# DUCC Installation and Verification
# Excerpt From Complete DUCC Documentation

Written and maintained by the Apache
UIMA™Development Community



Publication date: January 2014

# Overview

DUCC is a multi-user, multi-system distributed application. First-time installation is performed in two stages:

- Single-user installation: This provides single-user, single-system installation for testing, and verification. Simple development of small applications on small systems such as laptops or office workstations is possible after Single-user Installation.
- Multi-user installation: This provides secure multi-user capabilities and configuration for multi-system clusters.

First-time users must perform single-user installation and verification on a single system. Once this configuration is working and verified, it is straightforward to upgrade to a multi-user configuration.

**Note:** A key feature of DUCC is to run user processes in CGroups in order to guarantee each process always has the amount of RAM requested. Total RAM allocation to the managed process and its child processes that exceed requested memory will go to swap space. Without CGroups a process that exceeds its requested memory size by N% is killed (default N=5 in ducc.properties), and memory use by child processes is ignored.

DUCC is distributed as a compressed tar file. The instructions below assume installation from one of this distribution media. If building from source, the build creates this file in your svn trunk/target directory. The distribution file is in the form

```
uima-ducc-[version]-bin.tar.gz
```

where [version] is the DUCC version; for example, *uima-ducc-1.0.0-bin.tar.gz*. This document will refer to the distribution file as the "<distribution.file>".

# Software Prerequisites

Both single and multi-user configurations have the following software pre-requisites:

- A userid *ducc*, and group *ducc*. User *ducc* must the the only member of group *ducc*.
- Reasonably current Linux. DUCC has been tested on SLES 10.2, 11.1, and 11.2, and RHEL 6.3.

  *Note:* On some systems the default *user limits* for max user processes (ulimit -u) and nfiles (ulimit -n) are defined too low for DUCC. The shell login profile for user *ducc* should set the soft limit for max user processes to be the same as the hard limit (ulimit -u 'ulimit -Hu'), and the nfiles limit raised above 1024 to at least twice the number of user processes running on the cluster.
- For CGroupt support, libcgroup1-0.37+ on SLES and libcgroup-0.37+ on RHEL.
- DUCC has been tested and run on IBM and Oracle JDK 1.6 and 1.7.
- Python 2.x, where 'x' is 4 or greater. DUCC has not been tested on Python 3.x.

Multi-user installation has additional requirements:

- All systems must have a shared filesystem (such as NFS or GPFS) and common user space
- Passwordless ssh must be installed for user *ducc* on all systems.
- Root access is required to install a small setuid-root program on each system.

In order to build DUCC from source the following software is also required:

- A Subversion client, from `http://subversion.apache.org/packages.html`. The svn url is `https://svn.apache.org/repos/asf/uima/sandbox/uima-ducc/trunk`.
- Apache Maven, from `http://maven.apache.org/index.html`

The DUCC webserver server optionally supports direct "jconsole" attach to DUCC job processes. To install this, the following is required:

- Apache Ant, any reasonably current version.

To (optionally) build the documentation, the following is also required:

- Latex, including the *pdflatex* and *htlatex* packages. A good place to start if you need to install it is `https://www.tug.org/texlive/`.

More detailed one-time setup instructions for source-level builds via subversion can be found here: `http://uima.apache.org/one-time-setup.html#svn-setup`

## Building from Source

To build from source, ensure you have Subversion and Maven installed. Extract the source from the SVN repository named above.

Then from your extract directory into the root directory (usually current-directory¿/trunk), and run the command

```
mvn install
```

or

```
mvn install -Pbuild-duccdocs
```

if you have LaTeX insalled and wish to do the optional build of documentation.

If this is your first Maven build it may take quite a while as Maven downloads all the open-source pre-requisites. (The pre-requisites are stored in the Maven repository, usually your $HOME/.m2).

When build is complete, a tarball is placed in your current-directory/trunk/target directory.

## Documentation

After single-user installation, the DUCC documentation is found (in both PDF and HTML format) in the directory ducc_runtime/docs. As well, the DUCC webserver contains a link to the full documentation on each major page. The API is documented only via JavaDoc, distributed in the webserver's root directory `$DUCC_HOME/webserver/root/doc/api.`

If building from source, Maven places the documentation in

- `trunk/uima-ducc-duccdocs/target/site` (main documentation), and
- `trunk/target/site/apidocs` (API Javadoc)

## Single-user Installation and Verification

Single-user installation sets up an initial, working configuration on a single system. Although any user ID can be used to run DUCC in single-user mode, it is recommended to create user "ducc" to avoid confusion. No security is established in single-user mode, and all user processes run as the DUCC user ID.

Verification submits a very simple UIMA pipeline for execution under DUCC. Once this is shown to be working, one may proceed to upgrade to full installation.

## Minimal Hardware Requirements for single-user Installation

- One Intel-based or IBM Power-based system. (More systems may be added during multi-user installation, described below.)

- 8GB of memory. 16GB or more is preferable for developing and testing applications beyond the non-trivial.
- 1GB disk space to hold the DUCC runtime, system logs, and job logs. More is usually needed for larger installations.

Please note: DUCC is intended for scaling out memory-intensive UIMA applications over computing clusters consisting of multiple nodes with large (16GB-256GB or more) memory. The minimal requirements are for initial test and evaluation purposes, but will not be sufficient to run actual workloads.

## Single-user System Installation

1. Expand the distribution file:

   ```
   tar -zxf <distribution.file>
   ```

   This creates a directory with a name of the form "apache-uima-ducc-[version]".

   This directory contains the full DUCC runtime which you may use "in place" but it is highly recommended that you move it into a standard location; for example, under ducc's HOME directory:

   ```
   mv apache-uima-ducc-1.0.0 /home/ducc/ducc_runtime
   ```

   We refer to this directory, regardless of its location, as $DUCC_HOME. For simplicity, some of the examples in this document assume it has been moved to /home/ducc/ducc_runtime.

2. Change directories into the admin sub-directory of $DUCC_HOME:

   ```
   cd $DUCC_HOME/admin
   ```

3. Run the post-installation script:

   ```
   ./ducc_post_install
   ```

   If this script fails, correct any problems it identifies and run it again.

   Note that *ducc_post_install* initializes various default parameters which may be changed later by the system administrator. Therefore it usually should be run only during this first installation step.

4. If you wish to install jconsole support from the webserver, make sure Apache Ant is installed, and run

   ```
   ./sign_jconsole_jar
   ```

   This step may be run at any time if you wish to defer it.

That's it, DUCC is installed and ready to run. (If errors were displayed during ducc_post_install they must be corrected before continuing.)

The post-installation script performs these tasks:

1. Verifies that the correct level of Java and Python are installed and available.
2. Creates a default nodelist, $DUCC_HOME/resources/ducc.nodes, containing the name of the node you are installing on.
3. Defines the "ducc head" node to be to node you are installing from.
4. Sets up the default https keystore for the webserver.
5. Installs the DUCC documentation "ducc book" into the DUCC webserver root.
6. Builds and installs the C program, "ducc_ling", into the default location.
7. Ensures that the (supplied) ActiveMQ broker is runnable.

## Initial System Verification

Here we start the basic installation, submit a simple UIMA-AS job, verify that it ran, and stop DUCC. Once this is confirmed working DUCC is ready to use in an unsecured, single-user mode on a single system.

To run the verification, issue these commands.

1. cd $DUCC_HOME/admin
2. ./check_ducc -s

   Examine the output of check_ducc. If any errors are shown, correct the errors and rerun check_ducc until there are no errors.
3. Finally, start ducc: ./start_ducc -s

Start_ducc will first perform a number of consistency checks. It then starts the ActiveMQ broker, the DUCC control processes, and a single DUCC agent on the local node. Note that "single user mode" (-s) is specified for this first start. This inhibits the checks on the permissions on ducc_ling (described below).

You will see some startup messages similar to the following:

```
ENV: Java is configured as: /share/jdk1.6/bin/java
ENV: java full version "1.6.0_14-b08"
ENV: Threading enabled: True
MEM: memory is 15 gB
ENV: system is Linux
allnodes /home/ducc/ducc_runtime/resources/ducc.nodes
Class definition file is ducc.mixed.classes
OK: Class and node definitions validated.
OK: Class configuration checked
Starting broker on ducchead.biz.org
Waiting for broker ..... 0
Waiting for broker ..... 1
ActiveMQ broker is found on configured host and port: ducchead.biz.org:61618
Starting 1 agents
********** Starting agents from file /home/ducc/ducc_runtime/resources/ducc.nodes
Starting warm
Waiting for Completion
ducchead.biz.org Starting rm
     PID 14198
ducchead.biz.org Starting pm
     PID 14223
ducchead.biz.org Starting sm
     PID 14248
ducchead.biz.org Starting or
     PID 14275
ducchead.biz.org Starting ws
     PID 14300
ducchead.biz.org
    ducc_ling OK
    DUCC Agent started PID 14325
All threads returned
```

Now open a browser and go to the DUCC webserver's url, http://<hostname>:42133 where <hostname> is the name of the host where DUCC is started. Navigate to the Reservations page via the links in the upper-left corner. You should see the DUCC JobDriver reservation in state WaitingForResources. In a few minutes this should change to Assigned. (This usually takes 3-4 minutes in the default configuration.) Now jobs can be submitted.

The hostname and port are configurable by the DUCC administrator in ducc.properties

To submit a job,

1. $DUCC_HOME/bin/ducc_submit –specification $DUCC_HOME/examples/simple/1.job

Open the browser in the DUCC jobs page. You should see the job progress through a series of transitions: Waiting For Driver, Waiting For Services, Waiting For Resources, Initializing, and finally, Running. You'll see the number of work items submitted (15) and the number of work items completed grow from 0 to 15. Finally, the job will move into Completing and then Completed..

Since this example does not specify a log directory DUCC will create a log directory in your HOME directory under

`$HOME/ducc/logs/job-id`

In this directory, you will find a log for the sample job's JobDriver (JD), JobProcess (JP), and a number of other files relating to the job.

This is a good time to explore the DUCC web pages. Notice that the job id is a link to a set of pages with details about the execution of the job.

Notice also, in the upper-right corner is a link to the full DUCC documentation, the "DuccBook".

Finally, stop DUCC:

1. cd $DUCC_HOME/admin
2. ./stop_ducc -a

Once the system is verified and the sample job completes correctly, proceed to Multi-User Installation and Verification to set up multiple-user support and optionally, multi-node operation.

## Logs

The DUCC system logs are written to the directory

`$DUCC_HOME/logs`

In that directory are found logs for each of the DUCC components plus one for each node DUCC is installed on.

DUCC job/user logs are written by default to the user's HOME directory under

`$HOME/ducc/logs/<jobid>`

## Multi-User Installation and Verification

Multi-user installation consists of these steps over and above single-user installation:

1. Install and configure the setuid-root program, ducc_ling. This small program allows DUCC jobs to be run as the submitting user rather than user ducc.
2. Optionally install and configure CGroups.
3. Optionally update the configuration to include additional nodes.

Multi-user installation has these pre-requisites (DUCC will not work on multiple nodes unless these steps are taken):

- All systems in the DUCC cluster must have a shared filesystem and shared user space (user directories are shared over NFS or an equivalent networked file system, across the systems, and user ids and credentials are the same).
- Passwordless ssh must be installed for user ducc on all systems. Users do NOT require ssh access to the DUCC nodes.
- Root access is (briefly) required to install a small setuid-root program on each system.

## Ducc_ling Installation

Before proceeding with this step, please note:

- This step is required ONLY to install multi-user capabilities.
- The sequence operations consisting of *chown* and *chmod* MUST be performed in the exact order given below. If the *chmod* operation is performed before the *chown* operation, Linux will regress the permissions granted by *chmod* and ducc_ling will be incorrectly installed.

ducc_ling is a setuid-root program whose function is to execute user tasks under the identity of the user. This must be installed correctly; incorrect installation can prevent jobs from running as their submitters, and in the worse case, can introduce security problems into the system.

ducc_ling must be installed on LOCAL disk on every system in the DUCC cluster, to avoid shared-filesystem access to it. The path to ducc_ling must be the same on each system. For example, one could install it to /local/ducc/bin on local disk on every system.

To install ducc_ling (these instructions assume it is installed into /local/ducc/bin): As ducc, ensure ducc_ling is built correctly for your architecture:

1. cd $DUCC_HOME/duccling/src
2. make clean all

Now, as root, move ducc_ling to a secure location and grant authorization to run tasks under different users' identities:

1. mkdir /local/ducc
2. mkdir /local/ducc/bin
3. chown ducc.ducc /local/ducc
4. chown ducc.ducc /local/ducc/bin
5. chmod 700 /local/ducc
6. chmod 700 /local/ducc/bin
7. cp $DUCC_HOME/duccling/src/ducc_ling /local/ducc/bin
8. chown root.ducc /local/ducc/bin/ducc_ling
9. chmod 4750 /local/ducc/bin/ducc_ling

Finally, update the configuration to use this ducc_ling instead of the default ducc_ling:

1. Edit $DUCC_HOME/resources/ducc.properties and change this line:

   ```
   ducc.agent.launcher.ducc_spawn_path=${DUCC_HOME}/admin/ducc_ling
   ```

   to this line (Using the actual location of the updated ducc_ling, if different from /local/ducc/bin):

   ```
   ducc.agent.launcher.ducc_spawn_path=/local/ducc/bin/ducc_ling
   ```

What these steps do:

1. The first two step compile ducc_ling for your current machine architecture and operating system level.
2. The next two steps (mkdir) create directory /local/ducc/bin
3. The next two steps (chown) set ownership of /local/ducc and /local/ducc/bin to user ducc, group ducc
4. The next two steps (chmod) set permissions for /local/ducc and /local/ducc/bin so only user ducc may access the contents of these directories
5. The copy stop copies the ducc_ling program created in initial installation into /local/ducc/bin
6. The next step (chown) sets ownership of /local/ducc/bin/ducc_ling to root, and group ownership to ducc.

7. The next step (chmod) stablishes the *setuid* bit, which allows user ducc to execute ducc_ling with root privileges.
8. Finally, ducc.properties is updated to point to the new, privileged ducc_ling.

If these steps are correctly performed, ONLY user *ducc* may use the ducc_ling program in a privileged way. Ducc_ling contains checks to prevent even user *root* from using it for privileged operations.

Ducc_ling contains the following functions, which the security-conscious may verify by examining the source in $DUCC_HOME/duccling. All sensitive operations are performed only AFTER switching userids, to prevent unauthorized root access to the system.

- Changes it's real and effective userid to that of the user invoking the job.
- Optionally redirects its stdout and stderr to the DUCC log for the current job.
- Optionally redirects its stdio to a port set by the CLI, when a job is submitted.
- "Nice"s itself to a "worse" priority than the default, to reduce the chances that a runaway DUCC job could monopolize a system.
- Optionally sets user limits.
- Prints the effective limits for a job to both the user's log, and the DUCC agent's log.
- Changes to the user's working directory, as specified by the job.
- Optionally establishes LD_LIBRARY_PATH for the job from the environment variable

  `DUCC_LD_LIBRARY_PATH`

  if set in the DUCC job specification. (Secure Linux systems will prevent LD_LIBRARY_PATH from being set by a program with root authority, so this is done AFTER changing userids).

# CGroups Installation and Configuration

The steps in this task must be done as user root and user ducc.

To install and configure CGroups for DUCC:

1. Install the appropriate libcgroup package at level 0.37 or above.
2. Configure /etc/cgconfig.conf as follows:

```
# Mount cgroups
mount {
   memory = /cgroup;
}
# Define cgroup ducc and setup permissions
group ducc {
 perm {
     task {
        uid = ducc;
     }
     admin {
        uid = ducc;
     }
 }
 memory {}
}
```

3. Start the cgconfig service:

```
service cgconfig start
```

4. Verify cgconfig service is running by the existence of directory:

   ```
   /cgroups/ducc
   ```

5. Some versions of libcgroup are buggy. As user ducc, verify that the following command executes with no error:

   ```
   cgcreate -t ducc -a ducc -g memory:ducc/test-cgroups
   ```

6. Configure the cgconfig service to start on reboot:

   ```
   chkconfig cgconfig on
   ```

7. Update ducc.properties to enable CGroups. Note that if CGroups is not installed, the DUCC Agent will detect this and not attempt to use the feature. It is completely safe to enable CGroups in *ducc.properties* on machines where it is not installed.

## Set up the full nodelists

To add additional nodes to the ducc cluster, DUCC needs to know what nodes to start its Agent processes on. These nodes are listed in the file

`$DUCC_HOME/resources/ducc.nodes.`

During initial installation, this file was initialized with the node DUCC is installed on. Additional nodes may be added to the file using a text editor to increase the size of the DUCC cluster.

## Full DUCC Verification

This is identical to initial verification, with the one difference that the job "1.job" should be submitted as any user other than ducc. Watch the webserver and check that the job executes under the correct identity. Once this completes, DUCC is installed and verified.

## Enable DUCC webserver login

This step is optional. As shipped, the webserver is disabled for logins. This can be seen by hovering over the Login text located in the upper right of most webserver pages:

`System is configured to disallow logins`

To enable logins, a Java-based authenticator must be plugged-in and the login feature must be enabled in the ducc.properties file by the DUCC administrator. Also, ducc_ling should be properly deployed (see Ducc_ling Installation section above).

A beta version of a Linux-based authentication plug-in is shipped with DUCC. It can be found in the source tree:

`org.apache.uima.ducc.ws.authentication.LinuxAuthenticationManager`

The Linux-based authentication plug-in will attempt to validate webserver login requests by appealing to the host OS. The user who wishes to login provides a userid and password to the webserver via https, which in-turn are handed-off to the OS for a success/failure reply.

To have the webserver employ the beta Linux-based authentication plug-in, the DUCC administrator should perform the following as user ducc:

```
1. edit ducc.properties
2. locate: ducc.ws.login.enabled = false
3. modify: ducc.ws.login.enabled = true
4. save
```

Note: The beta Linux-based authentication plug-in has limited testing. In particular, it was tested using:

```
Red Hat Enterprise Linux Workstation release 6.4 (Santiago)
```

Alternatively, you can provide your own authentication plug-in. To do so:

1. author a Java class that implements
   org.apache.uima.ducc.common.authentication.IAuthenticationManager
2. create a jar file comprising your authentication class
3. put the jar file in a location accessible by the DUCC webserver, such as
   $DUCC_HOME/lib/authentication
4. put any authentication dependency jar files there as well
5. edit ducc.properties
6. add the following:
   ducc.local.jars = authentication/*
   ducc.authentication.implementer=<your.authenticator.class.Name>
7. locate: ducc.ws.login.enabled = false
8. modify: ducc.ws.login.enabled = true
9. save