



Private & Confidential

Intelligent Voice 4.7 Administration Guide

Date: 21st August 2018

Intelligent Voice Limited
St Clare House
30-33 Minories
London EC3N 1DD
t: +44 20 3627 2670
e: info@intelligentvoice.com
w: www.intelligentvoice.com



INTELLIGENT VOICE DISCLAIMER

© 2018 Intelligent Voice. All rights reserved.

Intelligent Voice believes the information in this publication is accurate as of its publication date. The information is subject to change without notice. The information in this publication is provided “as is”. Intelligent Voice makes no representation or warranties of any kind with respect to the information in this publication and specifically disclaims implied warranties of merchantability or fitness for a particular purpose. Using, copying, and distribution of any Intelligent Voice software described in this publication requires an applicable software license.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS THE PROPRIETARY AND CONFIDENTIAL INFORMATION OF INTELLIGENT VOICE LIMITED. YOU MAY NOT DISCLOSE, PROVIDE OR MAKE AVAILABLE THIS DOCUMENT, OR ANY INFORMATION CONTAINED IN THIS DOCUMENT, TO ANY THIRD PARTY, WITHOUT THE PRIOR WRITTEN CONSENT OF INTELLIGENT VOICE LIMITED.

The information in this document is provided for use with Intelligent Voice products. No licence, express or implied, to any intellectual property associated with this document or such products is granted by this document.

All Intelligent Voice products described in this document are owned by Intelligent Voice Limited (or those companies that have licensed technology to Intelligent Voice) and are protected by patents, trade secrets, copyrights or other industrial property rights.

The Intelligent Voice products described in this document may still be in development. The final form of each product and release date thereof is at the sole and absolute discretion of Intelligent Voice. Your purchase, license and/or use of Intelligent Voice products shall be subject to Intelligent Voice’s then current sales terms and conditions.

Contents

Introduction	6
Overview of Intelligent Voice components	6
Application Server (vrx)	6
MariaDB Database	6
Search Engine	7
Job queue server (Gearman)	7
IV “JumpTo” Web Application (PHP on Apache)	7
Processing tier workers (ASR, Diarization, VAD, Tagger, Cracker)	7
Cluster example	7
Web tier	8
Application Tier	8
Processing Tier	8
Installation and Upgrades	9
Installing new models	9
Security	9
SSL certificates	10
Passwords	10
SSH	10
Gearman	10
TCP Ports	11
Monitoring	11
Services	11
Audits	12
Logs	12
Application error logging available in the API	12
Backup	12
Audio files	13
Cache	13
Database	13
Single database instance	13
Galera Cluster	13
Biometrics	13

Indexes	13
Archiving and Restoration	14
Back up a single group on a standalone Intelligent Voice server	14
Restore a single group on a standalone Intelligent Voice server	14
Archive a single group on a standalone Intelligent Voice server:	14
Deleting Groups.....	14
Application Server settings	15
Caching	15
Indexing	15
Export options.....	15
Email Configuration.....	15
Diarization (Speaker Separation)	15
Multi-Channel and Stereo Processing.....	16
Cache Folder	17
Temp Directory	17
JumpTo Web settings	18
Email configuration	18
Processing services settings.....	18
Controlling the number of instances	18
Querying the IV database	18
Database user accounts	18
Example queries.....	19
Hide Groups	19
Identify Items Which Have No Text	19
Identify Whether Tags Were Created	19
Identify Number Of Tags Not Created For An Import.....	19
Show Job Payload	19
Add a new API account	19
Mount Shares/Drives from another server	20
Galera database cluster restart.....	21
Removing unwanted biometric data.....	22
Disable Tagging.....	23
Clear all unprocessed items from the Gearman queues	23

Rebuilding Indexes	24
Delta	24
Merge	24
Product Support	25
Appendix A. Useful Linux Commands.....	25
Disk Space	25
CPU Activity	25
Change Static IP Address	26
Add Additional Hard Drive Disk Space	27
Collect and compress log files	28
Appendix B. Reading data from S3 into Intelligent Voice	28
Mounting the bucket	29

Introduction

This administration guide is designed for administrators of an Intelligent Voice system. Some experience administering Linux servers is assumed but some useful Linux commands are included in Appendix A.

Overview of Intelligent Voice components

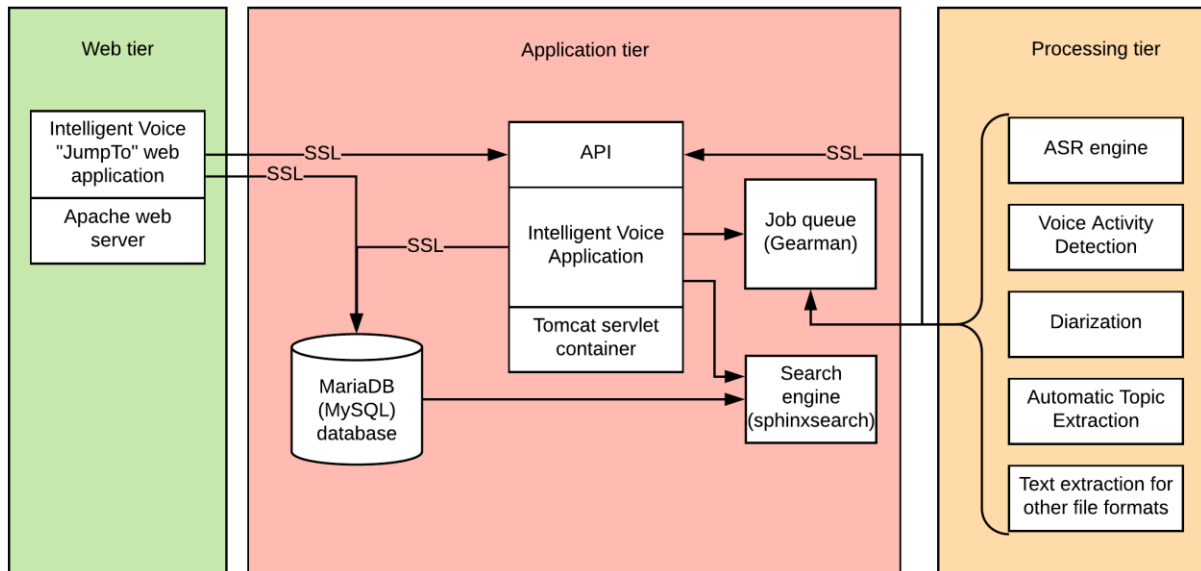


Figure 1 Intelligent Voice components grouped by tier

The Intelligent Voice system consists of a number of components, each of which can be installed on one or more physical or virtual servers. When installing on multiple servers, a common pattern is to split the running services into 3 tiers – see “Cluster example” below for more.

Intelligent Voice Application – vrx (running on Tomcat)

The IV application server (codename vrx).

This is a Java application which uses JAX-RS and Spring framework and runs in Apache Tomcat container. This is the core of the Intelligent Voice system and provides the API that powers the web application and the processing workers. The API is also available for the third party applications to build on.

The application server can be configured as failover and/or load balanced.

MariaDB Database

MariaDB 10, with optional Galera Cluster. MariaDB is a fork of MySQL created by the original author of MySQL.

All the data created by the application server is stored in this database. Galera Cluster support provides high availability and can be load balanced for scalable performance.

Search Engine

Search engine support provided by open source Sphinxsearch search engine with wrapper functions provided by IV for indexing and searching through web service requests. This can be distributed across multiple nodes for redundancy and performance.

Job queue server (Gearman)

Gearman job queue server provides a distributed shared job queuing mechanism. Jobs are inserted by the IV application server, and processed by workers which can be run on the same server or on a distributed cluster of servers.

IV “JumpTo” Web Application (using PHP on Apache)

The JumpTo web application provides an optional front end for the Intelligent Voice system. It is written in PHP and runs on Apache 2 web server.

Processing tier workers (ASR, Diarization, VAD, Tagger, Cracker)

- ASR – Automatic Speech Recognition
- VAD – Voice Activity Detection
- Diarization – Speaker separation
- Tagger – Automatic Topic Extraction
- Cracker – Text Extraction for other file formats (non-audio)

The workers are independent processes which take jobs from the job queue and return results via the web services API. The performance of the overall system is largely governed by the number of workers running. The workers are typically distributed over multiple systems for scaling and redundancy. The balance of workers must be adjusted to suit the available resources. For voice-only work, a typical balance would look like this:

1 x VAD Worker

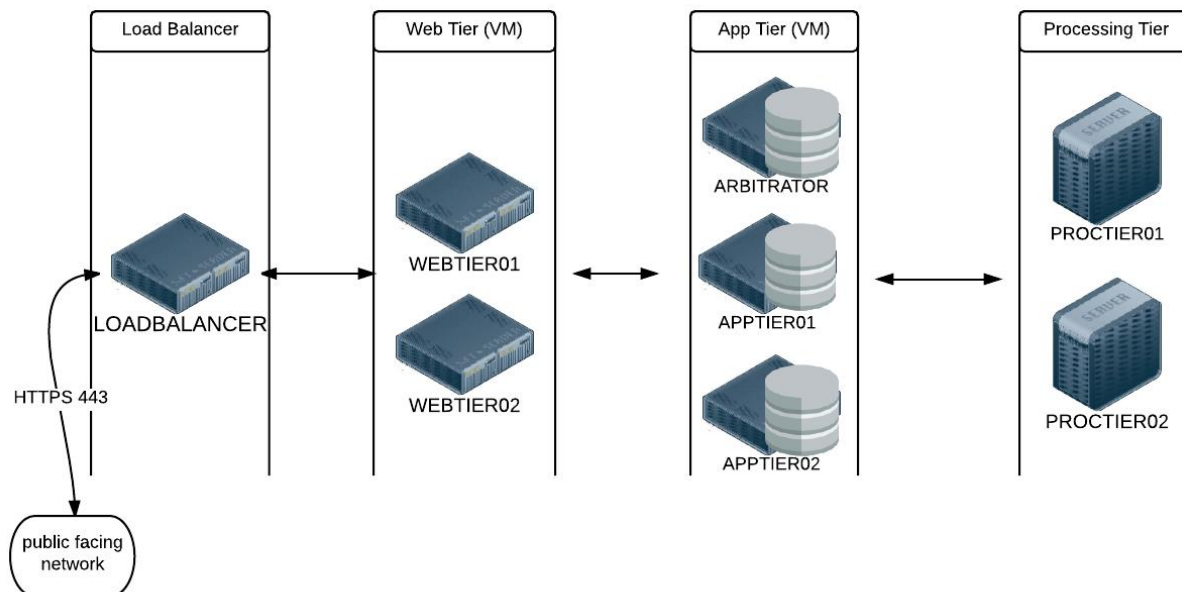
4 x ASR (Automatic Speech Recognition) Worker on GPU *or* 40x ASR Worker on CPU

24 x Diarization (Speaker identification) Worker

1 x Tagger Worker

Cluster example

If you are creating a cluster of machines, a common pattern is to split the IV services into 3 tiers.



Web tier

Server(s) that run the optional *JumpTo* website. If you are connecting IV directly to the internet, it is recommended you have a separate tier for the website for security (principle of defense-in-depth). Multiple web servers can be connected behind a load balancer for high availability

Application Tier

Servers that run the Intelligent Voice application server (vrx), the database, search indexes and job queues. For performance reasons it is generally recommended to have these services running on the same machine. Multiple instances can be run at once using the cluster database (MariaDB Galera).

Processing Tier

Servers that run process jobs for the application tier, such as ASR, diarization, voice activity detection (VAD), tagging and cracking (text extraction). These can all be run together on one or more machines, or run independently according to available resources. For example, the ASR can be run on GPU-equipped machines while Tagger and Diarization run on machines with more CPU and RAM. Processing services are configured to point at the job queue and application server API, but can be added and removed at any time, with no configuration needed on the application server. If a processing service is removed in the middle of a job, the job is returned the queue and can be processed later by another node. This allows for simple auto-scaling of the processing services.

Installation and Upgrades

See separate Installation guide for full details.

IV is distributed as one or more installer archive file(s). Within these file there is an install config file and a choice of install scripts to install everything on one server or individual components.

Upgrading an existing system can be done with the same installer, by disabling the overwrite option in the install config file.

Installing new models

Intelligent Voice includes tools to adapt language models. Details on this process are in the separate Language Model Adaption Guide.

There is an API method to install a model created with this process or a model distributed by Intelligent Voice. You can either call an API directly, or use the `install_model.py` script (which is included in the proc node installer) which calls the API.

There are 3 steps required for each model you want to install:

1. Put the model on the server in `/opt/models/ASR`
If it is compressed, extract it. The layout should include a folder named "model" inside a folder named with the complete model name. The contents should look like this:

```
total 144M
dr-xr-x---. 2 IV-ASR intvoice  49 Dec 15 16:29 conf
-r-xr-x---. 1 IV-ASR intvoice 143M Oct 18 14:06 final.mdl
dr-xr-x---. 3 IV-ASR intvoice  111 Dec 15 16:30 graph
dr-xr-x---. 3 IV-ASR intvoice  111 Dec 15 16:31 graph_ph
dr-xr-x---. 2 IV-ASR intvoice  125 Dec 15 16:31 ivector_extractor
-r-xr-x---. 1 IV-ASR intvoice  623 Oct 18 14:06 mfcc.conf
-r-xr-x---. 1 IV-ASR intvoice 1.2M Oct 18 14:06 tree
```

2. Run these two commands to set the permissions correctly on the model:

```
chown -R IV-ASR:intvoice /opt/models/ASR/IntelligentVoice_de-
DE_8kHz_56000_general_V4_ASrv5/
```

```
chmod -R 550 /opt/models/ASR/IntelligentVoice_de-DE_8kHz_56000_general_V4_ASrv5/
```

3. Import the model by running this command:

```
/opt/lmbuilder/install_model.py IntelligentVoice_de-DE_8kHz_56000_any_V4_ASrv5/
```

Security

See separate Intelligent Voice security guide for more detail.

Intelligent Voice consists of a number of services running on Linux servers and communicating with each other primarily via HTTPS (HTTP encrypted with TLS).

Security is designed around “defence in depth” principles using well tested and proven open source components where possible. Services are run with dedicated “system” user accounts (i.e. accounts with no login permissions) and minimal file access permissions. Communication between machines in a cluster is encrypted and restricted to named servers on a small number of TCP ports (detailed below), allowing the use of firewalls and monitoring.

User input is validated, sanitized and audited.

SSL certificates

The installer includes a certificate authority (ca.intelligentvoice.com) which is used by the installer to sign certificates used by the application server, JumpTo web application and the database. Other services are configured to trust this CA by default. This permits the IV system to work with enforced encryption out of the box even without internet access.

We recommend production systems replace the certificates generated by the installer with certificates created locally or trusted third parties, and then remove the included certificate authority. The easiest way to do this is to replace the files in /opt/jumpton/ssl on each server.

IV uses the standard methods of configuring certificates for tomcat, apache and mysql. For more advanced options such as the selection of ciphers, the documentation for tomcat, apache and mysql can be consulted.

Passwords

The IV installer sets up a number of passwords, mainly for database users accounts. You can set all these passwords as required in the installer config file install.cnf. It is highly recommended that you generate random passwords, and after installation remove this file and store in a secure location not on the IV server(s).

You can change the passwords at any time by editing the install.cnf and re-running the install (or most recent upgrade).

SSH

The recommended and most common way of administering IV servers is over SSH. Keys should be used instead of passwords.

NOTE: Prior to 4.0.1 the installer required SSH with passwords enabled, to set up keys for rsync. From version 4.0.1 this has been removed and it is now possible to use IV with SSH only permitting key access, or entirely without SSH installed.

Gearman

The only service deployed by IV that permits unencrypted connections is Gearman, the job queue service. This service can be configured to only listened on a local interface and remote processing

nodes configured to connect over a SSH tunnel. Instructions are in the security guide. A future release of IV will enforce TLS encryption for Gearman (INT-2430)

TCP Ports

The following services are listening on TCP ports. All other ports can be blocked by firewalls.

Service	port
mysql	3306
tomcat8	8443
apache2	443
gearman-job-server	4730

Monitoring

It is highly recommended that the IV system is continuously monitored by an alerting system such as Zabbix, Nagios, Prometheus, Cloudwatch, etc

IV servers are expected to have very high CPU usage while importing, processing and exporting.

IV servers are set to clear down temporary files and rotate logs. Alarms can be set on low disk space.

Services

If you are running IV on a single server or VM, all of the following services should be running. When running a cluster, the services can be run on different servers. The services running when using the layout given under “cluster example” is shown in the table below:

Service	Description	Web	App	Proc
mysql	MariaDB Database		✓	✓
sphinxsearch	Search engine (requires mysql)		✓	✓
tomcat8	Application server (requires mysql)		✓	
apache2	Web server (requires mysql)	✓	✓	
gearman-job-server	Job queue (requires mysql)		✓	
PhonemeLoader	Processes phoneme data from ASR and loads into the database		✓	
ASRWorker	Automatic Speech Recognition – transcribe speech to text			✓
DiarizationWorker	Identify speakers in a single recording channel			✓
VADWorker	detects voice in audio so ASR can skip non-voice sections			✓
CrackerWorker	Text extraction for emails, documents, chat logs, etc			✓
TaggerWorker	Automatic topic creation (requires mysql and sphinxsearch)			✓

The status of services can be checked with systemctl. For example, to check the status of sphinxsearch:

```
systemctl status sphinxsearch
```

Any services that are not running can be started systemctl, for example:

```
systemctl start sphinxsearch
```

Audits

Audit information collected by IV is available through the Audit API.

Sessions in JumpToWeb are audited in the dblog report (<https://iv-server-name/JumpToWeb/admin/reports/dblog>)

Logs

Service	Description
mysql	/var/log/mysql (Detailed logging can be enabled in /etc/mysql/my.cnf)
sphinxsearch	/var/log/sphinxsearch/searchd.log /var/log/sphinxsearch/query.log
tomcat8	/var/log/tomcat8/j2ws.log
apache2	/var/log/apache2/error.log (/var/log/httpd on Red Hat)
gearman-job-server	/var/log/gearman-job-server/gearman.log (does not log anything in normal use)
PhonemeLoader	/opt/jumpto/PhonemeLoader/log/PhonemeLoader.log
ASRWorker	/var/log/ASR/ASRWorker.log
DiarizationWorker	/var/log/Diar2/Diar2Worker.log
VADWorker	/var/log/VAD/VADWorker.log
CrackerWorker	/var/log/Cracker/CrackerWorker.log
TaggerWorker	/var/log/Tagger/TaggerWorker.log

Application error logging available in the API

The errors for items, recordings and segments can be viewed on a group-by-group basis using the Error API or the report in the JumpTo web application. They are also stored in the database.

(The basic unit in Intelligent Voice is an item. An item can have multiple recordings (e.g. when a call record has channels recorded to separate files) and these will be split into segments by speaker.)

Backup

There are three required items for backup: database, cache and audio files. You can also back up biometric files if you will use the biometric search tool, and the search indexes which makes restoration quicker.

Audio files

Audio files you have imported can be found in the original location you imported them from, if the import was done from a local path, or in the cache if done from a remote path.

You can get a list of all audio files imported using the bulk import tool on the import report:

<https://yourservername/JumpToWeb/admin/reports/jumptoweb/imports>

Cache

Backup everything in this location:

```
/var/jumpto/cache
```

Database

Single database instance

For a single database instance, backup can be done using the mysqldump tool. An example command to do this:

```
mysqldump -u root -p --all-databases | xz > iv-database-backup.xz
```

Galera Cluster

For a Galera cluster, using mysqldump to backup the database is sufficient to prevent data loss but restoring to a cluster can be complicated and the cluster performance may be affected by the backup.

For a guide to best practices for backing up a Galera cluster, see

<http://galeracluster.com/documentation-webpages/backingupthecluster.html>

Biometrics

Biometric files (.ser) can be preserved for later use with a manual Speaker Identification task. This will copy all biometric files to /backup:

```
find /var/jumpto/cache -type f -name "*.ser" -exec cp -rfp {} /backup \;
```

Indexes

Indexes can be recreated from the database. To save time on restore, everything under /var/sphinxsearch can be backed up.

Restoration is just copying the backed up index(es) back into place.

Indexes are built entirely from data contained in the database so they can also be restored by doing a main + delta rotate.

Archiving and Restoration

To archive an entire server follow the process to backup.

Back up a single group on a standalone Intelligent Voice server

1. Identify the group number (e.g. 101)
2. Use mysqldump to back up all the tables in the database starting with the group number (such as 101_evitems, 101_mail, etc)
3. Back up all the folders within the cache folder which start with the group number (such as 101000001, 101000002, etc)

Restore a single group on a standalone Intelligent Voice server

1. Identify the group number (e.g. 101)
2. Restore the database dump file
3. Restore the folders to the cache folder.

Archive a single group on a standalone Intelligent Voice server:

1. Identify the group number (e.g. 101)
2. Back up the group (see How to Back up a group on a standalone Intelligent Voice server)
3. Drop all the tables starting with the group number (such as 101_evitems, 101_mail, etc)
4. Delete all the folders within the cache folder which start with the group number (such as 101000001, 101000002, etc)

Deleting Groups

To do this manually the process is:

1. Drop all the tables starting with the group number (such as 101_evitems, 101_mail, etc)
2. Delete all the folders within the cache folder which start with the group number (such as 101000001, 101000002, etc).

A script to automate this process named “delete_group.sh” is included in the Intelligent Voice installer.

To use it, first make it executable:

```
chmod +x delete_group.sh
```

Then call it with the group number, for example group 1234:

```
./delete_group.sh 1234
```

This will create the log file delete_group1234.log

Application Server settings

The Intelligent Voice application server (“vrx”) is a Java J2EE application which runs on Apache Tomcat 8 and provides web services used by the other Intelligent Voice applications and the Intelligent Voice API.

vrx.properties is the main vrx settings file. It is deployed in the webapp (usually somewhere under /var/lib/tomcat8/webapps/) and needs to be copied to /opt/jumpto/vrx.properties to prevent it being replaced.

If there are multiple application servers, the settings must be changed on all applications servers simultaneously.

Caching

The following parameter should normally only be configured at installation time:
cachingOn=true - turns the cache on or off

The following parameters should not be changed while items are being imported:
cachefiles=/var/jumpto/cache/ - sets the cache location

Indexing

DisableIndexing=false - prevents new files being indexed.

Export options

HideBodyText=false – hide or show the transcript text in the body of the exported email (if hidden it will still be in the exported file so it can be indexed, but the text will not be visible)

Email Configuration

There are several settings in the “vrx” file for email configuration:

smtpServer	– the SMTP mail server
smtpPort	– the SMTP Port number for outgoing emails
smtpAuth	– SMTP Authentication tuned on or off
smtpFrom	– default JumpTo sender email used if IV auto generated email is empty
smtpUser	– default sender email
smtpPassword	– default sender password
smtpTo	– default recipient's email
smtpTLSEnabled	– SMTP TLS tuned on or off

Diarization (Speaker Separation)

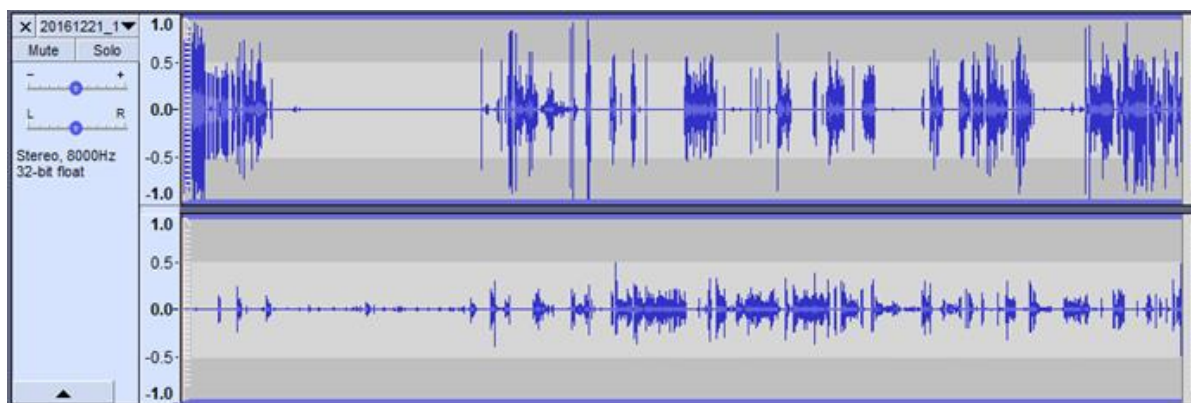
diarization.enabled=false – Disables Diarization job so no speakers are created. From 4.0.1 this option can also be set on individual items using the API.

Multi-Channel and Stereo Processing

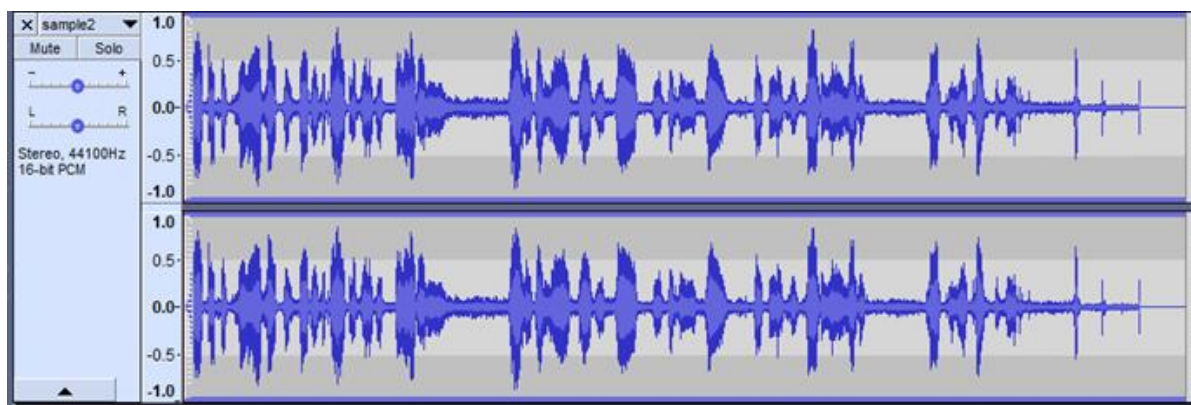
`treatAllFilesAsSingleChannel=false` – if set to "true" then files with more than 1 channel will be merged into a single channel otherwise for a file with multiple channels IV will create a single item with multiple recording entries, each of which will be processed separately by the ASR engine and identified separately in the output (as Channel 1, Channel 2, etc). From 4.0.1 this option can also be set on individual items using the API.

There are 3 types of files:

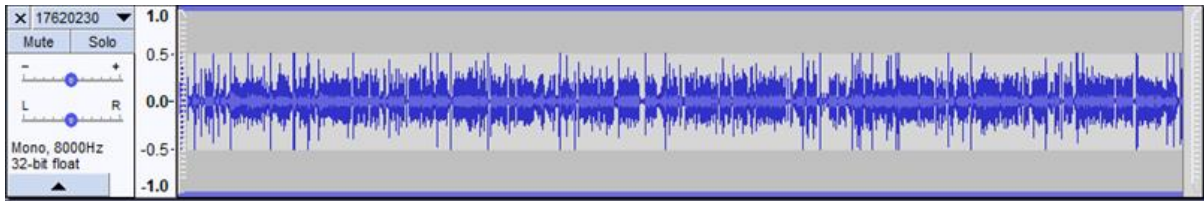
1. Multi-channel: Multiple speakers separated into multiple channels. If you open the file in Audacity you will see a different waveform in each channel. The settings for this are normally `treatAllFilesAsSingleChannel=false`, `diarization.enabled=false` but if you know one channel has multiple speakers you can try with `treatAllFilesAsSingleChannel=false`, `diarization.enabled=true`



2. Stereo: One or more speakers, but with the same audio in each channel. If you open the file in Audacity you will see the same or similar waveform in each channel. The settings for this are `treatAllFilesAsSingleChannel=true`, `diarization.enabled=true`



3. Mono: Only one channel. If you open the file in Audacity you will see only one channel. The settings for this are `treatAllFilesAsSingleChannel=false`, `diarization.enabled=true`



Cache Folder

1. Stop tomcat
2. Edit this file: /opt/jumpton/vrx.properties
3. Change this line: cache=/var/jumpton/cache

```
# Location for cache files
cachefiles=/mnt/data/sample/
cacheroot=/filesforbreaking/
```

4. Start tomcat

Temp Directory

The application server writes temporary files for audio file format conversion into a temporary directory. The default location for this is /tmp/tomcat8-temp. You may need to relocate this, e.g. for performance or storage space reasons.

1. Choose a new directory. This should be owned by and writable by the tomcat user.
2. Stop tomcat
3. Edit this file: /etc/default/tomcat8
4. Find these lines:

```
# Location of the JVM temporary directory
# WARNING: This directory will be destroyed and recreated at every startup !
#JVM_TMP=/tmp/tomcat8-temp
```

5. Uncomment the last line and set it the desired location.

```
# Location of the JVM temporary directory
# WARNING: This directory will be destroyed and recreated at every startup !
JVM_TMP=/mnt/data/sample/
```

6. Start tomcat

JumpTo Web settings

The JumpTo Web application is based on the Drupal PHP framework. Many of the settings are part of this framework and more information can be found at <https://www.drupal.org>.

Email configuration

Emails are used primarily for password reset for users. PHP must be configured to send email via a local MTA – sendmail is installed by the IV installer.

The sender address is configured through the JumpTo web site here:

<https://iv-server-name/JumpToWeb/admin/config/system/site-information>

Processing services settings

Settings for the processing services, such as the hostname of the application server and Gearman server to connect to, are set in `/opt/jumpto/config`

Controlling the number of instances

When GPUs are available for accelerated ASR, the ASR Worker automatically creates two processes for each GPU. Having a spare process allows the audio file to be transferred to the processing node and pre-processing work to be done ready for when a GPU is available.

For other workers, the number of processes must be configured to match the available resources. This is done by editing the worker files in `/opt/IV` and setting the `INSTANCES` variable at the top.

Querying the IV database

To run database queries on the command line the mysql command line client can be used like this:

```
mysql obsilon -uroot -p -e ""
```

for example:

```
mysql obsilon -uroot -p -e "select column from tablename;"
```

The IV application stores all application data in a schema called `obsilon`.

Database user accounts

The IV installer sets a password for the mysql “root” user and creates additional user accounts for the different services. You can set all these passwords as required in the installer config file `install.cnf`.

Example queries

Hide Groups

Groups can be hidden from all users of the JumpTo website by deleting their entry in the website table “jumpto_clients”. E.g. to hide groups 1, 2 and 3:

```
mysql obsilon -uroot -p -e "DELETE FROM obsilon.jumpto_clients WHERE idclients IN (1,2,3);"
```

Identify Items Which Have No Text

This command will display all the items which have no text. The subject field is displayed to trace back to the original item:

```
mysql -uroot -p -e " select id, EVSubject from obsilon.1_evitem where alltext_length = 1"
```

The reason 1 is used for alltext_length rather than 0 is a single space is appended to each transcript.

Identify Whether Tags Were Created

This command will show file name, start time, the last update time and whether tags were created:

```
mysql -uroot -p obsilon -e "select im.file, im.processing_start, max(i.Tagged) last_update, count(failed) tags_not_created from import_metadata im, import_item ii, 1_evitem i where ii.metadata_id = im.id and ii.item_id = i.id group by im.file"
```

Identify Number Of Tags Not Created For An Import

This command will show the start and finish time of an import and will also display the number of tags not created:

```
mysql -uroot -p obsilon -e "select im.file, im.processing_start, max(i.Tagged) last_update, count(failed) tags_not_created from import_metadata im, import_item ii, 1_evitem i where ii.metadata_id = im.id and ii.item_id = i.id group by im.file"
```

Show Job Payload

This query will show the actual job payload (in JSON format) from which the jobs in the queue can be seen:

```
SELECT CONVERT(data USING utf8) from gearman.gearman_queue where function_name = 'diar';
```

Add a new API account

To add a new API account “api_user” with password “fjlweru2-903”

```
mysql -uroot -p jumpto_admin -e "INSERT INTO users (user_name, user_pass) VALUES ('api_user', 'fjlweru2-903');INSERT INTO user_roles (user_name, role_name) VALUES ('api_user', 'auth-users');"
```

Mount Shares/Drives from another server

Mounting Windows network shares in Ubuntu is covered online here:

<https://wiki.ubuntu.com/MountWindowsSharesPermanently>

In brief, to configure a temporary mount for windows file share, create the directory to use as the mount point:

```
mkdir /mnt/path
```

NOTE: to use the share to import, export, or as a cache for the application server, then the share should be owned by tomcat8. Use tomcat8's user id and group id in the mount command.

To get tomcat8's user id and group id (user 109, group 116 in this case):

```
user@server:~$ grep tomcat8 /etc/passwd
tomcat8:x:109:116::/usr/share/tomcat8:/bin/false
```

Temporarily mount it with the following command:

```
sudo mount -t cifs //servername/path /mnt/path -o
username=*****,password=*****,iocharset=utf8,file_mode=0777,dir_mode=
0777,uid=109,gid=116
```

To configure a permanent mount point, add a line like this to the /etc/fstab file

```
//servname/path /mnt/cache cifs credentials=/etc/cifspwd,iocharset=utf8,
file_mode=0777,dir_mode=0777,uid=109,gid=116 0 0
```

And add the password to /etc/cifspwd, which should look like this:

```
username=myusername
password=mypassword
```

And protect the password file like this:

```
sudo chmod 600 /etc/cifspwd
```

If mount points have been configured in the fstab the following command can be used to remount:

```
sudo mount -a
```

Galera database cluster restart

To check the number of nodes in a cluster:

```
SHOW GLOBAL STATUS LIKE 'wsrep_cluster_size';
```

```
MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_cluster_size';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| wsrep_cluster_size | 3 |
+-----+-----+
1 row in set (0.00 sec)
```

To check if the cluster can accept queries:

```
SHOW GLOBAL STATUS LIKE 'wsrep_ready';
```

```
MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_ready';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| wsrep_ready | ON |
+-----+-----+
1 row in set (0.00 sec)
```

When the node returns a value of ON it can accept write-sets from the cluster. When it returns the value OFF, almost all queries fail with the error: ERROR 1047 (08501) Unknown Command

To stop a node within the cluster (for the purpose of maintenance, configuration change, etc) run:

```
sudo service mysql stop
```

```
chase@aaron2:~$ sudo service mysql stop
* Stopping MariaDB database server mysqld
chase@aaron2:~$
```

In this case the other nodes receive “good bye” message from that node, hence the cluster size is reduced.

To restart the cluster node:

```
sudo service mysql start
```

```
chase@aaron2:~$ sudo service mysql start
* Starting MariaDB database server mysqld
* Checking for corrupt, not cleanly closed and upgrade needing tables.
chase@aaron2:~$
```

The joiner node won't serve any requests until it is again fully synchronized with the cluster, so connecting to its peers isn't enough; state transfer must succeed first.

During clean shutdown the node will write its last executed position into the grastate.dat file:

```
chase@aaron2:~$ sudo nano /var/lib/mysql/grastate.dat
GNU nano 2.2.6 File: /var/lib/mysql/grastate.dat

GALERA saved state
version: 2.1
uuid: f7da3e0d-1a8f-11e5-82f0-ef5b17d66762
seqno: 782035
cert_index:
```

By comparing the seqno number of each node, the highest seqno amongst them is most likely the last one stopped. The cluster must be bootstrapped using this node.

If bootstrap is required the service fails to start with this error:

```
WSREP has not yet prepared node for application use
```

Full bootstrap sequence:

- Shutdown all nodes
- Start most recent node mysql with --wsrep-new-cluster:
`sudo service mysql start --wsrep-new-cluster`
- On other nodes, remove galera state files:
`sudo rm /var/lib/mysql/galera.cache /var/lib/mysql/g*.dat`
- Start other node(s) and arbitrator(s)

Additional cluster information can be found at <http://galeracluster.com/documentation-webpages/>

Removing unwanted biometric data

Biometric files (.ser) are preserved for later use with a manual Speaker Identification task.

If this is not required, the files can be deleted. An example command do this (on the app server) is:

```
find /var/jumpto/cache -type f -name "*.ser" -exec rm -f {} \;
```

Disable Tagging

The English language tagger is the default. If you are using voice transcription in another language you might wish to disable tagging. This can be done by stopping the TaggerWorker service during the import, and then removing all the tagger jobs afterwards (see section “Clear all unprocessed items from the Gearman queues” in this guide)

Clear all unprocessed items from the Gearman queues

An in-progress import can be stopped by removing the unprocessed jobs from the queues.

This action cannot be undone, so any of the items they must be imported again to be reprocessed.

On both APPLICATION TIERS stop gearman job service:

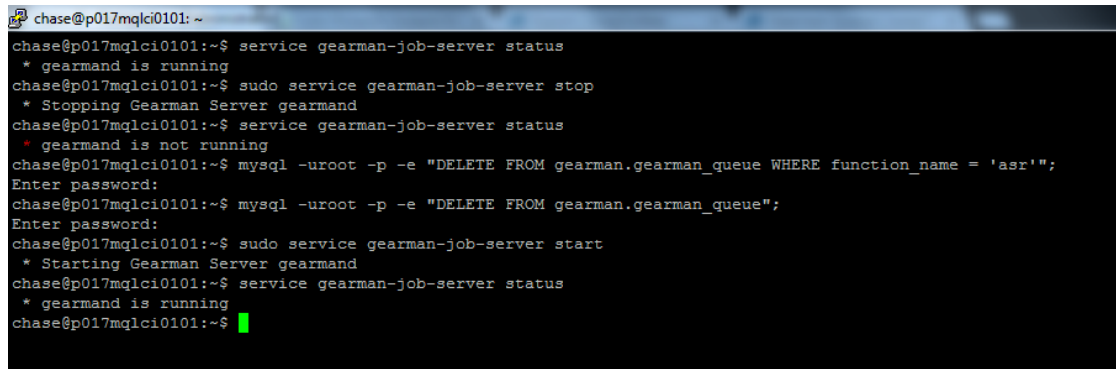
```
sudo service gearman-job-server stop
```

On the primary processing node run this removes just the ASR jobs:

```
mysql -uroot -p -e "DELETE FROM gearman.gearman_queue WHERE function_name = 'asr'";
```

This removes all jobs from the primary processing node:

```
mysql -uroot -p -e "DELETE FROM gearman.gearman_queue";
```



```
chase@p017mqlci0101: ~  
chase@p017mqlci0101:~$ service gearman-job-server status  
* gearmand is running  
chase@p017mqlci0101:~$ sudo service gearman-job-server stop  
* Stopping Gearman Server gearmand  
chase@p017mqlci0101:~$ service gearman-job-server status  
* gearmand is not running  
chase@p017mqlci0101:~$ mysql -uroot -p -e "DELETE FROM gearman.gearman_queue WHERE function_name = 'asr'";  
Enter password:  
chase@p017mqlci0101:~$ mysql -uroot -p -e "DELETE FROM gearman.gearman_queue";  
Enter password:  
chase@p017mqlci0101:~$ sudo service gearman-job-server start  
* Starting Gearman Server gearmand  
chase@p017mqlci0101:~$ service gearman-job-server status  
* gearmand is running  
chase@p017mqlci0101:~$
```

Then on the PROCESSING TIER restart the worker services:

```
sudo service ASRWorker stop
```

```
sudo service ASRWorker start
```

On both APPLICATION TIERS start the gearman job service:

```
sudo service gearman-job-server start
```

Rebuilding Indexes

Search in Intelligent Voice is powered by SphinxSearch open source search.

Indexes are automatically rotated at defined intervals (usually delta every 5 minutes and main every hour, but the interval can be customised in `/etc/cron.d/jumpton` on the app tier nodes).

Because of this, it should not normally be necessary to manually rebuild indexes. If a manual rebuild is required, these commands can be run (if running a cluster with multiple sphinxsearch servers, it should be run on all of them):

Delta

Command:

```
sudo php /opt/jumpton/scripts/mainindex.php delta
```

Expected output:

```
Rotating 4_delta_items
Rotating 4_delta_itemsmorph
Rotating 4_autocomplete
Rotating 7_delta_items
Rotating 7_delta_itemsmorph
Rotating 7_autocomplete
```

Merge

Command:

```
sudo php /opt/jumpton/scripts/mainindex.php merge
```

Expected output:

```
logs will be written to /var/log/sphinxsearch/
Setting new max id in 4_sph_counter_items: 1
Merging 4_items and 4_delta_items
Rotating 4_delta_items
Setting new max id in 4_sph_counter_morph_items: 1
Merging 4_itemsmorph and 4_delta_itemsmorph
Rotating 4_delta_itemsmorph
Rotating 4_autocomplete
```


Product Support

To submit a support request please [submit a support request](#) online or email support@intelligentvoice.com.

Alternatively you can speak to one of our support agents Monday-Friday 9:00am-5:30pm UK time on this phone number:

UK: +44 20 3697 0216

Appendix A. Useful Linux Commands

Disk Space

To check disk space use the following command:

```
df -h
```

```
demo@ivtest-780:~$ df -h
Filesystem                Size      Used Avail Use% Mounted on
/dev/mapper/ivtest--780--vg-root  1.9T    394G    1.5T   22% /
udev                      5.9G     4.0K    5.9G    1% /dev
tmpfs                      1.2G     428K    1.2G    1% /run
none                       5.0M      0      5.0M    0% /run/lock
none                       5.9G     60K    5.9G    1% /run/shm
/dev/sda2                  237M     39M    186M   18% /boot
/dev/sda1                  487M     2.1M    484M    1% /boot/efi
```

CPU Activity

To check CPU Activity use the following command:

```
top
```

Then:

```
1
```

This will show the individual usage for each CPU:

```
top - 10:14:31 up 7 days, 22:04,  2 users,  load average: 0.09, 0.12, 0.10
Tasks: 167 total,  1 running, 166 sleeping,  0 stopped,  0 zombie
Cpu0  :  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  :  0.3%us,  0.3%sy,  0.0%ni, 99.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  12239684k total, 12094252k used,  145432k free,  135464k buffers
Swap:  4128764k total,   376k used,  4128388k free,  8771588k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 1232 sphinxse  20   0 3111m  2.9g  2.9g  S   0 25.2  12:19.48 searchd
 1246 root      20   0 15984   656  480  S   0  0.0   1:28.38 irqbalance
 2178 tomcat7   20   0 4345m  2.1g  5988  S   0 18.3  813:36.02 java
 3636 demo     20   0 17340  1352  968  R   0  0.0   0:01.00 top
 6363 intvoice  20   0 66228   11m 1832  S   0  0.1   0:07.52 python
 6365 intvoice  20   0 66228   11m 1832  S   0  0.1   0:07.46 python
```

Change Static IP Address

Replace x with your new IP address and netmask:

```
sudo ifconfig eth0 xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx
```

```
demo@ivdemo50:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:15:5d:5a:81:57
          inet addr:192.168.90.149  Bcast:192.168.90.255  Mask:255.255.255.0
          inet6 addr: fe80::215:5dff:fe5a:8157/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:24028 errors:0 dropped:0 overruns:0 frame:0
          TX packets:827 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9413350 (9.4 MB)  TX bytes:98281 (98.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:86405 errors:0 dropped:0 overruns:0 frame:0
          TX packets:86405 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:10578253 (10.5 MB)  TX bytes:10578253 (10.5 MB)

demo@ivdemo50:~$ sudo ifconfig eth0 192.168.90.175
demo@ivdemo50:~$
```

Add Additional Hard Drive Disk Space

This assumes you're using Ubuntu and LVM (default for IV)

1. Shutdown the server, add new disks as appropriate and restart the server
2. To view devices and partitions, run: `cat /proc/partitions`
3. Create a new partition by running these commands in order:

```
sudo fdisk /dev/<newpartition>
```

```
n  
p  
1  
w
```

```
demo@ivdemo50:~$ cat /proc/partitions  
major minor #blocks name  
11      0    1048575 sr0  
 8      16   15728640 sdb  
 8      0   36700160 sda  
 8      1    248832 sda1  
 8      2         1 sda2  
 8      5   36448256 sda5  
252     0   32235520 dm-0  
252     1    4186112 dm-1  
demo@ivdemo50:~$ sudo fdisk /dev/sdb  
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel  
Building a new DOS disklabel with disk identifier 0xdb558daf.  
Changes will remain in memory only, until you decide to write them.  
After that, of course, the previous content won't be recoverable.  
  
Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)  
  
Command (m for help): n  
Partition type:  
  p   primary (0 primary, 0 extended, 4 free)  
  e   extended  
Select (default p): p  
Partition number (1-4, default 1): 1  
First sector (2048-31457279, default 2048):  
Using default value 2048  
Last sector, +sectors or +size{K,M,G} (2048-31457279, default 31457279):  
Using default value 31457279  
  
Command (m for help): w
```

4. Format New Partition

```
sudo fdisk /dev/sdb1
```

```
T
```

```
8e (Formats the partition to Linux LVM)
```

```
w
```

```
demo@ivdemo50:~$ sudo fdisk /dev/sdb  
  
Command (m for help): t  
Selected partition 1  
Hex code (type L to list codes): 8e  
Changed system type of partition 1 to 8e (Linux LVM)  
  
Command (m for help): w  
The partition table has been altered!  
  
Calling ioctl() to re-read partition table.  
Syncing disks.  
demo@ivdemo50:~$
```

5. check the volume group name:

```
sudo vgdisplay
```

6. add the volume group

```
sudo vgextend vgroupname (ivdemo50-vg) /dev/<newpartition>
```

```
demo@ivdemo50:~$ sudo vgextend ivdemo50-vg /dev/sdb
No physical volume label read from /dev/sdb
Physical volume "/dev/sdb" successfully created
Volume group "ivdemo50-vg" successfully extended
demo@ivdemo50:~$
```

```
sudo lvdisplay
```

```
sudo lvextend -L +10G /dev/VolGroupName/Name (increases by 10Gb)
```

```
demo@ivdemo50:~$ sudo lvextend -L +10G /dev/ivdemo50-vg/root
Extending logical volume root to 40.74 GiB
Logical volume root successfully resized
demo@ivdemo50:~$
```

7. Resize the filesystem:

```
sudo resize2fs /dev/VolGroupName/Name
```

Collect and compress log files

The following command will extract the last 100,000 lines from the vrx log file (j2ws.log) into a compressed file named j2ws.log.bz2 in the user's home directory:

```
sudo tail -100000 /var/log/tomcat8/j2ws.log | bzip2 -vc > ~/j2ws.log.bz2
```

Appendix B. Reading data from S3 into Intelligent Voice

The preferred method is to mount the S3 bucket on the filesystem using s3fs and import from there.

Configure your bucket for secure access by IV. You will need an AWS access key and a secret access key.

Mounting the bucket

NOTE: to use S3 to import, export, or as a cache for the application server, the mount point must be owned by the same user account as the application server (e.g. tomcat8 or tomcat). Use this user id and group id in the mount command.

Add your access key & secret access key to `/etc/passwd-s3fs`, which should look like this:

```
AWS_ACCESS_KEY:SECRET_ACCESS_KEY
```

And protect the file like this:

```
sudo chmod 600 /etc/passwd-s3fs
```

To get tomcat8's user id and group id (user 109, group 116 in this case):

```
user@server:~$ grep tomcat8 /etc/passwd
tomcat7:x:109:116::/usr/share/tomcat8:/bin/false
```

Temporarily mount it with the following command:

```
sudo s3fs -o allow_other,uid=109,gid=116,umask=227,use_cache=/tmp/cache
bucketname /mnt/s3 -o passwd_file=/etc/passwd-s3fs
```

To configure a permanent mount point, add a line like this to the `/etc/fstab` file

```
s3fs#bucketname /mnt/s3 fuse credentials=/etc/passwd-s3fs,
allow_other,uid=109,gid=116,umask=227,use_cache=/tmp/cache 0 0
```

Once the mount point has been configured in the `fstab`, it will mount when the server starts, or the following command can be used to mount:

```
sudo mount -a
```