**Last Revised on July 11th, 2017**

**Version 1.0**



Ansible Orchestration for

F5 Application Connector

**Document Owners:**

**Mark Lowcher**

Contents

[1 Revision History/Document Control 2-3](#_Toc487536331)

[1.1 Review / Retention Cycle: 2-3](#_Toc487536332)

[1.2 Updates 2-3](#_Toc487536333)

[1.3 Change History Detail: 2-3](#_Toc487536334)

[1.4 Purpose: 2-3](#_Toc487536335)

[1.5 Distribution 2-3](#_Toc487536336)

[2 Executive Summary 2-3](#_Toc487536337)

[2.1 Introduction 2-3](#_Toc487536338)

[2.2 Test Overview 2-3](#_Toc487536339)

[2.3 Scope 2-3](#_Toc487536340)

[3 Ansible Installation 2-4](#_Toc487536341)

[3.1 Install Openssl Server 2-4](#_Toc487536342)

[3.2 Update APT 2-4](#_Toc487536343)

[3.3 Install Ansible and Dependencies 2-4](#_Toc487536344)

[4 Ansible Orchestration 2-5](#_Toc487536345)

[4.1 Setting up the Connectivity between Ansible and the F5s. 2-5](#_Toc487536346)

[4.2 Ansible Setup Playbook 2-6](#_Toc487536347)

[4.3 Download the Playbooks for Application Connector 2-6](#_Toc487536348)

[4.4 Download the Application Center and AC\_Proxy Software 2-7](#_Toc487536349)

[5 Configuring the Application Service Center 2-7](#_Toc487536350)

[5.1 Running the Enable Application Connector Playbook 2-8](#_Toc487536351)

[5.2 Import the Application Center RPM 2-8](#_Toc487536352)

[5.3 Creating the External Monitor 2-11](#_Toc487536353)

[5.4 Running the Applicationconnector\_vips Playbook. 2-11](#_Toc487536354)

[6 Installing AC\_proxy in AWS Cloud 2-11](#_Toc487536355)

[6.1 Create a Key Pair in AWS 2-11](#_Toc487536356)

[6.2 Converting Your Private Key Using PuTTYgen 2-11](#_Toc487536357)

[6.3 Configuring PuTTY with the Private Key 2-12](#_Toc487536358)

[6.4 Launch an EC2 Instance in AWS 2-12](#_Toc487536359)

[6.5 Create Inbound Rules for the ACproxy 2-13](#_Toc487536360)

[6.6 Associate a Public IP to the ACproxy Instance 2-14](#_Toc487536361)

[6.7 Preparing Ubuntu and Ansible to Communicate Together 2-15](#_Toc487536362)

[6.8 Run the acproxy\_part1 Playbook 2-16](#_Toc487536363)

[6.9 Run the acproxy\_part2 Playbook 2-16](#_Toc487536364)

[6.10 Connect to the AC\_proxy Instances 2-17](#_Toc487536365)

Revision History/Document Control

Review / Retention Cycle:

This document will be reviewed and updated as needed. The record retention period for this document is “until superseded”. This document will be located at: <https://github.com/mlowcher61> in the documentation folder.

Updates

Updates to this document must be approved by the following: Mark Lowcher

Change History Detail:

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Date | Approver |  |
| Version 1.0 | 07/11/2017 | Mark Lowcher |  |

Note: This document was printed from an online system and must be used for reference purposes only. The online document is the current valid document. It is the user’s responsibility to ensure that they are using the latest copy, as found in <https://github.com/mlowcher61>

Purpose:

This document was created to show how to get Ansible on Ubuntu working with F5 modules and uri ReST and tmsh commands.

Distribution

The following are on distribution for this document, and will be copied as part of the document review prior to publication.

|  |  |
| --- | --- |
| Name | E-Mail ID |
| Mark Lowcher | [m.lowcher@f5.com](mailto:m.lowcher@f5.com) |
|  |  |

# *Executive Summary*

Introduction

This document describes how to install Ansible 2.3 on Ubuntu 16.04 and how to use Ansible to deploy Application Connector Service Center on an F5 and ACproxy in the AWS cloud.

Test Overview

All Ansible playbooks located at <https://github.com/mlowcher61> were tested on F5 version 12.1.2. They should also work on 11.6 but that was not tested.

Scope

The scope for this test is limited to specified features and functionality of F5 devices as outlined in this document.

Ansible Installation

Ansible must be installed on a Linux platform. We will spin up a new Ubuntu 16.04 within a hypervisor and run the following commands to get Ansible installed with all its dependencies. During initial setup of the Linux, create user ansible with your desired password.

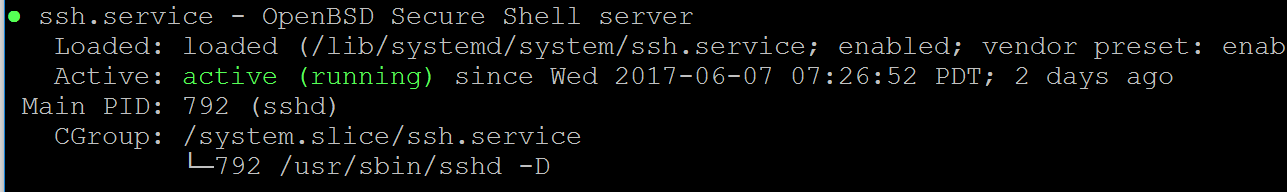
Once the Ubuntu instance has booted we will need to install openssl server from the hypervisor console so that we can access the Linux instance from an SSH session.

Install Openssl Server

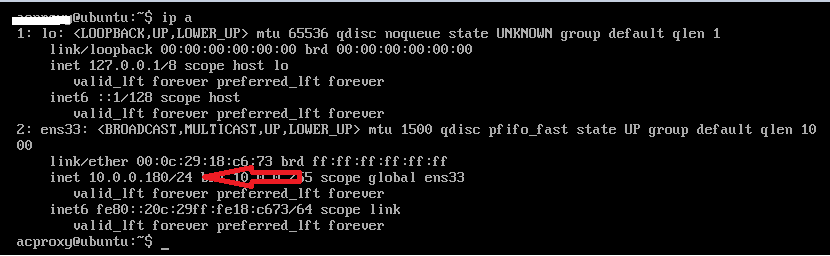
Run “sudo apt-get install openssh-server”

After the openssl server installation is complete, check the SSH status.

Run “sudo service ssh status” It should show as “active”



From the Linux console run “ip a” and determine what ip address was given through DHCP. You may want to shut down the Linux instance and put the network adapter into “bridged” mode.



Update APT

Run “sudo apt-get update”

Install Ansible and Dependencies

Perform the following from the home directory:

Run “cd ~”

Run “sudo apt-get install software-properties-common”

Run “sudo apt-add-repository ppa:ansible/ansible”

Run “sudo apt-get install python-pip”

Run “sudo pip install ansible”

Run “sudo pip install bigsuds

Run “sudo pip install f5-sdk

Run “sudo mkdir /usr/share/ansible”

Run “sudo chown ansible /usr/share/ansible”

Run “sudo chgrp ansible /usr/share/ansible”

Run “sudo mkdir ansible”

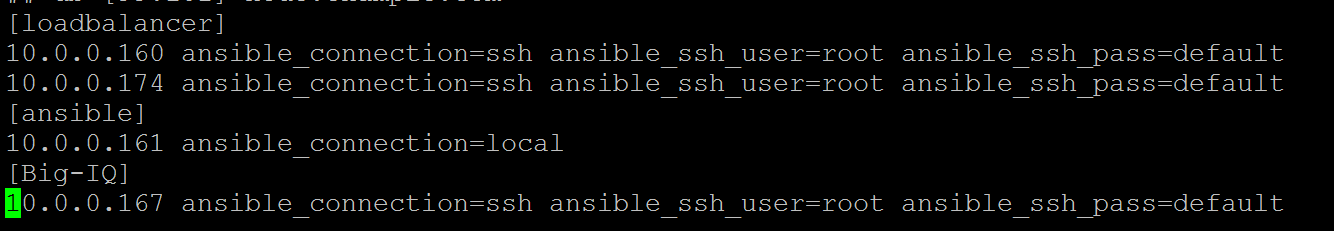
Run “sudo mkdir ./ansible/playbooks

Ansible Orchestration

Setting up the Connectivity between Ansible and the F5s.

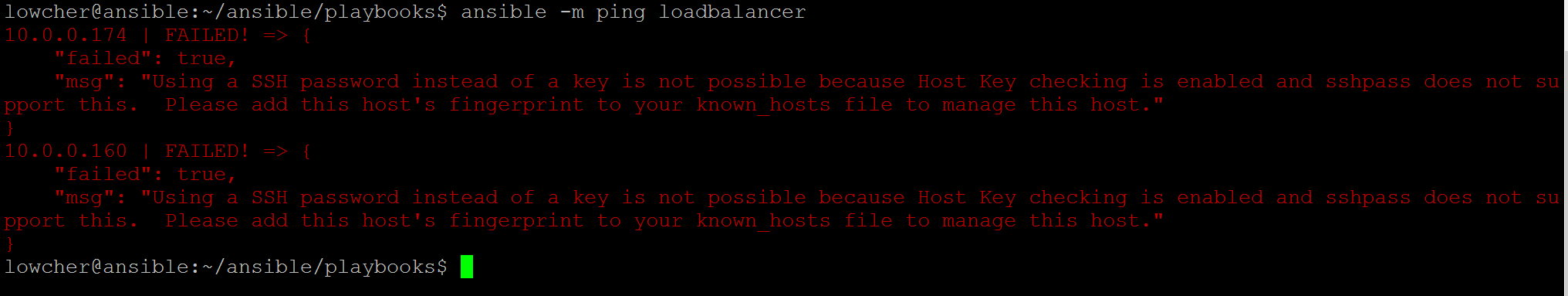
Now that Ansible is installed and we know the management ip address. We will configure Ansible for use.

First edit the /etc/ansible/hosts file and add your loadbalancer and ansible ip addresses and connection method. The example below shows setting up the environment with the default root user and default password of default.



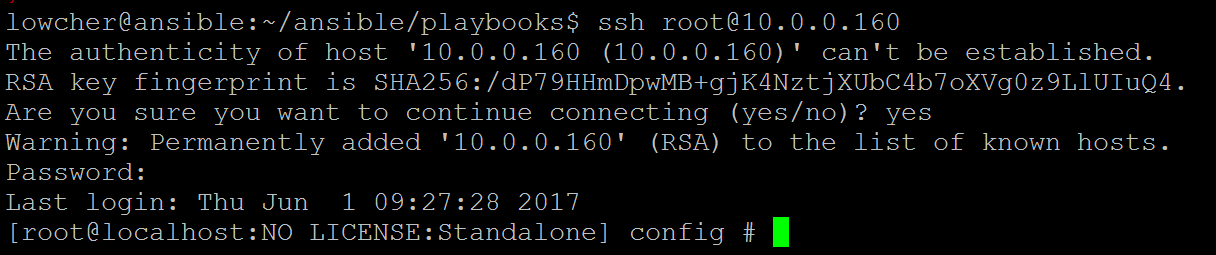
In our Ansible playbooks we will be able to target the individual loadbalancers by using “Loadbalancer[0]” for the .160 and loadbalancer[1] for the .174.

Test connectivity to the F5s using the ansible -m ping command

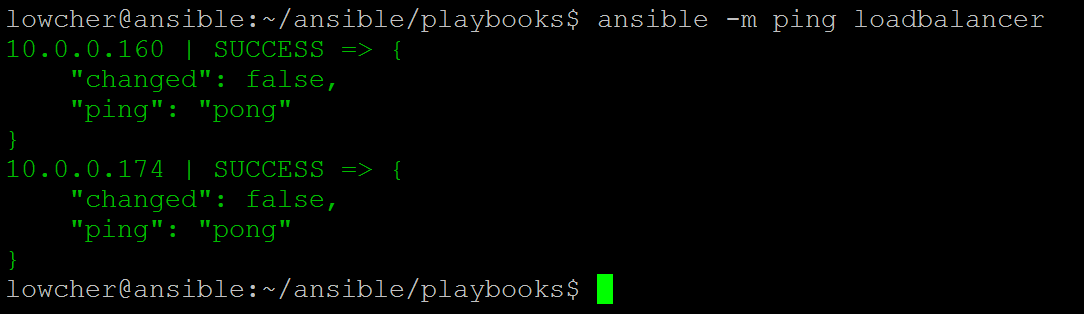


Since the Ansible workstation has never connected to the F5s, this will fail.

From Ansible, ssh as root to each of your loadbalancers with password “default” and type yes to continue.



Now retry the ansible -m ping loadbalancer command.



Now Ansible can communicate with the F5s.

Ansible Setup Playbook

From the <https://github.com/mlowcher61> site download all the Ansible playbooks located under f5-Ansible.

Upload all the playbooks to the ~/ansible/playbooks directory in the Ansible system.

Now run the f5\_ansible\_setup.yml playbook by running the following command.

Run “ansible-playbook f5\_ansible\_setup.yml.

Ansible is now ready to be used.

Download the Playbooks for Application Connector

You will find the Ansible playbooks used in this guide at <https://github.com/mlowcher61>

Download the following playbooks

* enable\_appconnector.yml
* appconnector\_vips.yml
* acproxy\_part1.yml
* acproxy\_part2.yml
* appconn\_monitor.yml

Due to a known bug with apt and Ansible, updating apt on a remote host using Ansible is not working. There are several times when “apt-get update” needs to be run on the linux machine that will be hosting the AC\_proxies. When this bug is resolved, the two Acproxy playbooks will be merged into one.

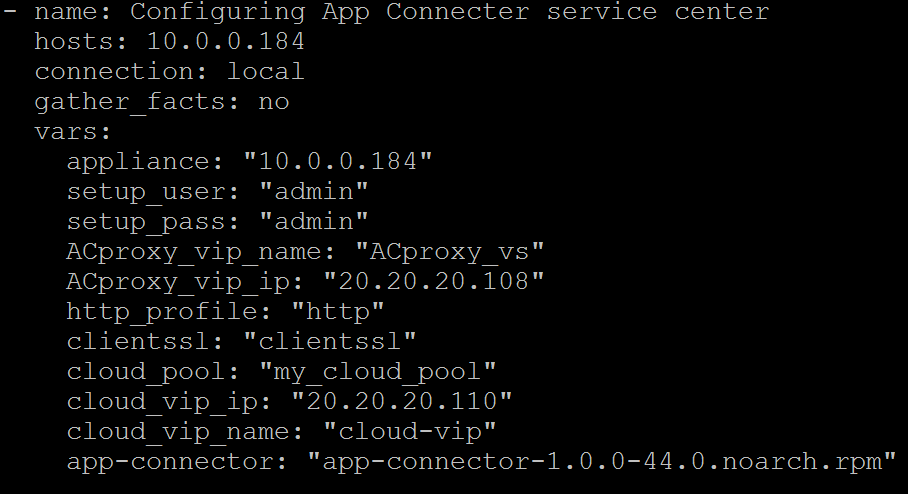
The enable\_appconnector playbook will enable Application Connector.

After running this playbook you must import and setup Application Connector from the GUI. At this time it is not possible to do it from Ansible. Although you can copy the files to the correct directory it will not show up in the GUI. This is a known issue and is supposed to be fixed in Ansible 2.4. If it is fixed this deployment guide will be updated.

The appconnector\_vips playbook will create the two virtual servers needed for Application Connector.

Open each of the three playbooks and become familiar with their contents. The “vars” section needs to be edited to match your environment. In the screenshot below most of the items are self explanatory, some worth mentioning are:

* “appliance” -- the management ip of the F5
* “app-connector” – the version of software you are using
* “clientssl” – the clientssl profile you will be using. You need to create your clientssl profile with your certificate and private key before running the app-connector playbook. You can do this manually or by using the following two playbooks located at <https://github.com/mlowcher61>
  + push\_cert\_key\_to\_F5 – this will install the certificate and key to the F5
  + create\_clientssl\_profile – This will create the clientssl profile using your cert and key.



Download the Application Center and AC\_Proxy Software

Download the app-connector RPM and acproxy software from <https://downloads.f5.com> .

Please make sure that the app-connector and f5-acproxy software are the same version and build number.

Example:

* f5-acproxy-1.0.0-build.44.tgz
* app-connector-1.0.0-44.0.noarch.rpm

Place the f5-acproxy-1.0.0-build.XX.tgz software in the ~/ansible/playbooks directory on the Ansible host. Now in the same directory create a file named docker.list and add the following line to it.

* deb https://apt.dockerproject.org/repo ubuntu-xenial main

Close and save the file.

Configuring the Application Service Center

The Application Connector service center needs to be running v13 or higher software. Spin up the F5 device that will be the Application Connector Service Center and perform initial configuration. You can do this using Ansible with the f5\_HA playbook located on Github at <https://github.com/mlowcher61> . If you will be using trunks on your f5s, use the f5\_HA\_trunks playbook.

This procedure is documented on Github in the mlowcher61 repository. Look under documentation for “[Ansible orchestration for F5 LTM and GSLB.docx](https://github.com/mlowcher61/F5-Ansible/blob/master/documentation/Ansible%20orchestration%20for%20F5%20LTM%20and%20GSLB.docx)”

Connectivity to your F5s from Ansible was discussed earlier in section 4.1.

You need to stage two files in the Ansible ~/ansible/playbooks directory.

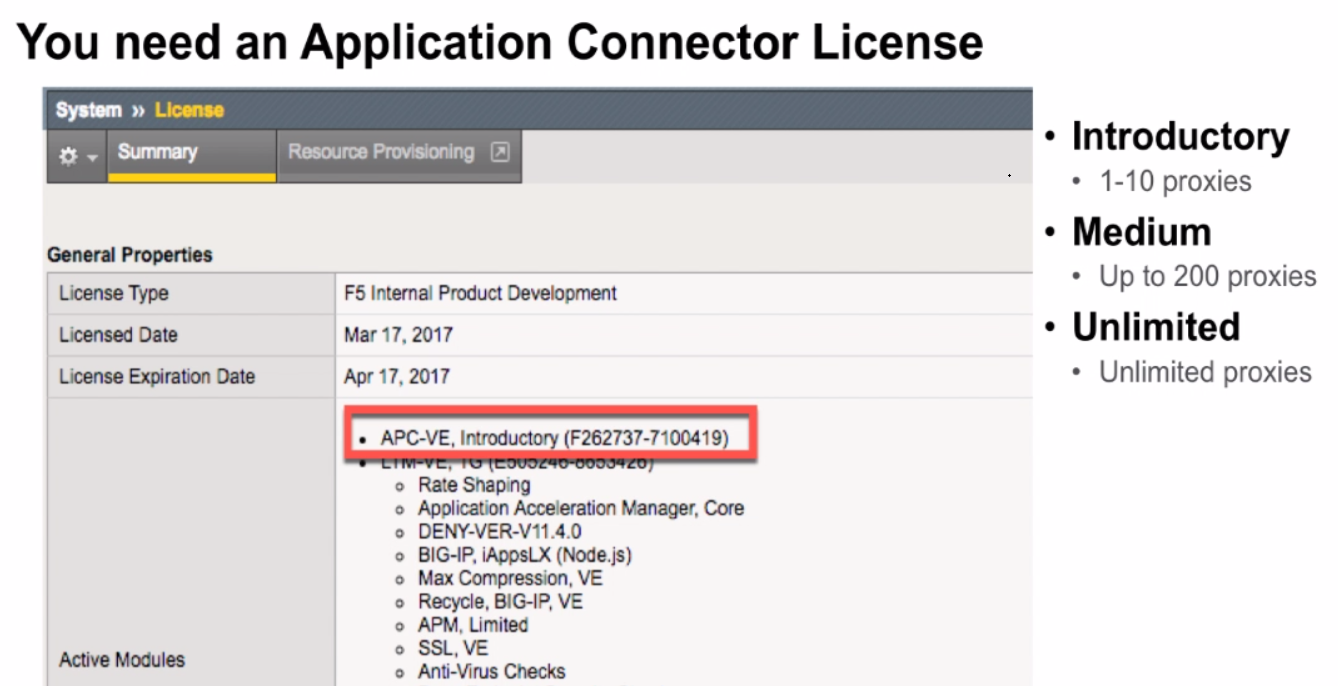
* The f5-acproxy-1.0.0-build.XX.tgz
* A file named docker.list with one line that reads:
  + deb https://apt.dockerproject.org/repo ubuntu-xenial main

Running the Enable Application Connector Playbook

Now that the F5 for Application Connector Service Center is licensed and has a base configuration and Ansible can successfully “ansible ping” to the F5. We will double check that all dependencies have been taken care of:

* F5 App Connector service center is running v13 or greater
* Verify that the F5 is licensed for Application Connector. Look under System -> License and find the entry in the license file that reads “APC”
* Verify the iRulesLX is provisioned under System -> Resource Provisioning





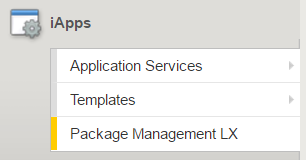
Now run the enable\_appconnector playbook.

* ansible-playbook enable\_appconnector.yml -vvvv

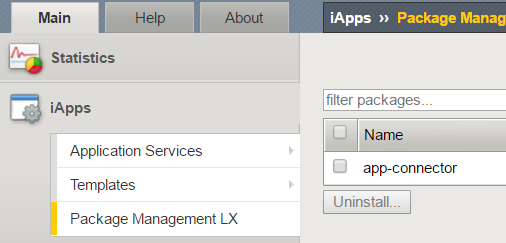
After running this playbook, you will need to go to the GUI of the F5 to perform the following tasks.

Import the Application Center RPM

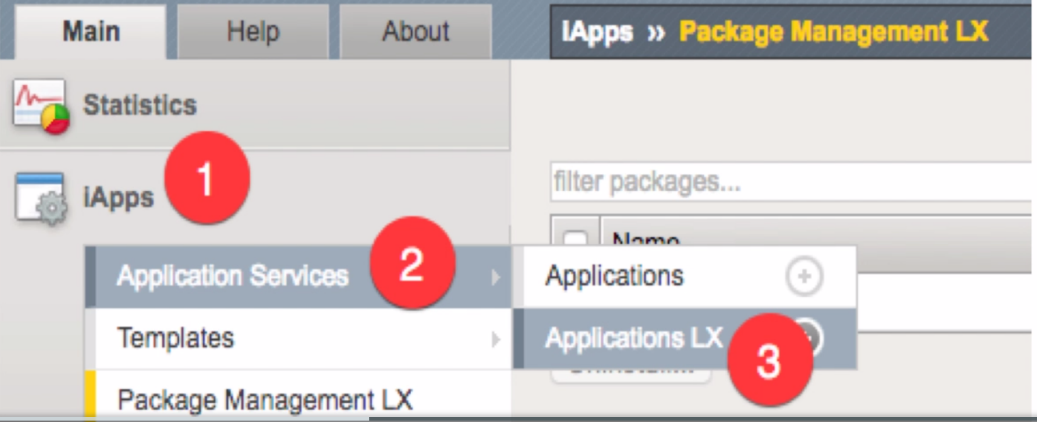
After running the enable\_appconnector playbook the GUI should now show “Package Management LX” under iApps



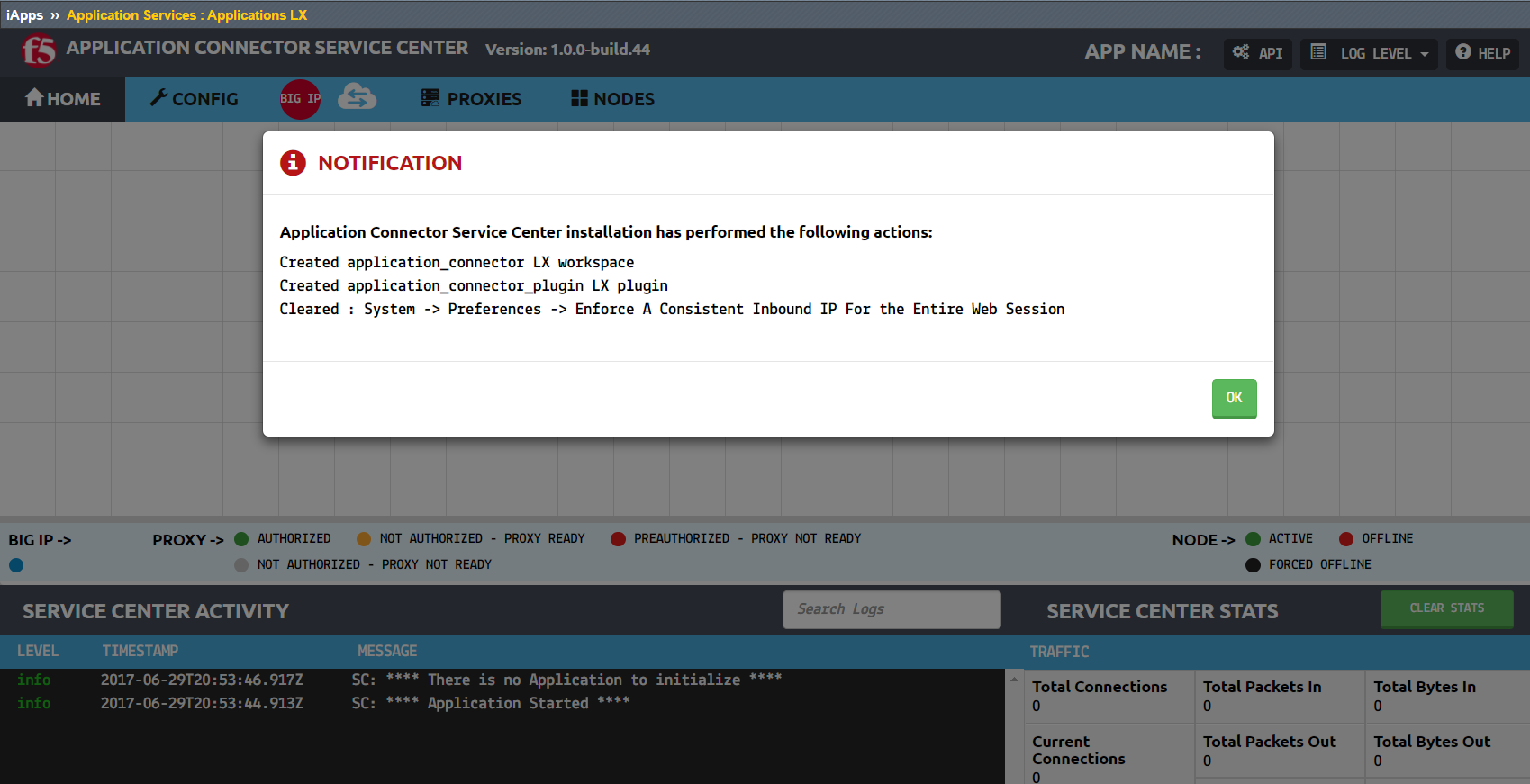
From here click import and then “choose file” and select the app-connector-1.0.0-XX.0.noarch.rpm from your desktop and click upload. You should now see the following:



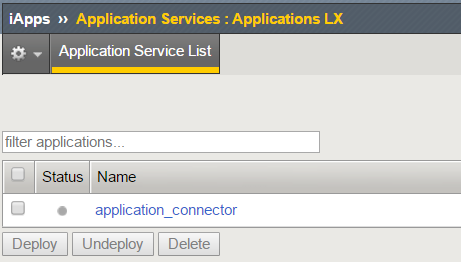
Now go to iApps -> Application Services -> Applications LX and click create on the far right.



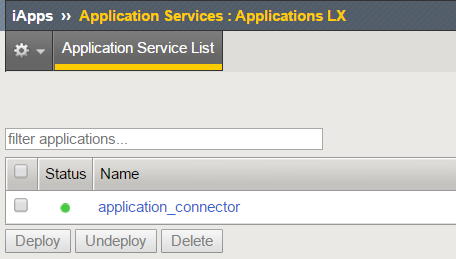
From the “Template” dropdown select “app-connector”. You will now see the following screen.



Click “Ok” to the notification message. Scroll to the bottom of the page and next to “App Name” put “application\_connector” and click “save”. You will now be back at the “Applications LX screen and see the following.



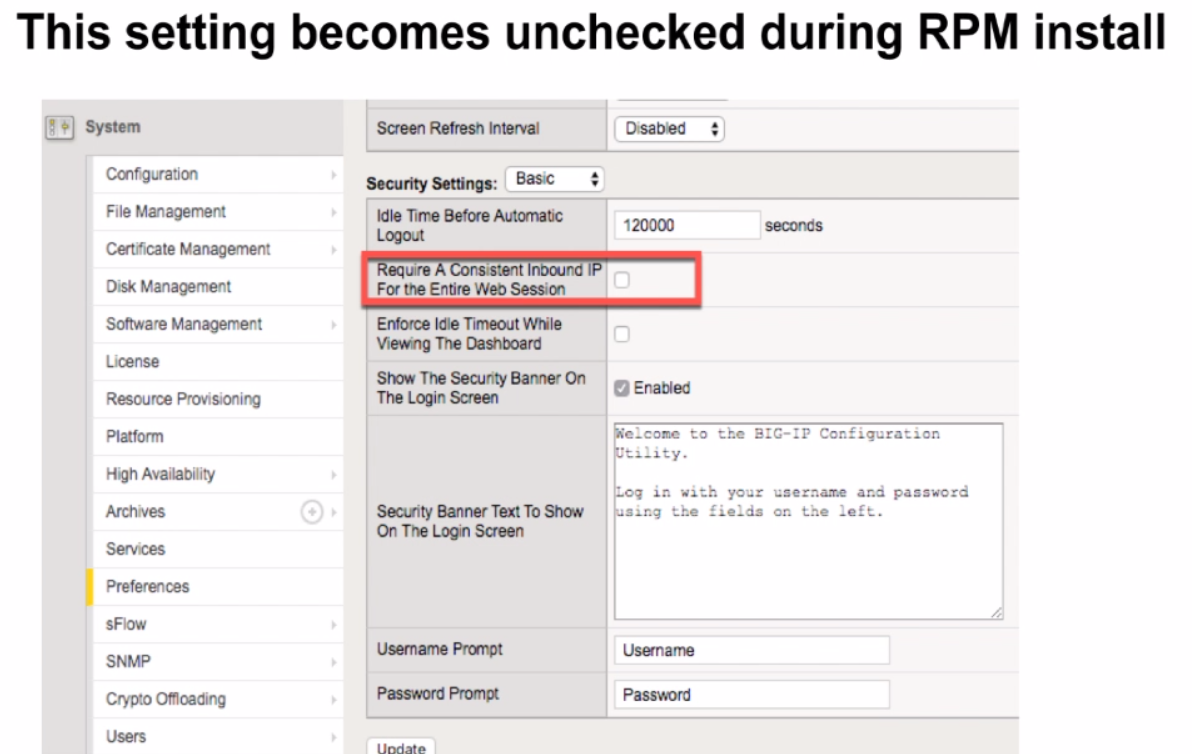
Check the box to the left of “application\_connector” and click deploy. The grey circle should now be green.



The Service Center is now installed. This process will uncheck the “Require A Consistent Inbound IP For the Entire Web Session”. If you have any issues with Service Center, verify that this setting under System -> Preferences is NOT checked. This can also be disabled using the tmsh command:

* tmsh modify sys httpd auth-pam-validate-ip off.

But it should be turned off automatically as we saw in the notification pop up a minute ago.



Creating the External Monitor

Open the appconn\_monitor playbook and edit the vars to match your environment.

Run the appconn\_monitor playbook.

Now we can run the appconnector\_vips playbook.

Running the Applicationconnector\_vips Playbook.

Open the appconnector\_vips playbook and edit the vars to match your environment.

Run the appconnector\_vips playbook.

Installing AC\_proxy in AWS Cloud

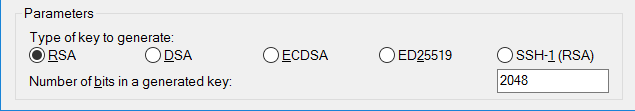
Create a Key Pair in AWS

Log into AWS console and click on “key pair” on the left of the screen. Click on “Create Key Pair” and give it a name such as “Marks-PublicKey”. Your public key is now stored on AWS to be used with your ec2 instances and your private key will automatically be downloaded to your computer.

Converting Your Private Key Using PuTTYgen

PuTTY does not natively support the private key format (.pem) generated by Amazon EC2. PuTTY has a tool named PuTTYgen, which can convert keys to the required PuTTY format (.ppk). You must convert your private key into this format (.ppk) before attempting to connect to your instance using PuTTY.

1. Start PuTTYgen (for example, from the **Start** menu, choose **ALL Programs > PuTTY > PuTTYgen).**
2. Under **Type of key to generate**, choose **RSA**



1. Choose **Load**. By default, PuTTYgen displays only files with the extension .ppk. To locate your .pem file, select the option to display files of all types.
2. Select your .pem file for the key pair that you specified when you launch your instance, then choose **Open**. Choose **OK** to dismiss the confirmation dialog box.
3. Choose **Save private key** to save the key in the format that PuTTY can use. PuTTYgen displays a warning about saving the key without a passphrase. Choose **Yes**.

**Note**

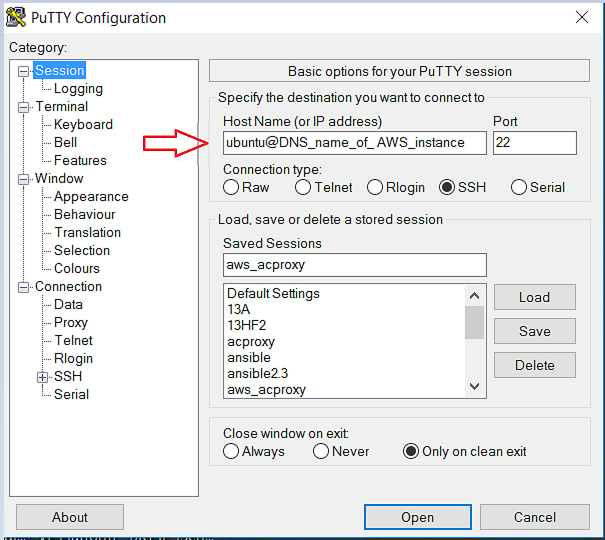
A passphrase on a private key is an extra layer of protection, so even if your private key is discovered, it can't be used without the passphrase. The downside to using a passphrase is that it makes automation harder because human intervention is needed to log on to an instance, or copy files to an instance.

Your private key is now in the correct format for use with PuTTY.

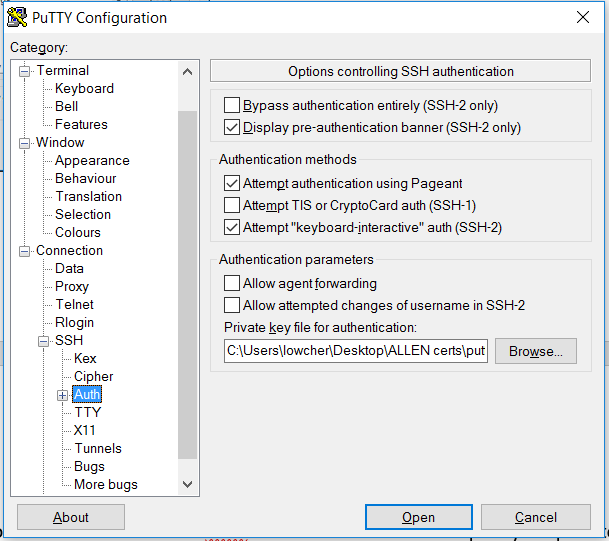
Configuring PuTTY with the Private Key

To SSH to your instance once it is online you will use your private key. If you will be connecting from a windows machine I suggest using Putty which is a free SSH application.

Open PuTTY and fill in the host name with “ubuntu@ and the dns name of your Ubuntu instance. And give the instance a name under “Saved Sessions”



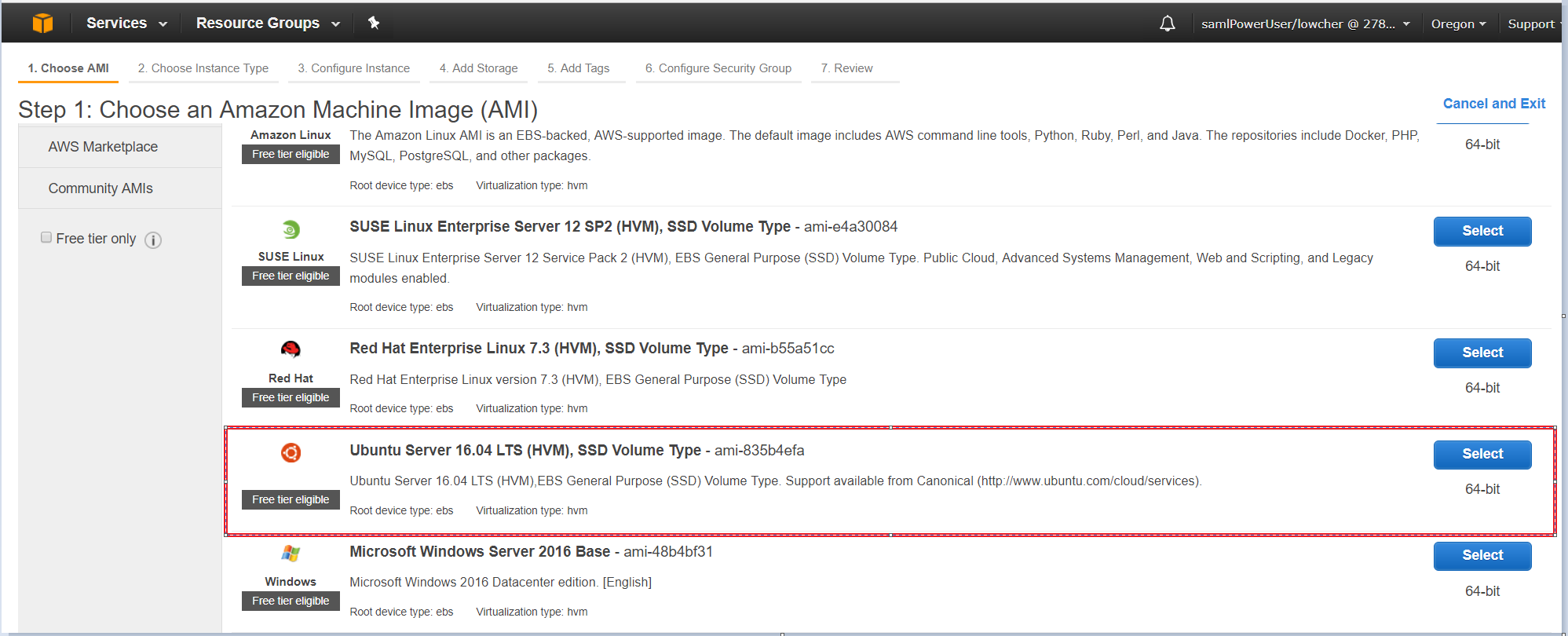
Now expand the “SSH” on the left and click on “Auth”. Click on browse and import your private key. Then click on “ “Session” at the top and save your PuTTY configuration.



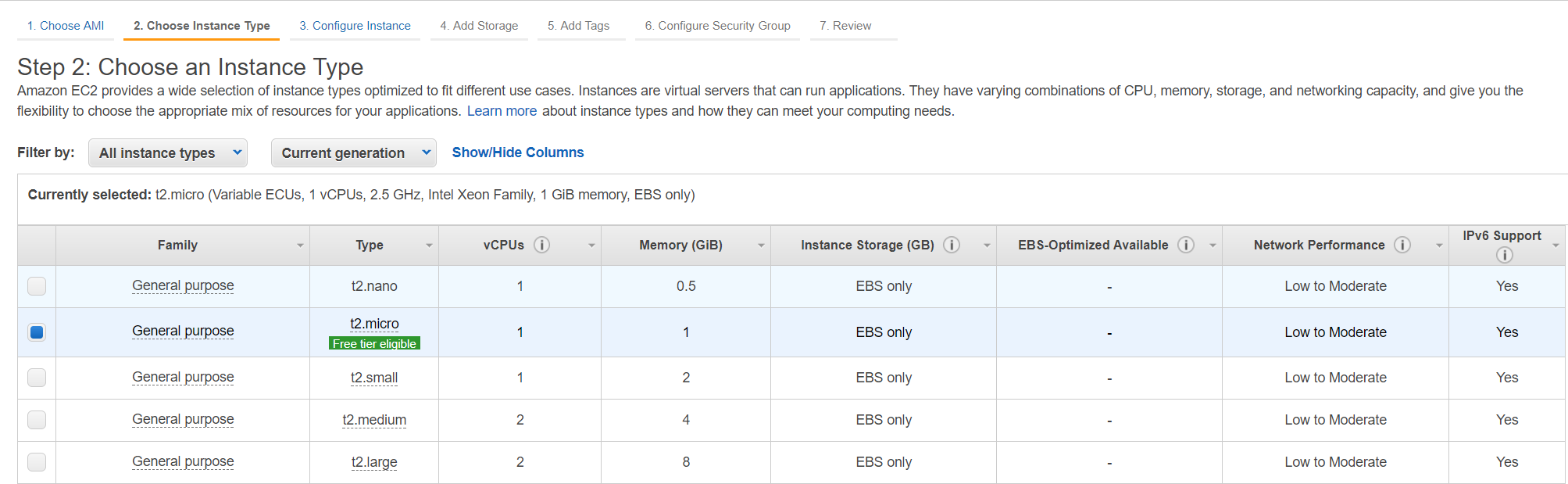
You will now be able to SSH to your Ubuntu instance.

Launch an EC2 Instance in AWS

From the AWS console home screen click on “EC2” and then click on “Launch Instance” Select the “Ubuntu Server 16.04”.



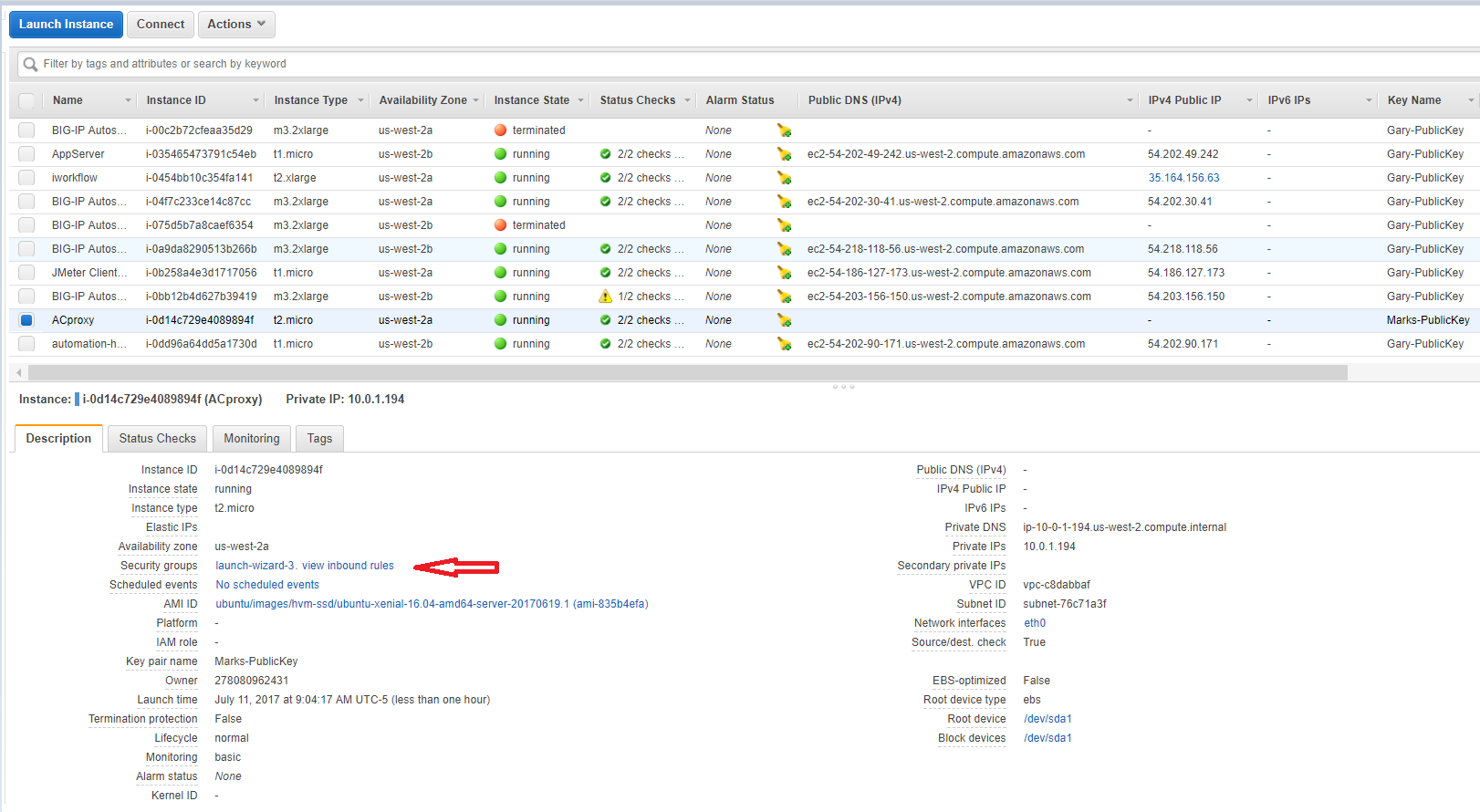
Choose the instance type you will require for your workload. Go with t2.micro at a minimum.



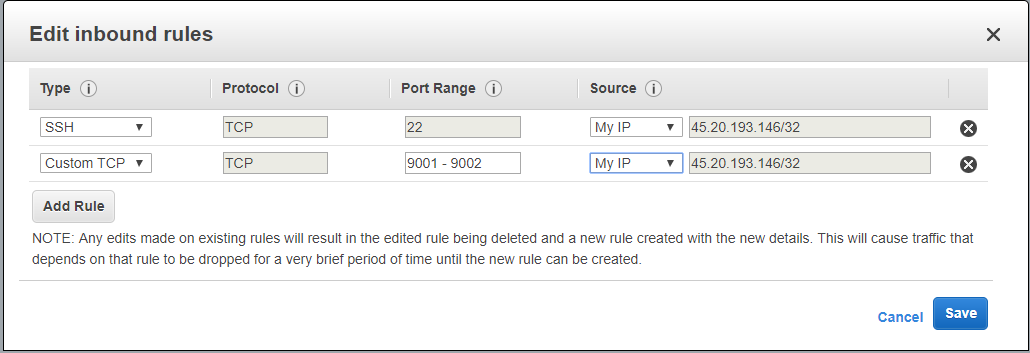
Click “Next: Configure Instance Details”. Use an existing VPC and subnet or create a new one then click “Review and Launch” and then “Launch” on the next page. Choose the public key you generated earlier and accept the disclosure and launch your instance. Click “View Instances” to see all you EC2 instances. Your new Ubuntu instance will not yet have a name. Click on the blank cell in the “Name” column and name it “ACproxy” or whatever name you would prefer. It will take a few minutes for the instance to show as running and that the status checks have completed.

Create Inbound Rules for the ACproxy

Click on the box to the left of “ACproxy”. This will show you the specifics of this instance.

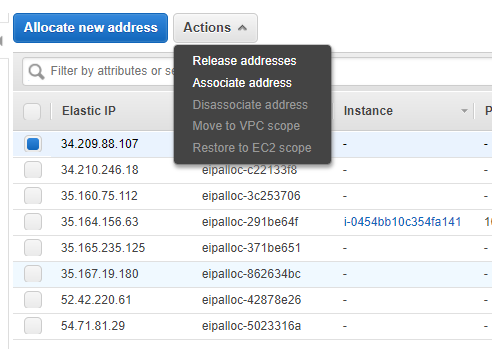


Click on the Security groups hyperlink and then click on “Inbound” near the bottom of the screen. There will be a rule in place to allow SSH port 22 from ANY source (0.0.0.0/0). It is a best practice to edit this field and select “My IP” form the “Source” drop down list. Also click on “Add Rule” and allow TCP ports 9001 and 9002 from your ip address then click save.



Associate a Public IP to the ACproxy Instance

Go back to EC2 instances and notice that column “Public DNS (IPv4) is empty. Click on “Elastic IPs” on the left and select a public ip to use for the ACproxy instance, then click on “Actions” and “Associate address”



On the next screen, select your ACproxy instance, the private ip from the drop down and check “Reassociation” and then “Associate”. Go back to your EC2 instances and you will see that you now have a public ip and a DNS name. Go back to PuTTY and enter the DNS name next to ubuntu@. You should now be able to SSH to your ACproxy instance. If you can’t, go to <https://console.amazon.com/vpc> and click on “Internet Gateways” and create a default route out of your VPC.

Preparing Ubuntu and Ansible to Communicate Together

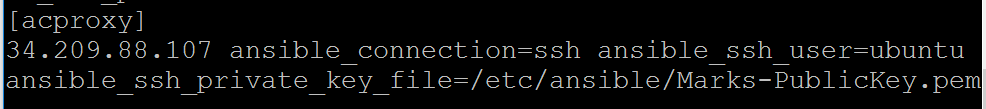
SCP your .pem file to your Ansible instance. Put it in the /etc/ansible directory.

**The Ansible instance does not have to be in the cloud it just must be routable to your ACproxy instance.**

Once the .pem files is in the /etc/ansible directory change the permissions so it has only read access for root with the following command.

* sudo chmod 400 <my\_keys.pem>

Now from the Ansible shell edit the /etc/ansible/hosts file. Create a heading named “ACproxy” and enter the ip address of the ACproxy underneath with the following Ansible arguments.

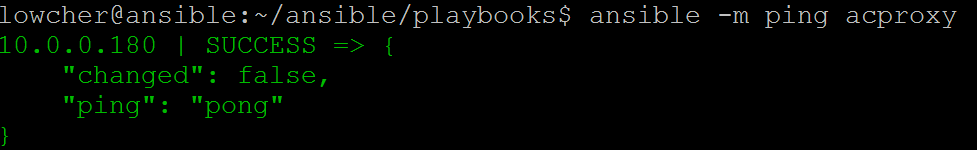


Now SSH to the Ubuntu from your desktop PuTTY session and accept the crypto fingerprint. We will now install the dependencies for Ansible to be able to communicate with the Ubuntu instance.

* sudo apt-get install python-pip --assume-yes
* sudo pip install --upgrade pip
* sudo apt-get install apt-transport-https ca-certificate
* sudo apt-get update

Now verify that Ansible can communicate with acproxy. From the Ansible command line run:

* ansible -m ping acproxy



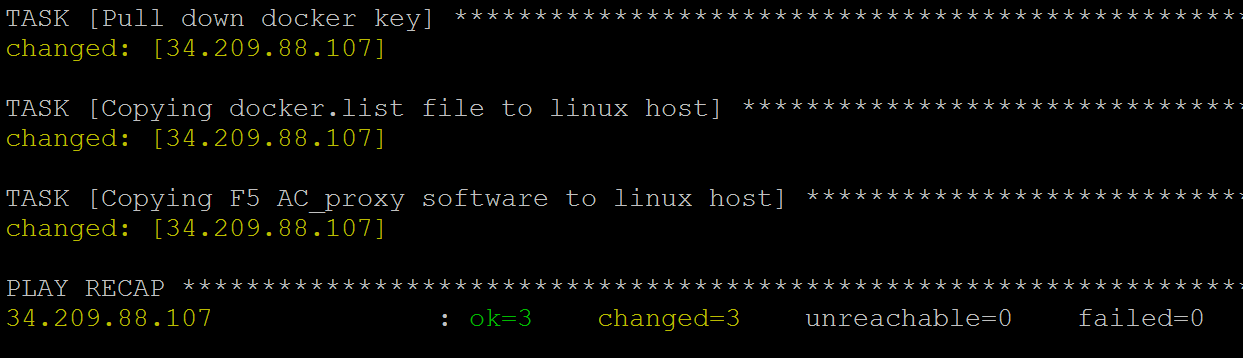
You should get a green SUCCESS message. Ansible is now ready to run playbooks against the acproxy.

Run the acproxy\_part1 Playbook

From the Ansible host within the ~/ansible/playbooks directory open the acproxy\_part1 playbook and edit the vars to match your environment. For host and appliance you can use “acproxy” as it is defined in the /etc/ansible/hosts file.

Run the acproxy\_part1 playbook using the following command:

* ansible-playbook acproxy\_part1.yml -vvvv



A successful run will show the above.

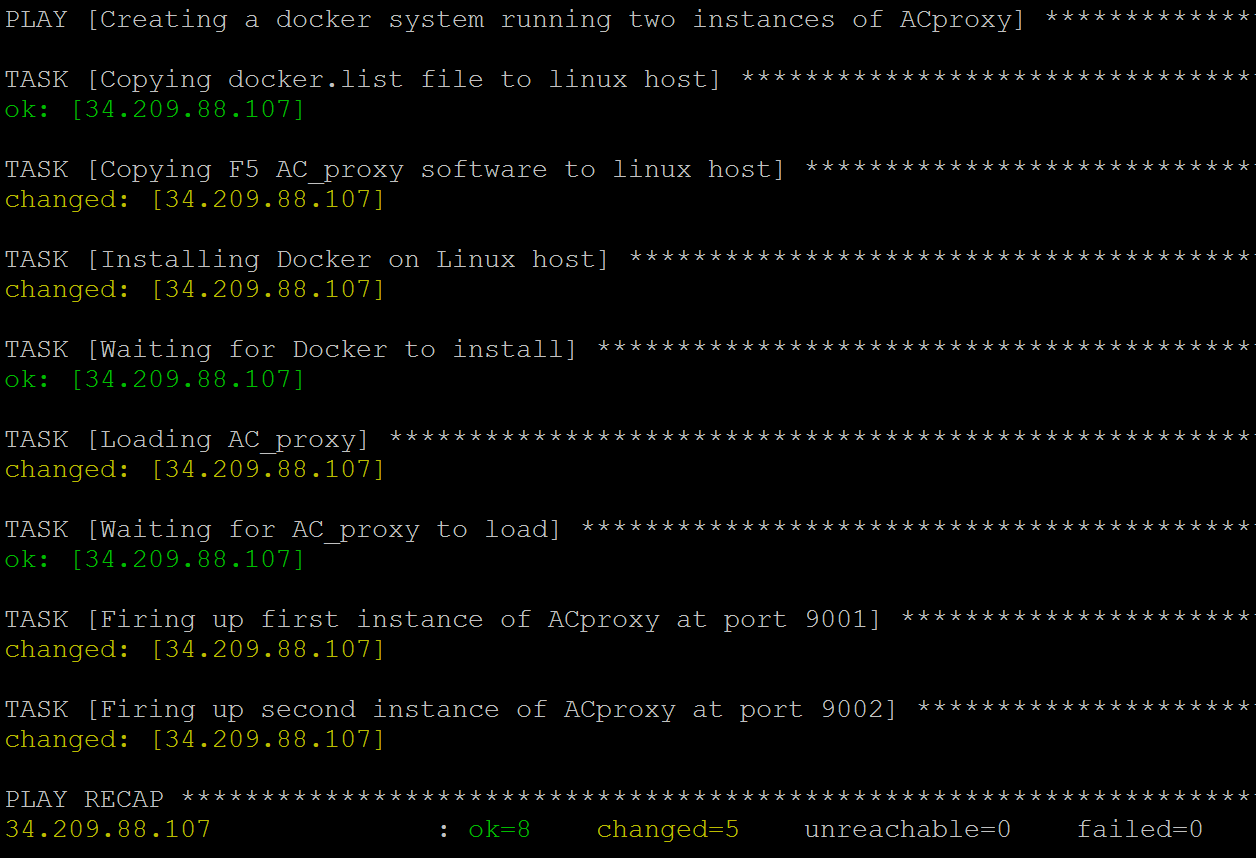
From the acproxy command line run the following command:

* sudo apt-get update

Run the acproxy\_part2 Playbook

Now edit the vars for the acproxy\_part2 playbook and run the acproxy\_part2 playbook

* ansible-playbook acproxy\_part2.yml -vvvv



A successful run should look similar to the screenshot above with no red and failed=0.

Connect to the AC\_proxy Instances

From a web browser go to <https://ip_of_acproxy:9001> and login as admin admin

Now try the second instance at <https://ip_of_proxy:9002> and login as admin admin

You now have two instances of AC\_proxy and you are ready to go.