3.0.0

Table of Contents

1Introduction 4

2Ubuntu Enterprise Cloud Installation 4

3Server Software Installation & Configuration 5

3.1 Install & Configure Eucalyptus Tools 5

3.2 Download and Install Ubuntu EMI 6

3.3 Create a Key-pair 7

3.4 Start a New Ubuntu Server Instance 7

3.5 Install Server Software 8

3.6 Create Storage Volume for PostgreSQL data 9

3.7 Create Buckets for Wars and Plugins 10

3.8 Configure PostgreSQL 11

3.9 Configure Apache2 11

3.10 Configure Tomcat6 12

3.11 Create and upload a new version of the EMI 12

3.12 Create Security Groups 13

3.13 Create the User Data Initialization Scripts 14

3.14 Deploy the Wars 16

4Running the Servers 16

4.1 Web Application 16

4.2 Worker 17

Glossary

|  |  |
| --- | --- |
| Term | Definition |
| Cloud | A federated set of physical machines that offer computing resources through virtual machines, provisioned and recollected dynamically. |
| Cloud Controller (CLC) | Eucalyptus component that provides the web UI (an https server on port 8443), and implements the Amazon EC2 API. There should be only one Cloud Controller in an installation of UEC. This service is provided by the Ubuntu eucalyptus cloud. |
| Cluster | A collection of nodes, associated with a Cluster Controller. There can be more than one Cluster in an installation of UEC. Clusters are sometimes physically separate sets of nodes. (e.g. floor1, floor2, floor2). |
| Cluster Controller (CC) | Eucalyptus component that manages collections of node resources. This service is provided by the Ubuntu eucalyptus cc package. |
| EBS | Elastic Block Storage. http://aws.amazon.com/ebs/ |
| EC2 | Elastic Compute Cloud. Amazon's pay by the hour, pay by the gigabyte public cloud computing offering. |
| EKI | Eucalyptus Kernel Image. |
| EMI | A Eucalyptus Machine Image (EMI) is a combination of virtual disk image(s), kernel, ram disk images, and an XML file containing metadata about the image. |
| ERI | Eucalyptus Ramdisk Image. |
| Eucalyptus | Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems. An open source project originally from the University of California at Santa Barbara, now supported by Eucalyptus Systems, a Canonical Partner. |
| Front-end | Physical machine hosting one (or more) of the high level Eucalyptus components (cloud, walrus, storage controller, cluster controller). |
| Node | A node is a physical machine that's capable of running virtual machines, running a node controller. Within Ubuntu, this generally means that the CPU has VT extensions, and can run the KVM hypervisor. |
| Node Controller (NC) | Eucalyptus component that runs on nodes which host the virtual machines that comprise the cloud. This service is provided by the Ubuntu package eucalyptus nc. |
| S3 | Simple Storage Service. Amazon's pay by the gigabyte persistent storage solution for EC2. http://aws.amazon.com/s3/. |
| Storage Controller (SC) | Eucalyptus component that manages dynamic block storage services (EBS). Each 'cluster' in a Eucalyptus installation can have its own Storage Controller. This component is provided by the 'eucalyptus sc' package. |
| UEC | Ubuntu Enterprise Cloud. Ubuntu's cloud computing solution, based on Eucalyptus. |
| VM | Virtual Machine. |
| VT | Virtualization Technology. An optional feature of some modern CPUs, allowing for accelerated virtual machine hosting. |
| Walrus | Eucalyptus component that implements the Amazon S3 API, used for storing VM images and user storage using S3 bucket put/get abstractions. |

# Introduction

This document describes the procedures to deploy the Cloud Processing Framework (CPF) to physical or cloud servers.

* Physical Servers
* Amazon EC2 Cloud
* A private Ubuntu Enterprise Cloud
* A private Openstack Cloud

The CPF can be deployed to any Tomcat 7+ servlet container and on any operating system. The procedures described here use Ubuntu[[1]](#footnote-1) as the operating system due to the ease of installation.

# Cloud/Server Installation

The CPF can either be deployed to physical servers or a shared cloud infrastructure.

If the servers are dedicated to running the CPF then deploying to physical servers may be a better option, as it is a simpler environment to install and manage.

A cloud infrastructure is appropriate if the servers are to be shared amongst other applications that require different operating systems or server software versions. The advantage of a cloud deployment is that the cloud instances can be easily moved from one server to another or additional servers added based on the load of the system. For CPF this could be used to add additional server instances for processing jobs overnight when other web applications aren't used as frequently.

## Physical Servers

For physical server deployment install Ubuntu or another operating system directly on the servers hard drives.

## OpenStack / Ubuntu Cloud Infrastructure

The Ubuntu Cloud Infrastructure (UCI) can be used to create a private cloud on one or more physical servers. UCI is part of the Ubuntu Server Linux distribution and is based on the OpenStack cloud software. The UCI can be deployed to one or more physical servers that will run the compute, storage and image controllers. The Ubuntu Cloud Infrastructure replaces UEC.

Documentation for UCI can be found on the following WIKI.

<https://help.ubuntu.com/community/UbuntuCloudInfrastructure>

Documentation for OpenStack can be found on the following sites.

<http://docs.openstack.org/>

<http://wiki.openstack.org/>

Several companies provide hosting solutions using the OpenStack software. The following page lists the companies that are involved in the OpenStack project. Some of these may provide hosting services similar to those provided by Amazon.

<http://openstack.org/community/companies/>

## Eucalyptus / Ubuntu Enterprise Cloud

The Ubuntu Enterprise Cloud (UEC) can be used to create a private cloud on one or more physical servers. UEC is part of the Ubuntu Server Linux distribution and is based on the Eucalyptus cloud software. The UEC can be deployed to one or more physical servers that will run the cloud, storage and node controllers. Ubuntu is migrating their cloud software away from UEC to the Ubuntu Cloud Infrastructure.

Documentation for UEC can be found on the following WIKI.

<https://help.ubuntu.com/community/UEC>

Follow these instructions are for new servers without an existing operating system.

<https://help.ubuntu.com/community/UEC/CDInstall>

Follow these instructions for existing server running Ubuntu Maverick Meerkat (10.10).

<https://help.ubuntu.com/community/UEC/PackageInstall>

## Amazon Web Services

The Amazon Web Services provide a cloud infrastructure that runs on Amazons servers rather than locally managed servers. The advantage of the Amazon Web Services is that you only pay (by the hour) for server instances that are currently running rather than for a dedicated physical server. Additional instances can be added within a few minutes to handle increased demand as required.

The disadvantages are:

* If a server instance is running 100% of the time then it could be more expensive than a physical server.
* The servers are hosted on Amazon's infrastructure, for some applications this could be a security concern.

For details on the Amazon Elastic Compute Cloud (EC2) that runs cloud operating system images see.

<http://aws.amazon.com/ec2/>

For details on the Amazon Simple Storage Service (S3) that can be used to store the CPF WAR and configuration files see.

<http://aws.amazon.com/s3/>

For details on the Amazon Elastic Block Store (EBS) that can be used to create virtual disc images to store the CPF database see.

<http://aws.amazon.com/ebs/>

For details on the Amazon Relational Database Service (RDS) that can be used to provide a managed Oracle database instead of installing a local PostgreSQL database see.

<http://aws.amazon.com/rds/>

# Install & Configure Eucalyptus Tools

The Eucalyptus open source project provides command line tools that are compatible with the Amazon web service APIs. These tools can be used for all of the cloud software described in this document as they provide cloud management APIs that are compatible with the Amazon Web Service API.

## Installation

Use the following command to install the Eucalyptus tools and the Simple Storage Service (S3) command line tools.

apt-get -y install euca2ools

apt-get -y install s3cmd

## Configuration

The Eucalyptus tools require several files and environment variables to be present to be able to connect to the server.

### Physical Servers

The Eucalyptus tools are not required for physical servers.

### OpenStack / Ubuntu Cloud Infrastructure

Use the following commands on a Nova server to download the configuration file and authentication credentials.

mkdir -p ~/.nova

cd ~/.nova

nova-manage project zipfile **<Name of Nova Project>** **<Name of Nova User>**

unzip nova.zip

chmod -R go-rwx ~/.nova

ln -s ~/.nova/novarc ~/.eucarc

echo ". ~/.eucarc" >> ~/.profile

. ~/.eucarc

Create the ~/.s3cfg configuration file. Replace the items in bold with the values for your environment.

[default]

access\_key = **ACCESSKEY**

secret\_key = **SECRET**

acl\_public = False

bucket\_location = US

debug\_syncmatch = False

default\_mime\_type = binary/octet-stream

delete\_removed = False

dry\_run = False

encrypt = False

force = False

gpg\_command = /usr/bin/gpg

gpg\_decrypt = %(gpg\_command)s -d --verbose --no-use-agent --batch --yes --passphrase-fd %(passphrase\_fd)s -o %(output\_file)s %(input\_file)s

gpg\_encrypt = %(gpg\_command)s -c --verbose --no-use-agent --batch --yes --passphrase-fd %(passphrase\_fd)s -o %(output\_file)s %(input\_file)s

gpg\_passphrase =

guess\_mime\_type = False

host\_base = **localhost:8773**

host\_bucket = **localhost:8773**

service\_path = **/services/Walrus**

human\_readable\_sizes = False

preserve\_attrs = True

proxy\_host =

proxy\_port = 0

recv\_chunk = 4096

send\_chunk = 4096

use\_https = False

verbosity = WARNING

### Eucalyptus / Ubuntu Enterprise Cloud

Use the following commands on a Eucalyptus server to download the configuration file and authentication credentials.

mkdir -p ~/.euca

cd ~/.euca

euca\_conf --get-credentials ~/euca.zip

unzip euca.zip

chmod -R go-rwx ~/.euca

ln -s ~/.euca/eucarc ~/.eucarc

echo ". ~/.eucarc" >> ~/.profile

. ~/.eucarc

Create the ~/.s3cfg configuration file. Replace the items in bold with the values for your environment.

[default]

access\_key = **ACCESSKEY**

secret\_key = **SECRET**

acl\_public = False

bucket\_location = US

debug\_syncmatch = False

default\_mime\_type = binary/octet-stream

delete\_removed = False

dry\_run = False

encrypt = False

force = False

gpg\_command = /usr/bin/gpg

gpg\_decrypt = %(gpg\_command)s -d --verbose --no-use-agent --batch --yes --passphrase-fd %(passphrase\_fd)s -o %(output\_file)s %(input\_file)s

gpg\_encrypt = %(gpg\_command)s -c --verbose --no-use-agent --batch --yes --passphrase-fd %(passphrase\_fd)s -o %(output\_file)s %(input\_file)s

gpg\_passphrase =

guess\_mime\_type = False

host\_base = **localhost:8773**

host\_bucket = **localhost:8773**

service\_path = /services/Walrus

human\_readable\_sizes = False

preserve\_attrs = True

proxy\_host =

proxy\_port = 0

recv\_chunk = 4096

send\_chunk = 4096

use\_https = False

verbosity = WARNING

### Amazon Web Services

Use the following steps to configure access to the Amazon Web Services.

mkdir -p ~/.ec2

cd ~/.ec2

1. Go to the Amazon Web Service Credentials Page  
   <https://aws-portal.amazon.com/gp/aws/securityCredentials>
2. Under Access Credentials click on Access Keys to get the Access Key Id and Secret Access Key. Record these for use in the ec2rc file below.
3. Under X.509 Certificates, Create a new certificate. Download the Private Key File and Download the X.509 Certificate files and copy them to ~/.ec2

Create the file ~/.ec2/ec2rc, replacing the items in bold with the appropriate values.

CREDENTIALS\_DIR=${HOME}**/.ec2/**

EC2\_REGION="${EC2\_REGION:**-us-east-1**}"

EC2\_CERT=${CREDENTIALS\_DIR}/**cert-ABCDEFGHIJKLMNOP.pem**

EC2\_PRIVATE\_KEY=${CREDENTIALS\_DIR}/**pk-ABCDEFGHIJKLMNOP**.pem

EC2\_ACCESS\_KEY=**ABCDEFGHIJKLMNOPQRST**

EC2\_SECRET\_KEY=**UVWXYZ0123456789abcdefghijklmnopqrstuvwx**

EC2\_USER\_ID=**950047163771**

EUCALYPTUS\_CERT=/etc/ec2/amitools/cert-ec2.pem

EC2\_URL=**https://ec2.${EC2\_REGION}.amazonaws.com**

S3\_URL=**https://s3.amazonaws.com:443**

Run the following commands to finish the configuration.

chmod -R go-rwx ~/.ec2

ln -s ~/.ec2/ec2rc ~/.eucarc

echo ". ~/.eucarc" >> ~/.profile

. ~/.eucarc

Run the following command to create the ~/.s3cfg configuration file. Enter in the Access Key and Secret Key from your Amazon account

s3cmd --configure

# Cloud Instance Installation & Configuration

The following instructions show how to create a new EMI that can be used for the CPF Web Application and CPF Worker Servers.

This document describes how to use command line tools to interact with the operating system and cloud. GUI and web tools can also be used for some of the tasks. These are outside the scope of this document.

## Download and Install Ubuntu EMI

Before creating a customized CPF EMI an operating system image must be downloaded and registered with the UEC Cloud Controller. These instructions assume that the CPF cloud image will be created to use the Ubuntu Server Edition release 10.10 (Maverick Meerkat) operating system. Newer versions may be used if required.

These instructions are derived from the following page.

<https://help.ubuntu.com/community/UEC/BundlingImages>

Run the following commands to install a Ubuntu Server 10.10 release for i386. If all the nodes are AMD 64bit compatible you can change ARCH to amd64.

ssh cpf@controller

RELEASE=**10.10**

ARCH=**i386** # or amd64

UEC\_IMG=ubuntu-$RELEASE-server-uec-$ARCH

URL=http://uec-images.ubuntu.com/releases/${RELEASE}/release/

cd /tmp

wget $URL/ubuntu-${RELEASE}-server-uec-${ARCH}.tar.gz

uec-publish-tarball $UEC\_IMG.tar.gz ubuntu-server-$ARCH-$RELEASE

After running the uec-publish-tarball command the EMI id will be displayed as shown below. This value is required

emi="**emi-F5521911**"; eri="none"; eki="eki-6A881E1E";

## Create a Key-pair

A key-pair is required when starting a new instance so that the server can be accessed via ssh.

ssh cpf@controller

Run the following command to see if the cpf-key key-pair has been created

euca-describe-keypairs | grep cpf-key

If there is no output run the following commands to create a key-pair.

cd ~/.euca

euca-add-keypair cpf-key > cpf-key.priv

chmod 600 cpf-key.priv

euca-describe-keypairs | grep cpf-key

The following output is returned. This information can be retrieved using the euca-describe-keypairs command.

KEYPAIR cpf-key **52:50:77:d3:d8:ac:28:2e:9d:93:a6:e1:d4:8e:73:4e:c2:a8:23:41**

## Start a New Ubuntu Server Instance

An instance of Ubuntu Server must be run in the cloud to install the software.

ssh cpf@controller

Run the following commands to run an instance of the image installed in 3.2.

RELEASE=10.10

ARCH=i386 # or amd64

EMI=`euca-describe-images |grep emi |grep ubuntu-server-$ARCH-$RELEASE | awk '{ print $2 }'`

euca-run-instances --instance-type m1.large --key cpf-key $EMI

The result of the command will show id of the created instance. The response does not immediately return the IP address of the created instance. Use the following command to check the status of the instance (not pending) and to get the allocated IP address.

euca-describe-instances

The result shown below shows the external IP address in bold. This is required to connect to the server via ssh to install software. When the status changes from pending to running then you can login to the cloud instance.

RESERVATION r-3AEF071C admin default

INSTANCE i-48A707AE emi-F4E318FE **192.168.1.200** 172.19.1.2 **running** 0 m1.large 2011-03-29T18:10:10.759Z cpf eki-6A571E13

The following command is used to login to the cloud instance created above, this must be run from the controller or on a server that has a copy of the ~/.euca/cpf-key.priv file. Replace controller with the IP address allocated to the running instance.

ssh -o StrictHostKeyChecking=no -i ~/.euca/cpf-key.priv ubuntu@**cpf-image**

## Install Server Software

The following instructions show how to install Apache2, Tomcat7, PostgreSQL 9.1 and the S3 file system tools to an Ubuntu server. The Ubuntu server can either be a physical server or a cloud image.

ssh -o StrictHostKeyChecking=no -i ~/.euca/cpf-key.priv ubuntu@**<hostname>**

Run the following commands to upgrade all components of the operating system to the most recent versions and install the server software. PostgreSQL, Apache2, and Tomcat will be configured to not auto start at boot time. The s3fs is a fuse file system must be installed from source.

cat > /tmp/install <<DELIM

apt-get -y update  
apt-get -y dist-upgrade  
apt-get -y install postgresql-9.1

apt-get -y install apache2 libapache2-mod-jk

apt-get -y install euca2ools

apt-get -y install tomcat7 tomcat7-admin

apt-get -y install build-essential

apt-get -y install libfuse-dev

apt-get -y install fuse-utils

apt-get -y install libcurl4-openssl-dev

apt-get -y install libxml2-dev

apt-get -y install mime-support   
find /etc/rc\* -name '\*apache2' -exec rm {} \;

find /etc/rc\* -name '\*postgresql' -exec rm {} \;

cd /usr/local/src  
wget http://s3fs.googlecode.com/files/s3fs-1.61.tar.gz  
tar xvzf s3fs-1.61.tar.gz

cd s3fs-1.61/

./configure --prefix=/usr

make

make install

rm -rf /usr/local/src/s3fs\* /tmp/install

DELIM

sudo bash /tmp/install

## Create Storage Volume for PostgreSQL data

Once an EMI has been created, bundled and registered with Eucalyptus the operating system disk image does not persist any changes to that disk from one run of the EMI to another (except for a EMI reboot).

The CPF PostgresSQL database requires a persistent disk to store the database data.

ssh cpf@controller

Run the following command to get the name of an availability zone.

euca-describe-availability-zones

Run the following command to create a 1GB volume on the cpf availability zone.

euca-create-volume --size **1** -z **cpf**

The volume identifier is returned.

VOLUME vol-59D30634 1 creating 2011-03-30T16:39:10.558Z

Attach the volume to the running instance, using the instance id and volume id. This will attach the volume to the device /dev/vda, Note the use of /dev/vd\* this is required to force it to be auto detected in the operating system.

euca-attach-volume --instance **i-48A707AE** --device **/dev/vda** **vol-59D30634**

The disk needs to be formatted and populated with directories and a PostgreSQL database.

ssh -o StrictHostKeyChecking=no -i ~/.euca/cpf-key.priv ubuntu@**cpf-image**

Use fdisk to create one primary partition.

sudo fdisk **/dev/vda** <<DELIM

n

p

1

t

83

w

DELIM

Run the following commands to format the disk and create the postgresql, apache2, tomcat6 log directories and the postgresql data.

cat > /tmp/data-install <<DELIM

mkfs.ext4 /dev/vda1

mount /dev/vda1 /data

mkdir -p /data/postgresql  
mkdir -p /data/postgresql/log  
mkdir -p /data/postgresql/main  
chown -R postgres:postgres /data/postgresql  
mkdir -p /data/apache2/log  
chown -R www-data:www-data /data/apache2  
mkdir -p /data/tomcat6/log  
chown -R tomcat6:tomcat6 /data/tomcat6

sudo -u postgres /usr/lib/postgresql/9.1/bin/initdb -D /data/postgresql/main

DELIM

sudo bash /tmp/data-install

## Create Buckets for Wars and Plugins

Storage buckets are used to store plugin jar, application war and worker war files that change frequently.

ssh cpf@controller

Create S3 buckets to store the CPF plugins, application war and worker war files.

s3cmd mb s3://cpfAppWars

s3cmd mb s3://cpfWorkerWars

s3cmd mb s3://cpfPlugins

These will be mounted on the running instances as part of the boot process.

## Configure PostgreSQL

ssh -o StrictHostKeyChecking=no -i ~/.euca/cpf-key.priv ubuntu@**cpf-image**

Change the password for the postgres database account, user account and stop the PostgreSQL service.

cat > /tmp/postgres-init DELIM1

sudo –u postgres - psql template1 <<DELIM2

ALTER USER postgres WITH PASSWORD '**cpf2011**';

\q

DELIM2

service postgresql stop  
passwd -u postgres

rm –f /tmp/postgres-init

DELIM1

sudo bash /tmp/postgres-init

Modify the PostgreSQL configuration to use the /data drive and listen on all addresses

sudo vi /etc/postgresql/8.4/main/postgresql.conf

The following shows the expected values for these properties.

data\_directory = '/data/postgresql/main'  
listen\_addresses = '\*'

password\_encryption = on

## Configure Apache2

Configure Apache2 to pass all requests to tomcat.

ssh -o StrictHostKeyChecking=no -i ~/.euca/cpf-key.priv ubuntu@**cpf-image**

Edit the default site.

vi /etc/apache2/sites-available/default

Add the following at the end of the <VirtualHost> definition.

JkMount /\* ajp13

## Configure Tomcat6

Tomcat needs to be configured to have a user account used to deploy wars and manage web applications. Tomcat must also be configured to give the web applications all permissions. The local host must be configured to point to the cpf war file directory.

ssh -o StrictHostKeyChecking=no -i ~/.euca/cpf-key.priv ubuntu@**cpf-image**

Create the tomcat users for the manager configuration.

vi /etc/tomcat6/tomcat-users.xml

<?xml version='1.0' encoding='utf-8'?>

<tomcat-users>

<role rolename="manager"/>

<user username="admin" password="**cpf2011**" roles="manager"/>

</tomcat-users>

Grant web applications all permissions.

vi /etc/tomcat6/policy.d/04webapps.policy

grant {

permission java.security.AllPermission;

};

Modify the localhost definition to use a different appBase and not unpack wars.

vi /etc/tomcat6/server.xml

<Host name="localhost" appBase="/cpf/wars"

unpackWARs="false" autoDeploy="true"

xmlValidation="false" xmlNamespaceAware="false">

## Create and upload a new version of the EMI

The following procedure can be used to create a new AMI from any running instance. This must be run after any change to the operating system or installation of new software which must be permanently saved.

The authentication credentials must be copied to the running instance to create a new EMI.

ssh cpf@controller

cd ~/.euca/

scp -o StrictHostKeyChecking=no -i cpf-key.priv -r ~/.euca/ ubuntu@192.168.1.200:

ssh -o StrictHostKeyChecking=no -i ~/.euca/cpf-key.priv ubuntu@**cpf-image**

**NOTE:** Increment the cpf\_ami\_version based on the current version in the bucket.

cat > /tmp/bundle <<DELIM

. ~ubuntu/.euca/eucarc

export cpf\_ami\_version=**1**  
export prefix=cpf  
export bucket=\$prefix.\$cpf\_ami\_version  
if [ \$(uname -m) = 'x86\_64' ]; then  
 arch=x86\_64  
else  
 arch=i386  
fi  
service apache2 stop  
service tomcat6 stop  
service postgresql stop  
rm -rf /home/ubuntu/.bash\_history /root/.bash\_history /var/cache/tomcat6/\*  
find /var/log -type f -delete  
aptitude clean  
umount /data  
umount /cpf

euca-bundle-vol \  
 --arch \$arch \

--destination /mnt \  
 --prefix \$prefix \  
 --size 2048 \  
 --exclude /data,/mnt,/home/ubuntu/.ssh,/root/.ssh,/root/.ec2,\

/home/ubuntu/.bash\_history,/root/.bash\_history

euca-upload-bundle \  
 --bucket \$bucket \  
 --manifest /mnt/\$prefix.manifest.xml

euca-register \  
 \$bucket/\$prefix.manifest.xml

euca-describe-images |grep \$bucket

DELIM

chmod 755 /tmp/bundle

sudo /tmp/bundle

This will return the EMI id of the created image. This will be used to launch the servers.

IMAGE **emi-D25E0A9C** cpf.1/cpf.manifest.xml admin available public i386 machine eki-B8FF1EB3

## Create Security Groups

Create Eucalyptus security groups for the CPF application and CPF worker instances. These can be used to restrict security access as required.

euca-add-group -d "CPF Application" cpf-app

euca-add-group -d "CPF Worker" cpf-worker

## Create the User Data Initialization Scripts

Eucalyptus provides a metadata service that can be used to send metadata to an instance when it is started. For Ubuntu servers this metadata can contain a bash script that can be executed on start-up. he following instructions show how to create these scripts to initialize the services and network mounts for the cpf-app and cpf-worker instances.

ssh cpf@controller

Run the following to create the bin/cpf-app-user-data.sh script for the cpf-app instance.

export VOLUME\_ID=**vol-c51ec6ac**

mkdir –p bin

cat > bin/cpf-app-user-data.sh << DELIM

#!/bin/bash

export EC2\_INSTANCE\_ID=\`curl -s http://169.254.169.254/1.0/meta-data/instance-id\`

export EC2\_CERT=/root/.euca/cpf-cert.pem

export EC2\_PRIVATE\_KEY=/root/.euca/cpf-pk.pem

mkdir /root/.euca

chmod 700 /root/.euca

cat > \$EC2\_CERT <<EOF

`cat $EC2\_CERT`

EOF

cat > \$EC2\_PRIVATE\_KEY <<EOF

`cat $EC2\_PRIVATE\_KEY`

EOF

euca-attach-volume $VOLUME\_ID -i \$EC2\_INSTANCE\_ID -d /dev/vda

while [ ! -e /dev/vda ]; do

sleep 1;

done

cat > /etc/passwd-s3fs <<EOF

$EC2\_ACCESS\_KEY:$EC2\_SECRET\_KEY

EOF

chmod 700 /etc/passwd-s3fs

cat > /etc/default/tomcat6 <<EOF

JAVA\_OPTS=" \\

-Djava.awt.headless=true \\

-Xmx512m \\

-XX:MaxPermSize=256m

"

EOF

mkdir -p /data

mount -t ext3 -o defaults /dev/vda1 /data

mkdir -p /cpf/wars

mount -t fuse -o allow\_other,url=$S3\_URL s3fs#cpfAppWars /cpf/wars

mkdir -p /cpf/plugins

mount -t fuse -o ro,allow\_other,url=$S3\_URL s3fs#cpfPlugins /cpf/plugins

ln -s /data/postgresql/log/ /var/log/postgresql

ln -s /data/apache2/log/ /var/log/apache2

ln -s /data/tomcat6/log/ /var/log/tomcat6

service postgresql start

service tomcat6 start

service apache2 start

DELIM

Run the following to create the bin/cpf-worker-user-data.sh script for the cpf-worker instance.

export CPF\_APP\_HOST=**cpf-app**

mkdir –p bin

cat > bin/cpf-worker-user-data.sh << DELIM

#!/bin/bash

cat > /etc/passwd-s3fs <<EOF

$EC2\_ACCESS\_KEY:$EC2\_SECRET\_KEY

EOF

chmod 700 /etc/passwd-s3fs

cat > /etc/default/tomcat6 <<EOF

JAVA\_OPTS=" \\

-Djava.awt.headless=true \\

-Xmx512m \\

-XX:MaxPermSize=256m \\

-DcpfWorker.webServiceUrl=http://$CPF\_APP\_HOST/cpf

"

EOF

mkdir -p /data

mkdir -p /data/postgresql/log/

mkdir -p /data/apache2/log/

ln -s /data/apache2/log/ /var/log/apache2

chown www-data:www-data /data/apache2/log/

mkdir -p /data/tomcat6/log/

ln -s /data/tomcat6/log/ /var/log/tomcat6

chown tomcat6:tomcat6 /data/tomcat6/log/

mkdir -p /cpf/wars

mount -t fuse -o allow\_other s3fs#cpfWorkerWars /cpf/wars

mkdir -p /cpf/plugins

mount -t fuse -o ro,allow\_other,url=$S3\_URL s3fs#cpfPlugins /cpf/plugins

service tomcat6 start

service apache2 start

DELIM

## Deploy the Wars

Copy the CPF application war (e.g. cpfApp.war) and CPF Worker war (e.g. cpfWorker.war) and any plugins to the controller.

ssh cpf@controller

s3cmd put cpfApp.war s3://cpfAppWars/cpfApp.war

s3cmd put cpfWorker.war s3://cpfAppWars/cpfApp.war

The Tomcat manager on the cpf-app or cpf-worker instances can be used to hot-deploy new versions of the application wars or worker wars. Alternatively the above process can be followed and then the tomcat server restarted or the application restarted using the Tomcat manager.

Plugins must be installed using process above and the CPF application and worker web applications must be restarted on all servers.

# Running the Servers

## Web Application

Only one instance of the web application can run at any one time. An attempt to run a second instance will cause it to block before the services are started, as it will not be able to attach the volume.

The bin/cpf-app-user-data.sh script performs the following tasks.

* Stores the X.509 certificates and access keys on the server
* Attaches the data volume to /dev/vda and waits for it to attach
* Updates the master-int-habc-cpf.dyndns.org and master-habc-cpf.dyndns.org DNS entries
* Sets the tomcat startup options in /etc/default/tomcat6
* Creates the mount points for /data, /cpf
* Mounts the EBS and S3 files systems to the above mount points
* Creates symbolic links for the log directories
* Starts the postgresql, tomcat6 and apache2 services

ssh cpf@controller

EMI=`euca-describe-images |grep cpf | head -1 | awk '{ print $2'} `

euca-run-instances \

--user-data-file bin/cpf-app-user-data.sh \

--group cpf-app \

--key cpf-key \

--instance-type m1.large \

$EMI

Run the following command until an IP address is allocated.

euca-describe-instances

Edit the worker script to use the IP address of the master.

**NOTE:** Dynamic DNS can be used if required.

vi bin/cpf-worker-user-data.sh

The following line should be modified to have the allocated IP address.

-DcpfWorker.webServiceUrl=http://**cpf-app**/cpf

**NOTE:** Wait until the instance is in a running state and that the CPF application is available at http://**cpf-app**/cpf.

## Worker

Multiple instances of the worker can be started concurrently. The master includes one copy of the worker so additional nodes are only required in heavy load situations.

The bin/cpfworker-user-data.sh script performs the following tasks.

* Stores the access keys on the server
* Sets the tomcat startup options in /etc/default/tomcat6
* Creates a /data directory to store the worker log files
* Creates the mount points for /mnt/master-wars and /mnt/habcdata
* Mounts the S3 files systems to the above mount points
* Creates a/mnt/wars directory with a symbolic link to the worker.war on /mnt/master-wars
* Creates symbolic links for the log directories
* Starts the tomcat6 and apache2 services

ssh cpf@controller

EMI=`euca-describe-images |grep cpf | head -1 | awk '{ print $2'} `

euca-run-instances \

--user-data-file bin/cpf-worker-user-data.sh \

--group cpf-worker \

--key cpf-key \

--instance-type m1.large \

$EMI

1. <http://www.ubuntu.com/download/server/download> [↑](#footnote-ref-1)