Troubleshooting 4.1.2 Guide

2015-08-04 Eucalyptus Systems

Contents

Troubleshooting Overview

This section covers common management or configuration problems and solutions for fixing these problems.

To troubleshoot Eucalyptus, you must know which machines each Eucalyptus component is installed on. You must have root access to each machine hosting the components. You must understand the network configuration connecting the Eucalyptus components.

Document version: Build 2704 (2015-08-03 20:58:57)

Eucalyptus Log Files

Usually when an issue arises in Eucalyptus, you can find information that points to the nature of the problem either in the Eucalyptus log files or in the system log files. This topic details log file message meanings, location, configuration, and fault log information.

Log File Location and Content

By default, the Eucalyptus log files are stored in `/var/log/eucalyptus/` on each machine that hosts a Eucalyptus component. If Eucalyptus is installed somewhere other than the filesystem root (`/`), log files are stored in `$EUCALYPTUS/var/log/eucalyptus/`.

CLC, Walrus, SC, and UFS Log Files

Cloud controller (CLC), Walrus, Storage controller (SC), and User-Facing Services (UFS) log files are as follows:

Log File	Description
`cloud-cluster.log`	This log contains information about your clusters as the CLC sees things: current status, current capacity, and any problems. These logs can help you detect if there is a capacity or communication issue associated with your clusters.
`cloud-fault.log`	This file is reserved for issues with known error codes and known resolutions.
`cloud-output.log`	This file contains all info-level logs for the Java component itself. If there are multiple Java components on a single host (for example, CLC and Walrus), the info-level logs for all of the components will go here.
`cloud-debug.log`	This file contains all messages generated from debug-level logging.
`cloud-error.log`	This file contains is enabled by default alongside info. Along with `cloud-output.log`, the `cloud-error.log` is one of the first places you should look.
`cloud-exhaust.log`	This file is full of errors and warnings.
`cloud-extreme.log`	This is legitimately a setting for developers only, because production usage would fill up the hard drive with log files very quickly.
`startup.log`	STDOUT and STDERR are redirected to this log file for system startup. This file contains system JVM startup output including system bootstrap information, component bootstrap configuration, local service discovery, and network interfaces.
`upgrade.log`	This file records the output from upgrade process. Same as seen on the command line when upgrading.

Log File	Description
`cloud-requests.log`	This file is only located on the UFS component and logs the system requests made to the different services: ec2, autoscaling, cloudformation, etc.
`jetty-request-[date].log`	This file is only located on the CLC component and tracks access information (credentials) associated with users and their accounts.
`/var/log/messages`	This file contains any general host problems. For example: networking issues, disk space, hardware failures.

CC Log Files

For the Cluster controller (CC), the general types of errors to look for are errors with node orchestration, communication issues between CC and NCs, and tunneling issues with multi-cluster configurations (to span security groups across AZs). Log files are as follows:

Log File	Description
`cc.log`	This is the canonical place for CC error messages, and is the common log for all info and warning messages as well. In the C code, we mostly follow syslog practices. You can change the CC logging level on the fly with a restart. Log messages tend to be readable and informative.
`cc-fault.log`	This file contains issues with known error codes and known resolutions.
`axis2c.log`	This file is for the web services stack on the CC. Web services calls get translated here. It is not too "user-friendly" for parsing, but you normally do not need to go through it. Most issues appearing in this file have to do with credential errors or OpenSSL issues.
`httpd-cc_error_log`	This file generally contains information about events around the web services stack. For example: component start or stop, IP tables, or networking errors.
`/var/log/messages`	This file contains DHCP bridge issues and general network-related issues.

NC Log Files

For the Node controller (NC) log files generally detail instance tasks, instance lifecycle, and instance operations. NC log files are as follows:

Log File	Description
`nc.log`	This file is the common file for all info, error, and warning messages. It is a good starting place for all issues on an NC.
`axis2c.log`	This file is for the web services stack on the NC. Web services calls get translated here. It is not too "user-friendly" for parsing, but you normally do not need to go through it. Most issues appearing in this file have to do with credential errors or OpenSSL issues.

Log File	Description
`httpd-nc_error_log`	This file generally contains information about events around the web services stack. For example: component start or stop, IP tables, or networking errors.
`nc-fault.log`	This file contains issues that have known error codes and known resolutions.
`euca_test_nc.log`	When the NC starts up, it runs through a self-test. This file contains log message from that process. It is useful to review if you have a fresh NC and you're seeing issues.
`/var/log/messages`	This file contains high-level KVM, libvirt, and general hypervisor issues. It also contains iSCSI/EBS issues (usually connecting instances to storage), and networking issues in certain Eucalyptus networking modes (most useful in Edge, Managed, and Managed-NoVLAN).
`/var/log/libvirt/*`	Various low-level libvirt errors and low-level QEMU and KVM errors.

API Services Running As Instances

The following log files are relevant to cloud administrators who have access to instances directly:

Log File	Description
`worker.log`	Logs on the Image Worker image are stored in `/var/log/eucalyptus-imaging-worker.`
`servo.log`	Logs on a given Elastic Load Balancer (ELB) are stored in `/var/log/load-balancer-servo.`

System Log Files

You might also find helpful information about the nature of an issue in the system logs. In particular, the following logs may be relevant:

- `/var/log/messages`
- `/var/log/libvirt/`

Log File Levels

All messages that show up as `FAULT`, `FATAL`, or `ERROR` require an action by the administrator.

FAULT Anything identified in a fault log has an identifiable cause and an identifiable solution, but one that Eucalyptus cannot fix by itself. The administrator needs to act immediately.

FATAL Any condition that indicates that Eucalyptus has failed (for example, OOM).

ERROR Any condition for which an operator must take immediate action to identify and/or remedy.

WARN An indication that the system could not perform a task, but does not necessarily indicate that immediate action by the operator is required. For example, when a user tries to allocate a bucket when their quota is exceeded, or when an action is being retried unsuccessfully, with the final timeout possibly giving ERROR instead of WARN.

INFO This is the default recommended log level. Any log message that contains useful information to see "what is happening" and generally indicates healthy activity. For example, anytime a user runs `euca-describe-instances` (that is, User A does Action B at Time T with

	Correlation-id I, and it succeeded or failed--grep for Correlation-id I in various logs for more info). This is useful for troubleshooting, but not necessarily for monitoring.
DEBUG	Detailed debug data is only available when the cloud is set to debug mode, and unlike INFO, it does not seek to aggregate messages. Instead, it writes them out the second they're generated. For example, entering or leaving a particular function. These messages are generally incomprehensible to administrators, but are useful to Eucalyptus engineers for debugging.
TRACE (backend) or EXTREME (frontend)	These are useful for engineers only in development. Unlike DEBUG, installations running in TRACE or EXTREME mode can actually degrade the system as a result of the monitoring activity, and could actually create failures. We recommend that you don't

Log File Configuration

For the CC and the NC, you can configure the log level using the `LOGLEVEL` parameter in `eucalyptus.conf`. This parameter will be picked up dynamically when the value is changed in the config file, without requiring a restart of the component.

For all other components, you can configure the log level by passing an appropriate `--log-level` argument in the init script. You can also dynamically change the level using `euca-modify-property` and set an appropriate value for `cloud.euca_log_level`. This takes precedence over the value specified in the init script.

Valid log levels are as follows, from most to least verbose:

- ALL
- EXTREME
- TRACE
- DEBUG
- INFO
- WARN
- ERROR
- FATAL
- OFF

If no value is specified, the default `INFO` is used.

Log File Format

Eucalyptus logs now have a standard format, which varies slightly per log level.

For log levels FATAL, ERROR, WARN and INFO:

```
YYYY-MM-DD HH:MM:SS LEVEL | message
```

For log levels DEBUG and TRACE:

```
YYYY-MM-DD HH:MM:SS LEVEL PROCESS:THREAD loggingMethodOrClass     | message
```

For log level EXTREME and ALL:

```
YYYY-MM-DD HH:MM:SS LEVEL PROCESS:THREAD loggingMethodOrClass
FILENAME:LineNumber | message
```

Fault Logs

Eucalyptus includes fault logs for easy identification of conditions outside of Eucalyptus's control that may cause it to fail. These messages are logged per component, and each fault is logged only once per component, in `/var/log/eucalyptus/[component]-fault.log`. The messages include a suggested resolution, and can be customized. Where they have been translated, Eucalyptus will use the system-configured `LOCALE` variable to serve appropriate messages.

Fault messages are based on XML-formatted templates, stored in a per-locale directory structure, with one file per fault message, and one file storing common strings. Default templates are shipped with Eucalyptus. These are stored in `/usr/share/eucalyptus/faults/` as follows:

```
/usr/share/eucalyptus/faults/en_US/0001.xml
...
/usr/share/eucalyptus/faults/en_US/1234.xml
/usr/share/eucalyptus/faults/en_US/common.xml
```

Using Localized Fault Logs

Localized messages are located in a per-locale directory under `/usr/share/eucalyptus/faults/`. If localized messages are available matching the system LOCALE, Eucalyptus will use those messages. If no LOCALE is set, Eucalyptus defaults to `en_US`.

Set the system LOCALE in `/etc/sysconfig/i18n` as follows:

```
LOCALE=ru_RU
```

Using Customized Fault Logs

To use your own customized messages, copy the message files to the appropriate directory under `/etc/eucalyptus/faults/` and edit them. Do not change the filenames. To test the fault template, run `euca-generate-fault`, providing the component name, fault ID, and any relevant parameters along with their values.

```
euca-generate-fault -c component_name fault_id [param] [value]
```

For example

```
euca-generate-fault -c nc 1008 daemon ntpd
```

The test fault should be logged in the appropriate component fault log (in this case, `/var/log/eucalyptus/nc-fault.log`

Eucalyptus uses customized messages where they are available, preferring a non-localized custom message over a localized default message. Localized messages should be in a per-locale directory under `/etc/eucalyptus/faults/`, with a directory name that matches the system LOCALE. If no LOCALE is set, Eucalyptus defaults to `en_US`.

Network Information

When you have to troubleshoot, it's important to understand the elements of the network on your system.

Here are some ideas for finding out information about your network:

- It is also important to understand the elements of the network on your system. For example, you might want to list bridges to see which devices are enslaved by the bridge. To do this, use the `brctl` command.
- You might also want to list network devices and evaluate existing configurations. To do this, use these commands: `ip`, `ifconfig`, and `route`.
- If you are running Eucalyptus in Managed networking mode, you can also use `vconfig` to evaluate VLAN configuration.
- You can get further information if you use the `euca-describe` commands with the `verbose` options. For example, `euca-describe-instances verbose` returns all instances running by all users on the system. Other describe commands are:
  - `euca-describe-volumes verbose`
  - `euca-describe-snapshots verbose`
  - `euca-describe-groups verbose`
  - `euca-describe-keypairs verbose`

Walrus and Storage

This topic contains information about Walrus-related problems and solutions.

Walrus decryption failed.

On Ubuntu 10.04 LTS, kernel version 2.6.32-31 includes a bug that prevents Walrus from decrypting images. This can be determined from the following line in cloud-output.log

```
javax.crypto.
BadPaddingException: pad block corrupted
```

If you are running this kernel:

1. Update to kernel version 2.6.32-33 or higher.
2. De-register the failed image (`euca-deregister`).
3. Re-register the bundle that you uploaded (`euca-register <bucket>/<manifest>`).

Walrus physical disk is not large enough.

1. Stop the CLC.
2. Add a disk.
3. Migrate your data.

Make sure you use LVM with your new disk drive(s).

Access and Identities

This topic contains information about access-related problems and solutions.

Need to verify an existing LIC file.

1. Enter the following command:

```
/usr/sbin/euca-describe-properties | grep ldap
```

The output from the example above shows the name of the LIC file and status of the synchronization (set to false).

```
PROPERTY  authentication.ldap_integration_configuration
{ 'sync': { 'enable':'false' } }
```

Windows Images

This topic contains information to help you troubleshoot your Windows images.

Properties

A typical size of Windows images is large and Eucalyptus has a set of properties that limit the size of various storage components. The first step in troubleshooting is to make sure that the values are large enough to store your Windows images. You can modify a property using

```
/usb/sbin/euca-modify-property -p <property>=<value>
```

The properties that might affect registering Windows images are:

- `walrus.storagemaxbucketsizeinmb`: max bucket size enforced by Walrus; should be larger than a Windows image
- `walrus.storagemaxcachesizeinmb`: total size of all images that is cached in Walrus; should be larger than the sum of all images (Windows/Linux) in Walrus
- walrus.storagemaxtotalsnapshotsizeingb: if a Windows image is a type of EBS-backed EMI, this should be large enough to store all EBS backed images
- `{PARTITION}.storage.maxvolumesizeingb`: if a Windows image is a type of EBS-backed EMI, this should be large enough to store the image

In addition, during the `euca-run-instances`, the CLC may time-out an instance while a large windows image (images in both Walrus and EBS) is being launched. We recommend that you raise the values of the following properties.

- `cloud.vmstate.instance_timeout`: maximum wait time, in minutes, before the instance becomes running. Am instance cannot stay in pending longer than this. Default: 60
- `cloud.vmstate.ebs_volume_creation_timeout`: maximum wait time, in minutes, before a volume backing a boot from EBS image is created. Default: 30
- `cloud.addresses.maxkillorphans`: The public IP assigned to an instance will be expired after the time limit. The exact time-out is {maxkillorphans*5} seconds (by default it's 50 seconds). If the volume backing an EBS image is not created in time, the public IP will be released from the instance.

Image Preparation

`euca-bundle-image` hangs	The time to bundle an image is proportional to the image size. Because the typical size of Windows image is big, give enough time until bundling is complete. As a rule of thumb, it may take up to 20 min. for bundling a 10 GB Windows image.
`euca-upload-bundle` fails	Make sure ‘walrus.storagemaxbucketsizeinmb’ is large enough. If not, ask your administrator.

Instance Launch and Login

Instance stays in pending	Typically, it takes longer to launch Windows images than Linux images as the delay is proportional to the image size. This can be especially long when the image is seeded on NCs the first time (images are cached in NCs and run within few seconds thereafter). As a rule of thumb, 10 GB Windows images may take up to 10 minutes to become ‘running’ when it is not cached in NCs.
Instance stay in pending and goes to shutdown	An instance may time-out if the Windows image is too big. Review and adjust the relevant properties.

Instance is running, but not accessible using Remote Desktop.	after instances become running, you should wait until Windows is fully booted. If the image is sysprepped, the booting time may take up to 10 min. Also you should make sure the followings are cleared: • The port 3389 is authorized in the security group • If the instance is attached to your active directory domain, the domain GPO shouldn't block the RDP port (3389) • The username should be authorized to log-in using Remote Desktop (refer to User guide: Windows integration service)
Finding the login username and password	Use `Administrator` and the password retrieved by `euca-get-password`. If the instance is attached to a domain, you may use your domain username and password (make sure the username is prepended with domain name, such as `YOUR_DOMAIN\Alice`).
Can't retrieve windows password using `euca-get-password`	Make sure the platform field of your windows EMI is set to 'windows', not 'linux' (use `euca-describe-images`). If not, the most likely reason is that the image name does not begin with 'windows'. You should bundle/upload/register the image with a proper name.
Instance is not attached to an Active Directory domain	• Make sure the parameters set in Windows integration service are correct. One way to verify them is to log in the instance using Administrator password and manually attach the instance to the domain (System Properties -> Computer Name) using the same parameters. • Make sure `VNET_DNS` in eucalyptus.conf is set to the domain controller (refer to User Guide: Configure Active Directory).

Disk and Volume

Ephemeral disks are not visible in the Windows	Open Disk Management console (**All Programs**->**Administrative Tools**->**Server Manager**->**Storage**) and find the uninitialized disks. You should create a partition on the disk and format it.
EBS volume is attached, but not visible in the Windows	Open Disk Management console (**All Programs**->**Administrative Tools**->**Server Manager**->**Storage**) and find the uninitialized disks. You should create a partition on the disk and format it. You don't have to repeat it when the volume is reattached later.
EBS volume is detached, but the disk drive (for example, `E:\`) is still visible in the Windows	For KVM hypervisor, you should perform "remove hardware safely" before detaching the volume.
`euca-bundle-instance` fails	Make sure the bucket specified with '-b' option doesn't already exist and the property 'walrus.storagemaxbucketsizeinmb' is large enough to store the image.

Instances

This topic contains information to help you troubleshoot your instances.

Inaccurate IP addresses display in the output of euca-describe-addresses.

This can occur if you add IPs from the wrong subnet into your public IP pool, do a restart on the CC, swap out the wrong ones for the right ones, and do another restart on the CC. To resolve this issue, run the following commands.

Note: A restart should only be performed when no instances are running, or when instance service interruption can be tolerated. A restart causes the CC to reset its networking configuration, regardless of whether or not it is in use. A restart of a CC in Managed and Managed (NoVLAN) modes that is managing active VMs can cause a temporary loss of network connectivity until the CC relearns the network topology and rebuilds the IP table entries.

```
/etc/init.d/eucalyptus-cloud stop
/etc/init.d/eucalyptus-cc stop
iptables -F
/etc/init.d/eucalyptus-cc restart
/etc/init.d/eucalyptus-cloud start
```

NC does not recalculate disk size correctly

This can occur when trying to add extra disk space for instance ephemeral storage. To resolve this, you need to delete the instance cache and restart the NC.

For example:

```
rm -rf /var/lib/eucalyptus/instances/*
service eucalyptus-nc restart
```

Elastic Load Balancing

This topic explains suggestions for problems you might have with Elastic Load Balancing (ELB).

Can't synchronize with time server

Eucalyptus sets up NTP automatically for any instance that has an internet connection to a public network. If an instance doesn't have such a connection, set the cloud property `loadbalancing.loadbalancer_vm_ntp_server` to a valid NTP server IP address. For example:

```
euca-modify-property -p
loadbalancing.loadbalancer_vm_ntp_server=169.254.169.254
PROPERTY loadbalancing.loadbalancer_vm_ntp_server
169.254.169.254 was {}
```

Need to debug an ELB instance

To debug an ELB instance, set the `loadbalancing.loadbalancer_vm_keyname` cloud property to the keypair of the instance you want to debug. For example:

```
# euca-modify-property -p
loadbalancing.loadbalancer_vm_keyname=sshlogin
PROPERTY loadbalancing.loadbalancer_vm_keyname sshlogin was
{}
```

High Availability

This topic contains information to help you troubleshoot your high availability deployment.

In the event that incorrect keys for a secondary CLC are used, Eucalyptus behaves as if that CLC no longer exists. The current primary CLC continues to operate as expected. In order to bring back the secondary CLC, perform the following tasks.

1. Stop the secondary CLC.

```
service eucalyptus-cloud stop
```

2. On the secondary CLC, delete all files from `/var/lib/eucalyptus/db`.
3. On the secondary CLC, delete all .pem and vtunpass files from `var/lib/eucalyptus/keys`.
4. Start the secondary CLC.

```
service eucalyptus-cloud start
```

5. Re-register the secondary CLC with the primary CLC.

```
/usr/sbin/euca_conf --register-cloud --partition eucalyptus
--host [Secondary_CLC_IP] --component [CLC_Name]
```

Recovering from a Failure: Walrus

Some sample scenarios in which we offer solutions.

In these examples, we will assume that Walrus `WS00` is the primary and `WS01` is the secondary Walrus server.

Software Failure Example

In this scenario, `WS01` refuses to go to `DISABLED` state. DRBD complains that it is in split brain mode. `drbdadm cstate r0` shows that DRBD is in WFConnection state.

If you are sure that data on `WS01` is out of date and can be discarded, execute the following commands to restore HA mode.

1. Shut down the eucalyptus-cloud process on WS01.
2. Ensure that the DRBD connection is down by typing "drbdadm disconnect r0" on any of the two Walrus hosts.
3. On the primary Walrus, WS00, set drbd as the primary by executing "drbdadm primary r0"
4. On the secondary Walrus, WS01, execute the following command:

```
drbdadm -- --discard-my-data connect
```

Warning: This command will discard all data on WS01 and synchronize data from WS00.

5. Monitor the state of DRBD by running:

```
watch -n 2 cat /proc/drbd
```

6. When the data on WS01 is synced, start the eucalyptus-cloud process on WS01.

Hardware Failure Example

In this example, the primary `WS00` needs to be taken out of service due to a hardware failure, such as a failed disk.

1. Shut down the eucalyptus-cloud process on WS00 if it is still running.
2. Monitor service status by running `euca-describe-services` on WS01 and ensure that WS01 has taken over as the new primary (state: ENABLED).
3. Shut down the host running WS00.
4. If the host running WS00 is to be replaced entirely or the OS reinstalled:
   - On the primary CLC, enter the following to deregister WS00:

```
euca_conf --deregister-walrusbackend --component WS00 partition <name of
partition>
      --host <WS00 host>
```

   - After Linux has been installed on the new WS00 host and it is ready for use, please reinstall the "eucalyptus-walrus" package.
   - Synchronize the DRBD configuration (/etc/drbd.conf and /etc/eucalyptus/drbd*) from the WS01 host.
   - On WS00, re-configure DRBD by following the Configure DRBD section of the Installation Guide and performing the steps that are relevant to the secondary Walrus server (WS00 is the new secondary Walrus server, in this example).
   - Re-register WS00 with a new host name if necessary. This will synchronize keys.
5. On WS00, execute the following command:

```
drbdadm -- --discard-my-data connect
```

Warning: This command will discard all data on WS00 and synchronize data from WS01.

6. Monitor the state of DRBD by entering:

   ```
   watch -n 2 cat /proc/drbd
   ```

   WS01 should be marked as the primary and WS00 is the new secondary. Wait until data is synchronized.

7. When the data on WS00 is synced from WS01, start the eucalyptus-cloud process on WS00.
8. Monitor service status by running "euca-describe-services" on the primary CLC and ensure that WS00 is DISABLED and WS01 is ENABLED.

At this point, the Walrus service is back in HA mode.

Index