

# Installation on Linux using scripts

## On this page

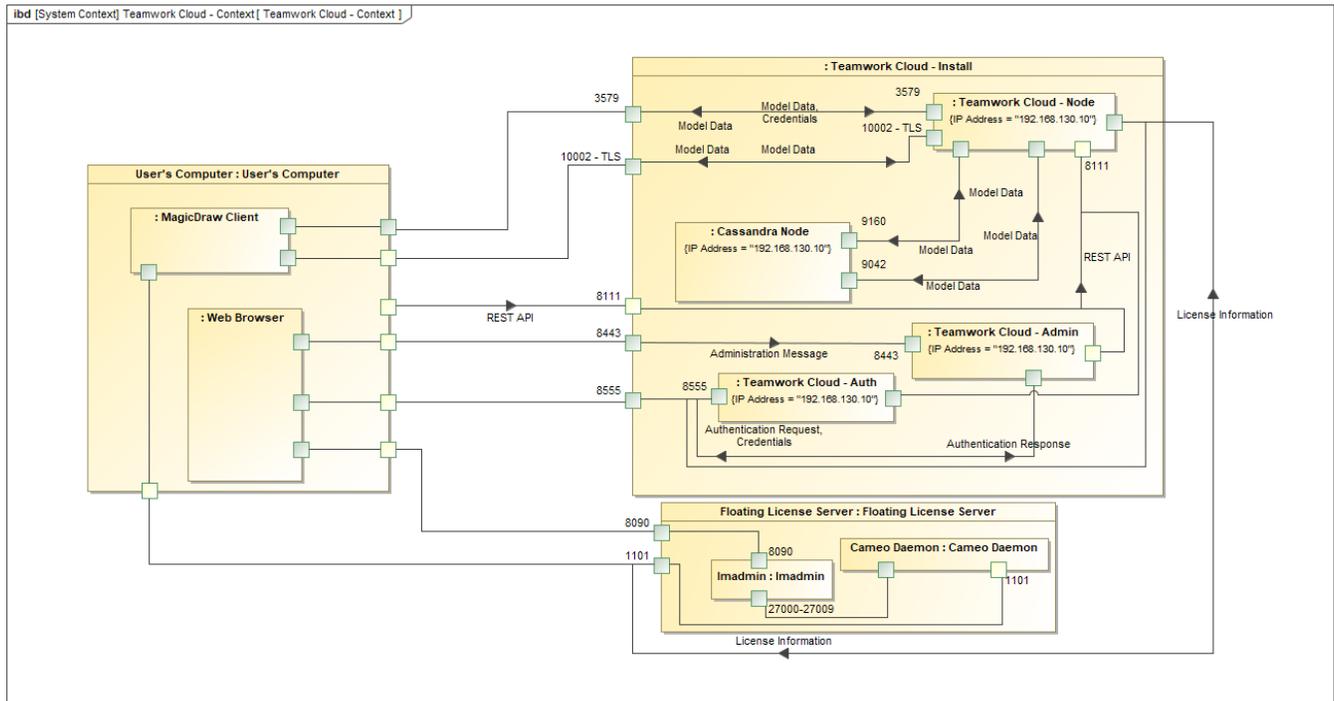
- [Recommended system requirements](#)
- [Minimum server system requirements](#)
- [Preparing the operating system](#)
- [Installing OpenJDK 8 \(for Cassandra\)](#)
- [Use this only if you need to install Oracle Java 8u202 \(used with Cassandra\) instead of OpenJDK](#)
- [Installing the FlexNet server \(Imadmin\)](#)
- [Installing Apache Cassandra 3.11.x](#)
- [Configuring cassandra.yaml](#)
- [Tuning Linux for Cassandra Performance](#)
- [Installing OpenJDK \(v11.0.12 was tested and recommended to use with Magic Collaboration Studio\)](#)
- [Installing Magic Collaboration Studio](#)
- [Post-Install Configuration](#)
- [Additional information that may affect installations in restricted environments](#)
- [Files installed on system locations](#)
- [Frequently Asked Questions](#)

## Scripts

The following are the authserver and installation script files used in this example:

- [install\\_flex\\_centos7.sh](#)
- [install\\_java8\\_u202.sh](#)
- [install\\_cassandra3\\_11\\_centos7.sh](#)
- [install\\_mcs2021xRefresh1\\_centos7.sh](#)
- [fixcassandraservice.sh](#)
- [Backup and restore data procedures and scripts](#)

This page shows how to install and configure Magic Collaboration Studio on Centos 7.x, deployed on a single server. It also provides the configuration for installing the Magic Collaboration Studio node as well as the underlying Cassandra node on the same server.



Magic Collaboration Studio installation and configuration on Centos 7.x on a single server.

## Recommended system requirements

System requirements are dictated by the intended deployment, taking into account the overall load that the environment will experience:

- Number of concurrent users
- Level of activity (commits) per day
- Overall number and size of the projects stored in Magic Collaboration Studio.

From a hardware perspective, the database (Cassandra) can be located on the same server as Magic Collaboration Studio, or separately on its own server. Storage requirements apply only to the node where the database is located.

Ideally, the node hosting Cassandra should be a physical node, since virtualization will introduce performance degradation. **Nodes running Cassandra should have DAS SSD drives (direct-attached)**. The best performance will be achieved using NVMe drives. Presently, there are hardware limitations on the size of the NVMe drives as well as the number of NVMe drives that can be installed on a single machine. Therefore, if the expected number and size of projects are significant, SAS SSD backed by a high-speed caching controller may be a more suitable choice. For ease of maintenance and reduction of risk, we recommend that the volumes reside on RAID-1 or RAID-10. If RAID is not used, the failure of a drive will result in a downed node, impacting the enterprise. By opting to deploy on RAID volumes, a drive failure will not affect the application and will allow the replacement of a drive with zero downtime.

Nodes hosting Magic Collaboration Studio nodes can be virtualized without any issues, provided the host is not oversubscribed on its resources.

#### Nodes containing both Magic Collaboration Studio and Cassandra

- 96 -128 GB ECC RAM
- >=16 processor threads (such as E5-1660)
- >1TB SSD DAS storage

#### Nodes containing only Cassandra

- 48 - 64 GB ECC RAM
- >=8 processor threads (such as E5-1620)
- >1TB SSD DAS storage

#### Nodes containing only Magic Collaboration Studio

- 48 - 64 GB ECC RAM
- >=8 processor threads (such as E5-1620)
- >250GB storage



#### **Multi-Node Clusters**

The recommended minimum sizing stated above applies to each node in a multi-node cluster.



#### **SAN Storage**

**SAN Storage should not be used on Cassandra nodes for data or commitlog volumes. This will result in severe performance degradation.**

**There is absolutely no amount of SAN tuning and OS tuning that will mitigate this.**

## Minimum server system requirements

- 8 Processor Cores - i.e. Quad-Core Hyper-threaded CPU (such as Intel E3-1230 or faster).
- 32 GB RAM (Motherboard with an ECC RAM is always preferred on any critical database server).
- Linux (RedHat/CentOS 7), 64 bit.

Please read the article for additional server recommendations for capacity and performance in the following link:

<https://docs.datastax.com/en/dse-planning/doc/planning/planningHardware.html>

If you use SATA drives and not SSDs, we recommend using a caching controller with BBU, configured for write back. In this configuration (single node Cassandra), we recommend using RAID - the aforementioned link refers to multi-node Cassandra deployments where native Cassandra replication is in place, which is not the case in this single node instance.

In order to install a full working environment, you would need:

- Open JDK 1.8.0\_x or Oracle Java (Java Hotspot) 1.8.0\_202 (for Cassandra)
- Open JDK 11.0.12 (for Magic Collaboration Studio)
- A FlexNet License Server
- Cassandra 3.11.x
- Magic Collaboration Studio

## Preparing the operating system

### Partitioning the drives

Prior to installing Cassandra, it is important to understand how Cassandra utilizes disk space in order to properly configure the host server.

Disk space depends on usage, so it's important to understand the mechanism. The database writes data to disk when appending data to the commit log for durability and when flushing memtables to SSTable data files for persistent storage. The commit log has a different access pattern (read/write ratio) than the pattern for accessing data from SSTables. This is more important for spinning disks than for SSDs.

SSTables are periodically compacted. Compaction improves performance by merging and rewriting data as well as discarding old data. However, depending on the type of compaction and size of the compactions, during compaction disk utilization and data directory volume temporarily increases. For this reason, be sure to leave an adequate amount of free disk space available on a node.

Cassandra's data and commit logs should not, under any circumstances, be placed on the drive where the operating system is installed. Ideally, a server would have 3-4 drives or partitions. The root partition, /, the OS partition, can be used as the target for the application. A /data partition should have adequate amounts of storage to accommodate your data. A /logs partition would hold your commit logs (and unless SSD, should be on a different physical disk than the data partition), and a /backup partition would be allocated for backups.

Please refer to <http://cassandra.apache.org/doc/latest/operating/hardware.html> for explanations on hardware selection.

In order to achieve adequate performance, separate partitions must be created, ideally on separate drives, to avoid i/o contention. We recommend 3 separate block devices (disks). The first block device will contain the operating system as well as a mount for the programs (*/opt/local*). The second block device (preferably SSD) will contain a mount point at /data - this is the device that must have high storage capacity for all of the data. The third block device will contain a mount point at /logs - this device should preferably be SSD but does not need to be of high capacity, since it will only store the commit logs, which are by default limited to 8GB (if using SSD, this can be a partition on the same block device as the data partition). All partitions should be formatted using the XFS file system, and there must not be a swap partition. The /backup partition can be a mount on a shared storage device, and should not be on the same physical drive as the /data partition.

The following is an example of the contents of */etc/fstab* after partitioning, where the partitions were created using LVM (without a mount for the /backup partition).

```
fstab

#
# /etc/fstab
# Created by anaconda on Tue May  2 16:31:05 2017
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/cl_twccentos7-root / xfs defaults 0 0
/dev/mapper/cl_twccentos7-data /data xfs defaults 0 0
/dev/mapper/cl_twccentos7-logs /logs xfs defaults 0 0
/dev/mapper/cl_twccentos7-opt_local /opt/local xfs defaults 0 0
```

- Disk 1 will contain the following partitions: */opt/local* (40GB) and / (rest of the drive capacity).
- Disk 2 (the disk with the highest capacity) will contain the */data* partition - as a minimum, we recommend 250GB. Due to the way compactions are handled by Cassandra, in a worst-case scenario, up to 50% of headroom may be needed.
- Disk 3 will contain the */logs* partition (at least 10 GB).
- You should also create an additional mount for backups. Unlike the data and commit log partitions, which should be on SSD storage, this mount can be of any type (including centralized storage such as a SAN or NAS). It should have at least the same capacity as the /data partition.

The aforementioned partitioning scheme is an example. Internal security protocols in your organization may dictate that other directories not be located in the main partition. During the installation, all applications will be installed in */opt/local*. Cassandra will install by default in */var/lib*. Application logs will be written to */home/twcloud*.

## Installing OpenJDK 8 (for Cassandra)

```
Installing OpenJDK 8 (for Cassandra)

yum -y install java-1.8.0-openjdk
```

### Use this only if you need to install Oracle Java 8u202 (used with Cassandra) instead of OpenJDK

From the [Java version list](#), please check that the recommended Oracle JVM version is compatible with the Magic Collaboration Studio version you are using. In order to consolidate all of the installed applications in a single location, we will be installing them under */opt/local/java*. To facilitate deployment, you may deploy using the associated script (**install\_java\_202.sh**). Oracle no longer allows direct download of their JDK, so it must be downloaded offline and placed in the same location as the **install** scripts. The installation script extracts it into the proper location, invokes the alternative command to point the system to this instance (you may need to select it when prompted), and creates entries in */etc/environment*. Upon completing the installation, issue the following command:

```
java -version
```

You should receive output such as the following:

```
java version "1.8.0_202"
Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.202-b08, mixed mode)
```

If properly installed, you will see Java identified as Java HotSpot(TM)

## install\_java\_202.sh

```
#!/bin/bash
echo "=====
echo "Installing Oracle Java 1.8.0_202"
echo "=====
echo ""
echo " Oracle Java can no longer be downloaded directly due to new authentication requirements"
echo ""
echo " After manually downloading jdk-8u202-linux-x64.tar.gz, copy it to this directory"
echo ""
echo " Latest version is available from https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html"
echo ""
echo " Archive downloads available from https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html"
echo ""
read -p -"Press any key to continue, Ctl-C to exit ...: " -nl -s
echo ""
echo "=====
sudo mkdir -p /opt/local/java
sudo tar xzf jdk-8u202-linux-x64.tar.gz -C /opt/local/java
cd /opt/local/java/jdk1.8.0_202/
sudo alternatives --install /usr/bin/java java /opt/local/java/jdk1.8.0_202/bin/java 2
sudo alternatives --config java
sudo alternatives --install /usr/bin/jar jar /opt/local/java/jdk1.8.0_202/bin/jar 2
sudo alternatives --install /usr/bin/javac javac /opt/local/java/jdk1.8.0_202/bin/javac 2
sudo alternatives --set jar /opt/local/java/jdk1.8.0_202/bin/jar
sudo alternatives --set javac /opt/local/java/jdk1.8.0_202/bin/javac
sudo echo 'JAVA_HOME=/opt/local/java/jdk1.8.0_202' > /etc/environment
sudo chown -R root:root /opt/local/java/jdk1.8.0_202
```

## Installing the FlexNet server (ladmin)

A FlexNet license server is required for Magic Collaboration Studio to operate. It can be installed on the same system, or on a separate machine. The automated deployment script (`install_flex_centos7.sh`) downloads all required components, deploys the server, creates the `systemctl` service entry to control it, and creates the necessary firewall rules to allow the required traffic. The firewall rules are created for both the internal and public zones, and the script may require modification depending on which zone the interface is located in. Additionally, if the firewall is not running when the installation script is executed, the rules will not be created. The script creates a user, `ladmin`, which runs the `ladmin` service. The FlexNet server requires the Redhat LSB core files as well as the `ld-linux` library in order to execute. The script is configured for Centos 7, but can be modified for a different version. In order to identify which LSB Core library is required, the following command can be issued:

```
sudo yum provides /lib/ld-1sb.so.3
```

The application should be installed in `/opt/local/FNPLicenseServerManager` (the installer's default location is `/opt/FNPLicenseServerManager` - so make sure that you change the location when prompted). All other default values presented by the installer should be accepted.

After the `ladmin` server has been installed it can be started by issuing the command:

```
sudo systemctl start ladmin
```

To check if the service is running, issue the following command:

```
sudo systemctl status ladmin
```

If the service fails to start, it is often because the built-in web server cannot resolve the hostname. To check if this is the case, issue the following commands:

```
cd /opt/local/FNPLicenseServerManager/logs
tail web.log
```

You will see output similar to the following:

```
[Tue May 02 18:43:27 2017] [alert] (EAI 2)Name or service not known:
mod_unique_id: unable to find IPv4 address of "yourhostname"
Configuration Failed
```

Where **yourhostname** is the name of the host. If this is the case, you will need to edit the `/etc/hosts` file and add an entry so the webserver can resolve the host. The line will be similar to the following:

```
192.168.130.10 yourhostname
```

## Installing Apache Cassandra 3.11.x

The deployment script for Cassandra removes Datastax Community Edition 2.2.x as well as OpsCenter and the Datastax Agent (which are not compatible with Cassandra 3.x), downloads and installs Cassandra the Cassandra tools from the Apache Software Foundation repository, and creates the necessary firewall rules to allow proper operation both for a single node or a cluster installation. To install, execute the installation script ([install\\_cassandra\\_3\\_11\\_centos7.sh](#)).

## Configuring `cassandra.yaml`

Now, proceed to edit `/etc/cassandra/default.conf/cassandra.yaml`:

```
sudo nano /etc/cassandra/default.conf/cassandra.yaml
```

 The script above makes all of the changes to the configuration files stated below. However, please verify that all of them have been made.

The first items we will be editing relate to the IP address of the Cassandra node and communication settings. In our diagram above, this IP address is **192.168.130.10**. You will need to search for 3 keys in the configuration file and modify them accordingly. The `seeds` parameter is a comma-delimited list containing all of the seeds in the Cassandra cluster. Since our cluster consists of only a single node, it contains only one entry - our IP address. The other 2 parameters contain the IP address on which Cassandra listens for connections and the IP address to broadcast to other Cassandra nodes in the cluster. The `broadcast_rpc_address` may be commented out using a `#` character. If so, remove the `#` and make sure there are no leading spaces.

Additionally, we need to set `rpc_address` to `0.0.0.0` (meaning, it will listen to rpc requests on all interfaces), and `start_rpc` to `true` (so it will process rpc requests).

```
seeds: "192.168.130.10"
listen_address: 192.168.130.10
broadcast_rpc_address: 192.168.130.10
rpc_address: 0.0.0.0
start_rpc: true
```

The next set of parameters controls thresholds to ensure that the data being sent is processed properly.

```
thrift_framed_transport_size_in_mb: 100
commitlog_segment_size_in_mb: 192
read_request_timeout_in_ms: 1800000
range_request_timeout_in_ms: 1800000
write_request_timeout_in_ms: 1800000
cas_contention_timeout_in_ms: 1000
truncate_request_timeout_in_ms: 1800000
request_timeout_in_ms: 1800000
batch_size_warn_threshold_in_kb: 3000
batch_size_fail_threshold_in_kb: 5000
```

If you have installed your commit log in its own partition, the default commit log size will be the lesser of  $\frac{1}{4}$  of the partition size of 8GB. In order to ensure that the recommended 8GB is used, you must uncomment the `commitlog_total_space_in_mb`, such that it will show as below. However, if you are uncommenting this value, please ensure that the partition has enough space to accommodate an 8GB commit log.

```
commitlog_total_space_in_mb: 8192
```

The next step is to point the data to the new locations. There are 4 entries that will be modified: `data_file_directories`, `commitlog_directory`, `hints_directory`, and `saved_caches_directory`. Search for these keys and edit them as follows:

```
data_file_directories:
- /data/data
commitlog_directory: /logs/commitlog
hints_directory: /data/hints
saved_caches_directory: /data/saved_caches
```

After you have made these changes, save the **cassandra.yaml** file. Now, start the related services, as follows:

```
systemctl start cassandra
```

Now, proceed to check if Cassandra is running. To do this, issue the following command:

```
nodetool status
```

If the service is running, you will receive output such as below:

```
Datacenter: datacenter1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address      Load          Tokens      Owns (effective)  Host ID                               Rack
UN  127.0.0.1     128.4 KB      256         100.0%           ea3f99eb-c4ad-4d13-95a1-80aec71b750f  rack1
```

If the service is fully operational, the first 2 characters on the last line will state "**UN**", indicating the node's status is **Up**, and its state is **Normal**.

## Tuning Linux for Cassandra Performance

 The installation script provided in the [Initial Installation](#) section below will tune the Cassandra performance automatically. Although, you can do it manually if you need to set other parameters after running the script.

There are multiple tunings that can be performed on Linux to improve the performance of Cassandra. The first step is to configure the TCP settings by adding the following tuning parameters to `/etc/sysctl.conf` file:

```
net.core.rmem_max=16777216
net.core.wmem_max=16777216
net.core.optmem_max=40960
net.core.default_qdisc=fq
net.core.somaxconn=4096
net.ipv4.conf.all.arp_notify = 1
net.ipv4.tcp_keepalive_time=60
net.ipv4.tcp_keepalive_probes=3
net.ipv4.tcp_keepalive_intvl=10
net.ipv4.tcp_mtu_probing=1
net.ipv4.tcp_rmem=4096 12582912 16777216
net.ipv4.tcp_wmem=4096 12582912 16777216
net.ipv4.tcp_max_syn_backlog=8096
net.ipv4.tcp_slow_start_after_idle = 0
net.ipv4.tcp_tw_reuse = 1
vm.max_map_count = 1048575
vm.swappiness = 0
vm.dirty_background_ratio=5
vm.dirty_ratio=80
vm.dirty_expire_centisecs = 12000
```

To apply the setting without requiring a reboot issue the command:

```
# sysctl -p
```

For a full list of steps to take to tune Linux, go to:

[https://docs.datastax.com/en/dse/5.1/dse-admin/datastax\\_enterprise/config/configRecommendedSettings.html](https://docs.datastax.com/en/dse/5.1/dse-admin/datastax_enterprise/config/configRecommendedSettings.html)

## Installing OpenJDK (v11.0.12 was tested and recommended to use with Magic Collaboration Studio)

### Installing OpenJDK 11 (used with Teamwork Cloud)

```
yum -y install java-11-openjdk
```

## Installing Magic Collaboration Studio

### Initial Installation

The deployment script for Magic Collaboration Studio ([install\\_mcs2021xRefresh1\\_centos7.sh](#)) creates a Magic Collaboration Studio user, under which the service will run, downloads all of the necessary files, and executes the installer.

Before running the script, download the Magic Collaboration Studio installer file, [install\\_mcs2021xRefresh1\\_centos7.sh](#), and place it in the same location as the script.

When you are installing Magic Collaboration Studio

1. Press **ENTER** until you are asked to configure the machine IP.
2. Configure the machine IP - enter the local IP address of the machine (e.g. 192.168.130.10).
3. Configure the cluster seed node IP - enter the local IP address of the machine (e.g. 192.168.130.10).
4. Configure the Magic Collaboration Studio service owner - enter **twcloud**.
5. Configure **JAVA\_HOME** - enter `"/etc/alternatives/jre_11"` without quotation marks.
6. Choose **Install Folder** - `/opt/local/MagicCollaborationStudio`.

Next, Magic Collaboration Studio Pre-Installation Summary will appear. It should look as follows:

```
=====
Pre-Installation Summary
-----
Please Review the Following Before Continuing:

Product Name:      Magic Collaboration Studio   Install Folder:
/opt/local/MagicCollaborationStudio   Machine ip:
"192.168.130.10"

Seed node ip:
"192.168.130.10"

JAVA_HOME:
"/etc/alternatives/jre_11"

Disk Space Information (for Installation Target):
Required:  395,614,661 Bytes
Available: 31,608,475,648 Bytes
```

Anywhere 192.168.130.10 is displayed, you must replace it with the IP address of your machine.

To start the authserver service, execute the following command:

```
sudo systemctl start authserver
```

To start the Magic Collaboration Studio service, execute the command:

```
sudo systemctl start twcloud
```

To start the WebApp service, execute the command:

```
sudo systemctl start webapp
```

To ensure the services start on reboot, execute the following commands:

```
sudo systemctl enable twcloud
sudo systemctl enable authserver
sudo systemctl enable webapp
```

## Post-Install Configuration

The installer has now created the preliminary Magic Collaboration Studio configuration. Your system is now functional and you can log in to the Admin console. For security reasons, we highly recommend making additional changes as described below.

- `/opt/local/MagicCollaborationStudio/configuration/application.conf` - the configuration file for the Magic Collaboration Studio service. If Magic Collaboration Studio is installed behind a proxy or firewall with NAT, upon the initial connection the MagicDraw client must know the external IP address to which it must connect. Search for **server-broadcast-host**, and enter the public IP address instead of the local IP address. We now need to point Magic Collaboration Studio to the Cassandra database. Search for **seeds =** , which is located in the connection section. Edit the value inside the quotes to point to the **listen\_address** you set in **cassandra.yaml** (i.e. seeds = ["192.168.130.10"]). A default password has been entered in the configuration file for its communication with the authorization server. It is recommended that it be changed. To do so, search for **CHANGE\_ME**, which is associated with the field **pswd**, and replace it with a password of your choosing.
- `/opt/local/MagicCollaborationStudio/AuthServer/config/authserver.properties` - the configuration file for Authorization service.
  - server.public.host**=public IP address (same as server-broadcast-host in **application.conf**). If you are accessing the server via an FQDN, use it instead of the IP address.
  - twc.server.host**=local IP address.
  - cassandra.contactPoints**=local IP address.
  - If you changed the **pswd** field in `/opt/local/MagicCollaborationStudio/configuration/application.conf` from the default, you must modify this file accordingly. Search for **authentication.client.secret**. Remove the leading **#** (to uncomment the directive), and replace the **CHANGE\_ME** value with the same value as that in **application.conf**.
  - If you are accessing the server by FQDN, you must edit the property **authentication.redirect.uri.whitelist** by adding an entry to whitelist the FQDN. For example: **authentication.redirect.uri.whitelist=https://192.168.130.10:8443/webapp/,https://FQDN:8443/webapp/,https://md\_redirect.**

Make sure that you add the public IP for both 8443 and 8111 ports to the whitelist.

- `/opt/local/MagicCollaborationStudio/WebAppPlatform/shared/conf/webappplatform.properties` - the configuration file for Magic Collaboration Studio Admin Console.
  - twc.admin.username** - Set it to the username of a local account with Administrator privileges (default is Administrator).
  - twc.admin.password** - Set it to the password corresponding to the Administrator user (default is Administrator).
  - If you changed the **pswd** field in `/opt/local/MagicCollaborationStudio/configuration/application.conf` from the default, you must modify this file accordingly. Search for **authentication.client.secret** and replace the **CHANGE\_ME** value with the same value as that in **application.conf**.

Once you have made the configurations, make sure to restart the affected services.

## Additional information that may affect installations in restricted environments

### Log Files

Magic Collaboration Studio executes under the Magic Collaboration Studio user, and by default will store log files of Magic Collaboration Studio and Authserver under this user's profile (`/home/twcloud`). There are 2 configuration files that control the location of these log files:

- `/opt/local/MagicCollaborationStudio/configuration/logback.xml` controls the location of the Magic Collaboration Studio log files, whereas
- `/opt/local/MagicCollaborationStudio/Authserver/config/logback-spring.xml` controls the location of the Authserver log files.

### `/opt/local/MagicCollaborationStudio/configuration/logback.xml`

In this file, there are settings for 2 log files that must be edited.

```

<appender name="SERVER-FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${user.home}/.twcloud/2021x/server.log</file>
  <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
    <fileNamePattern>${user.home}/.twcloud/2021x/server.%d{yyyy-MM-dd}.%i.log.zip<
/fileNamePattern>
    <maxFileSize>50MB</maxFileSize>
    <totalSizeCap>1024MB</totalSizeCap>
  </rollingPolicy>
  <encoder>
    <pattern>%-5.5level %date{YYYY-MM-dd HH:mm:ss.SSS} %message [%logger{200}, %thread{10}]\n<
/pattern>
  </encoder>
</appender>
<appender name="SERVER-STARTUP-CONFIG-FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${user.home}/.twcloud/2021x/startup-config.log</file>
  <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
    <fileNamePattern>${user.home}/.twcloud/2021x/startup-config.%d{yyyy-MM-dd}.%i.log.zip<
/fileNamePattern>
    <maxFileSize>50MB</maxFileSize>
    <totalSizeCap>1024MB</totalSizeCap>
  </rollingPolicy>
  <encoder>
    <pattern>%-5.5level %date{YYYY-MM-dd HH:mm:ss.SSSXXX} %message %n</pattern>
  </encoder>
</appender>

```

In each section, there are 2 settings that must be modified: **file** and **fileNamePattern**. The first setting (file) controls the absolute path to the latest log file. The second setting (**fileNamePattern**) controls the naming convention for the archiving of the log files. In most cases, it will suffice to replace the **\${user.home}** token with a different location, but you must ensure that the Magic Collaboration Studio user has ownership of the target directories.

### /opt/local/MagicCollaborationStudio/Authserver/config/logback-spring.xml

This file contains one section which must be modified.

```

<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${user.home.dir}/.authserver/2021x/authserver.log</file>
  <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
    <fileNamePattern>${user.home.dir}/.authserver/2021x/authserver.%i.log.zip<
/fileNamePattern>
    <minIndex>1</minIndex>
    <maxIndex>10</maxIndex>
  </rollingPolicy>
  <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
    <maxFileSize>30MB</maxFileSize>
  </triggeringPolicy>
  <encoder>
    <pattern>%-5.5level %date{YYYY-MM-dd HH:mm:ss.SSS} %message [%logger{0}, %thread{10}]\n<
/pattern>
  </encoder>
</appender>

```

The same changes and permissions apply to the changes to this file as to those for [/opt/local/MagicCollaborationStudio/configuration/logback.xml](#).

## Files installed on system locations

Daemon control files	Environment files	Cassandra installation
<ul style="list-style-type: none"> <li>• /etc/systemd/system/authserver.service</li> <li>• /etc/systemd/system/cassandra.service</li> <li>• /etc/systemd/system/twcloud.service</li> <li>• /etc/systemd/system/webapp.service</li> </ul>	<ul style="list-style-type: none"> <li>• /etc/cassandra/*</li> <li>• /etc/twcloud/*</li> </ul>	The script will place the data files in /data/data and commit logs in /logs/commitlog

## Frequently Asked Questions

**Q:** When accessing [https://ip\\_address:8443/webapp](https://ip_address:8443/webapp) I am displayed with a Tomcat 404 error

## HTTP Status 404 – Not Found

**Type** Status Report

**Message** /webapp

**Description** The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.

### Apache Tomcat/9.0.12

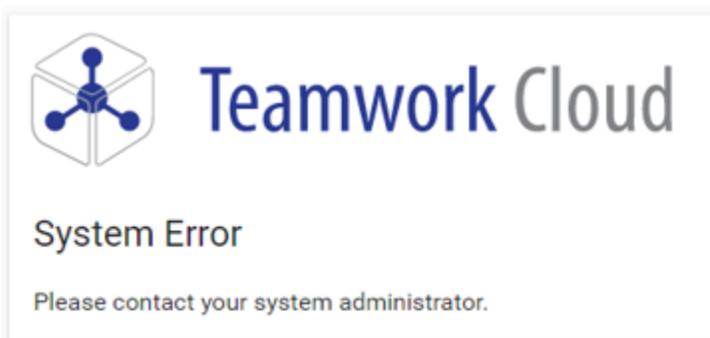
**A:** This error is typically caused by incorrect credentials for the authentication of webapp. This is accompanied by recurring error messages in web-app.log.

```
2019-10-12 14:49:14,625 [main] INFO com.nomagic.webappplatform.internal.version.TWCVersionValidator - Waiting for TWC/AuthServer to start, 120 of 120
```

```
2019-10-12 14:49:14,626 [main] ERROR com.nomagic.webappplatform.internal.version.TWCVersionValidator - Exception occurred during version checking
```

Verify that the credentials are correct in webappplatform.properties

**Q:** When Accessing the Magic Collaboration Studio Admin Console (WebApp), I get a system error when redirected to the Authserver login screen



**A:** This error is typically caused by an omission of the referring URL in the authentication server's whitelist, authentication.redirect.uri.whitelist, located in authserver.properties.

You will see a corresponding error in authserver.log.

```
ERROR 2019-10-04 17:26:52.258 AuthorizeException: invalid_request, Invalid redirect_uri parameter [AuthorizeController, ...]
```

Update the whitelist to include the referring URL. A common cause is accessing WebApp via a server name of FQDN when the whitelist only contains entries for the IP address.

**Q:** After entering my credentials in the Authserver login screen, I am not logged in, no error is displayed, and I am presented once again with the login screen



**A:** This error is typically caused by a mismatch in the client secret entries - authentication.client.secret located in authserver.properties and webappplatform.properties, and "pswd" in application.conf.

**You will see a corresponding error in authserver.log**

ERROR 2019-10-04 17:30:49.382 Invalid client secret ...