IBM Order Management 9.4

TCOI

Deployment Accelerator Installation on Softlayer

Agenda

- A basic introduction to chef, ruby and zookeeper.
- The 2 step model of TCO Installer.
- Setting up build server
- Building the images.
- Setting up the deployment server
- Deploying the environment from Images.

Basic overview of chef and its terminologies to be used throughout the session.

Chef

Chef is a tool that lets you programmatically provision and configure components. Treats infrastructure as any codebase.

Chef ensures that each node adheres to a certain policy.

Each node is assigned a role and that role determines the policy that node adheres to.

Chef client is installed on each node and when chef-client runs it pulls the policy from chef server.

These policies are nothing but a set of instructions written in ruby and are called **recipes** by chef.

A collection of recipes is a **cookbook**.

A **Role** determines what recipes or cookbooks will be executed on a chef node. The **runlist** defined in the definition of a role is a list of recipes/cookbooks to execute for that role.

TCO installer is a package containing cookbooks and recipes that enables fast deployment of OMS on Softlayer cloud.

Zookeeper is used for orchestration , ie , to ensure certain steps of installation happen in a specific order.

TCO Installer

TCO Installer – The problem and the solution

Problem -

Even the most basic uncustomized installation of SMCFS involves several steps.

Installing an application server.

Installing and configuring a database server.

Installing and configuring a JMS server.

Installation of SMCFS runtime itself, creating an EAR and deploying it, setting up one or more agent and integration servers etc.

These, in turn, entail manpower skilled in database, application server and JMS administration along with an SME in SMCFS. Therefore, even the most basic SMCFS setup requires considerable amount of time and skill resulting in a higher total cost of ownership of the product.

TCO Installer – The problem and the solution

Solution-

Through this feature, an attempt has been made to simplify the above process. Convention is prioritized over configuration and a set of pre-built pre-configured virtual machine image templates is created by the TCO Installer for each of the servers on IBM Softlayer, viz.:

1. Application server (with SMCFS EAR deployed)
2. Database server (with database and schemas created and factory-setup data loaded)
3. JMS (configured with the requisite queues for use with the installed SMCFS)
4. SMCFS runtime.

These images are then used by TCO installer to rapidly deploy multiple environments having same tested and verified configuration.

As explained earlier TCO installer consists of two steps or two ruby gem commands that can be

TCO Installer – Overview

TCO installer is a package containing cookbooks and recipes that enables fast deployment of OMS on Softlayer cloud.The entire process consists of two steps and hence requires manual execution of two commands –

- Build Step (provisionBuild) - The Build Process:

  In this step VMs are provisioned and image templates of the servers mentioned above are created. The following sub-steps need to be performed:

  a) Provision a VM and install WAS application server.

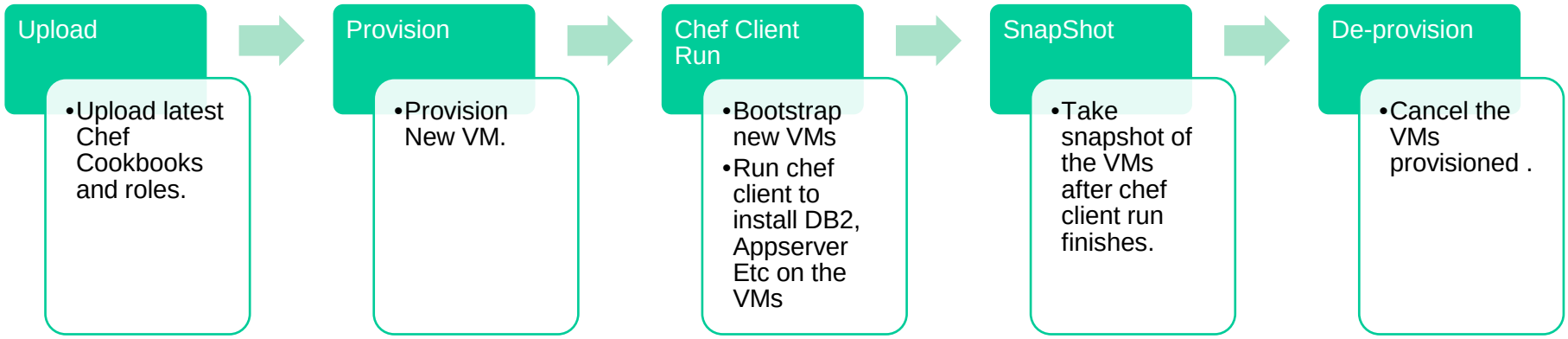
  b) Provision a VM and install DB2 database server on it. Setup DB2 and create requisite users, database, tablespace and schemas in preparation for SMCFS installation.

  c) Provision a VM and setup Websphere MQ on it. Setup the queue manager, queues and queue connectoin factories needed.

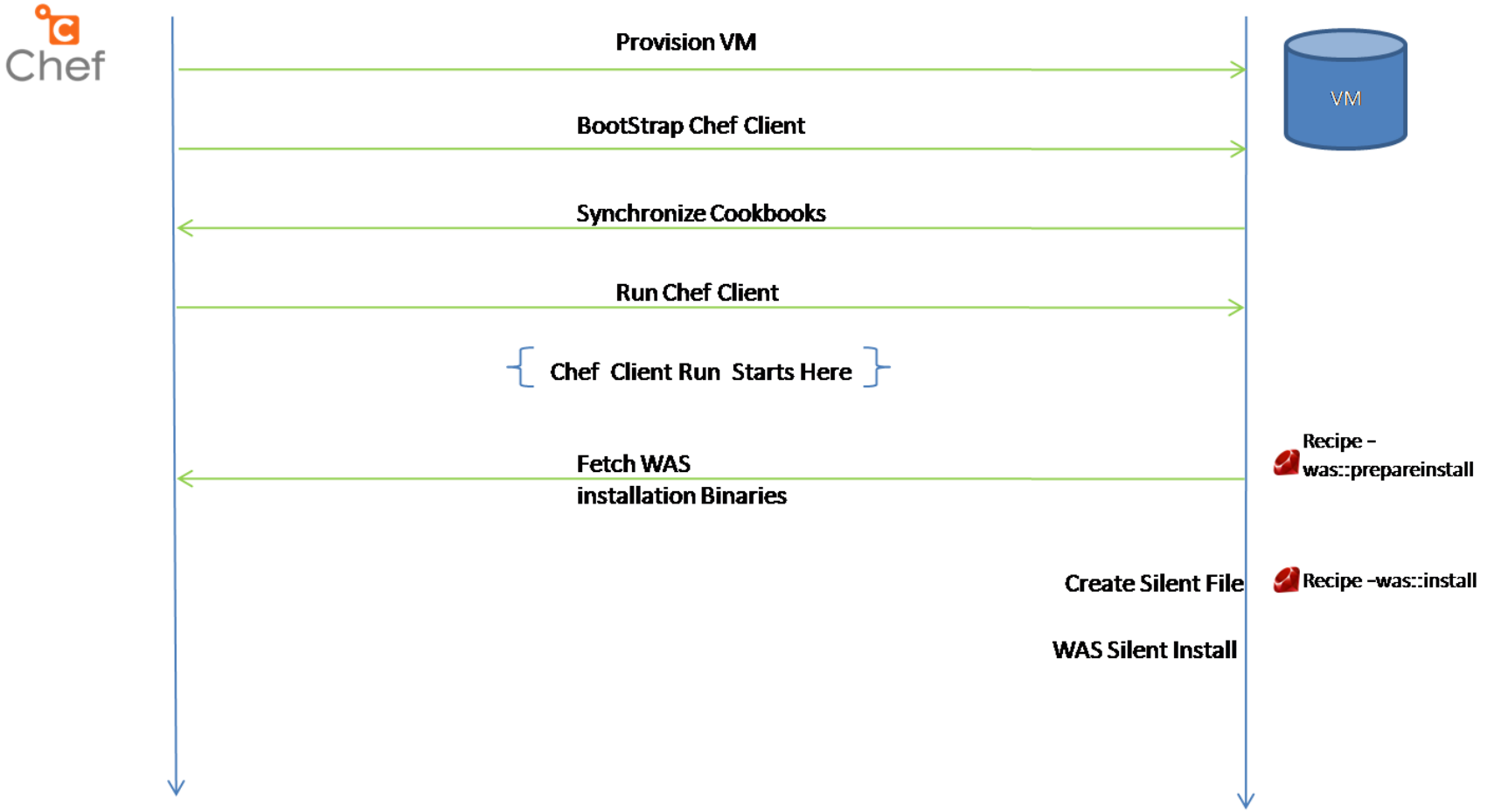
  d) Provision a VM and install SMCFS on it using the DB2 database created above. Configure the installed SMCFS to use Websphere MQ for its JMS. This will be the runtime VM that will be used as agent and integration servers as well.

  e) Create a snapshot of the above VMs to store as image templates.

Build flow -



Example – The appserver installation



TCO Installer – Overview

- Deploy step (createEnvironment) - In this step the image templates created in build step are used to provision a set of VMs that will provide a fully functional instance of SMCFS ready to be used. A set of scripts are provided through the cloud_deployment gem to assist in the same. Using the createEnvironment command and a configuration file (deploy.yml), the following will be provisioned:

  One or more WAS instances (with SMCFS deployed)

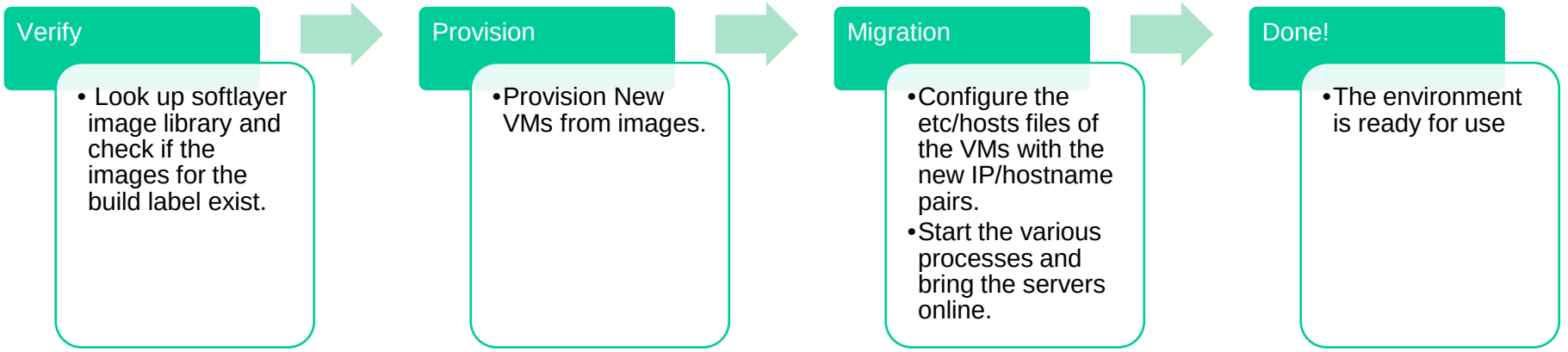
  One DB2 instance

  One MQ instance

  One or more agent server instances

  One or more integration server instances

Deployment flow -



TCO Installer package

The TCO installer package shipped with SSFS 9.4 , is a collection of -

- Cookbooks
- Role definitions
- Ruby Gems
- A user guide

Setting up the build server

As explained earlier TCO installer consists of two steps or two ruby gem commands

These commands are decoupled from each other and can be run from same linux machine(VM or Baremetal) or different machines.

Lets refer the Linux machine from which the provisionBuild Command is being run as our build server. This server requires following steps to set up –

Install the basic utilities and libraries like – java , ruby , zookeeper.

Install chef server

Install chef client

Configure chef server

Extract the deployment zip

Build and install the depac-common, depac-build gems

Create a file with name api-key.yml containing the Softlayer username and apikey

Configure the build configuration YAML file

provisionBuild

This command is used for the build step of TCO Installer. It's a ruby gem command and is bundled in the depac-build gem package.

Usage: provisionBuild -f <FILEPATH> [-l <LABEL>]

It requires two parameters to execute – a build label and path for the build configuration file.

The build label should be a unique identifier (timestamp , or some sequence eg – Build01) to easily identify the images created.

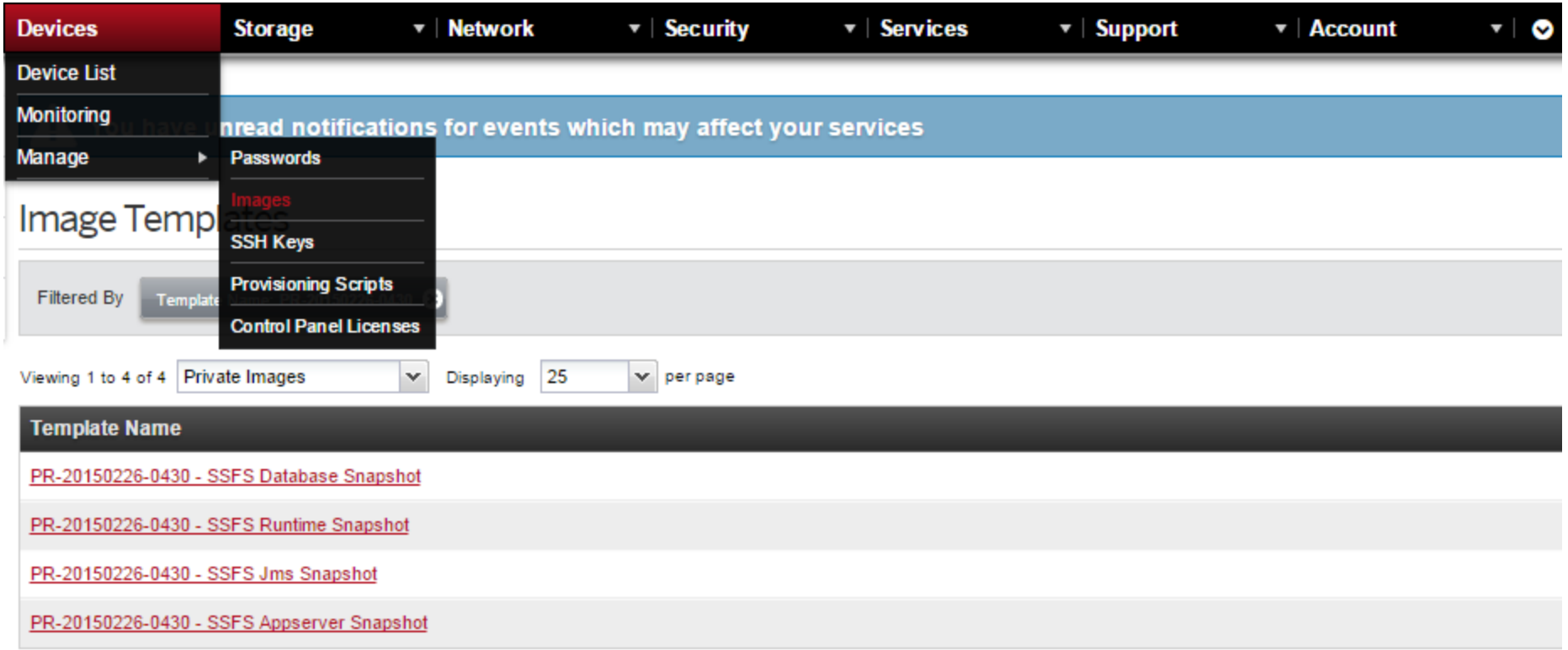
Build Configuration YAML

- A sample (build.yml.sample) of this file is shipped in the depac-build gem
- This file controls the flow of the provisionBuild
- This file also lets you choose the configuration and datacenter for the VMs created during build process
- Its completely documented with descriptive comments for each property.

Looking up the images on Softlayer Portal

The images created by provisionBuild are stored in softlayer ‘s image library. The images can be accessed from the softlayer portal (control.softlayer.com) or by softlayer api framework.

To browse the images created, you can list the images by accessing Devices->Manage-> Images menu options and filter the list by the build label



Setting up the deployment server

Lets call the Linux machine from which the createEnvironment ruby gem Command is being run, the deployment server. This server requires following steps to set up –

Install ruby.

Extract the deployment zip

Build and install the depac-common, depac-deployment gems. If the build server is also being used as deployment server then only build and install depac-deployment gem.

Create a file with name api-key.yml containing the Softlayer username and apikey

Configure the deployment configuration YAML file

createEnvironment

This command is used for the deployment step of TCO Installer. It’s a ruby gem command and is bundled in the depac-deployment gem package.

Usage: createEnvironment <build-label> <config-file-path>

It requires two parameters to execute – a build label and path for the deployment configuration file.

The build label should be a unique identifier (timestamp , or some sequence eg – Build01) to easily identify the images created.

Deployment Configuration YAML

- A sample (deploy.yml.sample) of this file is shipped in the depac-deployment gem
- The environment property in this file is used to uniquely identify a set of VMs as one environement, this environment label and build label is used to construct the hostname of VMs.
- This file lets you define the topology of the environment you wish to create by letting you define number of appserver nodes/agent nodes that are needed.
- This file also lets you choose the configuration and datacenter for the VMs created during build process
- Its completely documented with descriptive comments for each property.

Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, Coremetrics, DB2, PowerVM, Rational, WebSphere, and z/VM are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.