WSO2 Identity Server Demo-Suite

WSO2 offers a comprehensive open source product stack to cater to all needs of a connected business. With the single code base structure, WSO2 products are weaved together to solve many enterprise-level complex identity management and security problems. By believing in open standards and supporting most of the industry leading protocols, the WSO2 Identity Server is capable of providing seamless integration with a wide array of vendors in the identity management domain. The WSO2 Identity Server is one of the most powerful open source Identity and Entitlement Management server, released under the most business friendly Apache 2.0 license.

To try out these kind of standard solution patterns with the WSO2 Identity Server, a demo suite tool has been created. This tool provides a user friendly and rapid way to deploy artifacts which are needed for a particular demo.These artifacts include both WSO2 and third party server artifacts.

Getting Start

In order to try out each solution patterns using this demo-suite, follow the below steps.

1. Download WSO2 Identity Server 5.4.0 from here.
2. Download the demo-resource from here.
3. Run demo.sh script for any solution patterns as follows,
   ex: ./demo.sh solution-01 solution-02
4. A detailed description of the solution pattern which had been deployed, can be found in the "Solution Pattern Deployment and Results" section.

Structure of the tool

Here are the main resource folders in demo-suite tool.

demo-resources

- Solutions
- common-resources
- servers

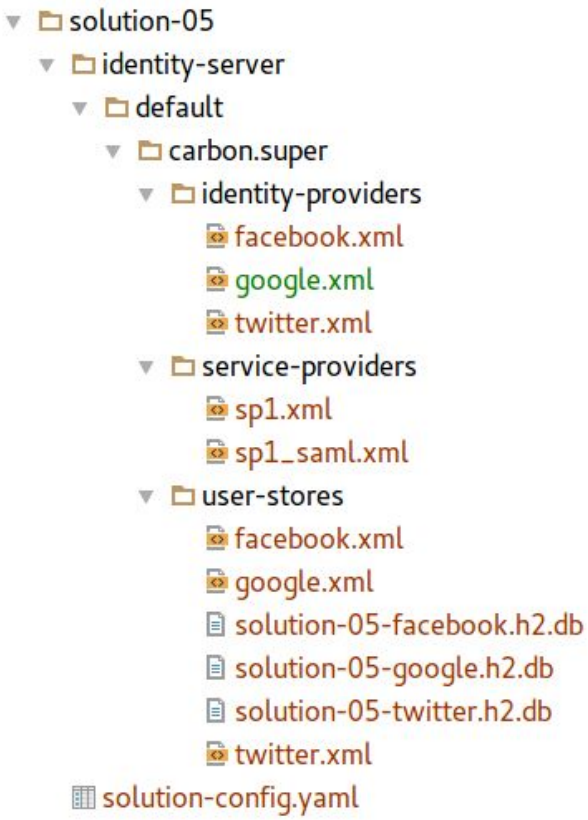
Solutions

We have created all the installation artifacts for lot of standard solution patterns under this folder and any one can create a new solution under this folder by following the expected structure as explained below.

Here is the folder and files structure of a solution.

solution-05 -> name of the solution pattern which is used as a prefix for all the artifacts to uniquely identify the solution in identity server.

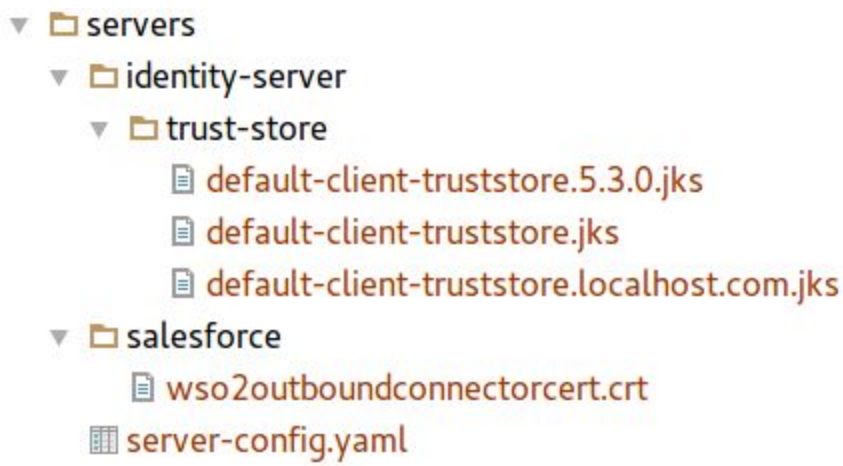
identity-server -> This name represent the identity server.
default -> This name refer the identity-server instance and it should be defined all the hostname, port and other configs under instance name in server-config.yaml file. We can define multiple Identity Server nodes by giving different instance names and configs.
carbon.super -> This represent the tenant domain name.
identity-providers -> All the identity provider artifacts should be under this folder. File name is not related to the artifact and it is just a meaningful name only.
service-providers -> All the service provider artifacts should be under this folder. Each service provider has its own inbound type and it is represented by a separate file by following this format [*service provider file name + "_" + protocol name*]
user-stores -> All the user store artifacts should be under this folder. File name is not related to the artifact and it is just a meaningful name only. It will generate *.h2.db file per user store which is refer by the user stores.
xacml-policy -> All the XACML policies will be installed from this folder. File names are not related to the XACML policies.
user-role -> Here we have simple property files to create user and roles in very simple manner. user.properties and role.properties are the 2 files and we can have any user and roles under these 2 files for each solution pattern.



There is a solution-config.yaml file which is provide the instruction to the client tool to know the sequence of the artifact to be run and inactive/active when it is required. Each solution contain this file.

Servers

Under this folder, there is a server-config.yaml to provide the custom configs in each Identity Server instances and the Tomcat. Demo resources provide a tomcat container out of the box and anyone can point to the remote Tomcat server as well by this config file.



Common-Resources

This folder contains all the common resources for solution deployment. All the client web application will be here and DB backup also under this folder.

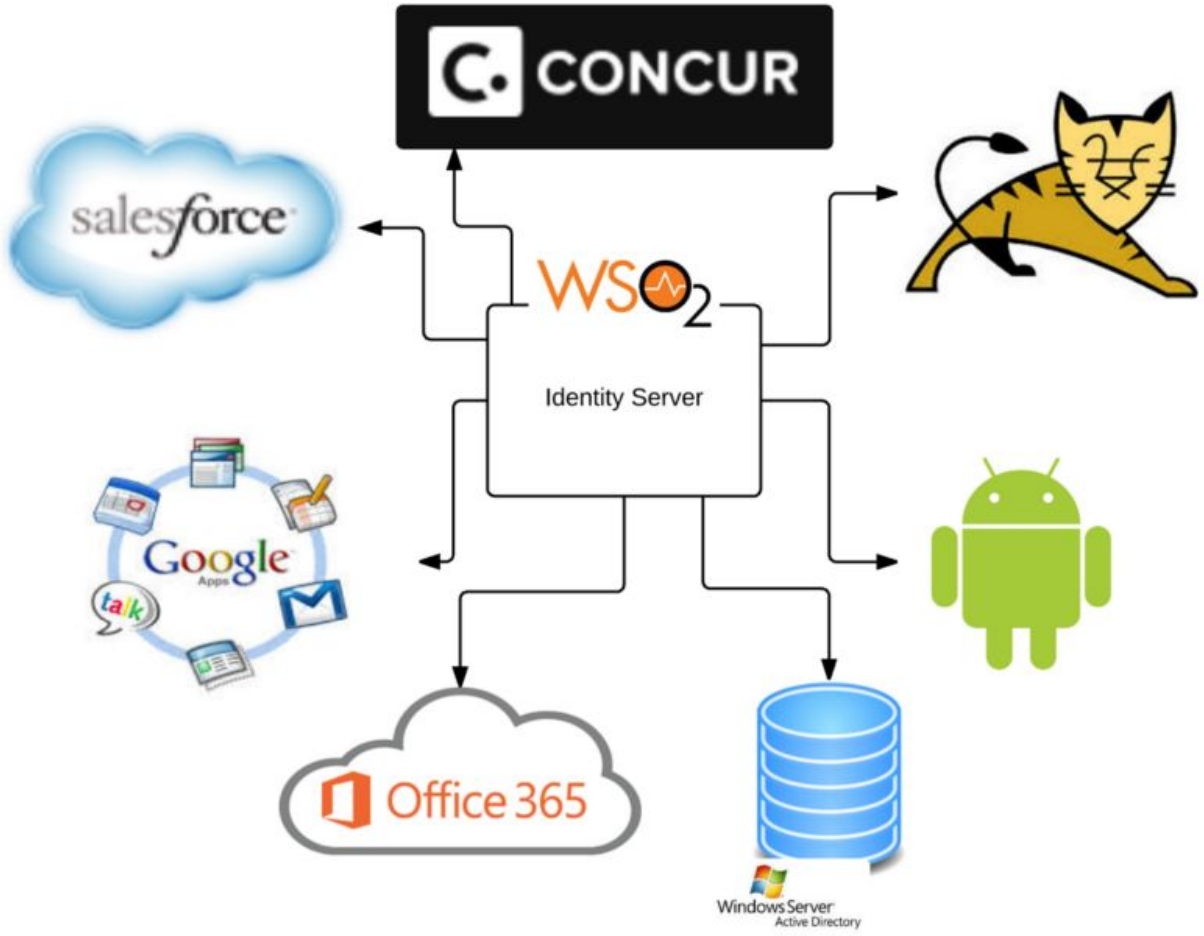
We have implemented 2 simple client applications which is based on OAuth2 and SAML protocols.

1. **francesca.com**
   This web application authenticate using **SAML** protocol and once we deployed this for a solution pattern, it can be accessed as in below.
   ex:
   Solution : solution-03
   Client application URL : http://localhost:8080/solution-03-francesca.com

2. **lebens.com**
   This web application authenticate using **OAuth2** protocol and once we deployed this for a solution pattern, it can be accessed as in below.
   ex:
   Solution : solution-05
   Client application URL : http://localhost:8080/solution-05-lebens.com

Solution Pattern Deployment and Results

[solution-01]
Single Sign On between multiple heterogeneous identity federation protocols



Problem

1. The business users need to access multiple service providers supporting multiple heterogeneous identity federation protocols.
2. Some service providers are on-premise while others are in the cloud. For example Google Apps (SAML 2.0), Salesforce (SAML 2.0), Office 365 (WS-Federation) are cloud based while JIRA, Drupal, Redmine are on-premise service providers.
3. A user logs into any of the service providers should be automatically logged into the rest.

Solution

Represent each service provider in the WSO2 Identity Server and configure the corresponding inbound authenticators (SAML, OpenID, OIDC, WS-Federation).

Once you deploy the solution-01, it will install the 2 service providers (francesca.com - SAML and lebens.com - OAuth2) in Identity Server and deploy the 2 client web applications in tomcat as well.

http://localhost:8080/solution-01-francesca.com
http://localhost:8080/solution-01-lebens.com