Stratos 2.0 User Guide

Contents

Getting Started

Demo Setup

In this demo we show how to use Stratos2 interactive cli to subscribe to a php cartridge, which require a mysql cartridge for database access. Before anything we need to first setup our IaaS, that is Openstack running on a physical machine or on an Oracle virtualbox instance.

What do you need for the demo.

You need to download the s2demo.zip file downloadable along with the alpha pack. Note that this pack is separate from the alpha pack. It is a bundled up virtualbox image already installed with Openstack. It also contains the demo_setup folder which contains the software that goes with the alpha pack.

This s2demo pack is intended for someone who needs immediately get to speed up with the Stratos2. For someone who need to follow all the steps of installing Stratos2, alpha pack itself is the better option.

Setup the IaaS

Essentially you don't want to spend your time in setting up Openstack yourself as the virtualbox image is already installed with openstack. It also contains the demo_setup folder which we used to setup the demo.

Setup the Demo

You need virtualbox-4.1 or compatible version for running this image. Load the image into virtualbox and then start an instance.
Log into the instance as
username wso2
password g

%cd demo_setup

Edit the conf/setup.conf file and change the values according to your environment. Note that you can keep most of the values as it is.

Now execute the setup script

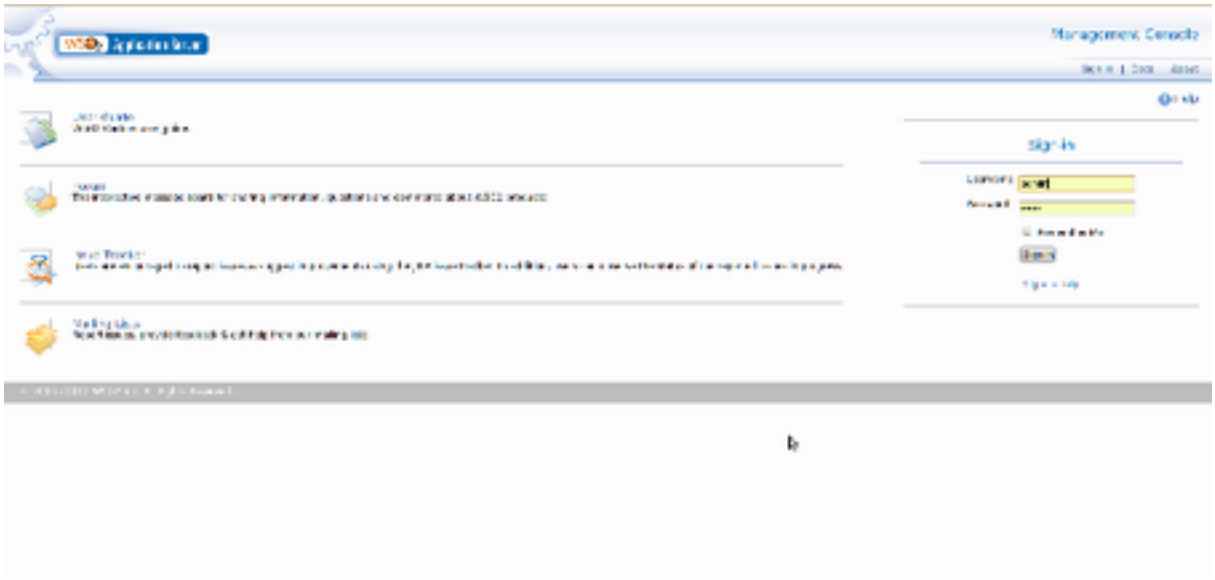
%./setup-demo.sh

This will configure all servers and start all servers. At the end of the script, you'll get an URL, which will be useful in tenant creation process, described in the next section.

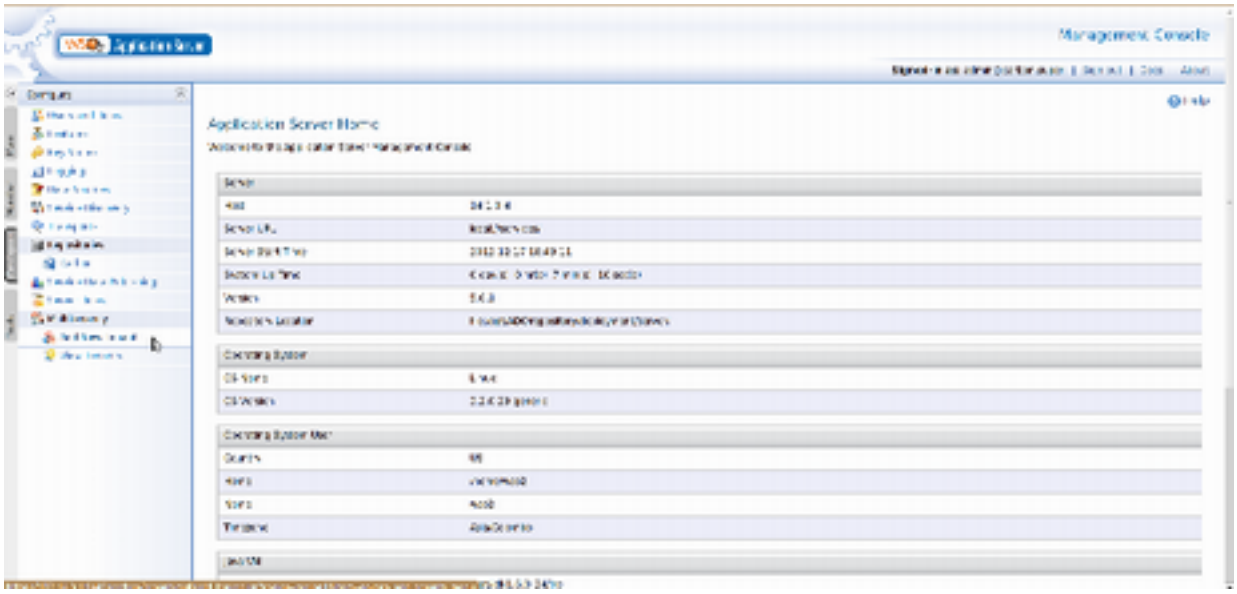
Tenant Creation

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

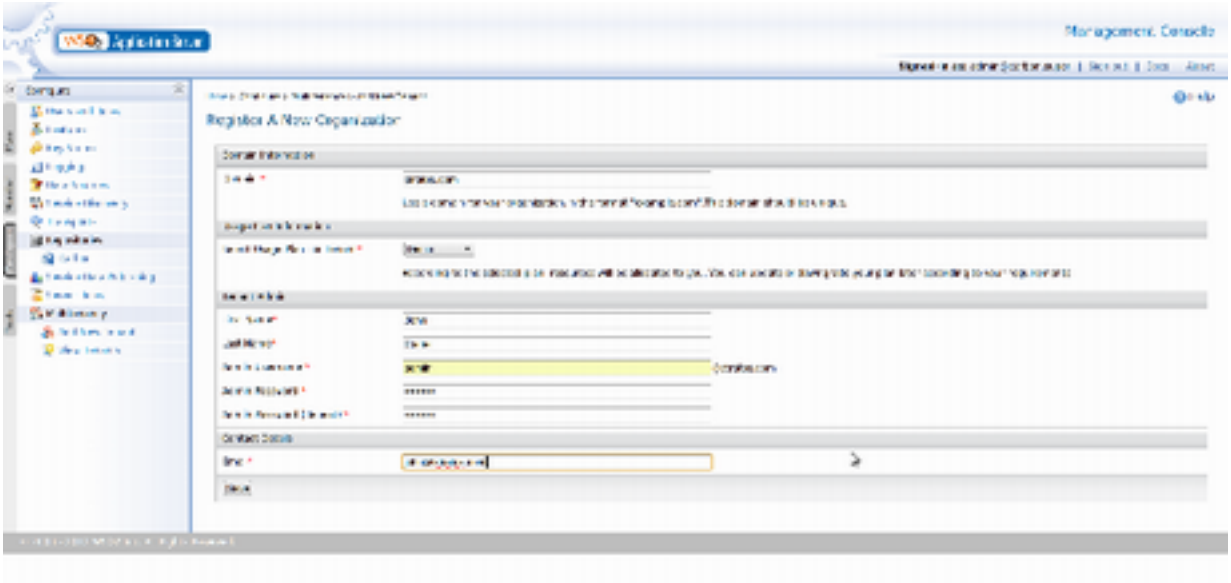
- access the URL obtained after running the 'setup-demo.sh' script. This will redirect you to a management console.



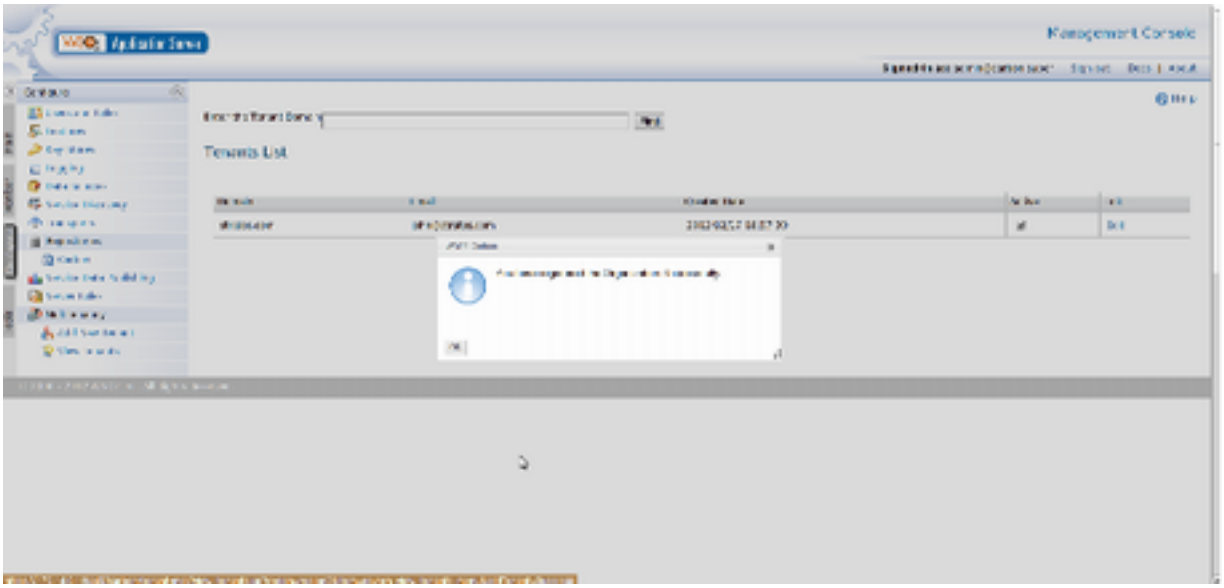
- Login to the management console using "admin" as the username and "admin" as the password.



- 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 “stratos.com” and corresponding ‘admin’ user “admin@stratos.com” is successfully created in Stratos 2 environment and you can proceed with the next steps of the Stratos 2 journey.

CLI Tool User Guide

You can start play with the demo setup. For that you need the interactive client CLI tool. You can find the CLI tool in /home/wso2/demo_setup/software/CLI.tar.gz file. Extract this to a place of your choice and then follow as below.

Usage

- You have to export the host and port of ADC as environment variables before using the CLI Tool.
  E.g. "export STRATOS_ADC_HOST=203.143.18.246 STRATOS_ADC_PORT=9445"

Console Mode

- In this mode you can call the jar to log-in to Stratos(ADC),
  java -cp <jar_name> <main_class> [username] [password]
  E.g. java -cp org.wso2.carbon.hosting.mgt.cli-1.0.0-Tool.jar org.wso2.carbon.hosting.mgt.cli.CliTool admin@stratos.com password

  If you wants to give the keystore path,
  java -cp <jar_name> <main_class> [username] [password] -keystore <path_to_keystore>

- You can provide a simple bash script to make this call more user friendly,
  E.g. Put following command in a script, say in stratos.sh,

  java -cp <jar_name> <main_class> $1 $2 $3 $4

  Then log-in call would be,

  sh stratos.sh [username] [password]
  or
  sh 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

- Logged-in user will be given console, stratos> (like mysql).
  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,

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>
-min : minimum number of instances. Default is 1
-max : maximum number of instances. Default is 1
-connect : Connect one cartridge to another. this need to use with -alias
eg: subscribe php myapp -connect mysql -alias mydb

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

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

Usage for action help:
help [action]

Single Call Mode

You can call the jar directly with action. It will not show the “stratos>” prompt and fulfill the request of command immediately for particular action command.

- In this mode he can call the jar as follows
  java -cp <jar_name> <main_class> [username] [password] [action commands]

If user wants to give the keystore path,
java -cp <jar_name> <main_class> [username] [password] [action commands] -keystore <path_to_keystore>

SugarCRM Express Guide

- Login to Stratos2.0 using CLI

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

stratos>subscribe php mycrm -connect mysql -alias crmDB -repoURL https://github.com/lakwarus/sugarcrm.git

Where php is PHP cartridge, mycrm is alias/name for PHP application, mysql is MYSQL cartridge, crmDB is alias for Database 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.slive.com:8280
(this might take a minute... depending on repo size)

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

Wordpress Guide

- Login to Stratos2.0 using CLI
CLI$ ./stratos.sh admin@telecom.org password
Successfully Authenticated!

stratos>subscribe php myweb -connect mysql -alias mywebDB

Where php is PHP cartridge, myweb is alias/name for PHP application, mysql is MYSQL cartridge, mywebDB is alias for Database.

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!
http://git.slive.com/telecom.org/myweb.git repo Created!
Your application will publish here http://myweb.php.slive.com:8280

- Upload wordpress

$ git clone *http://git.slive.com/telecom.org/myweb.git*
$ cd myweb
$ ls
simplesamlphp sql www
copy wordpress files into www folder. Wordpress can download from here [1]
$ git add www/*
$ git commit -a -m “initial commit”
$ git push

- Setting up database
stratos> info mywebDB
Cartridge Info

Cartridge: mysql
Alias: mywebDB
Host: mywenDB.mysql.slive.com / 192.168.17.140
Password: cilxskqh
Admin URL: https://mywenDB.mysql.slive.com/phpmyadmin
Status: ACTIVE
Active Instances: 1

- Goto https://mywenDB.mysql.slive.com/phpmyadmin (Login using above info)
- Create a database.

- Goto http://myweb.php.slive.com:8280
- Run installer (give database details as above created)

Now http://myweb.php.slive.com:8280 website is ready to use.

Known Issues

https://wso2.org/jira/browse/SPI-21

References

[1] http://wordpress.org/latest.tar.gz

PHP cartridge and Identity Server Integration Guide

- Login to Stratos2.0 using CLI

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

stratos>subscribe php mysaml

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

And it give following output;

Subscribing to mycrm cartridge.
Success!
http://git.slive.com/telecom.org/mysaml.git repo Created!
Your application will publish here http://mysaml.php.slive.com:8280

- Upload simplesamlphp
$ git clone http://git.slive.com/telecom.org/mysaml.git
$ cd mysaml
$ ls
simplesamlphp sql www
copy simplesamlphp files into simplesaml folder. Simplesamlphp can download from here [1]
Set following configuration at simplesamlphp/metadata/saml20-idp-remote.php

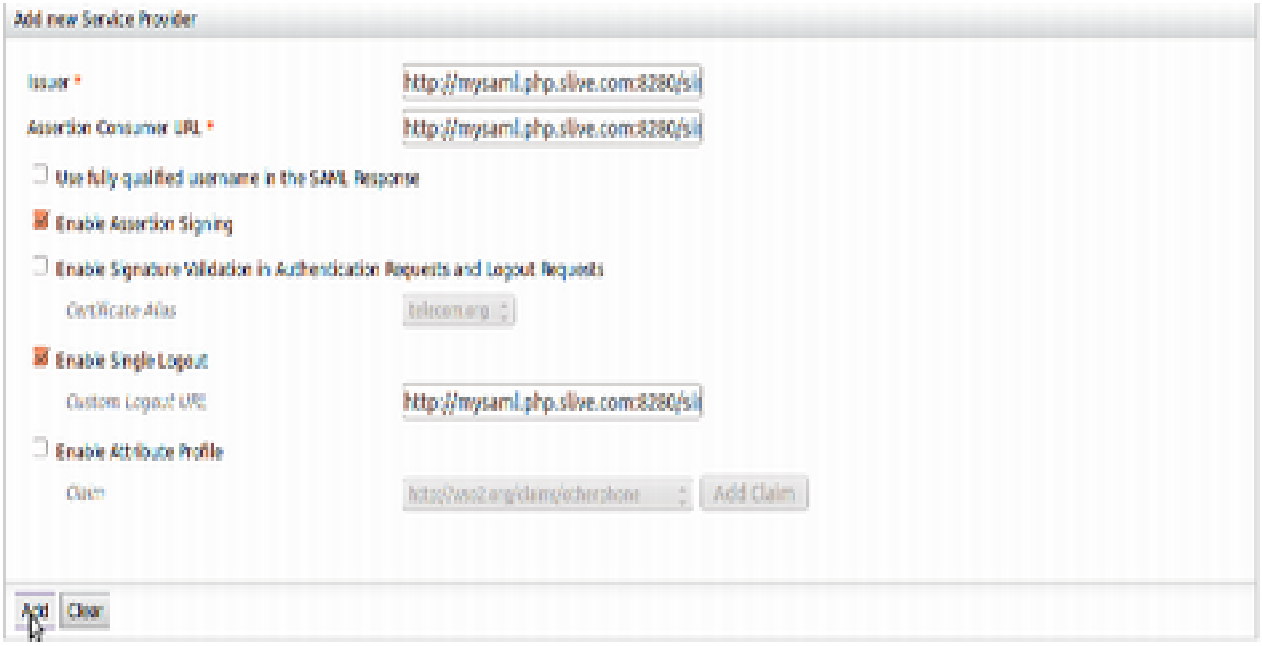
```
$metadata['https://idp.slive.com:8243/samlsso'] = array(
      'name' => array(
            'en' => 'WSO2 IS - guest users',
      ),
      'description'          => '',

  'SingleSignOnService'  => 'https://idp.slive.com:8243/samlsso',
  'SingleLogoutService'  => 'https://idp.slive.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

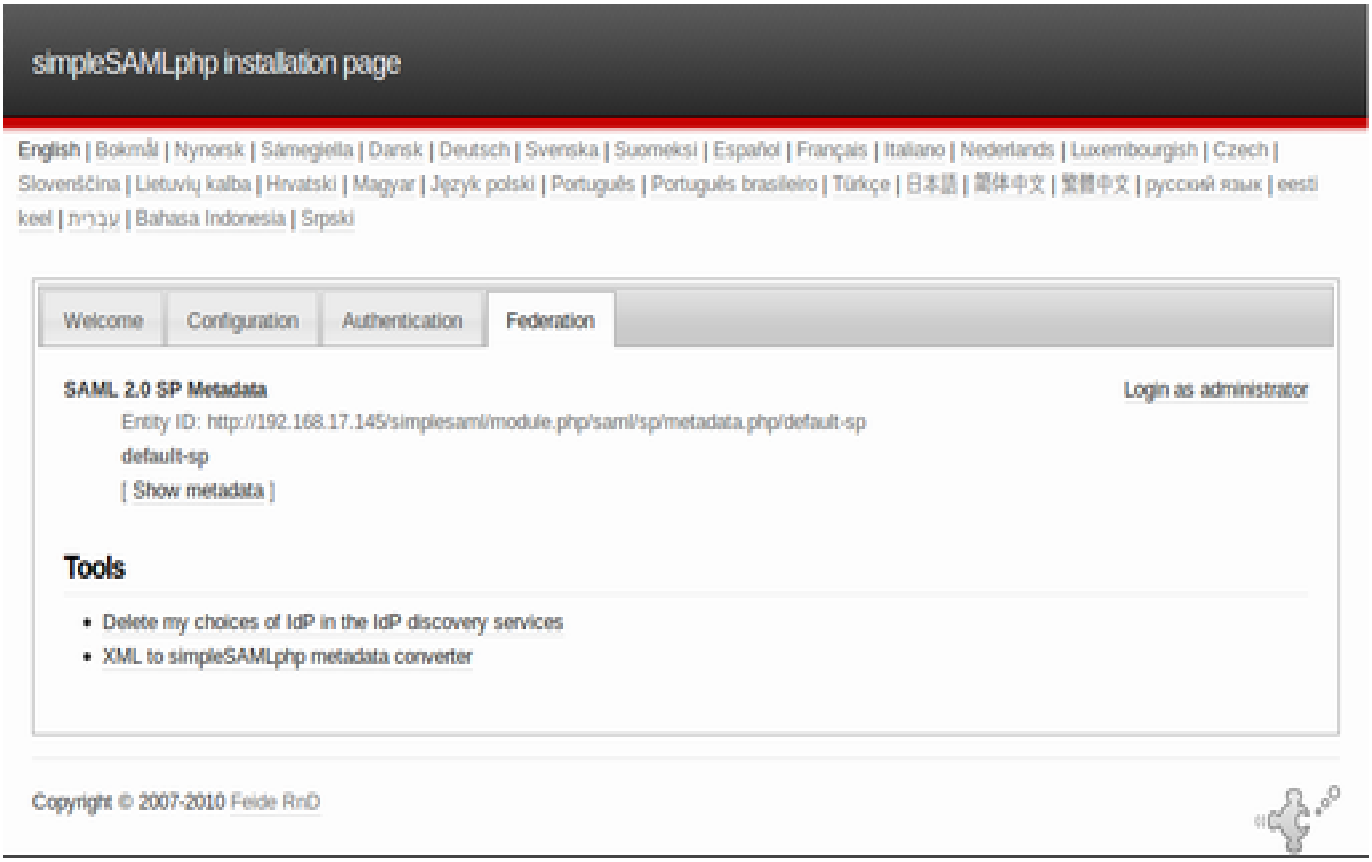
- Login to IS using tenant info.

Goto https://idp.slive.com:8243/
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 describe how can get above
details

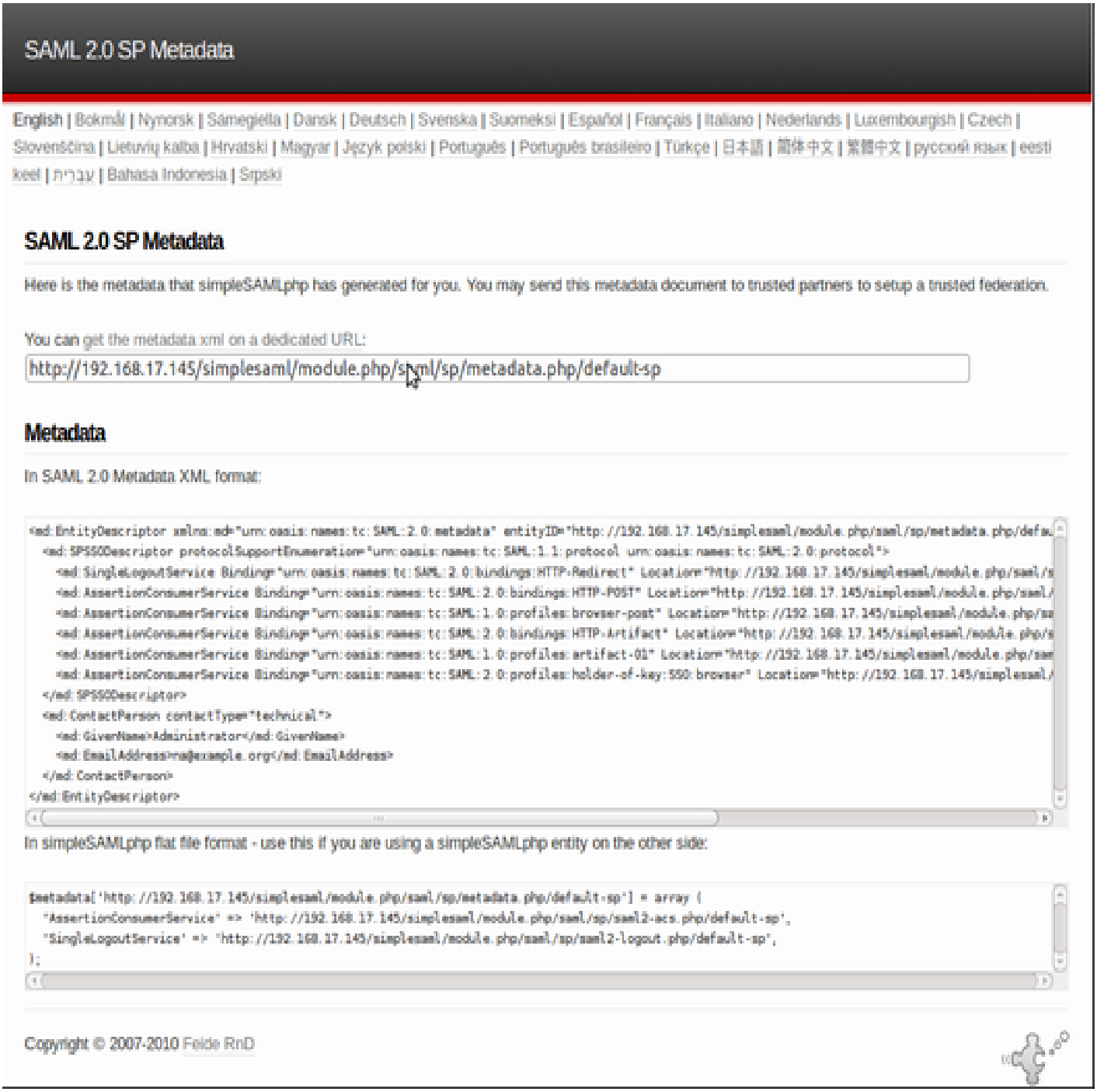


- Getting SP Metadata
Goto http://mysaml.php.slive.com:8280/simplesaml

Click on Show metadata on Federation
tab



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

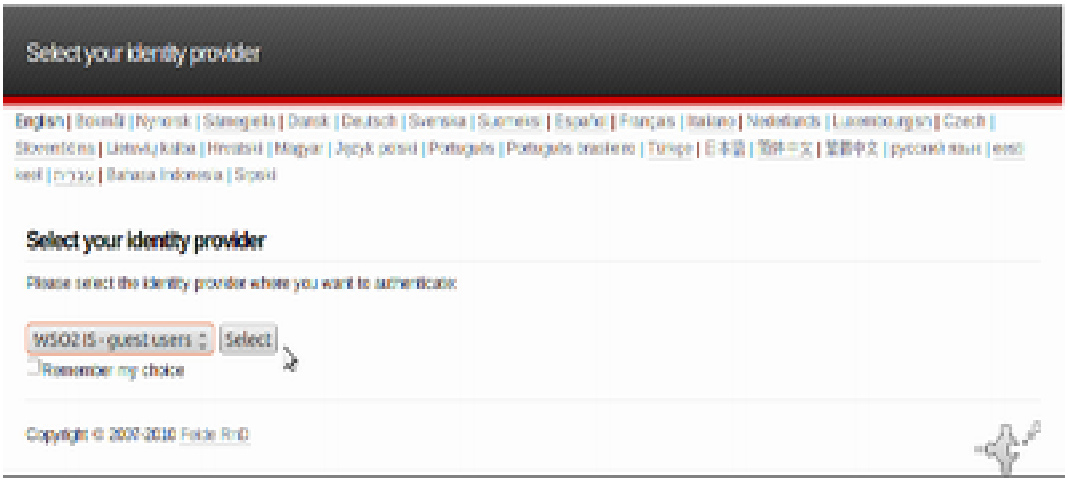


- Test SSO

Logout from IS Session

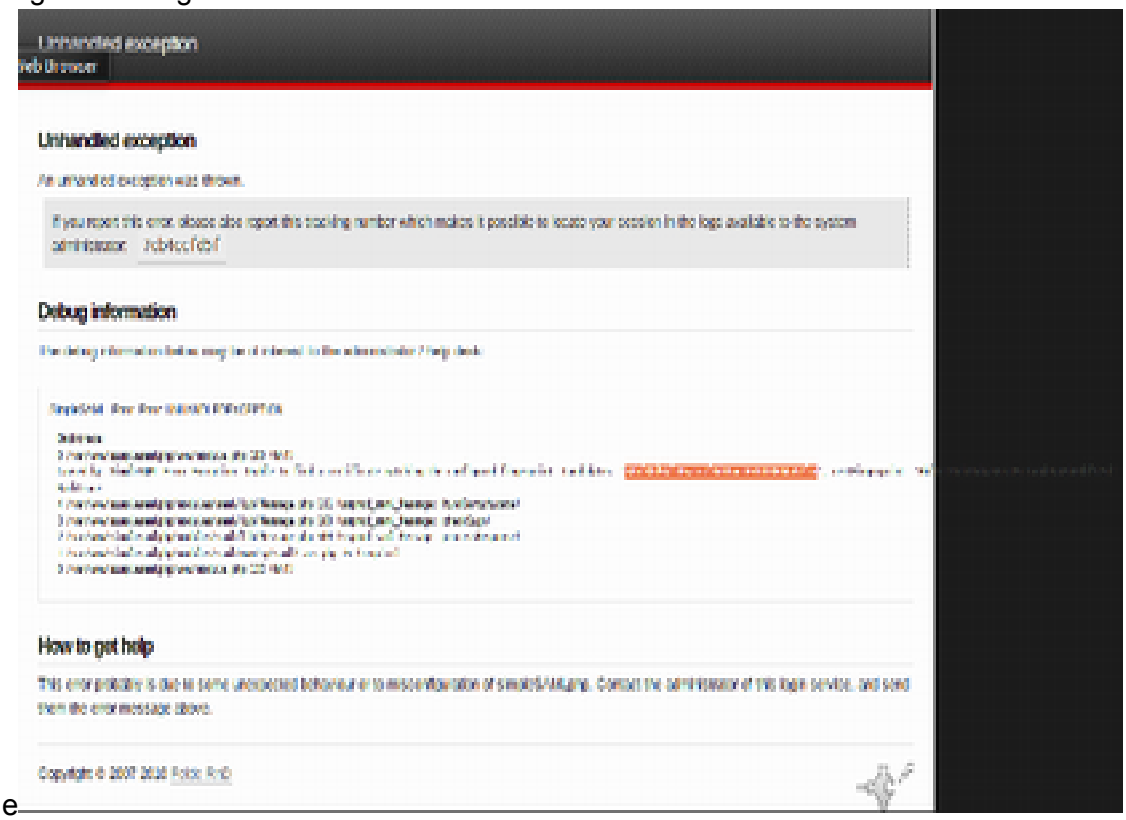
Goto http://mysaml.php.slive.com:8280/simplesaml

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.slive.com:8243/samlsso**'] = array(
    'name' => array(
        'en' => 'WSO2 IS - guest users',
    ),

```
    'description'          => '',

  'SingleSignOnService'  => 'https://idp.slive.com:8243/samlsso',
  'SingleLogoutService'  => 'https://idp.slive.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.

Known Issues

Known issue 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 out the following line

```
$msg = base64_decode($msg);
```

References

[1] http://simplesamlphp.org/download

Custom Domain Mapping Guide

You needs have the Stratos Cli Tool before add a custom domain mapping.

We will provide an interactive cli tool for tenant to add domain mappings. Here is an example for a tenant is adding domain mappings for different cartridges. At the time of subscription tenant does not have to provide own domain information. It is only after subscription you can add the domain mapping through cli tool.

- Say tenant **abc.com** is subscribed to **as** and **php** cartridges where it doesn't need any information about domain mapping when he subscribes.
    - There is a separate command for domain mapping. If tenant needs to add a domain mapping for **as** cartridge,

```
stratos> domain-mapping as
```

- Next he has to provide own domain for mapping,

```
Enter own domain name: mydomain.com
```

- Then it will be displayed the Stratos domain he has to CNAME for.

Deployment Synchronize with Github

In this section describe how to use existing github repository with Stratos2.0 Environment.

- Login to Stratos2.0 using CLI

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

stratos>subscribe php myphpapp -repoURL https://github.com/lakwarus/myphpapp.git

Where **php** is PHP cartridge, **myphpapp** is alias/name for PHP application and **https://github.com/lakwarus/myphpapp.git** is github repo.

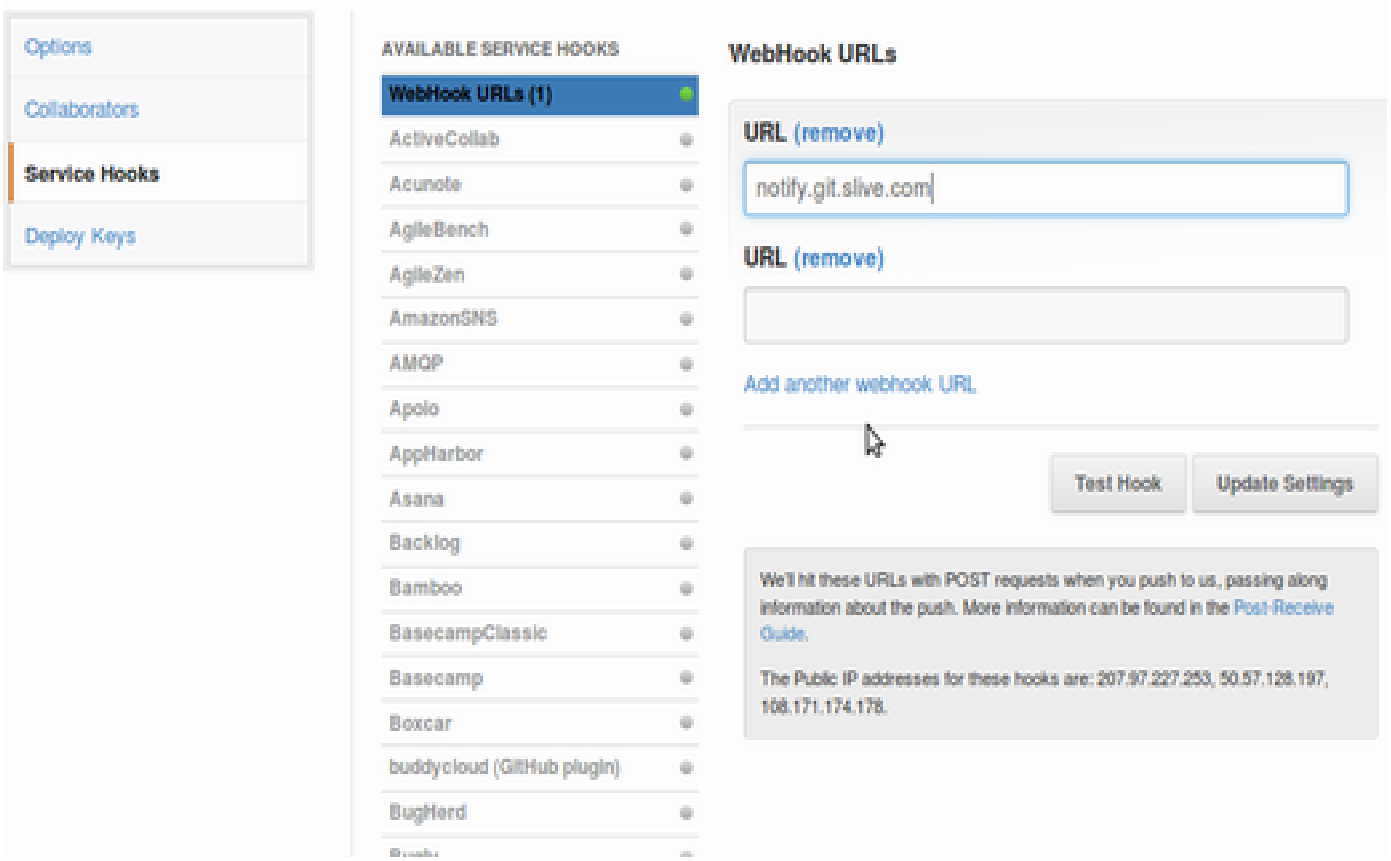
And it give following output;

Subscribing to myphpapp cartridge
Success!
Your application is being published here http://myphpapp.php.slive.com:8280
(this might take a minute... depending on repo size)

- myphpapp application is now ready to use.

- Setting Service Hook at Github

Goto github website and select repository
Goto setting -> Service Hooks -> WebHook URL and add notify.git.slive.com and do update settings.



Autoscaling User Guide

Prerequisite

- You should configure WSO2 Cloud Controller and WSO2 ELB for autoscaling (Please refer relevant user guides).

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.

1. ***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.

1. ***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.

1. ***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.

1. ***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.

1. ***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.

1. ***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.

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

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

subscribe php nirmalphp -min 1 -max 5