Speech Transcription Platform Installation Instructions

This document minimally describes the process of installing the speech transcription platform (STP) starting with a freshly installed operating system for the purposes of demonstrating a working proof-of-concept system and basic system evaluation on a single host machine. Since installing the various components on a single host machine is not a typical setup, this document is not intended to reflect best practices or offer advice or suggestions for installations in different scenarios. The instructions as set out in this document have been tested and verified to work on a host machine installed inside the VirtualBox virtualisation software and thus this is the recommended way to perform the evaluation of the system.

Some basic system requirements are:

- At least 4GB of RAM to build and install the system. **(required)**
- 2 or more CPUs to build and run the system **(recommended)**
- At least 8GB of RAM to run the speech recognition service **(required)**
- An internet connection throughout the installation process **(required)**

After setup, if the host machine is rebooted, the docker containers need to be rerun **in the following order** in order to resume testing/evaluation:

1. Speech server docker (Section 9)
2. Services docker (Section 11)
3. Speech scheduler (Section 12)
4. Application server docker (Section 13)

1. Install Ubuntu 16.04

Use a web browser to navigate to the Ubuntu 16.04.2 LTS website -- https://www.ubuntu.com/download/desktop -- and download the install media. Use the USB or DVD installation method to install Ubuntu 16.04.2 to the host server.

2. Prepare Ubuntu 16.04 environment

Install the following software packages from the standard Ubuntu repositories as follows using `sudo` (you will need to input your password):

```
$ sudo apt-get clean all
$ sudo apt-get update
$ sudo apt-get install git python-bcrypt docker.io apache2 curl tmux
python-requests chromium-browser virtualenv build-essential gcc
```

3. Configure Docker (optional)

Docker files are stored on the root partition by default which in general has limited space. In this section we are going to change the docker image location to your home partition. If your root and home are on the same partition you may ignore this step.

Stop the running docker service:

```
$ sudo service docker stop
```

Edit the `/etc/defaults/docker` file and add the following option:

```
DOCKER_OPTS="-g /home/docker"
```

Create a new docker location on the home partition:

```
$ sudo mkdir /home/docker
```

Restart the docker service:

```
$ sudo service docker start
```

4. Create and configure user

Create a platform user named **stp** and give it “sudo” and “docker” access as follows (you will be prompted to input your password and create a password for the new user):

```
$ sudo addgroup --gid 1050 stp
$ sudo adduser --uid 1050 --gid 1050 --home /home/stp --shell /bin/bash stp
$ sudo adduser stp sudo
$ sudo adduser stp docker
```

This user has **UID** (user identification) and **GID** (group identification) of **1050**. You will need to configure these IDs when running the docker containers. If these IDs are already taken on your system, select available ones and adapt subsequent instructions accordingly.

5. Login as user: stp

On the host server log in as **stp**. All further steps assume that you are logged in as this user.

6. Download the platform source

As user **stp** create a workspace directory (this will be `/home/stp/work/`) and download the platform components to sub-directories:

```
$ mkdir ~/work
$ cd ~/work
$ mkdir gui app services docs
$ pushd gui
$ git clone https://bitbucket.org/ntkleynhans/stp_ui.git
$ cd stp_ui
$ git checkout parly
$ popd
$ pushd app
$ git clone https://bitbucket.org/ntkleynhans/parliament_platform.git stp
$ popd
$ pushd services
$ git clone https://bitbucket.org/ntkleynhans/tech_services.git
$ popd
$ pushd docs
$ git clone https://bitbucket.org/ntkleynhans/stp_docs.git
$ popd
```

7. Configure Apache web-server

The application and speech servers require Apache2 which should already be installed (see step 2). Next, enable a few Apache modules:

```
$ sudo a2enmod proxy
$ sudo a2enmod proxy_http
$ sudo a2enmod proxy_balancer
$ sudo a2enmod lbmethod_byrequests
```

8. Install the speech server

We are first going to build the speech server. Go the the source directory and link the docker build script:

```
$ cd ~/work/services
$ ln -s tech_services/install/Dockerfile
```

To build the docker image run, with your UID and GID (see section 4, **create and configure user**):

```
$ docker build -t speech --build-arg UID=1050 --build-arg GID=1050 .
```

Next, we are going to use the database creation tools to create a few databases located in `~/stp/`. For the root password we are going to choose **root123** but you should use the password of your choice:

```
$ mkdir ~/stp
$ cd ~/work/services
$ ./tech_services/speech_server/tools/authdb.py ~/stp/speech_admin.db root123
$ ./tech_services/speech_server/tools/authdb.py ~/stp/speech_auth.db root123
```

Next, setup the services databases and services:

```
$ ./tech_services/speech_server/tools/servicedb.py ~/stp/speech_services.db
```

Setup the jobs databases:

```
$ ./tech_services/speech_server/tools/jobsdb.py ~/stp/speech_jobs.db
```

Add a user (username `appserver` and password `apppass`) to the speech server using the following command:

```
$ ./tech_services/speech_server/tools/adduser.py ~/stp/speech_auth.db appserver
apppass
```

NOTE: You must remember these speech server login credentials as we need to add them to the application server configuration file.

Edit the hosts Apache configuration file (`/etc/apache2/apache.conf`), as root, and add the following `ProxyPass` commands:

```
ProxyPass "/speech" "http://127.0.0.1:9950/wsgi"
ProxyPassReverse "/speech" "http://127.0.0.1:9950/wsgi"
```

We are going to run the speech server on port 9950 and forward normal port 80 HTTP traffic on /speech to 9950. So `http://127.0.0.1/speech` will be forwarded to `http://127.0.0.1:9950/wsgi`

Restart the Apache service for the changes to take effect:

```
$ sudo service apache2 restart
```

9. Start the speech server

With the speech server docker container built and speech services added to the database you can start the speech server:

```
$ docker run --name speech --env UID=1050 --env GID=1050 --env SO_SNDTIMEO=600
-v ~/stp:/mnt/stp -d -p 9950:80 speech:latest
```

Optionally, one can test whether the speech server is working by logging in and out using `curl` (**Note:** Once you have successfully logged in using curl, **you must log out again** before continuing to install and run the application server):

Logging in:

```
$ curl -i -k -v -H "Content-Type: application/json" -X POST -d '{"username":
"appserver", "password": "apppass"}' http://127.0.0.1/speech/jobs/login
```

Which should result in a JSON message with a random token, e.g.:

```
{"templogin": false, "token": "MWI0Y2U3NWMtOTY2OS00Yzk2LThmYTMtNjYxMmMzZGFlMTNl"}
```

Logging out:

```
$ curl -i -k -v -H "Content-Type: application/json" -X POST -d '{"username":
"appserver", "password": "apppass"}' http://127.0.0.1/speech/jobs/logout2
```

Which should result in a JSON message:

```
{"message": "User logged out"}
```

10. Install Sun Grid Engine

The speech scheduler requires the Sun Grid Engine (SGE) to queue requested speech jobs. Install the SGE gridengine packages:

```
$ sudo apt-get install gridengine-master gridengine-client gridengine-common
gridengine-exec
```

During installation you'll get the following prompts:

- **`Postfix Configuration`** -- select "Local only" and set system mail name to your domain name or for example "mydomain.org"
- **`Configuring gridengine-common`** -- for Configure SGE automatically: select "yes"

- **Configuring gridengine-common** -- the SGE Cell name: can be left as "default"
- **Configuring gridengine-client** -- set SGE master hostname to "localhost"

Fix the /etc/hosts file and restart SGE

Edit `/etc/hosts` file and add your system's ***hostname*** to the loopback IP address (127.0.0.1), if another entry exists for your hostname you should also remove that:

Change:

```
127.0.0.1   localhost
```

to

```
127.0.0.1   localhost   hostname
```

After editing this file you will need to restart the SGE gridengine-master service:

```
$ sudo service gridengine-master restart
```

To check for the gridengine-master service run the following (check for **sge_qmaster** process):

```
$ ps axf
 1072 ?        Sl     0:00 /usr/lib/gridengine/sge_qmaster
```

Setup the *qmaster*

For the following steps we will configure for the user on the host system named `stp`.

Add `stp` to the managers list:

```
$ sudo qconf -am stp
```

Add `stp` to the operators list:

```
$ sudo qconf -ao stp
```

Add a new scheduler configuration:

```
$ sudo qconf -Msconf
~/work/services/tech_services/scheduler/tools/sge_config/grid
```

Add a new host group:

```
$ sudo qconf -Ahgrp
~/work/services/tech_services/scheduler/tools/sge_config/hostlist
```

Add a new queue:

```
$ sudo qconf -Aq ~/work/services/tech_services/scheduler/tools/sge_config/queue
```

Add the administration and submit host:

```
$ sudo qconf -as localhost
$ sudo qconf -ah localhost
```

Setup the *execd*

Add a new exec host:

```
$ sudo qconf -Ae
~/work/services/tech_services/scheduler/tools/sge_config/hostexec
```

Add to a list attribute of an object:

```
$ sudo qconf -aattr hostgroup hostlist localhost @allhosts
```

Enable the queue:

```
$ sudo qmod -e speech.q@localhost
```

Restart the services

Run the following commands to restart the gridengine services:

```
$ sudo service gridengine-master restart
$ sudo service gridengine-exec restart
```

You should see the following processes running:

```
$ ps axf
 1072 ?        Sl     0:00 /usr/lib/gridengine/sge_qmaster
 4177 ?        Sl     0:00 /usr/lib/gridengine/sge_execd
```

11. Install the docker speech services

Copy Kaldi source

Copy the `kaldi.tar.bz2` archive to the speech services docker build location:

```
$ cp kaldi.tar.gz
~/work/services/tech_services/scheduler/speech_services/docker/
```

Build and run the services docker container

Move to the speech services docker build location:

```
$ cd ~/work/services/tech_services/scheduler/speech_services/docker/
```

Build the docker services by running the following command:

```
$ docker build -t services --build-arg UID=1050 --build-arg GID=1050 -f
docker-services .
```

To run the newly built docker service:

```
$ docker run -u dac --name services -v ~/stp:/mnt/stp -dt services:latest
/bin/bash
```

Build and run the aligner evaluation service container

Build the docker aligner (only needed for alignment evaluation) by running the following command:

```
$ docker build -t align_noskips --build-arg UID=1050 --build-arg GID=1050 -f
docker-align-noskips .
```

Run the aligner evaluation service:

```
$ docker run -u dac --name align_noskips -v ~/stp:/mnt/stp -dt
align_noskips:latest /bin/bash
```

Here we assume `~/stp` is where the speech resources will be placed.

Copy resources in place

Lastly, the alignment and recognition resources should be copied and extracted to `~/stp`.

Extract alignment and recognition resources:

```
$ cp align.tbz2 ~/stp && cd ~/stp/ && tar -xjf align.tbz2
$ cp recognize.tbz2 ~/stp && cd ~/stp/ && tar -xjf recognize.tbz2
```

12. Run the speech scheduler

Edit scheduler configuration

Make sure the `jobsdb, services` and `storage` variables point to the correct location.
Edit the JSON configuration:
`~/work/services/tech_services/scheduler/config/scheduler.json`

Example:

```
      "jobsdb": "/home/stp/stp/speech_jobs.db",
      "services":
"/home/stp/work/services/tech_services/scheduler/speech_services",
      "storage": "/home/stp/stp/jobs/",
      "DIR_TRANSLATE" : ["/mnt", "/home/stp"]
```

Launch the scheduler

The `python-requests` module should have been installed in step 2. You must start the scheduler on the host system to routinely monitor the speech queue. We would suggest using a terminal multiplexer such as `tmux` to start the scheduler in a persistent shell:

```
$ tmux
$ python scheduler.py config/scheduler.json
```

Then detach from `tmux` with `Ctrl-b d` (if unsure, refer to the tmux man page).

13. Install the platform application server

We are first going to build the speech server. Go the the source directory and link the docker build script:

```
$ cd ~/work/app
$ ln -s stp/install/Dockerfile
```

Update the platform dispatcher configuration
`~/work/app/stp/app_server/config/dispatcher.conf` so the application server can make use of the speech server.

Example:

```
"speechserver" : {
                  "username" : "appserver",
```

```
"password" : "apppass",
"login" : "jobs/login",
"logout" : "jobs/logout",
"logout2": "jobs/logout2",
"discover" : "jobs/discover"
}
```

To build the docker image run:

```
$ docker build -t stp --build-arg UID=1050 --build-arg GID=1050 .
```

Create databases

Set up two new authentication databases, for the main (projects, editor) and admin services, using the `authdb.py` tool :

```
$ python ~/work/app/stp/app_server/tools/authdb.py ~/stp/auth.db
$ python ~/work/app/stp/app_server/tools/authdb.py --rootpass root123
~/stp/admin.db
$ python ~/work/app/stp/app_server/tools/projectdb.py ~/stp/project.db
```

Fix Host Apache configuration

Edit (as root) the hosts Apache configuration file (`/etc/apache2/apache.conf`) and add the following `ProxyPass` commands:

```
ProxyPass "/app" "http://127.0.0.1:9999/wsgi"
ProxyPassReverse "/app" "http://127.0.0.1:9999/wsgi"
```

Restart the Apache service:

```
$ sudo service apache2 restart
```

Run application server docker images

First, determine the host server's ***external IP address.*** This can be don using the ifconfig command, example output for a server running in VirtualBox looks like this (with the external interface's IP in bold):

```
...
enp0s3Link encap:Ethernet  HWaddr 08:00:27:89:ae:be
      inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
      inet6 addr: fe80::4ed:be80:f325:d41b/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:211595 errors:0 dropped:0 overruns:0 frame:0
```

```
      TX packets:96644 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:180647400 (180.6 MB)  TX bytes:6215819 (6.2 MB)
...
```

Run the docker image making sure to use the external IP determined above when setting the APPSERVER and SPEECHSERVER variables as follows:

```
$ docker run --name stp --env UID=1050 --env GID=1050 --env
APPSERVER=http://10.0.2.15/app --env SPEECHSERVER=http://10.0.2.15/speech --env
SO_SNDTIMEO=600 -v ~/stp:/mnt/stp -d -p 9999:80 stp:latest
```

14. Install the GUI

Create a location in the Apache document root. This is generally in `/var/www/html` but may vary depending on your setup:

```
$ sudo mkdir /var/www/html/speechui/
```

Verify that the base URLs in `/home/stp/work/gui/stp_ui/js/vars.js` are set up correctly, they should all refer to the loopback IP address `127.0.0.1`, for example:

```
...
var ABASE_URL = "http://127.0.0.1/app/admin"
...
var PBASE_URL = "http://127.0.0.1/app/projects"
...
var EBASE_URL = "http://127.0.0.1/app/editor"
...
```

Copy the GUI source code to the newly created location:

```
$ sudo cp -r /home/stp/work/gui/stp_ui/* /var/www/html/speechui/
```

Change ownership and permissions:

```
$ sudo chown -R www-data:www-data /var/www/html/speechui/
$ sudo chmod -R o-rwx /var/www/html/speechui/
```

Create an alias by editing the Apache configuration file `/etc/apache2/apache2.conf` and adding the following:

```
Alias /speechui /var/www/html/speechui
<Directory /var/www/html/speechui>
```

```
  Options -Indexes -FollowSymLinks
</Directory>
```

Restart the Apache service:

```
$ sudo service apache2 restart
```

Using the **Chrome Web browser** you can access the GUI :
http://127.0.0.1/speechui/home/

Note: the UI currently only supports Chrome.