Getting Started with WSO2 Private PaaS on Amazon EC2

This tutorial will help you create your own Private Platform as a Service on Amazon EC2. The tutorial walks you through the following steps:

- Step 0: Finding the Documentation
- Step 1: Provisioning the EC2 Instance from the WSO2 Private PaaS AMI
- Step 2: Provisioning the WSO2 Private PaaS Environment
- Step 3: Log Into the Private PaaS Administration Console and Create a Tenant
- Step 4: Provision Application Platform Services
- Step 5: Deploy Applications
- Step 6: Evaporating and Rehydrating the Cloud
- Step 7: Monitor Private Cloud

Step 0: Finding the Documentation

The first step to take before getting started with WSO2 Private PaaS is to locate the documentation! I know you may not RTFM, but in this case, you will need a good guidebook. To find the documentation:

1. Browse to http://wso2.com/cloud/private-paas/
2. The WSO2 Private PaaS platform page describes the platform goals and architecture. The page lists key downloadable assets on the right hand side, which can be used to deploy WSO2 Private PaaS in an on-premise environment. The page lists links to environment installation tools, command line interface tools, core Platform assets, and deployable cartridges.
3. You will also notice a section describing the identifiers for Ready to Use Amazon Machine Images
    a. In this AMI section, click the EC2 Demo Guide link.
4. The Quick Start Guide describes:
    a. How to start your private PaaS by provisioning the WSO2 Private PaaS demo image as an EC2 instance
    b. How to register a PaaS tenant
    c. How to configure cartridges and provision application platform services
    d. Let's start building our Platform as a Service environment

Step 1: Provisioning the EC2 Instance from the WSO2 Private PaaS AMI

In the first video, we showed you how to find the WSO2 Private PaaS documentation and locate the installation instructions. In this step, we will provision an EC2 instance from the WSO2 Private PaaS AMI. After following this video, you will be able to access a private Cloud environment containing all the bits required to provision and start your private Platform as a Service.

1. Start by logging into the Amazon Web Service Console
2. To initialize the AMI for your private cloud environment, you will need to specify multiple Amazon EC2 environment parameters during the provisioning process.
3. I have created a worksheet that will help you collect the required EC2 environment parameters.
    a. The worksheet is publicly hosted on github at https://github.com/karux/wso2-privatepaas/blog/master/worksheet-1-provision-wso2privatepaas-ec2.htm
    b. To get ready, fork the wso2-privatepaas project and open up the worksheet

Worksheet #1 – Provision WSO2 Private PaaS on EC2

Current Default Entries for Table Below

WSO2 Private PaaS EC2 Image Identifier: **ami-962872c4**
Current EC2 Availability Zone: ap-southeast-1b
Size: m1.xlarge

Enter the region of the IaaS you want to spin up instances : ap-southeast-1

boot.sh PAAS INSTALLATION SCRIPT PROMPTS

You selected Amazon EC2.
Are you in a EC2 VPC Environment? [y/n] : y

EC2 Image Identifier	
Stratos Domain:	
Enter EC2 identity:	
Enter EC2 Credentials:	
Enter EC2 owner id:	
Enter EC2 keypair name:	
Enter EC2 availability zone:	
Enter EC2 security group IDs:	
Enter EC2 VPC Subnet ID:	
Enter the region of the IaaS you want to spin up instances:	

Network Access Addresses

Public DNS:	
Public IP:	

4. Get ready to launch an EC2 instance based on the WSO2 Private PaaS AMI.
    a. First, make sure you record the correct AMI identifier.
    b. Second, select an appropriate security group.
        i. For demonstration purposes, we will select a security group with wide-open privileges. A best practice is to at a minimum restrict external Internet traffic to your IP address.
        ii. Record the security group ID parameter. Note, the recorded parameter is NOT the security group name.
    c. Third, select the access credentials that will control the PaaS environment.
        i. A best practice is to use the Identity Access Management console to create a distinct user for PaaS administration.
        ii. Once you select the user, the access key will be shown. The access key should be used as the 'EC2 identity' parameter.
        iii. The EC2 credential parameter is obtained by clicking on the 'Manage Keys' button. I'm not going to click on the button during this video, because I want to keep my credentials private.
    d. Fourth, enter the domain for your private PaaS in the 'Stratos Domain' parameter row.
        i. The domain is used by the Elastic Load Balancer to route requests into the PaaS run-time environment.
        ii. If you don't own your own domain, choose a meaningful identifier.
        iii. After the instance is running, you should configure your DNS to associate the domain name
        iv. with the EC2 public IP.
        v. If you don't have DNS administrator access, you can change your desktop or laptop host file to make the domain name association.
    e. Fifth, create an EC2 key pair to provide you shell access into the PaaS environment.
        i. After you create the key pair, download the private key file and appropriately set the file permissions
            1. Hint chmod 0400 [privatekeyfilename].pem
        ii. Record the full key pair name on the worksheet.
    f. We have now reached a point where EC2 administration pre-work is complete, and we are ready to spawn our base Private PaaS instance.
        i. As you build and tear down demonstration environments, you can re-use security groups, credentials, and keypairs across Private PaaS environments.
    g. Navigate to the EC2 Instance Management screen to launch your first PaaS instance.
        i. Click on the 'Launch Instance' button to start the instance creation wizard.
        ii. Choose 'Community AMIs' and enter the demo instance identifier into the search box.

iii. Verify the search query returns 'WSO2 Private PaaS 4.0 Demo', and click the select button
iv. Choose the m1.xlarge instance type recommended by the documentation. This instance type will be used for the main PaaS components responsible for administration, logging, monitoring, and deployment.
v. On the 'Configure Instance Details' and 'Add Storage' pages, accept the default values and move on by clicking the Next button.
vi. On the Tag Instance page, you may choose to fill in a friendly identifier.
    1. WSO2 Private PaaS does not require you to enter any tags.
    2. I'm going to enter a sample tag to help me track the instance.
vii. Move on to 'Configure Security Groups'
    1. Select the existing security group recorded on our worksheet
viii. Click the Launch button to start the EC2 instance spin up process.
    1. Select the key pair recorded on our worksheet
    2. Don't forget to acknowledge the security key notice
    3. Click 'Launch Instance' to finalize the launch process
ix. To finalize the EC2 provisioning homework, view the instance details and record a few more environment parameters on our worksheet.
    1. Record the owner ID.
    2. Record the availability zone
    3. Record the VPC subnet ID.
        a. Make sure you do NOT select the VPC ID, or the PaaS won't be able to spin up services.
    4. By now, your base instance should be running and ready administrative access.
        a. Let's test out access to our new private cloud!
        b. You can log into the instance using the private key, Ubuntu user name, and EC2 public hostname or EC2 public IP address.
        c. I am going to use the ssh-ec2-paas.sh script that you can download from github.
        d. The script takes the full private key filename and EC2 public hostname of your instance.
        e. Record the Public DNS and Public IP on your worksheet
        f. In your instance, you should be able to list the 'private-paas' directory

In this step we:

- Created an EC2 instance from the base WSO2 Private PaaS AMI.
- Provisioned the EC2 environment with our security group, and keypair.
- Verified access to our running EC2 instance containing the private-paas bits.
- We are now ready to provision and start our private Platform as a Service.

Step 2: Provisioning the WSO2 Private PaaS Environment

During this video session, we will provision our Private Platform as a Service.

In prior steps we:

- Located the installation documentation
- Created an EC2 instance from the base WSO2 Private PaaS AMI, provisioned the EC2 environment with our security group and key-pair, and showed how to login to the running instance.

1. To start the provisioning process, first ssh into your EC2 instance as the Ubuntu user.
2. Change into the private-paas directory
3. Set the PaaS configuration files to your custom domain
    a. Copy the fixup-domain.sh file from the wso2-privatepaas GitHub project [https://github.com/karux/wso2-privatepaas] into the private-paas directory. The script performs a search and replace on the wso2.com domain string with your custom domain.
    b. Execute fixup-domain.sh by passing the default domain wso2.com and your custom domain.
        i. For example. `./fixup-domain.sh wso2.com example.org`
4. In the private-paas directory, you will find the installation script, boot.sh
5. Execute boot.sh and don't forget to run as super user by using the sudo command
6. The script firsts asks you for the PaaS domain name. Enter a domain that you administer, or pick any memorable domain that you will place in your client's host file.
7. The script will continue now install the cloud metadata repository and puppet deployment master
8. The script can provision multiple pre-built middleware cartridges. We will instruct the script to install the WSO2 Application Server, not install business process server, install WSO2 Enterprise Service Bus, and install WSO2 API Manager.
9. Confirm that we are provisioning the PaaS in an EC2 VPC environment
10. Enter the EC2 identity recorded on our worksheet during the prior session. The identity and credentials were found in the AWS IAM console
11. Enter the EC2 credentials. I've blacked out my credentials to keep them private, and you will notice the terminal window does not echo the credential string. After you paste once, blindly hit enter
12. Enter the EC2 owner identifier, which we located in the EC2 instance console
13. Enter the EC2 keypair name (without the pem extension), which we located in the EC2 instance console
14. Enter the EC2 availability zone
15. Enter the EC2 security group identifier. Remember the text to enter is the identifier, and not the assigned name.
16. Enter the EC2 VPC subnet identifier. Remember the identifier is for the subnet, not the entire VPC. If you enter the wrong value, the services will not start.
17. Choose your availability zone, in our case Singapore, from the list shown, and enter the string for your IaaS region.

18. The installation script will now execute a lengthy provisioning process. Go get a cup of your favorite beverage. My favorites are black coffee or lemonade.
    a. [after synchronizing, cut 4:07-5:47 or speed up 8x]
19. The installation script is now configuring the WSO2 Application server and WSO2 Enterprise Service Bus cartridges.
20. After the cartridges are provisioned, the script will start the key services: Business Activity Monitor, Gitblt repository, and WSO2 Identity Server core services,
21. The installer will now deploy Platform as a Service policies for partitions, load balancing, and cartridges.
22. The installer will now activate all services

In this step, we provisioned the WSO2 Private PaaS environment. We are now ready to add tenants and create application platform service groups.

Step 3: Log Into the Private PaaS Administration Console and Create a Tenant

During this video session, we will configure our Private Platform as a Service with tenants. A tenant is the organization, business unit, or team that will be using the PaaS.

In prior steps we:

- Located the installation documentation
- Created an EC2 instance from the base WSO2 Private PaaS AMI, provisioned the EC2 environment with our security group and key-pair, and showed how to login to the running instance.
- Provisioned our Private Platform as a Service

1. After creating an instance and provisioning your Private Platform as a Service with the boot.sh installation script, your EC2 instance console should show multiple cloud instances that provide load balancing, application server, and enterprise service bus services.
2. Access the PaaS administrator web application by pointing your browser to the public dns or public ip, followed by the default port (9443), slash console slash login
3. The default username and password combination will be admin : admin
4. In our getting started instructions, you will find a section called 'Registering a tenant and configuring cartridges.
5. We will create a tenant using the Private PaaS GUI. Other options include using the command line interface or directly calling the REST API.
6. Browse to the Tenant Management page
7. Click the Add Tenant button to create your first tenant
8. Enter a tenant domain name and the owner's contact details.
9. Create a second tenant.

In this step, we quickly created two tenants. The tenants are isolated consumers of the PaaS.

Step 4: Provision Application Platform Services

During this video session, we will provision application platform services

In prior steps we:

- Located the installation documentation
- Created an EC2 instance from the base WSO2 Private PaaS AMI, provisioned the EC2 environment with our security group and key-pair, and showed how to login to the running instance.
- Provisioned our Private Platform as a Service
- Added tenants

1. Access the PaaS administrator web application by pointing your browser to the public dns or public ip, followed by the default port (9443), slash console slash login
2. The default username and password combination will be admin : admin
3. In our getting started instructions, you will find a section called 'Registering a tenant and configuring cartridges.
4. We will configure a cartridge using the Private PaaS GUI. Other options include using the command line interface or directly calling the REST API.
5. Because the default cartridges are multi-tenant, we will follow the subscribing to a multi-tenant cartridge instructions
6. An important pre-requisite for multi-tenant cartridges is to provision a Git repository that will be associated with the application platform service.
    a. The Git Repository Folder structure will vary based on which multi-tenant cartridge type is deployed. Because I am demonstrating how to turn an WSO2 application server cartridge into an cloud application container service, I will create the first directory structure listed on the documentation page.

7. I am going to provision a Git repository on GitHub.
    a. I have logged into GitHub using my GitHub account
    b. Navigated to the repository sub-tab
    c. And select 'create new repository'
    d. The repository name should describe your application platform service. We will be hosting a multi-tenant SaaS application called app1
    e. We will initialize the git repo with a README and license. Because the repo will be holding run-time artifacts, do NOT select .gitignore Java.
    f. After the repository is created, copy the github URL. We will use the github url to push the directory structure into the repository.
    g. I have posted a bash script that will create the required directory structure. The script is located at `https://github.com/karux/wso2-privatepaas/blob/master/create.AppServer.repo.sh`
        i. The script simply clones the repo into our local directory, adds the directories to our repo structure, and then pushes the directory structure to GitHub.
        ii. To run the script, enter the script filename, the copied GitHub URL, and the project name.

create.AppServer.repo.sh Script

```
# this script initializes the git repository with
# the directory structure required by the WSO2 AppServer cartridge
# Parameters:
#       $1 git repo url
#       $2 application service name
git clone $1
mkdir ./$2/axis2modules
touch ./$2/axis2modules/.gitkeep
mkdir ./$2/axis2services
touch ./$2/axis2services/.gitkeep
mkdir ./$2/jaggeryapps
touch ./$2/jaggeryapps/.gitkeep
mkdir ./$2/jsservices
touch ./$2/jsservices/.gitkeep
mkdir ./$2/webapps
touch ./$2/webapps/.gitkeep
cd $2
git add ./axis2modules/.gitkeep
git add ./axis2services/.gitkeep
git add ./jaggeryapps/.gitkeep
git add ./jsservices/.gitkeep
git add ./webapps/.gitkeep
git commit -m 'establish appserver cartridge directory structure'
git push
```

8. Browse to the Cartridge Management page
9. Click the Subscribe to Cartridge button to create your first application platform service
    a. The page lists two available cartridge types: appserver and esb
    b. We are going to select the appservergroup-5.2.1 cartridge
    c. Enter a subscription alias that describes the platform service. I am going to name the service myJava
    d. Enter the repository location by copying the repository URL
    e. Click subscribe to cartridge
    f. Stratos will now spin up a worker node, a management node, and register the nodes with the elastic load balancer
    g. The My Cartridges page now specifies the configured application platform service, associated instances, and repository reference.
    h. At any time, we can push artifacts stored in the GitHub repo to our running service instances.
10. Selecting the application service platform will display the information page.
    a. Note the public IP and hostname for the management and worker nodes. Configure your domain naming service to associate the IP address with the hostname.

In this session we used the Stratos Manager User Interface to add an Application Server services. The service will be used to host web applications in the cloud.

Step 5: Deploying a Cloud Application

During this video session, we will deploy a web application to our PaaS.

In prior steps we:

- Located the installation documentation
- Created an EC2 instance from the base WSO2 Private PaaS AMI, provisioned the EC2 environment with our security group and key-pair, and showed how to login to the running instance.
- Provisioned our Private Platform as a Service
- Added tenants
- Added an application platform service. The service hosts web applications in the cloud.

1. After you clone the wso2-privatepaas repository (https://github.com/karux/wso2-privatepaas), you will notice the testWebApp subdirectory
2. Change into the testWebApp subdirectory and build the test application
    a. `mvn install`
3. Copy the `./target/paasdemoapp.war` file into the cartridge webapp subdirectory. For example
    a. `cp ./target/paasdemoapp.war ../wso2-paas-cartridge-app1/webapps`

4. Commit the web application war file to the cartridge synchronization repository
    a. `change into the ../wso2-paas-cartridge-app1/webapps directory`
    b. `git add paasdemoapp.war`
    c. `git commit`
    d. `git push`
5. Navigate to the access url of the management console
6. Select the main tab, Manage Applications, List section in the application server management console
    a. You will notice the paasdemoapp has been automatically deployed.
    b. Click the go to URL link in the Actions column to view the deployed web application
7. NOTE: if you remove the application war file from the git repository, the PaaS will undeploy the application

Step 6: Evaporating and Rehydrating the Cloud

Stopping back-end application platform service instances

Passivating the LB and Controller
Stopping the EC2 Instances

Start LB and Controller instance
Re-set DNS for EC2 public IP addresses