



IBM Software Group



Software as a Service

Deploying multi-tenant SaaS applications on IBM middleware in the Amazon Elastic Compute Cloud

IBM Developer Skills

ON DEMAND BUSINESS™

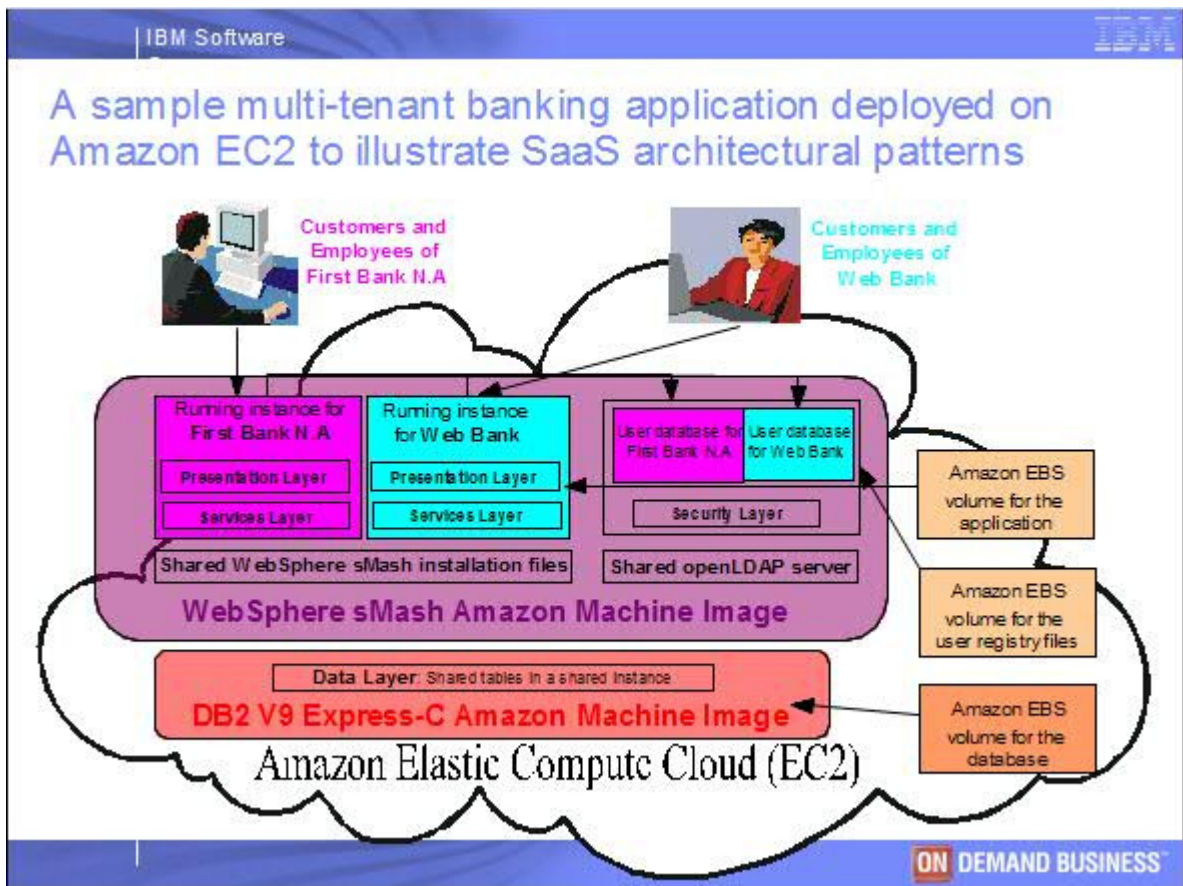
© 2006 IBM Corporation

Welcome to the IBM Software-as-a-Service demonstration series.

In this [series](#), we will demonstrate some architectural patterns exploiting features in IBM middleware for building software-as-a-service solutions.

IBM has recently [announced](#) the [availability](#) of multiple IBM middleware products for the Amazon Elastic Compute Cloud (a.k.a. EC2).

In this demo, we will show how the newly available WebSphere sMash and DB2 Express-C Amazon Machine Images (a.k.a AMIs) can be used to deploy a sample multi-tenant banking application to EC2. This sample banking application has been used to illustrate multiple SaaS architectural patterns in earlier demos in this [series](#) and is available for download from the website associated with this demo.



Here we show the sample multi-tenant banking application and the IBM middleware products in Amazon Machine Images used to deploy them in EC2.

Amazon's EC2 is a web service that provides resizeable computing capacity in a cloud on which user applications and the underlying middleware can be run.

The sample application runs on IBM WebSphere sMash and DB2 Express-C middleware products preinstalled in Amazon Machine Images. Websphere sMash is an application server which can be used to create, assemble and deploy applications written in dynamic scripting languages such as Groovy, quickly and easily. DB2 Express is IBM's entry-level database server for transaction processing. The sample application also uses OpenLDAP which is a free, open source implementation of the Lightweight Directory Access Protocol, or LDAP commonly used for accessing user information from a registry.

We show how this multi-tenant banking application can share its components and deployment infrastructure between customers and employees of two fictitious tenant banks: First Bank N.A and Web Bank.

The sample application includes

- A User Interface layer and a Services layer which run in separate runtime instances of WebSphere sMash for each bank. This pattern is explained in greater details in an earlier demo in this series titled “Creating dynamically scripted multi-tenant applications using WebSphere sMash, Part 1, using separate application instances for different tenants”.

The sample application also includes

- A data layer comprised of shared tables in a shared DB2 database instance.
- A security layer comprised of a shared openLDAP server with isolated user databases for each tenant.
- Two Amazon Elastic Block storage volumes: One volume houses the sMash application files and the openLDAP database files. The second one houses the DB2 database tables.

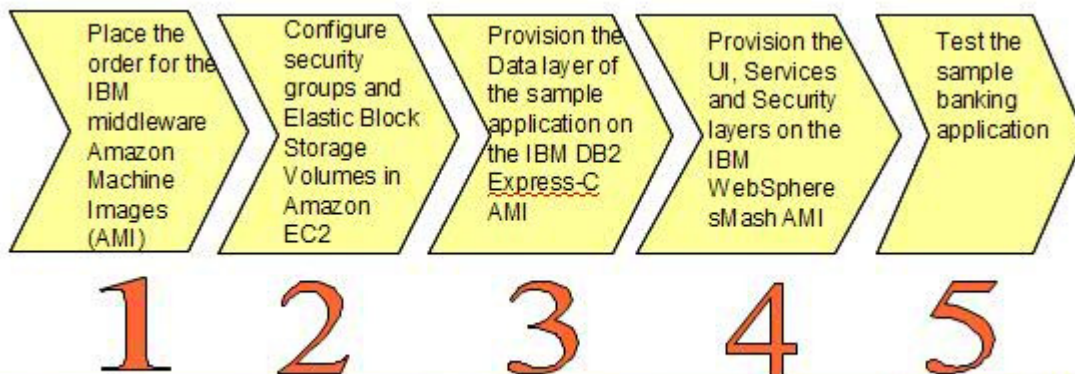
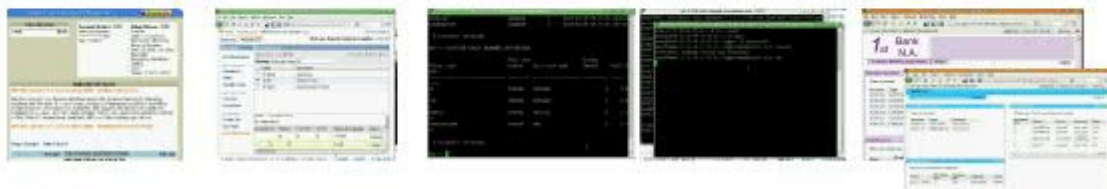
Next we show these functional layers of the sample multi-tenant application.

<<Live walkthrough>>

- This is the UI layer for First Bank N.A. It is built with DOJO widgets embedded in Groovy template files running in WebSphere sMash. It includes multiple iFrames for mimicking portlets. The UI layer is best viewed in the Firefox browser.
 - This is the UI layer for Web Bank. Note that different CSS files are used to provide a different look and feel for each tenant bank
 - This is the UI layer for the employee role. Users in different roles such as customer, employees etc see different pages at runtime.
- The Services layer is built with RESTful services implemented using groovy scripts running in WebSphere sMash. Here we invoke the get customer profile service from a browser. Here we can see the Groovy script implementation of this service. These scripts invoke the sMash data api to access a shared database
- The Security layer includes two separate openLDAP databases configured for each bank: bank1-ldap and bank2-ldap. Here we can see the sMash configuration for openLDAP for First Bank N.A. and Web Bank.
- The Data Layer includes shared tables with DB2 V9 XML columns for storing customizable tenant specific data. Since the WebSphere sMash data model doesn't support DB2 XML natively yet, we created a relational view from the DB2 XML column as shown here.

The design patterns described above are shown in greater details in an earlier demo in this series titled “[Building Web delivered SaaS applications on open-source and entry-level IBM middleware](#)”.

Deployment Steps



Next we discuss some high-level steps for deploying this application to Amazon EC2: In the first step, we will place the order for the IBM WebSphere sMash and DB2 Express-C AMIs.

In the second step, we will configure Amazon EC2 security groups and Elastic Block Storage volumes.

In the third step, we will deploy the data layer components of our sample application to the DB2 Express-C AMI.

In the fourth step, we will deploy the UI, services and security layer components of our sample application to the WebSphere sMash AMI.

In the fifth step, we will access the instances for each tenant bank from a URL.

Step 1: Place the order for the AMIs:

We start from the [IBM developerWorks Cloud Computing Resource Center](#)

We select the DB2 Express-C AMI.

We click on the start developing link which takes us to Amazon Web Services where we sign in with our existing Amazon account.

The Amazon payment screen appears where we place the order.

Amazon payment process succeeds and displays an Activation key.

We repeat the above steps for the Websphere sMash AMI and obtain another Activation key.

Step 2: Configure Security Groups and EBS volumes:

Next we use the Amazon Web Services Management Console to create security groups and Elastic Block Storage volumes.

We create two security groups db2 and smash.

We authorize the db2 security group to allow all traffic from smash security group only.

We authorize the smash security group to access ports 8080 and 8081 from any machine.

We create three EBS volumes for the LDAP database, the sMash applications and the DB2 database.

Step 3: Provision the data layer components on the DB2 Express-C AMI

Next we provision the data layer components on the DB2 Express-C version 9.5 32 bit AMI.

We find this AMI in the AWS Management Console.

Verify the product code as 53C681D4.

And launch it.

While launching, we specify
the number and the type of instance,
a pre-created key pair and
the security group db2 created in the previous step.

We verify that the instance is running and
get the DNS name for the instance.

We copy our EC2 certificate and private key to /mnt folder in the instance.

We use ssh to connect to the instance.

We go through a set of guided steps to configure the instance at first boot.

Major steps include accepting the license agreements,
specifying the location of the EC2 certificate and keys that we copied to this instance.
specifying and mounting the EBS volume we created in the previous step.
creating DB2 users and
starting the DB2 server

We create a database called SAAS DB
We connect to the database and
We import the DDL file for the sample application.

Step 4: Provision the UI, services and security layer components on the WebSphere sMash AMI

Next we provision the UI, services and security layer components on the WebSphere sMash AMI.

As with the DB2 AMI, we find the AMI and launch it using the AWS Management Console.

We connect to the WebSphere sMash AMI instance.

We attach the two EBS volumes with the sMash instance.

We mount the two EBS volumes in the instance at /home/smash/apps and /home/ldap.

We install openLDAP and copy its configuration file to the EBS volume.

We create two directories in the EBS volume for storing two openLDAP databases for the two tenant banks.

We add the configuration for the sample application to the openldap configuration file.

We launch the openLDAP server.

We add a default set of users and groups for each tenant bank.

We log on as the smash user and unzip the smash application zip files for the two tenant banks into the EBS volume.

We enter the database information in the sMash configuration file for the two tenant banks.

We run the zero resolve and run commands to start the sMash application for First Bank N.A (bank1).

We open another terminal and go to the smashbank2 folder and run the same commands to start the sMash application for Web Bank (bank2).

We update the hosts file to introduce two fictitious URLs for the two tenant banks (bank1.com and bank2.com)

We create two Apache HTTP server document root directories and a configuration file with two virtual hosts and reverse proxy rules for each tenant bank.

We start the Apache HTTP server.

Step 5: Test the sample multi-tenant banking application

Next we test the sample application.

We add the host names for the two tenant banks in our hosts file.

We enter the application URL for each First Bank N.A.

We enter the application URL for Web Bank.

Conclusion

- **Rapidly deploy multi-tenant applications built on WebSphere sMash and DB2 Express-C to the Amazon Elastic Compute Cloud (EC2)**
 - Easily scale to more tenants by
 - Sharing WebSphere sMash and DB2 components or
 - Adding new EC2 instances
- **Pay for IBM middleware and Amazon EC2 instances on an hourly usage basis**

In conclusion, we have shown how a multi-tenant SaaS application built on IBM WebSphere sMash and DB2 can be rapidly deployed to the Amazon Elastic Compute Cloud using a few, simple steps.

Developers can easily scale their applications by repeating these simple deployment steps on additional EC2 instances.

When using the “Developer AMIs” developers get charged at an hourly rate only for the usage of Amazon’s EC2 resources. In the near future, developers will also be able to deploy to “Production AMIs” for an hourly charge.