public key location).

Make sure conf/setup.conf is according to your environment. Specially “General configuration” section and “Openstack” section. You can accept the default for most of the parameters.

- Go inside wso2s2-openstack-1.0.0 and run setup.sh (as given below) as root user to install Stratos2 Paas with Openstack demo. When you enable Stratos2 demo it will upload images of all out of the box Carbon and Non Carbon cartridges provided into the glance image server.

sudo JAVA_HOME=/opt/jdk1.6.0_24 ./setup.sh -d openstack -p"all"

You can see the logs at

$ tail -f /var/log/s2/s2.log
$ tail -f /var/log/s2/s2-openstack.log

- If you want to re-do the Stratos2 setup again by running the setup.sh, first you need to clean the existing setup by using clean.sh and then if you enabled Stratos2 Openstack demo clean_openstack.sh. The commands are as below.

  sudo ./clean.sh -a<hostname> -b<host_user> -c<Openstack mysql root user> -d<Openstack mysql password>

  eg.
  sudo ./clean.sh -as2demo -bwso2 -croot -dopenstack
  sudo ./clean_openstack.sh

Amazon EC2

For Amazon EC2 we provide a Puppet based deployment mechanism for Stratos 2. Puppet is an IT automation software that helps system administrators to manage infrastructure throughout its lifecycle. For EC2 IaaS you need access to EC2 environment by creating an Amazon AWS account.

Users can follow the given below steps to install Stratos2 in Amazon EC2 using this Puppet based deployment methodology.

Setting up puppet master (with nginx and mongrel)

- Installing puppet master packages (puppetmaster 2.7 , mongrel, nginx)

  apt-get install puppetmaster mongrel nginx

- Following commands are used to stop puppetmaster and nginx

  /etc/init.d/puppetmaster stop
  /etc/init.d/nginx stop

- Apply following changes as given, respectively to the configuration files mentioned.

  /etc/default/puppetmaster :

```
DAEMON_OPTS="--ssl_client_header=HTTP_X_SSL_SUBJECT"
SERVERTYPE=mongrel
PUPPETMASTERS=5
PORT=18140
```

/etc/pupet/puppet.conf :

```
[main]
logdir=/var/log/puppet
vardir=/var/lib/puppet
ssldir=/var/lib/puppet/ssl
rundir=/var/run/puppet
factpath=$vardir/lib/facter
templatedir=/etc/puppet/templates/
modulepath=/etc/puppet/modules/
report=false

[stratos2]
modulepath=/mnt/puppet/stratos2/modules/
templatedir=/mnt/puppet/stratos2/templates/
manifest=/mnt/puppet/stratos2/manifests/site.pp
manifestdir=/mnt/puppet/stratos2/manifests/
report=false

[master]
node_name = facter
```

/etc/puppet/autosign.conf :

```
*.wso2.com
```

/etc/puppet/auth.conf :

```
path ~ ^/catalog/([^/]+)$
method find
allow $1
allow star.s2.wso2.com

path ~ ^/node/([^/]+)$
method find
allow $1

path /certificate_revocation_list/ca
```

```
method find
allow *

path /report
method save
allow *

path /file
allow *

path /file_metadata
auth any
method find
allow *

path /certificate/ca
auth no
method find
allow *

path /certificate/
auth no
method find
allow *

path /certificate_request
auth no
method find, save
allow *

path /
environment stratos2
allow *

path /
auth any
```

/etc/puppet/fileserver.conf :

```
[files]
  path /etc/puppet/
  allow *

[plugins]
  allow *
```

```
[stratos2]
 path /mnt/puppet/stratos2/
 allow *
```

- Configurations for Nginx is as below.

/etc/nginx/nginx.conf:

```
user www-data;
worker_processes 5;

error_log /var/log/nginx/error-puppet.log;
pid /var/run/nginx.pid;

events {
       worker_connections  1024;
}

http {
       default_type  application/octet-stream;

sendfile       on;
tcp_nopush     on;

keepalive_timeout  120;
tcp_nodelay        on;

upstream puppetmaster {
   server 127.0.0.1:18140;
   server 127.0.0.1:18141;
   server 127.0.0.1:18142;
   server 127.0.0.1:18143;
   server 127.0.0.1:18144;
}

server {
listen                  8140;
ssl                     on;

ssl_certificate
/var/lib/puppet/ssl/certs/s2demo.s2.wso2.com.pem;
ssl_certificate_key
/var/lib/puppet/ssl/private_keys/s2demo.s2.wso2.com.pem;
```

```
  ssl_client_certificate  /var/lib/puppet/ssl/ca/ca_crt.pem;
  ssl_crl                 /var/lib/puppet/ssl/ca/ca_crl.pem;

  ssl_ciphers             SSLv2:-LOW:-EXPORT:RC4+RSA;
  ssl_session_cache       shared:SSL:8m;
  ssl_session_timeout     5m;
  ssl_verify_client       optional;
  rewrite_log             on;
  client_max_body_size        50m;
  root                    /var/empty;
  access_log              /var/log/nginx/access.log;

  location / {
proxy_pass          http://puppetmaster;
  proxy_redirect      off;
  proxy_set_header    Host             $host;
  proxy_set_header    X-Real-IP        $remote_addr;
  proxy_set_header    X-Forwarded-For  $proxy_add_x_forwarded_for;
  proxy_set_header    X-Client-Verify  $ssl_client_verify;
  proxy_set_header    X-SSL-Subject    $ssl_client_s_dn;
  proxy_set_header    X-SSL-Issuer     $ssl_client_i_dn;
                         proxy_read_timeout  120;
                }
        }
  }
```

- Set server host name to s2demo.s2.wso2.com (if you want to change the domain name make sure to change the nginx ssl settings) add the host entry to /etc/hosts accordingly

```
hostname s2demo.s2.wso2.com
hostname > /etc/hostname
```

- Starting the puppet master

/etc/init.d/puppetmaster start
/etc/init.d/nginx start

Setting up puppet agent

- Installing packages

$apt-get install puppet

- Generating the keys in the server (puppet master)

  $puppet cert --generate star.s2.wso2.com

  Copy the generated key in /var/lib/puppet/ssl/private_keys/**star.s2.wso2.com.pem** to clients (pupet agent) /var/lib/puppet/ssl/private_keys/

  Copy the generated key in /var/lib/puppet/ssl/ca/signed/**star.s2.wso2.com.pem** to clients /var/lib/puppet/ssl/certs/

- Change the **/etc/puppet/puppet.conf** as below.

```
[main]
logdir=/var/log/puppet
vardir=/var/lib/puppet
ssldir=/var/lib/puppet/ssl
rundir=/var/run/puppet
factpath=$vardir/lib/facter
templatedir=$confdir/templates
server=s2demo.s2.wso2.com
waitforcert=60
report=false

[master]
environment=stratos2
modulepath=/etc/puppet/$environment/modules
templatedir=/etc/puppet/$environment/templates
manifest=/etc/puppet/$environment/manifests/site.pp
manifestdir=/etc/puppet/$environment/manifests/

[agent]
environment=stratos2
certname = star.s2.wso2.com
node_name = facter
node_name_fact = fqdn
```

Setting up Stratos2

- Create a folder path for stratos2 puppet manifests.

  $ mkdir -p /mnt/puppet/stratos2

- Extract the puppet master files (manifests, modules,templates) to /mnt/puppet/stratos2

- Change the hosts template file in /mnt/puppet/stratos2/templates/hosts.erb.

Public Stratos 2.0 Setup on Amazon EC2

We provide preconfigured EC2 image which is publicly accessible through the EC2. Please refer user guide for more information how to use the preconfigured image to run stratos2.

- Stratos 2.0 is tested on VMWare vCloud, to work on you need vCloud v1.5. You may use the vCloud API SDK v5.1, or the dedicated API SDK v1.5.

Command Line Interface (CLI)

You need to download the wso2s2cli-1.0.0.zip which is the CLI tool and unarchive it into any of the machines that you use to interact with the Stratos2 cloud.

Export the following variables

```
$ export STRATOS_SC_HOST=192.168.92.11
$ export STRATOS_SC_PORT=9445
```

You also need ***java 1.6*** or later in your path

```
$ unzip wso2s2cli-1.0.0.zip
$ cd wso2s2cli-1.0.0
```

Before using the CLI tool you need to create a tenant using the Stratos Manager user interface at **https://s2manager.wso2.com:9445/**

You need an entry in your **/etc/hosts** file in the machine your browser run, which map **s2manager.wso2.com** to the dns/ip of the ec2 or virtualbox machine.

Connect using
Username : admin
Password : admin

Please refer to the Stratos 2.0 User Guide that ship along with this document on detail information on how to use the CLI tool.

Stratos Manager User Interface

Stratos manager user interface can be used to interact with the Strato2 cloud in much the same way as the CLI tool. However Stratos Manager provide additional functionalities like registering the tenants for Stratos2 and extended facilities for using WSO2 carbon products.

Configuration in Detail

Stratos IaaS configuration file (repository/conf/cloud-controller.xml)

Stratos DevOps edits cloud-controller.xml to configure IaaS. This is more of a stratos level configuration file.

```
<cloudController xmlns:svns="http://org.wso2.securevault/configuration">

<svns:secureVault
provider="org.wso2.securevault.secret.handler.SecretManagerSecretCallbackHandl
er" />

      <dataPublisher enable="false">
      <!-- BAM Server Info - default values are 'admin' and 'admin'
                        Optional element. -->
                  <bamServer>
      <!-- BAM server URL should be specified in carbon.xml -->
                        <adminUserName>admin</adminUserName>
                        <adminPassword
svns:secretAlias="cloud.controller.bam.server.admin.password">admin</adminPass
word>
```

```
                </bamServer>
<!-- Default cron expression is '1 * * * * ? *' meaning 'first second of every
minute'.Optional element. -->
                <cron>1 * * * * ? *</cron>
                <!-- Cassandra cluster related info -->
                <!--cassandraInfo>
                      <connectionUrl>localhost:9160</connectionUrl>
                      <userName>admin</userName>
                      <password
svns:secretAlias="cloud.controller.cassandra.server.password">admin</password>
                </cassandraInfo-->
      </dataPublisher>

      <topologySync enable="true">
                <!-- MB server info -->
                <mbServerUrl>MB_HOSTNAME:MB_LISTEN_PORT</mbServerUrl>
                <cron>1 * * * * ? *</cron>
      </topologySync>

<!-- Specify the properties that are common to an IaaS here. This element
 is not necessary [0..1]. But you can use this section to avoid specifying
 same property over and over again. -->

<iaasProviders>
     <EC2_PROVIDER_STARTiaasProvider type="ec2" name="ec2 specific details">

<className>org.wso2.carbon.stratos.cloud.controller.iaases.AWSEC2Iaas</classNa
me>
    <provider>aws-ec2</provider>
    <identity
svns:secretAlias="elastic.scaler.openstack.identity">EC2_IDENTITY</identity>
   <credential
svns:secretAlias="elastic.scaler.openstack.credential">EC2_CREDENTIAL</credent
ial>
                      <scaleUpOrder>EC2_SCALEUP_ORDER</scaleUpOrder>
                      <scaleDownOrder>EC2_SCALEDOWN_ORDER</scaleDownOrder>
                      <property name="jclouds.ec2.ami-query"
value="owner-id=XX-XX-XX;state=available;image-type=machine"/>
            <property name="availabilityZone" value="EC2_AVAILABILITY_ZONE"/>
                      <property name="securityGroups"
value="EC2_SECURITY_GROUPS"/>
            <property name="instanceType" value="EC2_INSTANCE_TYPE"/>
            <property name="keyPair" value="EC2_KEYPAIR"/>
                <imageId>EC2_IMAGE_ID</imageId>
                </iaasProviderEC2_PROVIDER_END>
                <iaasProvider type="openstack" name="openstack specific
details">
```

```
<OPENSTACK_PROVIDER_STARTclassName>org.wso2.carbon.stratos.cloud.controller.ia
ases.OpenstackNovaIaas</className>
                        <provider>openstack-nova</provider>
                        <identity
svns:secretAlias="cloud.controller.openstack.identity">OPENSTACK_IDENTITY</ide
ntity>
                        <credential
svns:secretAlias="cloud.controller.openstack.credential">OPENSTACK_CREDENTIAL<
/credential>
                        <property name="jclouds.endpoint"
value="OPENSTACK_ENDPOINT" />
            <property name="jclouds.openstack-nova.auto-create-floating-ips"
value="false"/>
                        <property name="jclouds.api-version" value="2.0/" />
                        <scaleUpOrder>OPENSTACK_SCALEUP_ORDER</scaleUpOrder>

<scaleDownOrder>OPENSTACK_SCALEDOWN_ORDER</scaleDownOrder>
                        <property name="X" value="x" />
                        <property name="Y" value="y" />
                        <imageId>OPENSTACK_IMAGE_ID</imageId>
                  </iaasProviderOPENSTACK_PROVIDER_END>
      </iaasProviders>

</cloudController>
```

Cartridge configuration files

For cartridge configuration there can be a single file or several files Cartridge Deployer creates <cartridge>.xml to capture metadata about cartridge including images id's

(Eg.repository/deployment/server/cartridges/php.xml)

```
<!-- Use below section to specify properties that are needed in order to start
Cartridges.  -->
      <cartridges>

      <!-- You can have 1..n cartridge elements. -->
      <cartridge type="php" host="php.STRATOS_DOMAIN" provider="zend-provider"
version="5.5" multiTenant="false">
<!-- cartridge element can have 0..n properties, and they'll be overwritten by
the properties specified under iaasProvider child elements of cartridge
element. -->

            <displayName>PHP</displayName>
            <description>PHP Cartridge</description>
```

```
<!-- A cartridge element should add a reference to an existing IaaS provider
(specified in the above &lt;iaasProviders&gt; section) or it can create a
completely new IaaS Provider (which should have a unique "type" attribute. -->

            <iaasProvider type="openstack" >
                  <imageId>nova/3a4000c8-0973-421a-94f8-dbfe8b7b5250</imageId>
                  <property name="keyPair" value="stratos-demo"/>
                  <property name="instanceType" value="nova/1"/>
                  <property name="securityGroups" value="default"/>
                  <!--<property name="payload" value="resources/as.txt"/>-->
            </iaasProvider>

            <!--<iaasProvider type="ec2" >
                  <imageId>us-east-1/ami-ef49e786</imageId>
                  <property name="keyPair" value="aa"/>
                  <property name="securityGroups" value="default"/>
                  <property name="instanceType" value="m1.large"/>
                  <property name="payload" value="resources/as-ec2.txt"/>
            </iaasProvider>-->

            <deployment baseDir="/var/www">
                  <dir>www=copy#app#files#here</dir>
                  <dir>simplesamlphp=copy#saml#libraries#here</dir>
                  <dir>sql=copy#saml#libraries#here</dir>
            </deployment>

            <portMapping>
            <http port="80" proxyPort="8280"/>
            <https port="443" proxyPort="8243"/>
            </portMapping>

            <!--<appTypes>
                  <property name="axis2services" isBothmapping="false"/>
                  <property name="webapps" isBothmapping="true"/>
                  <property name="jaxwebapps" isBothmapping="true"/>
                  <property name="jaggeryapps" isBothmapping="true"/>
            </appTypes>-->
      </cartridge>

      </cartridges>
```

For carbon cartridges, cartridge type should be the same as the value for CartridgeAlias element in repository/conf/carbon.xml file in the carbon cartridge.

Services configurations files

Service configurations can be a single file or several files. Includes domain, sub-domain, and cartridge type per service.

(Eg. repository/deployment/server/services/as.xml)

```
<service domain="wso2.appserver.domain" subDomain="__$default"
tenantRange="*">
       <cartridge type="appserver"/>
       <host>appserver.s2.wso2.com</host>

<payload>/opt/wso2cc-1.0.0/repository/resources/payload/as-default.zip</payloa
d>
</service>

Again for carbon cartridges, cartridge type should be the same as the value
for CartridgeAlias element in repository/conf/carbon.xml file in the carbon
cartridge.
```

Setting up a gitblit server and creating git repositories in an EC2 instance

1. Download Gitblit Go from http://gitblit.com/, and upload the zip file to the relevant EC2 instance.
2. Unzip and open the configuration guide data/gitblit.properties.
3. Set server.httpPort = 8280 (or a suitable port for http). The properties server.httpBindInterface and server.httpsBindInterface should be empty.
4. Navigate to the parent directory and run the command java -jar gitblit.jar --baseFolder data. Now you can access the gitblit server from <ec2 ip>:<http port>. For complete set up guide, refer http://gitblit.com/setup.html
5. Create a repository and add a user to the repository. The URL of the repository can be obtained by clicking on the repository name. Please note that currently only **http external**

git repository urls are supported, and the url should not contain the user name. (example git repo url: http://75.101.228.144:8080/git/s2-beta-test.git. Here, 75.101.228.144 is the ip of the instance and 8080 is the http port)

6. The repository should be initialized by cloning it into your local machine, adding an initial commit and pushing it to the repository. (for an example, a README file). Else, the pull commands will fail.

Putting a post commit hook to a repositories:

A post commit hook should be added to the git repository to notify commit events to Stratos Controller. Gitblit supports groovy scripts for execution after an artefact is pushed to the repository.

1. Open data/gitblit.properties and add groovy.scriptsFolder = <path_to_folder_with_script>. This folder should contain the groovy script. Script should have the .groovy extension.
2. Add the script name to groovy.postRecieveScripts = <name_of_script>. This will add scripts common to all repositories. To add repository specific scripts, remove the entry from groovy.postRecieveScripts, select edit repository --> hook scripts to select the relevant script. For more details, refer Groovy Hook Scripts section in http://gitblit.com/setup.html.
3. The script should contain a curl call with the repository URL to the repository notification service in Stratos Controller. A sample script is shown below:

```
#!/usr/bin/env groovy

def response = "curl -k -X POST
https://ec2-75-101-228-144.compute-1.amazonaws.com:9445/repo_notification -d
payload={'repository':{'url':'http://75.101.228.144:8080/git/s2-beta-test.git'}}".execute().text
println response
```

Here, the first bold section is the end point reference for the repository notification service, and the second is the json format string which includes the repository url.

Troubleshooting

While subscribing if you get a message similar to following,

```
The authenticity of host '172.14.1.3 (172.14.1.3)' can't be established.
ECDSA key fingerprint is 84:da:0c:75:0f:6d:62:80:89:be:05:ad:c2:c8:42:4f.
Are you sure you want to continue connecting (yes/no)?
```

* Make sure you have started servers as “wso2” user and server directories contain required permission to wso2 user.

References

[1] http://damithakumarage.wordpress.com/2013/03/20/easy-installation-of-openstack-essex-on-a-single-node/